



*Simplay CEC Explorer  
Instruction Manual*

*Version 2.71 2010/04/26*

*Simplay Labs, LLC*

## Preface

### Notice

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, NO WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

Simplay Labs, LLC and its affiliates disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information contained in this specification.

Copyright © 2010 by Simplay Labs, LLC. All rights reserved. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein. Unauthorized use or duplication prohibited. “Simplay”, “Simplay HD”, “Simplay Labs” and all associated logos are trademarks of Simplay Labs, LLC or its affiliates. “HDMI” and all associated logos are trademarks of HDMI Licensing, LLC. Third-party trademarks and service marks are property of their respective owners.

### Document Revision History

0.1	2008/01/16	Initial version
0.2	2008/03/07	Draft version
1.0	2008/04/25	Issue version, added hardware setup scheme
1.1	2008/07/01	added “package content”, “firmware configuration”, “quick start guide”
1.2	2008/07/07	chapter 1.2 new figures
1.3	2008/07/07	updates in chapter 2.3
1.4	2008/08/29	updates according to CEC Explorer Version 2.0
1.5	2008/09/09	updated 3.1.2.4 ... 3.1.2.6
1.6	2008/11/20	Added Chapter 3 (ATC Test Tool)
1.7	2008/11/22	Added Chapter 6.2 (Terms and Conditions)
1.8	2009/01/05	Removed information regarding serial connection
1.9	2009/02/06	Added “Options how to connect the Explorer”
2.0	2009/02/11	Added descriptions in chapters 3.x
2.52	2009/04/27	Completely revised version
2.53	2009/05/06	Chapter Network settings updated
2.60	2009/07/20	Revised network connection method
2.61	2009/08/25	Added cable labels description
2.70	2010/02/20	Updated CEC v1.4 contents, and added CDC CTS description
2.71	2010/04/26	Minor updates on the electrical testing section

# Table of Contents

<b>PREFACE</b>	<b>2</b>
Notice.....	2
Document Revision History .....	2
<b>TABLE OF CONTENTS</b>	<b>3</b>
<b>1 INTRODUCTION</b>	<b>6</b>
1.1 Scope 6	
1.2 Hardware Interfaces.....	7
<b>2 INSTALLATION</b>	<b>9</b>
2.1 Package Content .....	9
2.2 System Requirements .....	9
2.3 Quick Start Guide .....	9
2.4 CEC Explorer – Connection Modes .....	10
2.4.1 CEC Explorer connected directly to a test PC.....	10
2.4.2 CEC Explorer connected to LAN using DHCP.....	13
2.4.3 CEC Explorer connected to LAN using fixed IP address.....	13
2.4.4 Network Troubleshooting.....	14
2.5 Device Configuration.....	15
2.5.1 Performing Firmware Updates .....	16
2.5.2 Configuring the Simplay CEC Explorer network settings.....	18
2.5.3 Configuring the Simplay CEC Explorer for additional features.....	19
<b>3 SIMPLAY CEC EXPLORER APPLICATION</b>	<b>20</b>
3.1 The menu bar .....	22
3.1.1 Application menu .....	22
3.1.2 Action Menu.....	28
3.1.3 Test Scripting Menu .....	35
3.1.4 Auto Comm Menu.....	37
3.1.5 Help Menu .....	38
3.2 The tool bar.....	39
3.3 Header- and Opcode Data Group Box .....	40
3.4 Operand Data Group Box .....	41
3.5 Send Box.....	43
3.6 Help Box / Auto Response Box.....	44
3.7 The Pulse Sequence Graph .....	44
3.8 The Message History List Box .....	45
3.9 The Operands Edit Field.....	47
<b>4 SCRIPTING INTERFACE</b>	<b>49</b>
4.1 Introduction .....	49
4.2 Script Structure .....	49
4.3 The <TestExecution> tag.....	51
4.4 Script Execution Mode .....	52
4.5 Tag types and their attributes.....	52

4.5.1	<TestExecution> tag	52
4.5.2	<Routine> tag	53
4.5.3	<DefineVar> tag	53
4.5.4	<DoOperation> tag	54
4.5.5	<UserList> tag	54
4.5.6	<GetUserListEntry> tag	55
4.5.7	<DoSub> tag	55
4.5.8	<Return> tag	56
4.5.9	<Loop> tag	56
4.5.10	<Break> tag	57
4.5.11	<Continue> tag	57
4.5.12	<Exit> tag	57
4.5.13	<If/ElseIf/Else> tag	58
4.5.14	<AutoResponse> tag	59
4.5.15	<Extract> tag	59
4.5.16	<ChangeDeviceMask> tag	60
4.5.17	<LockPulseList> tag	60
4.5.18	<GetBitTime> tag	61
4.5.19	<UnlockPulseList> tag	61
4.5.20	<GetFrame> tag	62
4.5.21	<LogError> tag	62
4.5.22	<LogWarning> tag	63
4.5.23	<LogMessage> tag	63
4.5.24	<SendCecFrame> tag	64
4.5.25	<SetHPD> tag	65
4.5.26	<SetPhysAddr> tag	65
4.5.27	<SetPower> tag	66
4.5.28	<UserAction> tag	66
4.5.29	<UserInput> tag	67
4.5.30	<UserDecision> tag	68
4.5.31	<Wait> tag	69
4.6	System Variables	70
4.7	Cambered Brackets	70
4.8	Operations	71
4.9	Conditional Statements	71
<b>5</b>	<b>SIMPLAY ATC CEC TEST TOOL</b>	<b>73</b>
5.1	Introduction	73
5.2	The CEC ATC Test Dialog	73
5.2.1	Control- and Information Unit	73
5.2.2	The Tab Widget	78
5.3	The CDC ATC Test Dialog	86
5.3.1	Control- and Information Unit	86
5.3.2	The Tab Widget	90
5.4	Test Setup	95

5.5	Blue Boxes.....	97
5.5.1	The Critical Flag.....	99
5.6	Test Execution.....	99
5.6.1	Preconditions.....	100
5.6.2	Auto Responses.....	100
<b>6</b>	<b>APPENDIX</b>	<b>102</b>
6.1	Support.....	102
6.2	Terms and Conditions.....	102
6.2.1	FreeRTOS.....	102
6.2.2	STR91x Firmware Library.....	102
6.2.3	LWIP Ethernet Stack.....	102
6.3	Abbreviations.....	103
6.4	List of Figures.....	105
6.5	List of Tables.....	107
6.6	References.....	107

# 1 Introduction

## 1.1 Scope

Current devices in the market are promoted with the benefit of 'CEC-usage', that is devices interact with each other via the *Consumer Electronics Control* line (according to the HDMI Specification Version 1.3a). This manual describes the usage of the 'Simplay CEC Explorer' which enables the user to test devices for their CEC capabilities.



Figure 1: Front Panel of CEC Explorer Hardware



Figure 2: Back Panel of CEC Explorer Hardware

## 1.2 Hardware Interfaces

### Front Panel:

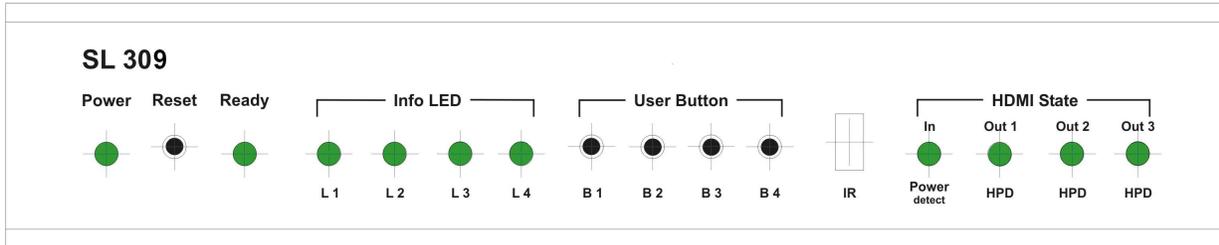
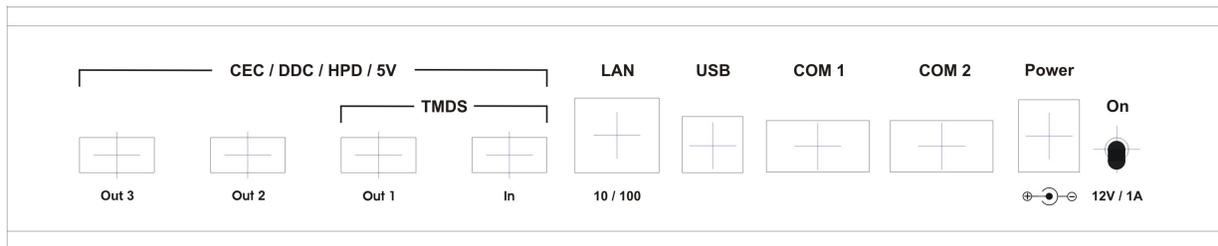


Figure 3: Schematic presentation of CEC Explorer's Front Panel

- Power: LED indicating the device's power state
- Reset: Reset button
- Ready: LED indicating the device's operating state
- L1 ... L4: LEDs indicating internal system states
- B1 ... B4: Buttons with context dependent functions
- IR: Infrared sensor receiving commands from an infrared remote control
- Power detect: LED indicating the detection of a 5V power signal from a connected DUT at the CEC Explorer's HDMI In port
- HPD Out1...Out3: LEDs indicating that the Hot Plug Detect (HPD) signals of the HDMI Out1...Out3 ports are active

## Back Panel:



**Figure 4: Schematic presentation of CEC Explorer's Back View**

- HDMI port In: to be connected to a source device, e.g. a DVD player
- HDMI port Out1: to be connected to a sink device, e.g. a TV set
- HDMI port Out2: used during ATC test for connection of an oscilloscope
- HDMI port Out3: currently unused
- LAN: Ethernet port (10/100 Mbit/s) used for connection to PC
- USB: currently unused (reserved for future use)
- COM1: currently unused
- COM2: used during ATC tests for connection of an oscilloscope
- Power: should be connected to the supplied 12V power adaptor only
- Power switch: top position is ON

## **2 Installation**

### **2.1 Package Content**

Please check if the package contains the following components:

- Simplay CEC Explorer Device SL309
- Simplay CEC Explorer Instruction Manual (electronic form)
- CD containing Simplay CEC Explorer application, LAN Update application and Instruction Manual
- Power Supply 110-230V / 12V 1.5A (MSH-Power-01)
- Power Cable US norm (PL-US-01)
- Power Cable EU norm (PL-Euro-01)
- HDMI Cable (PL-HDMI-01)
- Ethernet Cable (PL-Ethernet-01)
- HDMI-BNC Cable (S-HtB-01)
- COM-BNC Cable (S-CtB-01)

### **2.2 System Requirements**

The CEC Explorer application requires a PC with Ethernet interface running Windows XP. If you intend to use other operating systems, please contact Simplay Labs.

### **2.3 Quick Start Guide**

- Connect the Simplay CEC Explorer to your LAN using the enclosed Ethernet cable.
- Connect the enclosed power supply to the Simplay CEC Explorer device.
- Power on the Simplay CEC Explorer device, the power LED should lit up.
- Start the Simplay CEC Explorer application.
- Click on “Configure Application” (in the ‘Application’ menu item), change to the ‘Comm Settings’ tab and push the “Find Simplay devices” button. Select the target device according to the serial number of the Explorer’s hardware and close the dialog.
- Select “Open Port” (in the ‘Test Scripting’ menu item). On success, the GUI is enabled and ready for operation.

- Connect the DUT (device under test) to the HDMI In or HDMI Out1 port of the Simplay CEC Explorer depending on the type of device.
- The Simplay CEC Explorer may now act as an arbitrary sink or source device. In addition, it features a pass through mode from HDMI In to HDMI Out 1 port, enabling it to be hooked up in between source and sink device for simply observing CEC data traffic like in this example:

DVD-Player <-> Simplay CEC Explorer <-> TV-Set

## **2.4 CEC Explorer - Connection Modes**

The preferred network connection mode for Simplay CEC Explorer is “Direct Connection”, that is, to connect the Explorer to the PC using a regular (non-crossed) Ethernet cable. The Explorer does support network connection through corporate or private network. However, due to network traffic, the timing accuracy for CEC messages cannot be guaranteed. This may cause certain test cases to fail due to timing sensitive constraints for CEC.

### **2.4.1 CEC Explorer connected directly to a test PC**

A direct connection (without hub or switch) to a test PC is a preferred connection. If the Test PC supports MDI(X) (Medium Dependent Interface) at its Ethernet port, a regular (non-crossed) Ethernet cable may be used. If MDI(X) is not supported, a Crossover Ethernet cable is required (not included in the kit).

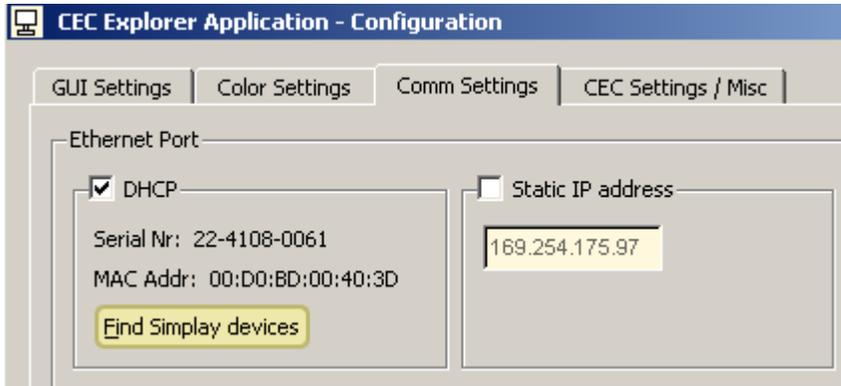
There are two modes the Test PC and CEC Explorer have to be configured, i.e.:

- a) AutoIP mode for the CEC explorer and DHCP mode for the Test PC
- b) Both configured for fixed IP address.

#### **2.4.1.1 Direct Connection using AutoIP mode and DHCP for the Test PC (preferred method #1)**

If the AutoIP and DHCP for the Test PC have been chosen for the direct connection method, the CEC Explorer network settings need to be configured for using AutoIP (see chapter 2.5).

The settings for the CEC Explorer application are the same as for DHCP (Figure 5).



**Figure 5: Application Configuration for usage of AutoIP**

To set the configuration and find CEC Explorer device, click on “Configure Application” (in the ‘Application’ menu item), change to the ‘Comm Settings’ tab and click on the “Find Simplay devices” button. Select the target device according to the serial number of the Explorer’s hardware and then close the dialog.

#### **2.4.1.2 Direct Connection configured for fixed IP addresses (preferred method #2)**

If the CEC Explorer is connected directly to a test PC, then using a fixed IP addresses for PC and CEC Explorer is the preferred solution. In this case, the Test PC does not search for a DHCP server and it is ready for communication immediately.

When using a connection based on a fixed IP address, make sure that the IP addresses of both PC and CEC Explorer belong to the same sub net, e.g.:

- CEC Explorer: 192.168.10.10
- PC: 192.168.10.11

Fixed IP addresses should be aligned with the topology of your local area network. Please contact your IT department for suitable IP addresses. Subsequently, the application will connect to the CEC Explorer hardware directly, without searching for the device based on UDP broadcasts.

Note that the configuration on the PC side is relevant only for a specific Ethernet port. Therefore, it is no problem to include the PC DHCP-based into a corporate network via WLAN and to connect a CEC Explorer to the PC using a cable if this port is configured for using a fixed IP address.

In Windows XP, the configuration of the Ethernet port can be checked via ‘Control Panel’ ==> ‘Network Connections’ ==> Properties of “Local Area Connection” (Figure 6). The ‘Properties’ button of the ‘Internet Protocol (TCP/IP)’ opens a sub dialog (Figure 7) to

configure the IP settings for DHCP (“Obtain an IP address automatically”) or for a fixed IP address.

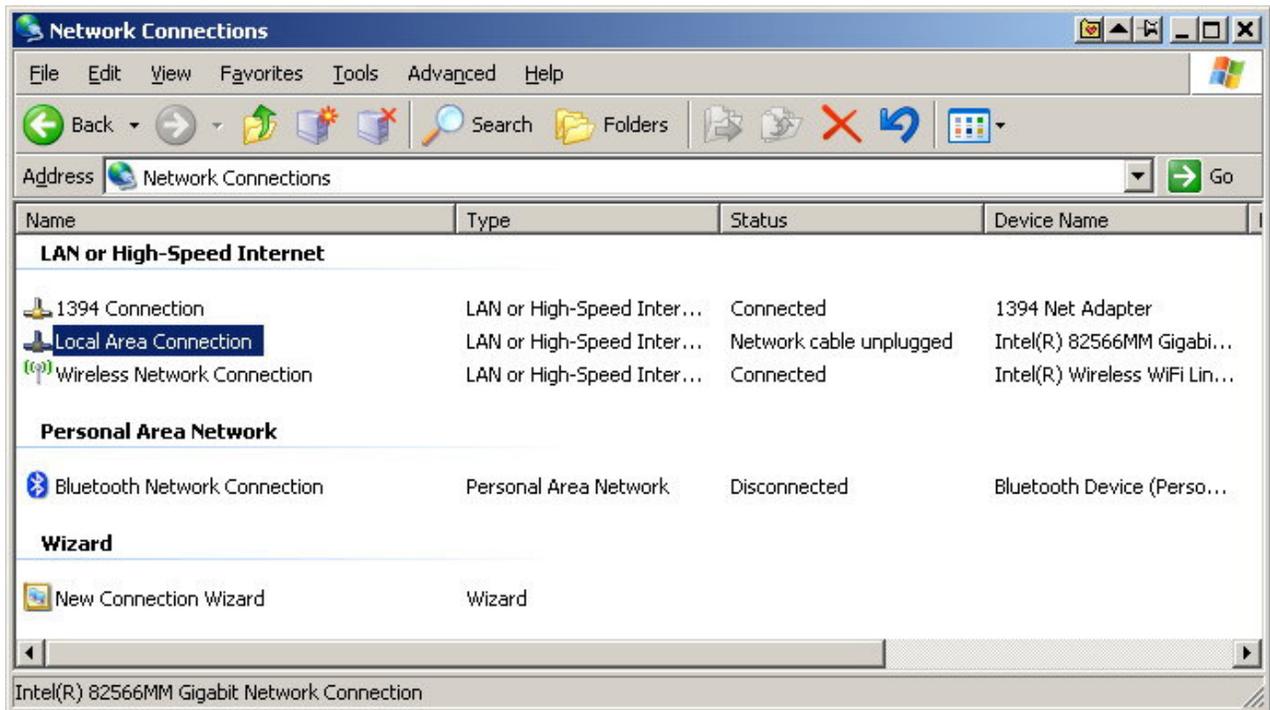


Figure 6: Network Connections settings in the Control Panel

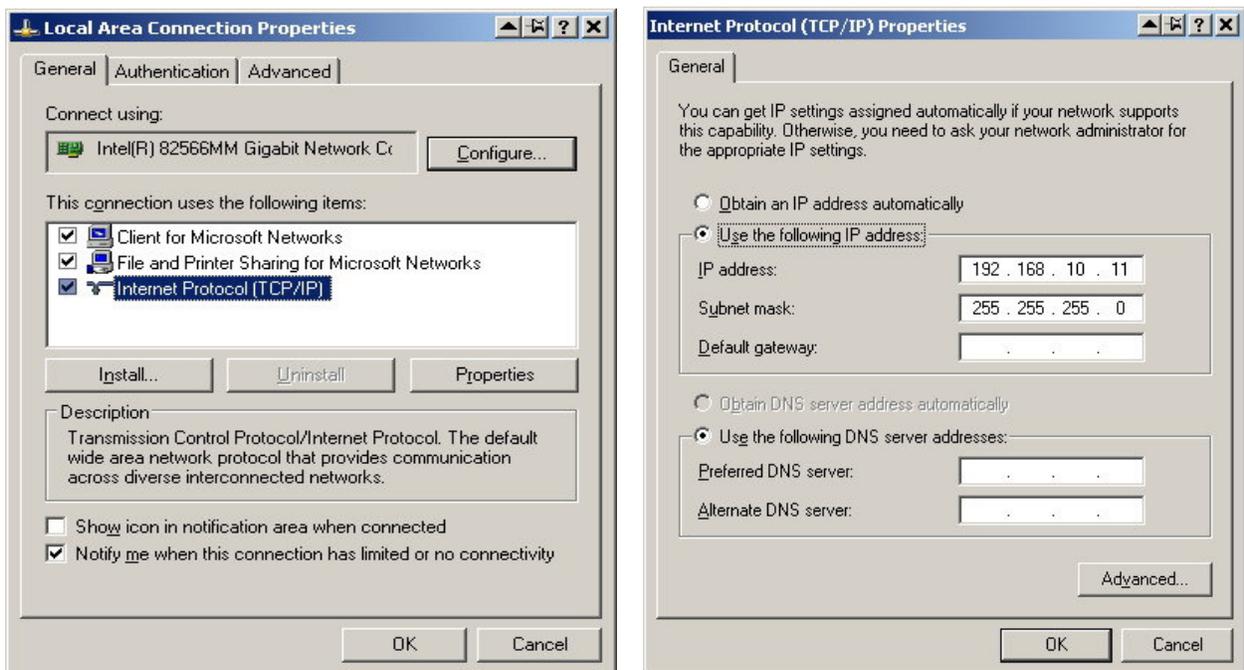


Figure 7: Properties of “Internet Protocol (TCP/IP)”

**Section 2.4.2 and 2.4.3 are the alternative connection methods. Please note that using any one of these connection method may affect the CTS test results due to local network traffic.**

### 2.4.2 CEC Explorer connected to LAN using DHCP

When delivered with factory settings, the CEC Explorer hardware is configured for DHCP which is the preferred connection. In this mode, the CEC Explorer is connected to a local area network (LAN) with an existing DHCP server using the supplied Ethernet cable. Note that the CEC Explorer application searches for CEC Explorer devices using UDP broadcasts. These broadcasts, however, may be blocked by network components such as routers or firewalls causing problems in terms of device detection. In that case, connect test PC and CEC Explorer to the same switch or hub, and connect the hub to your LAN, thus ensuring undisturbed communication between both devices.

In case your corporate network does not block the communication, any PC within the network running the CEC Explorer application, is able to connect to the CEC Explorer device.

### 2.4.3 CEC Explorer connected to LAN using fixed IP address

If the CEC Explorer shall be accessible by any PC in your corporate network, or in case the DHCP mode causes problems as described in 0, you should configure the CEC Explorer hardware using a fixed IP address (see Chapter 2.5). Please check a suitable IP address with your IT department. This IP address should be added as an exception to the corporate firewall so that any PC in the network can access it.

On using a fixed IP address, enter this address also into the CEC Explorer application using the 'Configure Application' menu item, 'Comm Settings' tab (**Error! Reference source not found.**). In this mode, the application will contact the CEC Explorer hardware directly, without searching for the device based on UDP broadcasts:



**Figure 8: Application Configuration for usage of a fixed IP address**

## 2.4.4 Network Troubleshooting

### DHCP

The most flexible way is to connect the CEC Explorer directly to a local area network, configured for DHCP mode. If you have problems to establish a stable connection between PC and CEC Explorer, there may be network components between these devices blocking the communication. In certain corporate network settings, direct communication between any two devices is prohibited. In most cases, connecting PC and CEC Explorer to the same hub or switch and connecting the hub to the local area network will help because the hub/switch directly routes the packets between PC and CEC Explorer.

### Fixed IP address

If you need access to the CEC Explorer from anywhere in your company or even from outside the company via the Internet, you have to configure the CEC Explorer for using a fixed IP address. In this case, please contact your IT department regarding the selection of a suitable IP address and regarding the configuration of your corporate firewall. The CEC Explorer application uses the following ports for the communication with the CEC Explorer device:

- UDP port 21324
- TCP port 21325

### Windows Firewall

It is preferred to switch off the Windows firewall because it delays the communication between PC and CEC Explorer, but you may use it if required. Note that opening the port in the CEC Explorer application may take up to 10 seconds if the Windows firewall is activated.

## 2.5 Device Configuration

Besides the normal *Operation Mode*, the Simplay CEC Explorer device can be set to a *Configuration* mode. This might be needed for:

- Updating the firmware
- Changing the feature bits
- Changing the network settings

In order to set the Simplay CEC Explorer to *Configuration-* or *Firmware Update* mode, either button B1, B2, or B3 has to be pressed during power-up. The button defined the network connection used during the configuration

- [Button 1] **selects the previously configured mode with the LAN-Update application.** i.e.:
  - 1) fixed IP address & net mask, or
  - 2) DHCP mode (factory setting), or
  - 3) Auto-IP mode.
- [Button 2] **enforces DHCP mode** (CEC Explorer connected to a LAN network).
- [Button 3] **enforces Auto-IP mode** (CEC explorer connected directly to the Test PC).

Press and hold the button before switching on the device, and do not release the button before the 'Network Ok' status is indicated (Table 1):

L1	L2	L3	L4	Status
x				Connecting to network
	X			DHCP: asking for IP address
X	X			Network Ok

Table 1: LED status at the CEC Explorer's front panel while powering up

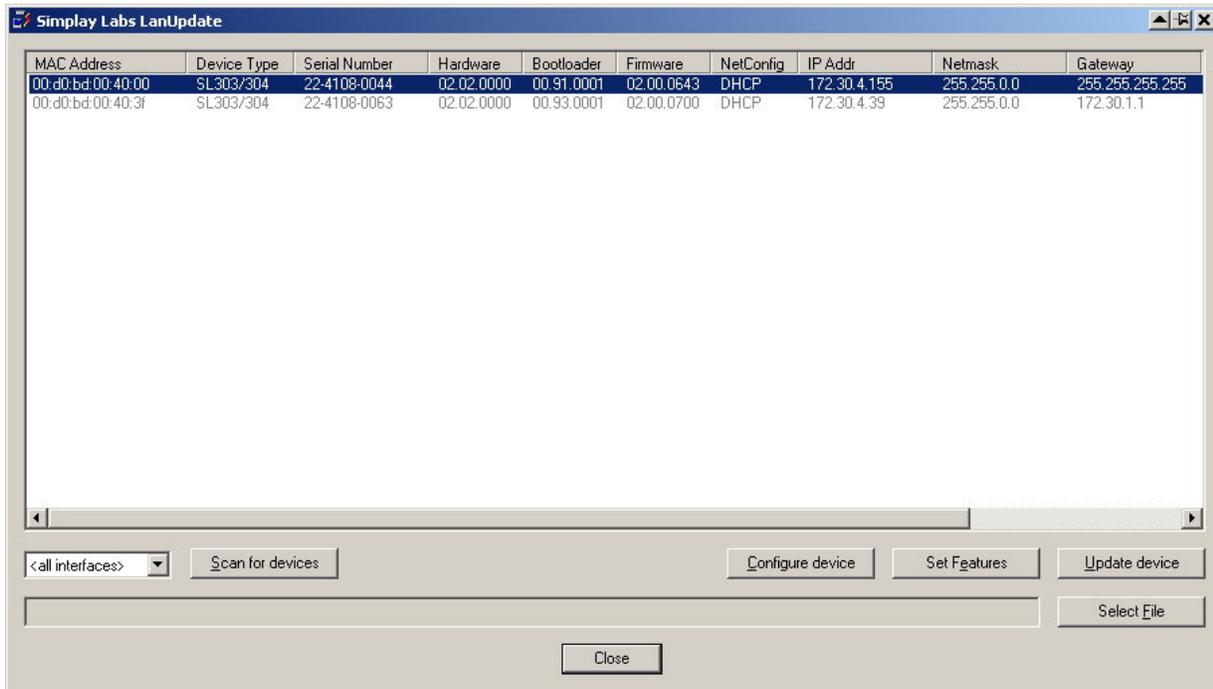
Possible error states:

- If connection to the network fails, only L1 lights up.
- If DHCP fails, L1+L2 lit after a long timeout. That is, the device did not get an IP address, but the LAN-Update application may still be able to access the device.

If the Simplay CEC Explorer is in *Configuration* mode (L1 + L2 are lit as described above), the device can be accessed by the LAN Update application, "LanUpdate\_1.2.exe", running in the local area network.

### 2.5.1 Performing Firmware Updates

- Set the Simplay CEC Explorer to *Configuration* mode as described in 2.5.
- Start the Simplay Labs LAN Update application (LanUpdate\_1.2.exe)



**Figure 9: LAN Update Application, main dialog**

- Press the button “Scan for devices” and select the device to be updated (Figure 9)
- Press the button “Select File” to select the firmware file (\*.fw) that shall be used for the update (Figure 10)
- Press the button “Update device” to start the update process. After completion, the device will restart automatically.

L1	L2	L3	L4	Status
x	x			Waiting for command
		x		Update running
x	x	x		Update successful
x	x			Update not successful

**Table 2: LED status at the CEC Explorer’s front panel while updating firmware**

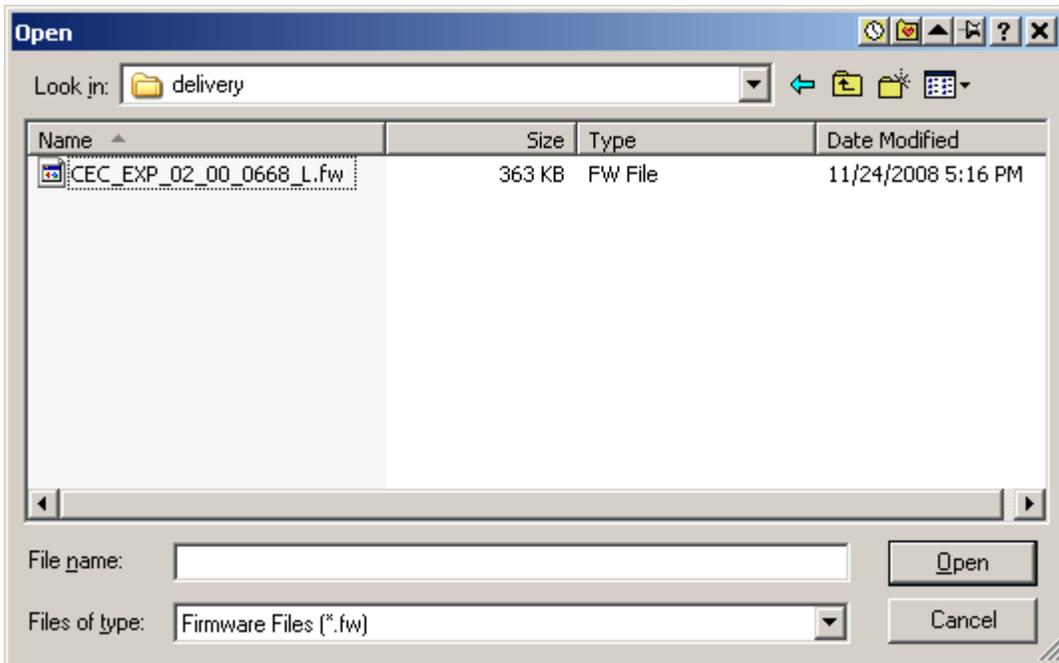


Figure 10: LAN Update Application, file dialog for firmware file

### 2.5.2 Configuring the Simplay CEC Explorer network settings

- Set the Simplay CEC Explorer to *Configuration* mode as described in 2.5.
- Start the Simplay Labs LAN Update application (LanUpdate\_1.2.exe)
- Press the button “Scan for devices” and select the device that shall be configured (Figure 9)
- Press the button “Configure device” to open the configuration dialog (Figure 11)

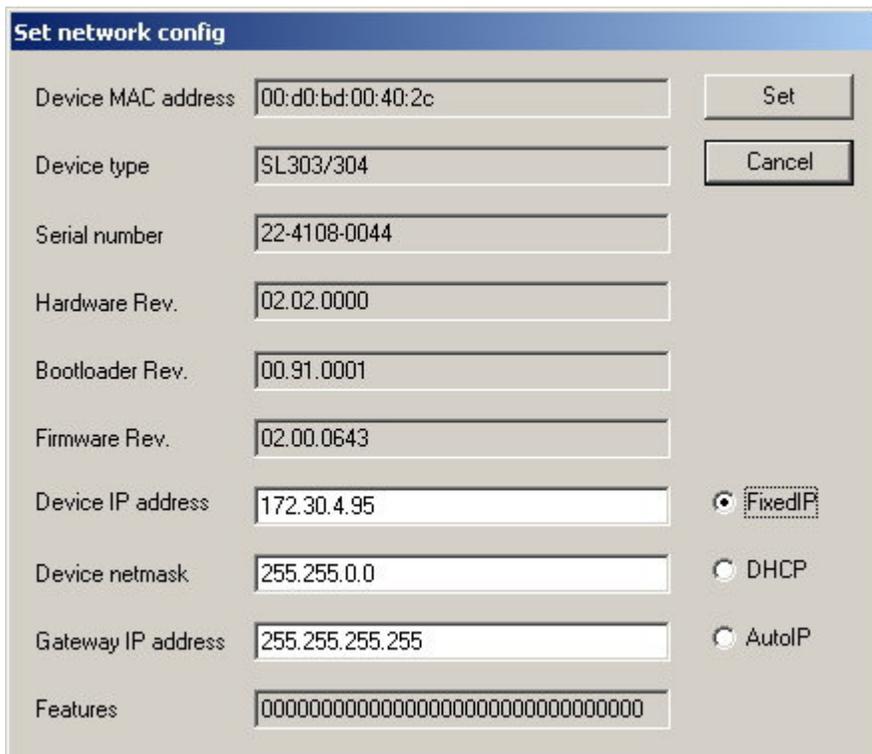


Figure 11: LAN Update Application, configuration dialog

- Select the check box “FixedIP”, “DHCP” or “AutoIP”
- In case of “FixedIP”, set the desired IP address and net mask (the Gateway IP address is currently not used)
- Press the button “Set” and confirm the following message box
- Wait until L1 + L2 are lit up again, then press OK
- Reset the CEC Explorer



### 2.5.3 Configuring the Simplay CEC Explorer for additional features

- Set the Simplay CEC Explorer to *Configuration* mode as described in 2.5.
- Start the Simplay LAN Update application (LanUpdate\_1.2.exe)
- Press the button “Scan for devices” and select the device to be updated (Figure 9)
- Press the button “Select File” and select the feature bit file (\*.fb) that shall be used for the update

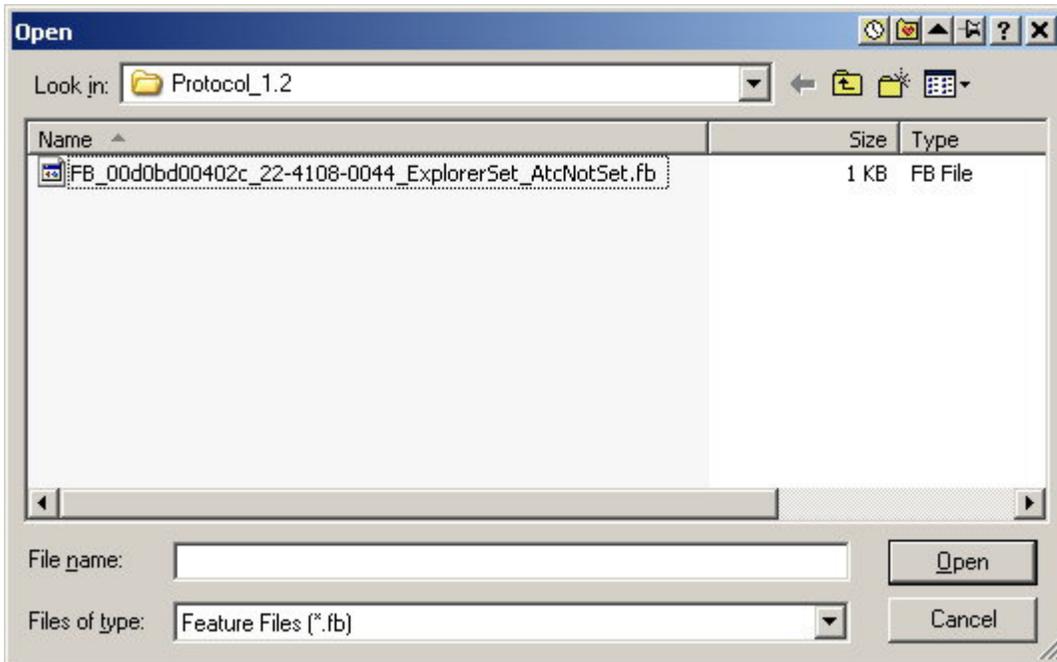


Figure 12: LAN Update Application, file dialog for feature bits

- Press the button “Set Features” to start the update process. A message box indicates the completion of the process
- Restart the device manually after the update.

Note: Feature Bit files are compiled for usage with a specific MAC address. Therefore, if you have several Simplay CEC devices, check that you’ve selected the correct one. If the file does not match the selected hardware, an error message will be displayed.



### 3 Simplay CEC Explorer Application

The Simplay CEC Explorer application is a powerful GUI that works only in context with the CEC Explorer hardware. It is used for monitoring the data traffic between interconnected devices on the CEC bus. In addition, it can be used for:

- conveniently creating CEC frames to be sent
- parsing of received CEC frames
- emulating all device types of a CEC network by means of their logical addresses
- setting the 5V power signal to each of the three HDMI Out ports, separately
- setting HPD signal and physical address to be assigned at the HDMI In port
- setting up auto communication (in terms of auto responses)
- setting up more complex communication flows (see chapter 4: scripting interface)
- logging, saving, and reloading of the message history
- a detailed analysis of pulse durations
- creating new or editing existing frames with individual bit timings
- performing ATC tests (if featured)

The main GUI is reflected in Figure 13. Implemented in C++, it is based on Trolltech's Qt toolkit (version 4.4.3), which is a cross-platform application development framework widely used for the development of GUI programs. Although the actual GUI sources can be used for both Windows- and Unix OS, up to now, the current version has only been developed for and tested on the Windows XP system.

The main GUI contains several elements represented by encircled digits in Figure 13. It comprises

- a menu bar (digit 1) with the five menu items 'Application', 'Action', 'Test Scripting', 'Auto Comm', and 'Help' (section 3.1)
- a tool bar (digit 2) for quick access of certain menu items (section 3.2)
- a box for specifying *header- and opcode data* (digit 3) of CEC frames (section 3.3)
- a box for specifying *operand data* (digit 4) of CEC frames (section 3.4)
- a box handling all aspects of data sending (digit 5) and mask setting (section 3.5)
- a box for providing auto communication settings (digit 6) or instantaneous help for GUI elements (section 3.6)
- a graphical presentation (digit 7) of the CEC frame's pulse sequence (section 3.7)
- a list box for the message history (digit 8) showing the main frame data since the port has been opened (section 3.8)
- an edit field for displaying the meaning of operands (digit 9) of selected frames (section 3.9)

Due to integrated splitters, one may customize the appearance of the main GUI to a certain degree. Just move the mouse in between the group boxes and change their sizes as required.

In addition, there are several sub dialogs that may be opened via the main dialog to get access to the other functionalities provided by the CEC Explorer application. These are:

- A pulse editor to create customized or edit existing pulse sequences (section 3.1.2.8).
- A configuration dialog for customizing menu items, pulse graph colors and communication settings (section 3.1.1.1).
- An XML-Tracer window that enables the user to track XML-encoded test scripts during script execution (section 3.1.3.4).
- A CDF dialog to perform ATC tests (section 5).

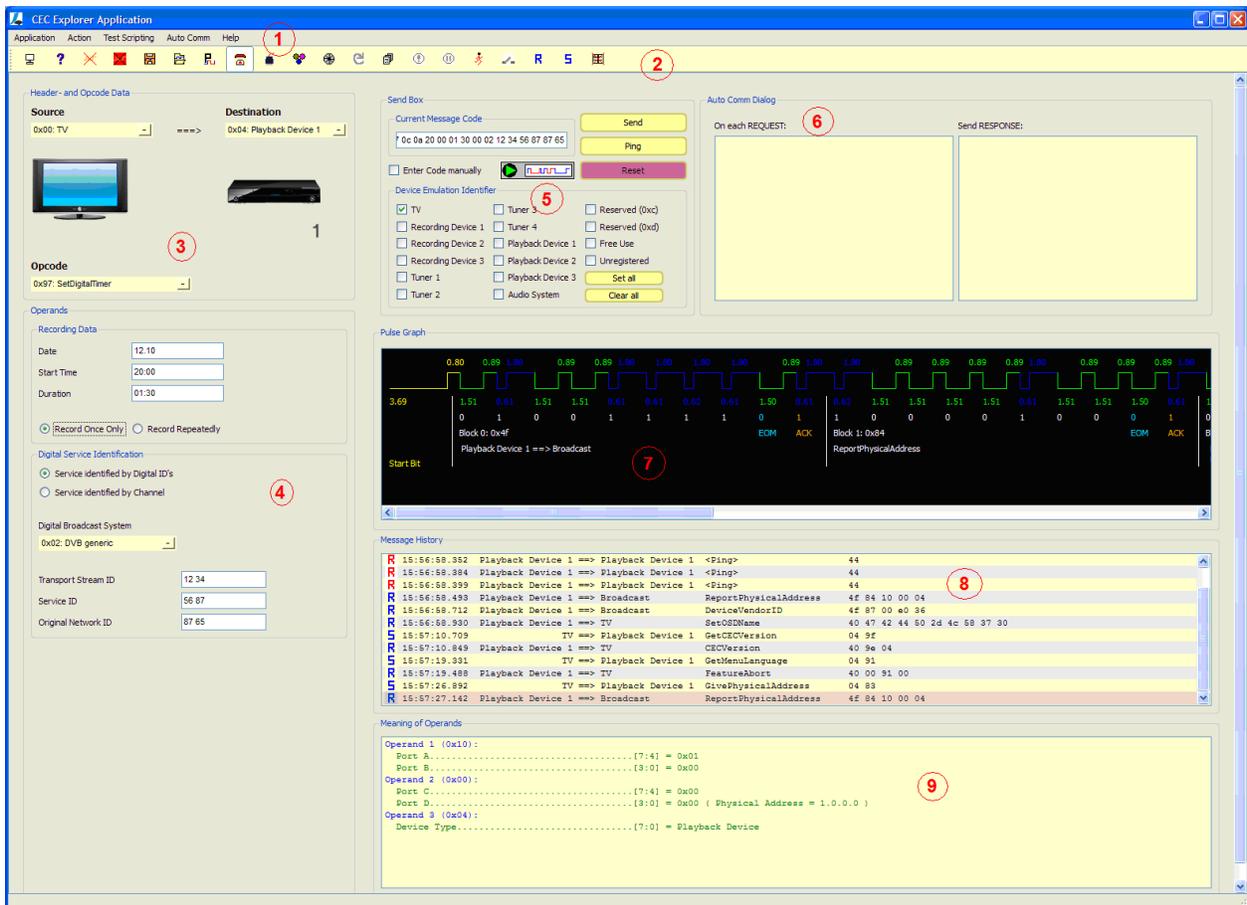


Figure 13: GUI of CEC Explorer application

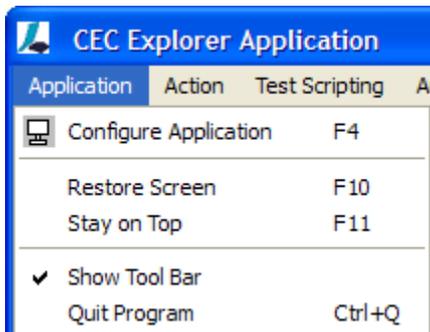
The CEC Explorer's functionalities are described in more detail within the following sections dealing with the individual menu items and their associated actions.

### **3.1 The menu bar**

Most of the menu bar's items are also accessible via toolbar buttons (section 3.2) or shortcut keys. The latter are shown in the corresponding figures of the menus.

#### **3.1.1 Application menu**

The *Application* menu (Figure 14) consists of five items, the shortcuts of which are displayed in context with the items.



**Figure 14: Application menu**

##### **3.1.1.1 Configure Application (Key F4)**

Pops up the *Configuration Dialog* with four tabs (Figure 16 - Figure 19) enabling the user to reconfigure certain aspects of the GUI and the communication link. These settings are stored in an ini file (*cec\_test.ini*) to be available at restart of the application. The ini file is located in the same folder as the application itself. If there is no such file, the application starts with default settings.

The '*Configure Application*' item's availability is independent of the port state, i.e. it may be opened at any time.

###### **3.1.1.1.1 GUI Settings Tab**

This tab enables the user to determine the view of all *Combo Boxes* in the main GUI. Furthermore, the visibility of tool bar items can be configured.

Many operands of a CEC frame are selected via pop-up list boxes (*Combo Boxes*), where each line contains the hex code and the corresponding description. By means of three Check Boxes, the user may determine whether the entries in all *Combo Boxes* shall be sorted by name (or by opcode), whether or not the opcodes shall be shown at all, and whether the opcode is shown at start or at end of line.

Notwithstanding, the behavior of individual *Combo Boxes* can also be set via their own context menu (Figure 15). However, individual changes are not stored in the ini file and must be reset at the application's restart when desired.

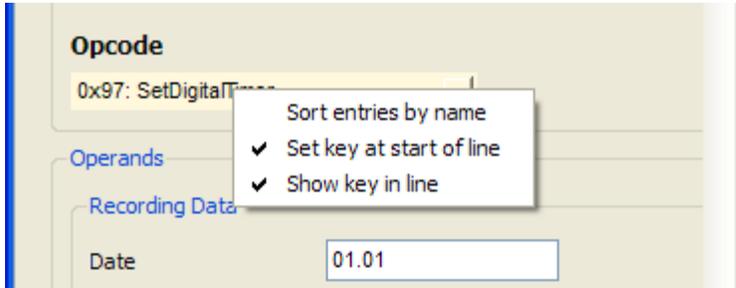


Figure 15: Individual context menu of Combo Box elements

The list box labeled 'Visible Tool Bar items' contains a number of checkable items representing all possible symbols in the application's tool bar. The check states of these items determine which of them are visible in the toolbar enabling the user to customize its tool bar. Items can be checked or unchecked by a single mouse click.

In addition, one may also select to show either small or big (i.e. labeled) toolbar icons.

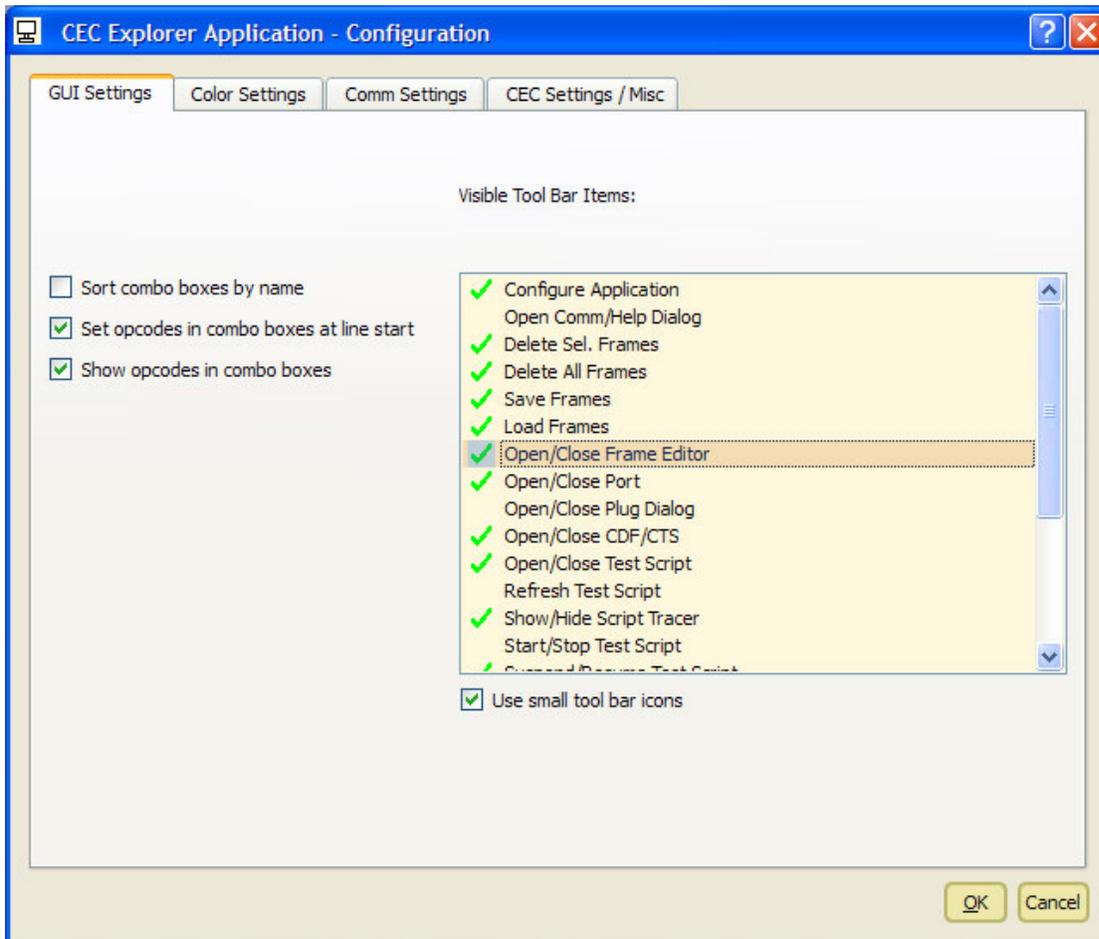


Figure 16: Configuration Dialog (GUI Settings tab)

### 3.1.1.1.2 Color Settings Tab

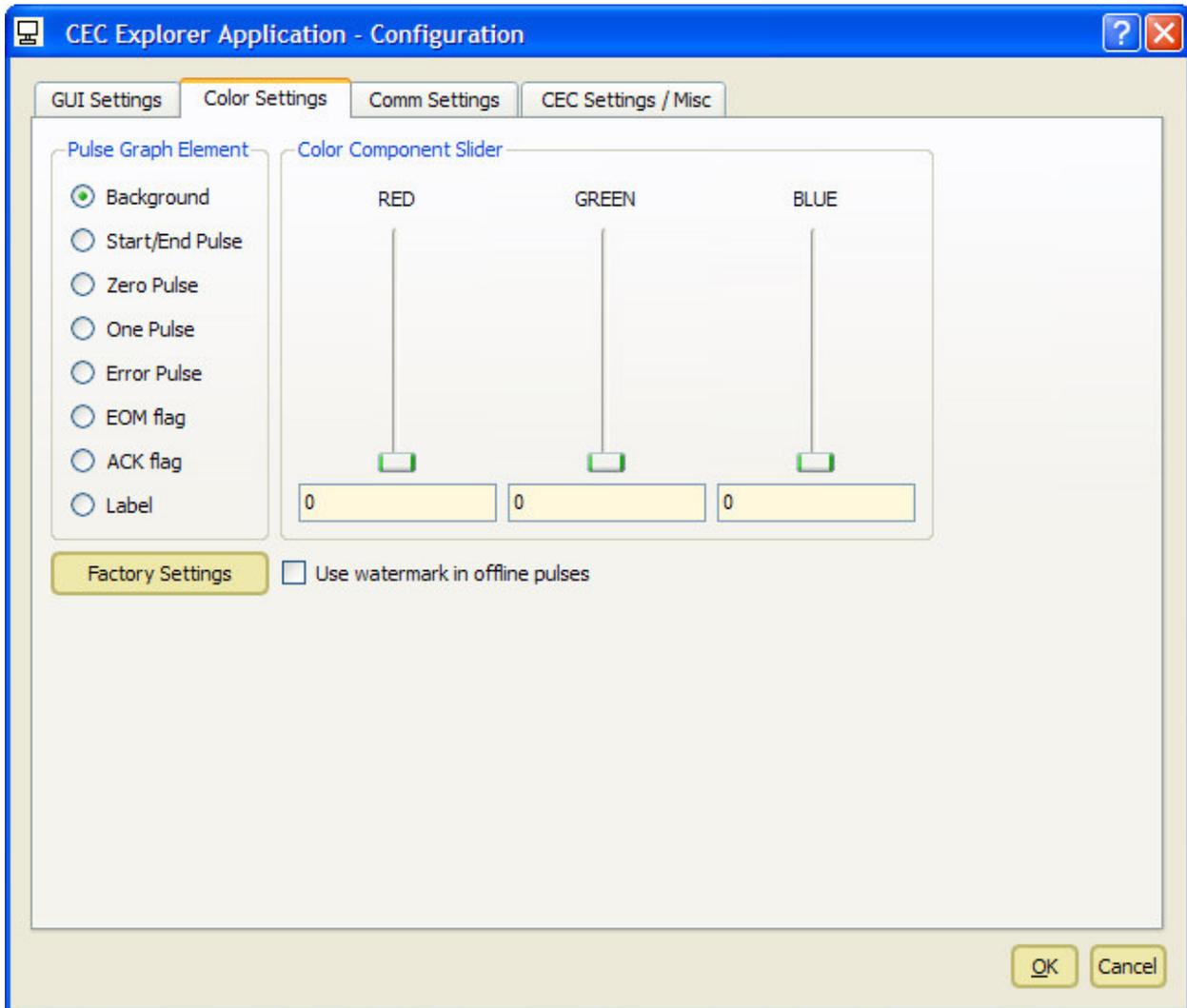


Figure 17: Configuration Dialog (Color Settings tab)

This tab enables the user to determine the view of the Pulse Graph (Figure 34:) in the main GUI. The *'Pulse Graph Element'* box determines the aspect to change and the sliders (or edit fields) can be used to determine the color of the selected component.

Clicking the *'watermark'*-Checkbox causes a little 'Offline' label to appear in the top left corner of each pulse sequence that couldn't be sent via the port.

The *'Factory Settings'* button resets the colors to the original values. Open a pulse sequence while changing the colors to directly check the new settings in time. Cancelling the dialog will reset colors to the settings at the dialog's pop up.

### 3.1.1.1.3 Comm Settings Tab

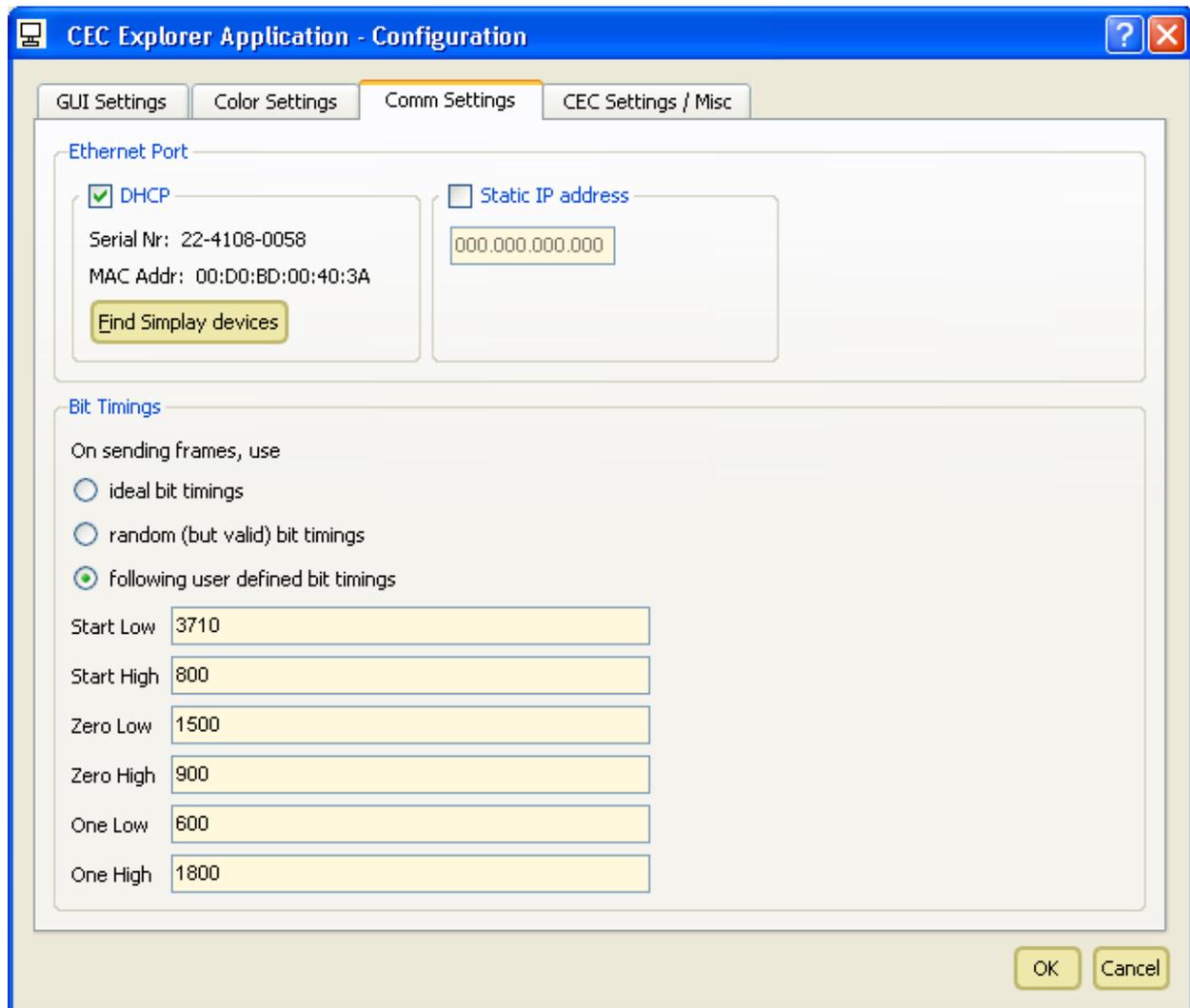


Figure 18: Configuration Dialog (Comm Settings tab)

This tab enables the user to set certain aspects of the communication as already partly described in section 2.4.

The interface selection is disabled when the port is currently open. If you use DHCP, select the target device by clicking on “*Find Simplay devices*”. If you work with fixed IP addresses for the CEC Explorer device and the PC, please enter the fixed IP address of the CEC Explorer into the field “*Static IP address*”.

The “*Bit Timings*” group box describes the default timings to be used for the normal send operation mode outside of the *Frame Editor* (section 3.1.2.8). That is, each pulse is sent with either ‘ideal bit timings’, random bit timings, or user defined bit timings. Note that the CEC Explorer creates random bit timings which are valid according to the specification [1]. The hardware, however, may slightly change the timings depending on the HDMI

cable length which cannot be influenced by the CEC Explorer application (thus, there might be time warnings for some pulses). Similarly, you cannot rely upon the other timing types since real timings do always slightly deviate from the desired ones. The deviation is in the range of several micro seconds, at maximum.

Note that each time value must be in between 10 and 14220 micro seconds (which is the unit used here).

### 3.1.1.1.4 CEC Settings Tab

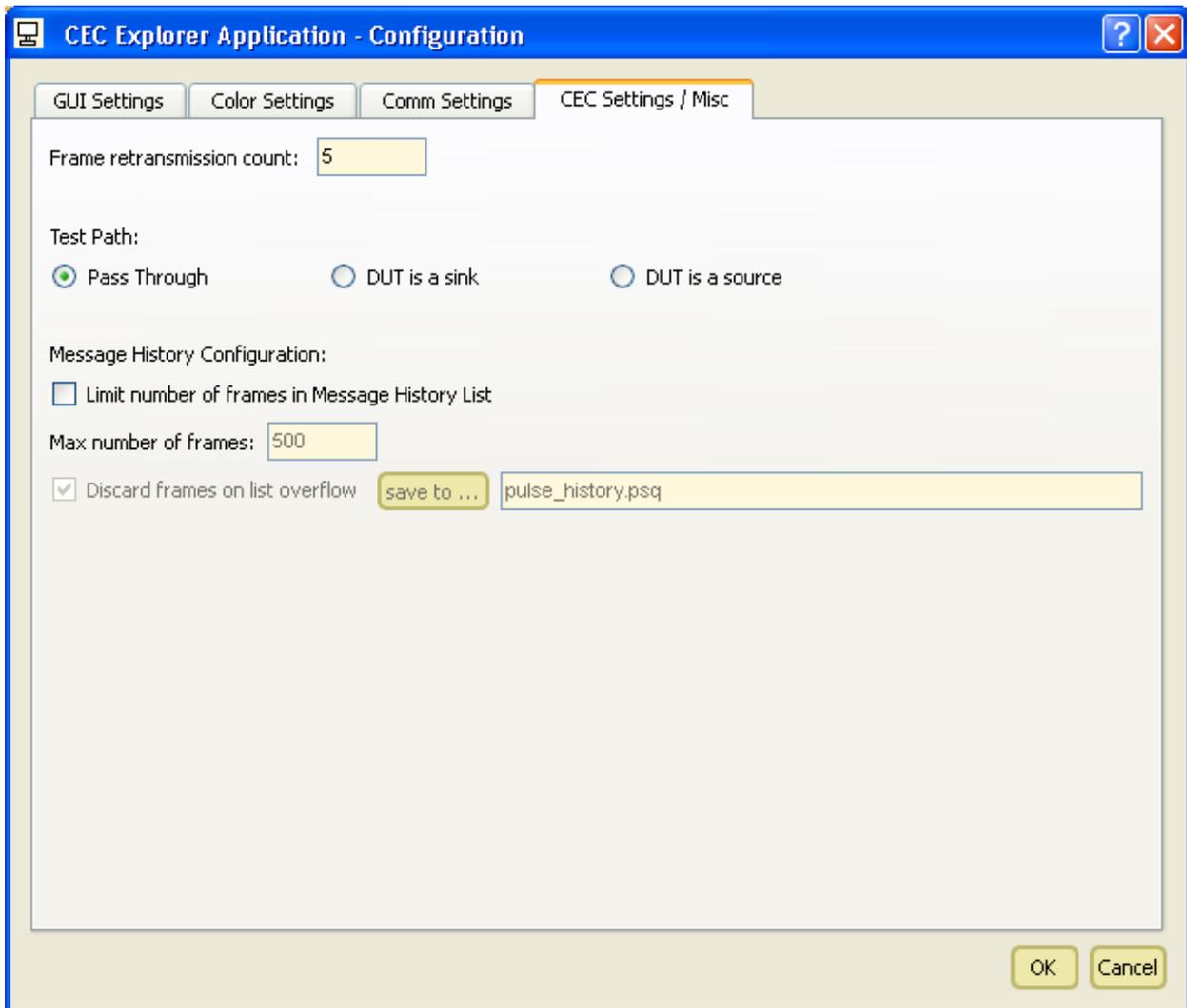


Figure 19: Configuration Dialog (CEC Settings tab)

This tab is used to set some CEC properties and the configuration of the *Message History List* (3.8).

The *Frame retransmission count* defines how often a retransmission shall be tried in case of errors. According to the HDMI CEC Specification this value should be up to 5. However, the value to be entered here may range from 0 to 255.

The *Test Path* determines the physical connections of the ports with regard to the CEC-, DDC-, HPD- and Power signal line. Please select the test path as follows:

- *Pass Through*: on using the CEC Explorer as a sniffer tool, connect a source to the HDMI In port and a sink to the HDMI Out1 port. In this mode, the CEC Explorer connects the CEC and DDC signals of the HDMI In and -Out1 port.
- *DUT is a sink*: use this mode for connecting a sink to the HDMI Out1 port.
- *DUT is a source*: use this mode for connecting a source to the HDMI In port.

Note that during ATC tests, the path settings mentioned above are configured automatically.

The check box "*Limit number of frames in Message History List*" determines the capacity of the *Message History List*. If checked, the number of shown frames is limited to the number reflected in the "*Max number of frames*" edit field. On overrun, the oldest frames are either discarded (if the check box "*Discard frames on list overflow*" is checked) or saved to the file defined in the "*save to*" field (if the check box is unchecked).

On setting the maximum number of frames to 0, no frames are written to the list. However, one can still determine to write frames to the specified file. Note that the limitation increases the overall performance of the CEC Explorer application.

#### **3.1.1.2      *Restore Screen (Key F10)***

As already mentioned, the single group boxes on the display are organized within a splitter enabling the user to simply drag the elements out of the application by means of the mouse device. This menu item restores the original screen design. Note that it is only available when the port is open.

#### **3.1.1.3      *Stay On Top (Key F11)***

This togglable item causes the application to become the top most window even if another application gains focus. The setting is not stored in the ini file.

#### **3.1.1.4      *Show Tool Bar***

This togglable action shows or hides the tool bar of the application. You may also access the item by clicking the menu bar with the right mouse button.

#### **3.1.1.5      *Quit Program (Ctrl + Q)***

Quits the program, and stores certain settings in the application's ini file. The ini file is used for the reconfiguration of the application on startup. Ini file settings concern window geometries, configuration settings, ATC test settings, and port settings, for example.

### 3.1.2 Action Menu

The *Action* menu (Figure 20) consists of eight items, the shortcuts of which are displayed in context with the items. Apart from 'Open Port', all these items are disabled when the port is not open.

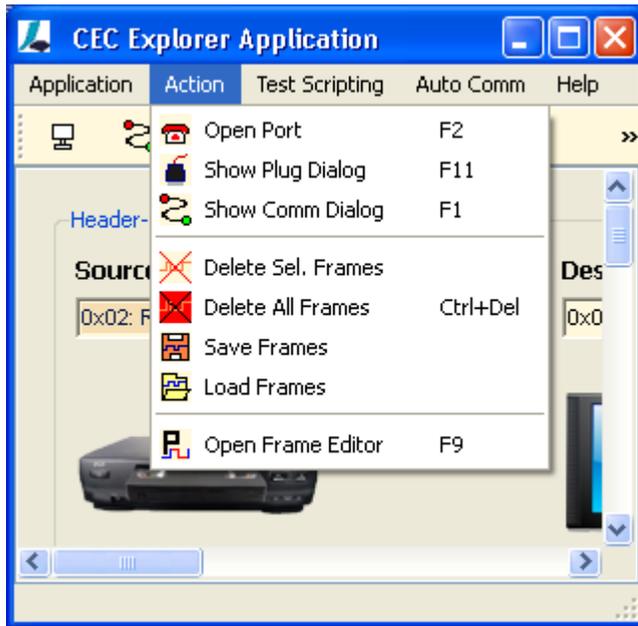


Figure 20: Action Menu

#### 3.1.2.1 Open/Close Port (Key F2)

This toggable item opens or closes the port in accordance with the port settings. If opening fails, the user is notified and may be prompted to check the connection or to perform a hardware reset. If opening succeeds, monitoring of the CEC line starts immediately, reflected by potential entries in the *Message History List*. In addition, all menu items are enabled which are associated with an open port

#### 3.1.2.2 Show/Hide Plug Dialog (Key F11)

This dialog shows the HPD- and 5V status of all CEC ports (Figure 21). Column 1 refers to the port type: one HDMI In and three HDMI Out ports. Column 2 shows the current status of the port, i.e. the power signal at the HDMI In- and the HPD signal(s) at the HDMI Out port(s). If there is a HPD signal at the Out ports, the assigned physical address is also reflected.

Column 3 enables the user to exercise control at the ports: checking the check boxes sets the HPD signal at the HDMI In port with the defined physical address in the transmitted EDID, and the 5V power signal(s) at the HDMI Out port(s).

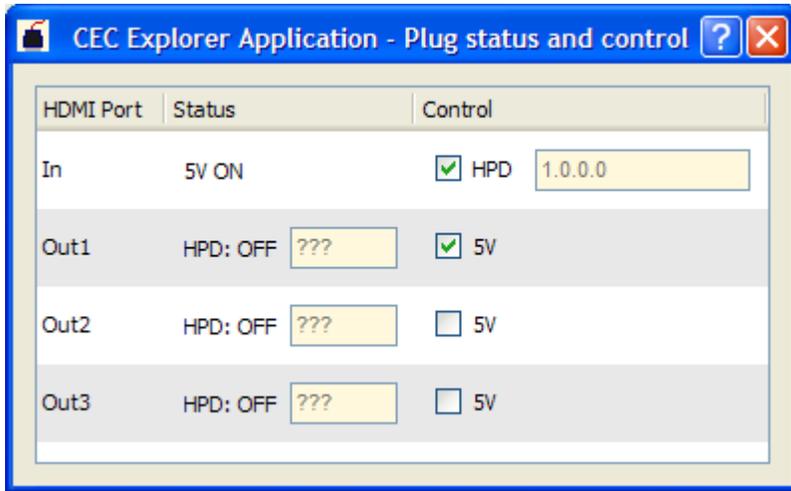


Figure 21: Plug Dialog

### 3.1.2.3 Show Comm/Help Dialog (Key F1)

This togglable action is used to either show the *Auto Comm-* or the *Help Dialog* at the top right corner of the application. The Auto Comm Dialog is reflected in Figure 22.

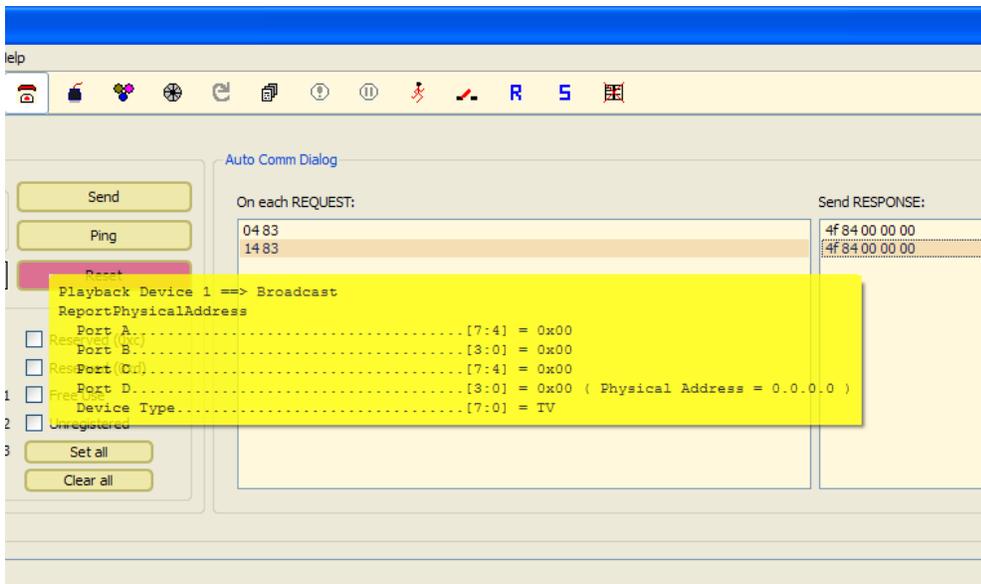


Figure 22: Auto Communication Dialog

It consists of two lists. The list on the left hand side contains all hex encoded messages the user wants to respond to. The list on the right hand side contains the desired response that is automatically sent out whenever the defined request of the same line has been received.

Note that the *Auto Communication* responds to both received and self-sent CEC frames enabling the CEC Explorer to automatically send a <User Control Released> frame after each <User Control Pressed> frame, for example.

A single request may occur more than once in the 'Request List' enabling the user to send a chain of different frames as response (in the order of appearance). Note that the usage of wildcards is possible in order to generalize incoming requests. The following wildcards can be used:

- \* stands for an arbitrary number of characters (request list only)
- ? stands for a 4-bit replacement (request list only)
- # stands for a variable that can be re-used in the response list, i.e. the corresponding character is used within the response.

Although the list contains the hex coded CEC frames only, the user may catch up on the message by checking the tool tip text of a line (indicated yellow in Figure 22). The handling of list entries is also discussed in section 3.1.4.

The Help Box (Figure 23) is alternatively shown in the top right corner of the dialog. It provides instantaneous help for most of the GUI elements. In this context, it is sufficient to hover the mouse cursor over an element and the Help Box is updated.

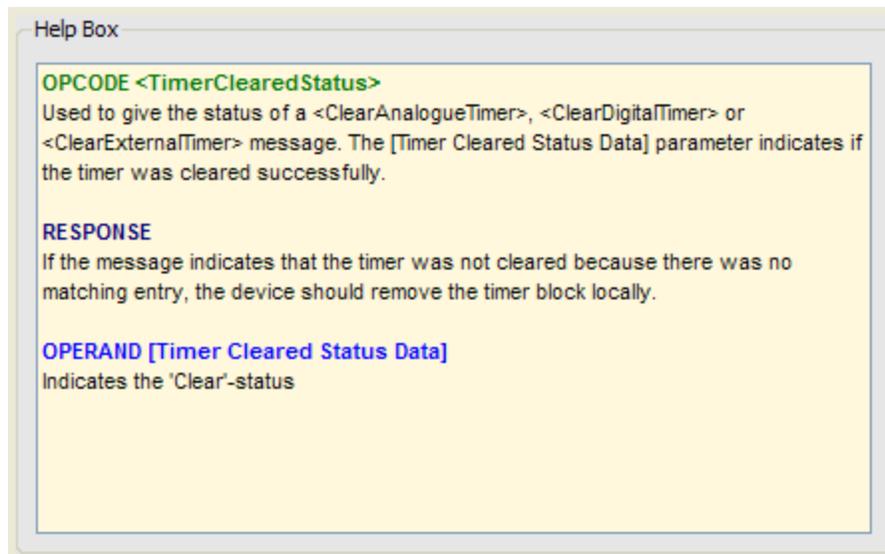


Figure 23: Help Box

There are two special notifications within the Help Box: Hovering over the *Message History List* reflects the current number of entries, and hovering over a red colored (erroneous) "*Current Message Code*" shows the nature of the warning/error.

#### **3.1.2.4 Delete Sel. Frames (Key Delete)**

The *Message History List* stores all sent and received messages since the port has been opened (in addition, it stores offline pulses that were never sent). The maximum number of entries can be configured (section 3.1.1.1.4) or is limited by available memory. Note that the performance of the CEC Explorer suffers from a list with too many entries (>2000).

A single or several entries can be selected by means of the keyboard or mouse device. Clicking this menu item then deletes the selected rows from the list. The Delete key has the same effect unless the GUI's focus lies on the *Auto Comm Dialog*. In that case, the selected Auto Comm entries are deleted instead.

#### **3.1.2.5 Delete All Frames (Key Ctrl + Delete)**

This command deletes all entries at once without the necessity to mark them in advance.

#### **3.1.2.6 Save Frames (Key Ctrl + S)**

This command saves either the selected or the complete entries of the *Message History List*. The user is requested in advance if a selection exists. The application saves the raw timing data as binary files (\*.psq for "pulse sequences"). Note that the key shortcut "Ctrl + S" is either linked to saving the *Message History List* or the *Auto Comm List* (depending on the current focus).

#### **3.1.2.7 Load Frames (Key Ctrl + L)**

This command loads previously saved pulse data from the psq-file format into the application. If the current *Message History List* contains entries, the user is queried for handling the existing pulses (save, discard or append new data).

Depending on the number of entries, loading a psq file may take some time. The '*Always show latest Frame*' flag is disabled for the time of loading in order not to force a permanent update of the list which would decrease performance.

Note that limiting the number of frames might interact with frame loading, i.e. the application may load more frames than the list is configured for, which eventually causes the application to immediately delete the first frames again after loading is finished. Also take care not to load a psq file which is currently used for collecting frames due to list size overrun. This would duplicate a corresponding frame sequence in the target file.

#### **3.1.2.8 Open/Close Frame Editor (Key F9)**

The *Frame Editor* (Figure 24) is a powerful sub dialog to create customized or edit existing pulses. It contains several group boxes which in turn contain several GUI elements.

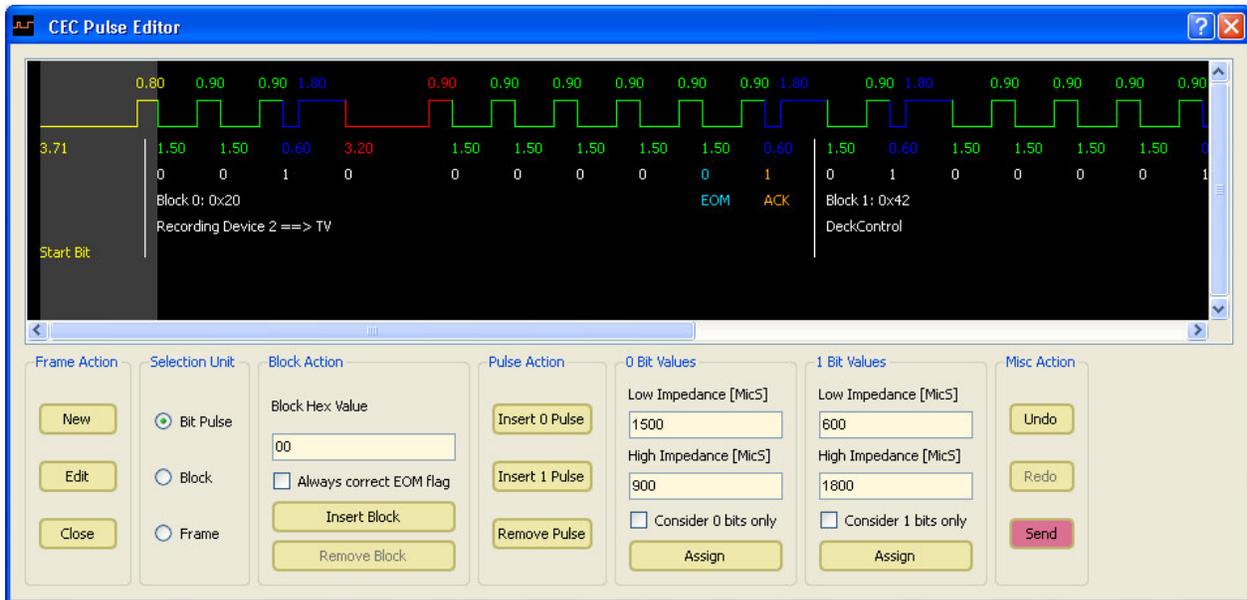


Figure 24: Frame Editor

### Frame Action:

- New: creates a new pulse sequence from scratch. It contains the start pulse only
- Edit: creates a copy of the currently visible pulse sequence in the main GUI
- Close: closes the current pulse sequence

### Selection Unit:

A certain unit of the pulse sequence is always selected. Here you can determine to either select a **bit pulse** (low + high pulse), a **block** (CEC block of 10 bit pulses in between vertical lines), or the **complete frame**. The actual selection can then be changed by using the cursor keys "Left" and "Right" which moves one bit pulse (or block) unit to the left or right, respectively. Use the Home- or End key to move to the start or end of the sequence. Use Ctrl + Cursor keys to move the bit pulse selection block-wise. You can also select a unit with the left mouse button. The selection is always moved to the middle of the *Frame Editor*, i.e. the scroll bar is adapting. Note that the *Frame Editor* can be resized in the horizontal direction only.

### Block Action:

Enables the user to insert or remove complete CEC blocks of 10 bit pulses. This works only in *Block-Selection* mode. Exception is, however, when the *Frame Editor* is in *Bit-Pulse-Selection* mode and the acknowledge- or start bit is selected: then you can insert a block to the right also (these restrictions are handled by the dialog itself which disables/enables the corresponding buttons). The value of the inserted block corresponds to the block hex value (in the above shown figure:

0x00). An input mask prevents entering invalid values. One may also insert a new block in *Frame-Selection* mode. This one is simply appended to the end.

Bit timings for new blocks are derived from the "*0 Bit Values*" and "*1 Bit Values*" Group Boxes (see below). Blocks are always inserted on the right of the selected block. The selection does not change. The check box 'Always correct EOM flag' causes the sequence to correct all EOM flags to 0 (apart from the last one unless it doesn't exist, i.e. the block is too short). When a block is removed, it is always the selected one, and the block that was formerly on the right from it, is the newly selected block (unless there was no block).

#### Pulse Action:

Enables the user to insert or remove single bit pulses (always on the right of the selected bit pulse). You can insert a 0- or a 1-pulse according to the bit timings defined in the next two group boxes (see below). Pulse Actions are restricted to the *Bit-Pulse-Selection* mode. The removed pulse is the selected one. The bit pulse on the right is the newly selected bit pulse (similar to block removal).

#### 0 Bit Values / 1 Bit Values:

Determines the timing of low- and high-impedance of a bit pulse. Clicking 'Assign' will assign these values to the current selection. If the check boxes 'Consider 0/1 bits only' are checked, only the pulses that already have the corresponding value, are considered within the selection. Assume, the user wants to change the value of all low impedances within existing 0-Bit-Pulses of a complete pulse sequence. Then one might select the complete frame, set the new value for the low impedance of 0-Bit-Pulses and check 'Consider 0 Bits only'. If you now click 'Assign', all corresponding values are changed. If the check box is unchecked, then all bit pulses would be changed to a 0-Bit-Pulse. In case the values are outside the specification for 0- and 1- bit pulses, the edit fields are colored red. However, the user is not prevented to assign erroneous data.

#### Misc Action:

Contains three buttons to *Undo/Redo* previous actions as usually performed in GUI applications. Currently, one might undo/redo up to 5 steps at maximum. The *Send* button directly sends the current frame of the *Frame Editor* according to WYSIWYS (What you see is what you send).

The *Frame Editor* is an '*Always on Top*' dialog that may be closed without losing the present data. Also note that vertical lines of the current pulse sequence can be dragged within certain limits by means of the left mouse button. Each change of the pulse sequence can be undone and each change automatically causes re-parsing of the complete sequence to get the new data description.

There are several shortcut keys for the *Frame Editor* navigation which are shortly summarized in Table 3:

Key Sequence	Effect
Cursor key right:	move selection to the right
Ctrl + Cursor key right:	move bit pulse selection 10 steps to the right
Cursor key left:	move selection to the left
Ctrl + Cursor key left:	move bit pulse selection 10 steps to the left
Home:	move selection to start of sequence
End:	move selection to end of sequence
Cursor key up:	assigns bit value 1 to the current selection (this can be a complete block or frame). Timings are retrieved from the corresponding edit fields, and the ' <i>Consider 1 bits only</i> ' check box is considered.
Ctrl + Cursor key up:	assigns bit value 1 to all 1-Bit-Pulses of the current selection (can be a complete block or frame). Timings are retrieved from the corresponding edit fields. If the check box 'Consider 1 Bits only' is checked, the Ctrl key has no use.
(Ctrl +) Cursor Key down:	as described for the 'up' key but 0-Bit Pulses are assigned.
Key 0:	inserts a 0 Pulse (if possible)
Key 1:	inserts a 1 Pulse (if possible)
Key 5:	inserts a block to the right (if possible)
Key Delete:	removes selection
Key F1:	creates a new pulse sequence from scratch
Key F2:	edits the current pulse sequence from the main GUI
Key F4:	sets focus to the 'Hex Value' field
Key F5:	sets focus to the '0 Bit Low Impedance' field
Key F6:	sets focus to the '0 Bit High Impedance' field
Key F7:	sets focus to the '1 Bit Low Impedance' field
Key F8:	sets focus to the '1 Bit High Impedance' field
Key F9:	sets focus to the <i>Frame Editor</i> dialog to enable other shortcut keys (otherwise, if a line edit field is in focus, key shortcuts are filtered by it)
Escape:	closes the pulse sequence
Return:	sends the pulse sequence directly
Ctrl + Z:	Undo last action
Ctrl + Y:	Redo last action

**Table 3: Shortcut keys for the Frame Editor**

Note that the bit timing values of the *Frame Editor* are not stored in the ini file.

### 3.1.3 Test Scripting Menu

The CEC Explorer enables the user to define a conditional test flow by means of a proprietary, XML-based language. The language contains all essential elements of a programming language which's why complex scripts can be written to control a device in several ways. The syntax of the script language is described elsewhere (section 4). This chapter is therefore restricted to the description of the seven menu items and their associated actions (Figure 25). Note that all these items are disabled when the port is not open.

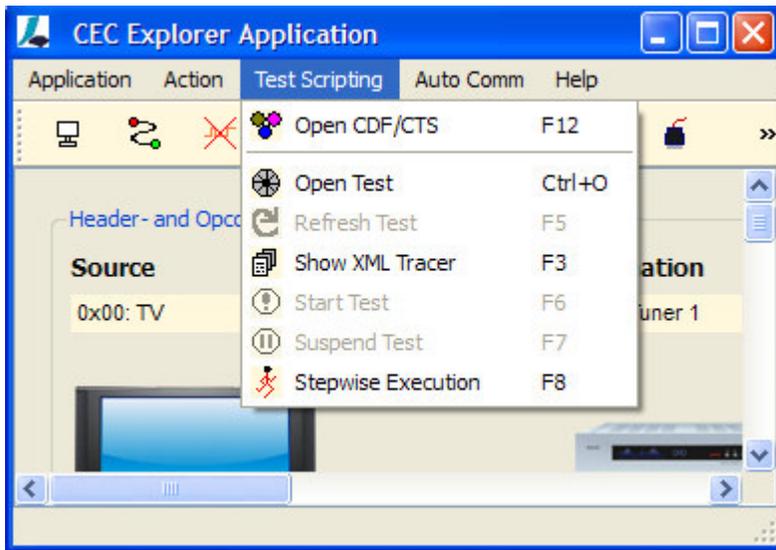


Figure 25: Test Scripting menu

#### 3.1.3.1 Open/Close CDF/CTS (Key F12)

This item opens/closes the CDF/CTS dialog to perform ATC tests. This is described in detail in chapter 5.

#### 3.1.3.2 Open/Close Test (Key Ctrl + O)

Opens/closes an XML-encoded test script for execution, which includes parsing of the XML file. In case of not-well formed XML code, the script is not loaded and the user is notified. The XML files can be viewed in a browser using a style-sheet file which converts it into HTML. The path to the last opened test case is stored in the ini file. Opening a test script prevents execution of ATC tests. Therefore, close an open test script before running ATC tests.

### 3.1.3.3 Refresh Test (Key F5)

Reparses the XML-file, i.e. the script file can be changed in a separate XML editor and reloaded to the application. If reloading fails (due to XML-errors), the test is closed.

### 3.1.3.4 Show/Hide XML Tracer (Key F3)

Opens/closes a split tracer window. The upper part shows the complete XML file content of the currently loaded script, and the lower part shows the log output during script execution. This output can also be stored in a log file if the script is set up correspondingly (see section 4).

During script execution, the currently executed command is highlighted, and the script can be tracked which is especially helpful when executing a script in *'stepwise'* mode. Closing the tracer does not unload the data. When a script starts, the log output window is cleared.

### 3.1.3.5 Start/Stop Test (Key F6)

Starts/stops the execution of the test script. During execution, the test cannot be closed but the test can be stopped or suspended at each time. The same button is used for stopping the test (item label changes). The script runs autonomously and continuously unless a user interaction is encoded in the XML file or the script execution is temporarily suspended. At script end or on stopping the script, the error/warning statistics is dumped to the XML tracer (lower field) if it was enabled via the test case (see section 4).

### 3.1.3.6 Suspend/Resume Test (Key F7)

Suspends the script execution until *'Resume'* is pressed (same item). The test can be stopped during suspension. On stepwise execution, the Suspend/Resume actions are not available. Instead this item can be used to perform a single step (see 3.1.3.7).

### 3.1.3.7 Stepwise/Continuous Execution (Key F8)

This item actually enables the user to debug a script file. When changing to *Stepwise Execution* mode, the functionality of the *'Suspend/Resume'* menu item changes to show a *'Single Step'* label. The script is suspended after each test step and must be re-triggered by means of pressing the *'Single Step'* item in the menu bar (or Key F7). The current selection within the XML tracer points to the next step to be executed. Most test steps cause a log output in the XML tracer (lower field).

### 3.1.4 Auto Comm Menu

As mentioned before (section 3.1.2.3), the CEC Explorer is provided with an integrated *Auto Comm* dialog enabling the user to specifically respond to a detected frame. In case of a frame match, the application sends out the response as soon as the CEC line becomes available.

The *Auto Comm* menu (Figure 26) consists of six items, the shortcuts of which are displayed in context with the items. None of these items are available when the port is not open.

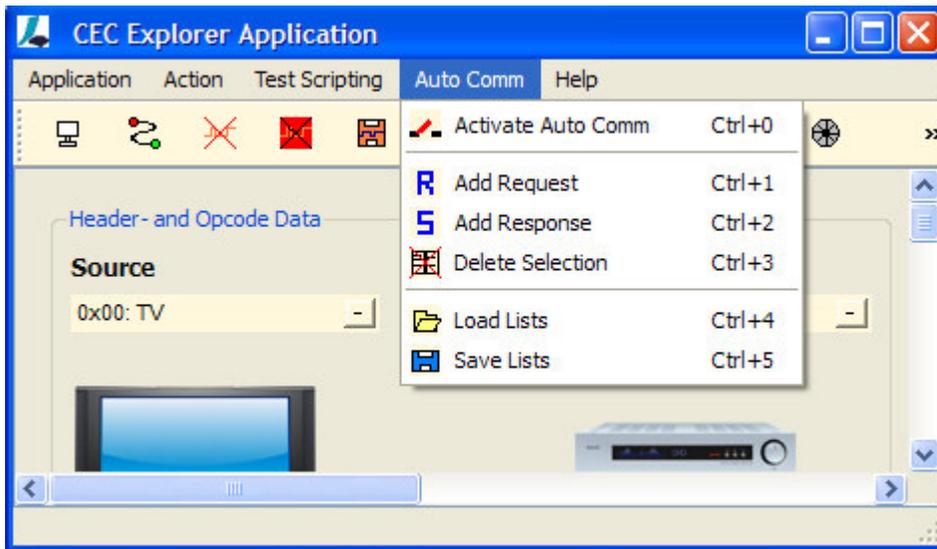


Figure 26: Auto Comm menu

#### 3.1.4.1 Activate Auto Comm

Auto communication must be activated manually, i.e. this feature is disabled as long as this item is not enabled. It is a precondition that the 'Request List' contains as much entries as the 'Response List'. Otherwise, auto responses cannot be activated.

#### 3.1.4.2 Add Request

This item adds an entry to the 'Request List'. The entry is grabbed from the *Current Message Code Field* in the main dialog (section 3.5). Once added, it can be edited by double clicking the list item. On pressing the 'Return' key, the persistent item editor is closed again and the new entry is immediately checked for correct syntax. In case of an error, the old entry is restored. Editing deactivates auto communication.

Requests can also be added via the context menu of the *Message History List*. For this purpose, the user must select a single list entry and click the right mouse button. The 'Add as request to Comm Dialog' item adds the selected frame to the 'Request List'.

The 'Request List' may contain an arbitrary number of '\*' and '?' wildcards but there must be only one '#' wildcard in each item.

#### **3.1.4.3 Add Response**

This item adds an entry to the 'Response List'. Likewise, the entry is grabbed from the *Current Message Code Field* in the main dialog and can then be edited by double clicking the new list item.

Like requests, responses can also be added via the context menu of the *Message History List*. The corresponding item is labeled 'Add as response to Comm Dialog'.

The 'Response List' must not contain any '\*' or '?' wildcards but there can be an arbitrary number of '#' wildcards in each item. These are replaced by the 4-bit-character at the corresponding '#'-position of the request. If there is no such character in the request, the response is not sent.

#### **3.1.4.4 Delete Selection**

This deletes the currently selected items from the *Auto Comm Dialog*. However, the user may also delete selected items by means of the 'Delete' key when the focus is on either list. Note that if the focus is on the *Message History List*, the 'Delete' key will handle selected items of that list.

#### **3.1.4.5 Load Lists**

Once created, lists can be stored in a file using a text format (\*.clf mime). This enables the user to reload a formerly created list on restart of the application. Editing these files in a different application is possible but may lead to parsing errors on loading if the application cannot read the entries.

#### **3.1.4.6 Save Lists**

This will save the entries of either list in a text file, the name of which can be chosen in a file dialog.

#### **3.1.5 Help Menu**

The Help Menu (Figure 27) consists of only one item. It is available independent of the port's state.



Figure 27: Help menu

### 3.1.5.1 About CEC Explorer

Pops up an *About Dialog* containing version information of software and firmware. The latter is not available before the port has been opened. There is also copyright information (Figure 28).



Figure 28: About Dialog

## 3.2 The tool bar

The tool bar (Figure 29) contains buttons for quick access of menu items. The functions are reflected both symbolically and textually (if configured). The tool bar can be re-positioned to any side of the main dialog using the drag-and-drop property of the mouse device. Status bar information and tool tips notify the user about button's functionalities.

The actual functionalities have already been explained in section 3.1 of this document. Toggable buttons are re-labeled on clicking to reflect the current functionality. The configuration menu (section 3.1.1.1.1) enables the user to customize the toolbar. This includes checking / unchecking the desired / undesired items and selecting big or small icons (without text). The tool bar position is not stored in the ini file.

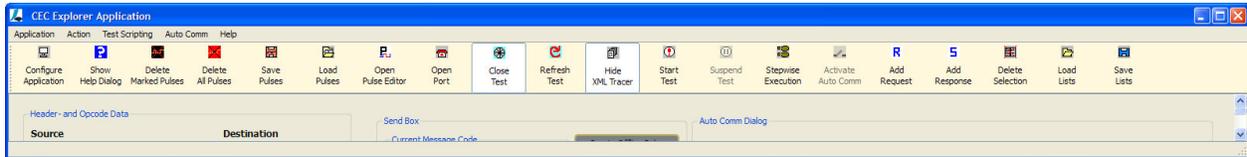


Figure 29: The Tool Bar

### 3.3 Header- and Opcode Data Group Box

The *Header- and Opcode Data* Group Box (Figure 30) in the top left corner of the dialog enables the user to select source, destination, and opcode of a CEC frame, which represent the first two bytes of such a frame.

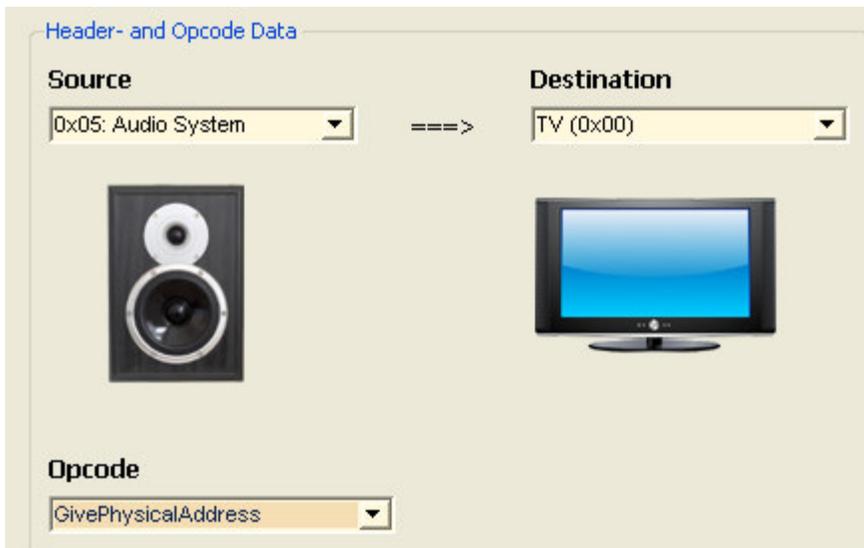


Figure 30: Header and Opcode Data Group Box

All three components can be changed via Combo Boxes, containing the actual description as well as the code behind it. Note that the three Combo Boxes in Figure 30 reflect the three possible configurations of Combo Boxes, i.e. the code at start of line (Source), the code at end of line (Destination), and the line without code (Opcode). Sorting the contents by name and setting the code at line end enables the user to quickly access an entry via keyboard. The behavior can be changed individually via the Combo Boxes' context menu.

The images below source- and destination Combo Box represent symbols for the selected devices and may contain numbers in case there is more than one device of a given type. For quick access, the user may scroll the mouse wheel directly on the image to change it. The following symbols may occur:



Television (unique)



Audio System (unique)



Playback Device (1 – 3)



Recording Device (1 – 3)



Tuner (1 – 4)



Free Use (unique)



Broadcast (unique)



Unregistered (unique)



Reserved (1 – 2)

### **3.4 Operand Data Group Box**

The appearance of the Operand Data Group Box in the bottom left corner of the dialog changes dynamically with the selected opcode. Since each opcode is associated with a special set of operands, the corresponding GUI elements appear enabling the user to conveniently select the possible operand data.

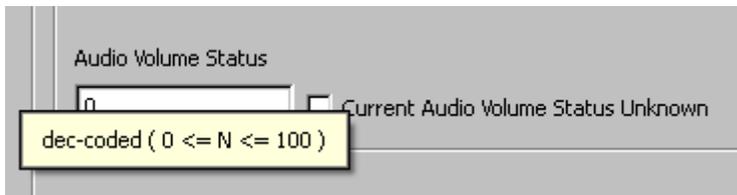
The example, reflected in Figure 31: , refers to the Opcode '*<Set External Time>*', that has been selected before. According to the CEC specification [1], the user may select *Recording Data* (date, start time, duration, sequence), and the *External Source Specifier* for this opcode.

The Operand Data Group Box contains some sort of intelligence, i.e. certain GUI elements are enabled/disabled (visible/hidden) in dependence of other settings. Furthermore, most of the edit fields contain validators and input masks to avoid input of erroneous data. Reserved or 'Future Use'-values are also not accessible (header data is an exception). In case the user deliberately wants to send invalid data, the '*Current Message Code*' box (section 3.5) can be edited manually, or the *Frame Editor* must be used..

The 'Current Message Code' box contains the current CEC frame data and is updated whenever one of the settings changes.

Each edit field in the operand group box that requires manual input of data can be requested for expected data by setting the focus on the field and pressing Shift + F1. A tool tip text then points to the valid data you may enter (Figure 32).

**Figure 31: Operand Data Group Box**



**Figure 32: Tool Tip Text pointing to expected data**

### 3.5 Send Box

The Send Box (Figure 33: Send Box) in the top centered part of the dialog contains all elements dealing with sending CEC frames or other data via the Ethernet port.

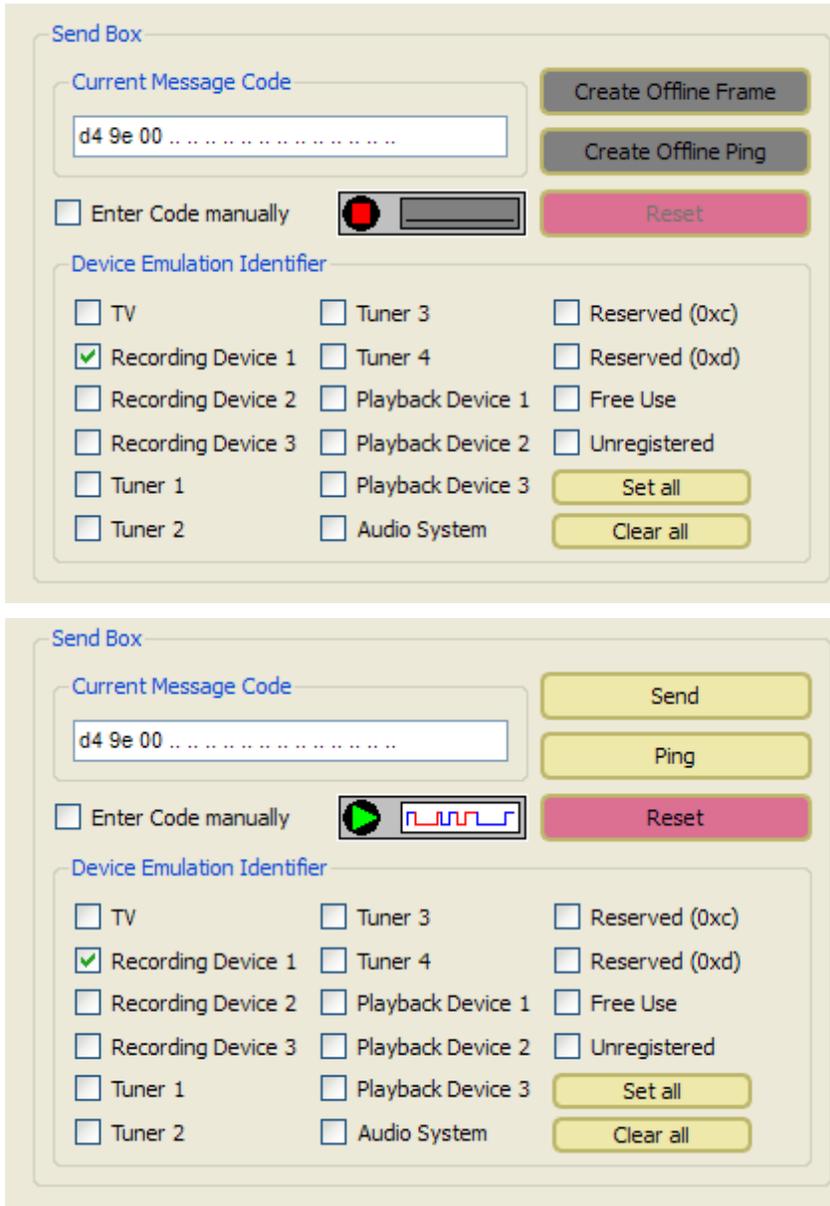


Figure 33: Send Box, inactive (top) and active (bottom)

The edit field '*Current Message Code*' is a normally non-editable field that always reflects the CEC frame that would result from the current GUI settings on the left side. On clicking the Check Box '*Enter Code manually*', one may also type the hex-coded message directly. However, the maximum size of a frame cannot exceed 16 bytes.

Although validators and input masks are used in the Operand Group Box, the resulting message may still become invalid. In that case, this field is either colored yellow (indicating a warning) or red (indicating an error).

In case the port is not open, the actual send buttons are grayed and labeled '*Create Offline Frame/Ping*'. The Reset button is disabled. Furthermore, a little image next to the '*Reset*' button shows a red square as a symbol for being offline. When a port has been opened, the buttons change their color and label as shown in the figure above. The '*Reset*' button for a hardware reset is enabled.

The '*Send*' button immediately sends out the current frame (as shown in the *Current Message Code* field). Frame data is either sent out using ideal bit timings (as specified in the CEC specification [1]), incidentally chosen bit timings or user-defined bit timings. This is determined in the application's configuration dialog (section 3.1.1.1.3).

The '*Ping*' button uses the currently selected source/destination couple to send out a '*One-Byte-Frame*' containing the header only. For quick access, one might also press the *Enter*-button to send a frame, and *Ctrl + Enter* to send a *Ping*. Note that the *Enter* key is only enabled if no persistent editor has been opened in the *Auto Comm* dialog. On pressing *Shift + Enter*, the user can resend the currently selected pulse(s) of the *Message History List*.

The "*Device Emulation Identifier*" check boxes define which devices are emulated by the CEC Explorer, meaning which received commands are acknowledged by the CEC Explorer hardware. An example: if the check box "*Tuner 2*" is selected, the CEC Explorer will acknowledge all messages that are received with the destination address of Tuner 2. The checkbox "*Unregistered*" is an exception: if this checkbox is selected, all broadcast messages will be rejected by the CEC Explorer. Note that one might also use the '*Reserved*' addresses.

The '*Reset*' button sends a reset command to the CEC test board. This might help if the communication has stopped. If this doesn't help, the board must be reset and/or the port has to be closed.

### **3.6 Help Box / Auto Response Box**

These boxes are located in the top right corner of the dialog. Only one of them is visible at a given time as can be selected via a toggleable menu item. Both dialogs have already been described above (3.1.2.3 and 3.1.4).

### **3.7 The Pulse Sequence Graph**

The *Pulse Sequence Graph* (Figure 34) in the centre of the dialog reflects the pulse durations of a pulse sequence, i.e. a CEC frame. Furthermore, it contains all characteristic data of a CEC frame. In order to show the complete sequence, the graph is embedded in a horizontal scroll area. The *Pulse Sequence Graph* shows the pulses at its top, where the pulse durations in milliseconds are written above and below the pulses, respectively.

By default, start pulses are marked yellow, whereas pulse durations for '0'- and '1'-bit representations are marked green and blue. Below the pulse graph, the bit pattern is reflected. The first eight bits of each CEC block contain the actual data (marked white), bit 9 stands for the EOM flag (light blue), and bit 10 represents the ACK bit (brown).

Below the bit pattern, the block number, the byte presentation of the bit pattern and the EOM/ACK-assignments are shown. In the most bottom row of the *Pulse Sequence Graph*, the interpretation of the signals is reflected. In case, the pulse sequence contains an error, it is shown in the bottom left corner of the graph. Offline pulses are assigned a watermark in the top left corner (if configured). However, colors can also be configured individually (see 3.1.1.1.2) which is stored in the ini file.

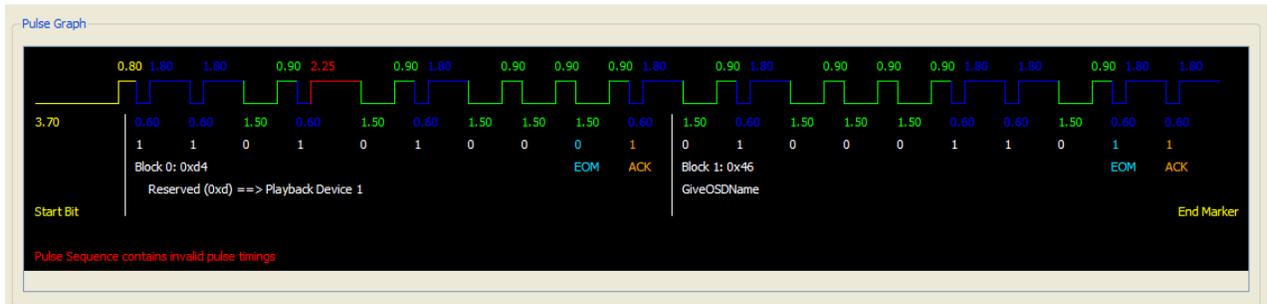


Figure 34: Pulse Sequence Graph

Since the pulse sequence of each message in the *Message History List Box* (section 3.8) is stored until deletion, it can be recovered later by simply clicking on the corresponding entry in the list. The user may also scroll with the mouse wheel either on the *Pulse Graph* to navigate through the *Message History List*.

### 3.8 The Message History List Box

The *Message History List Box* is placed just below the *Pulse Sequence Graph* (section 3.7). An excerpt of the list with representative data is shown in Figure 35.

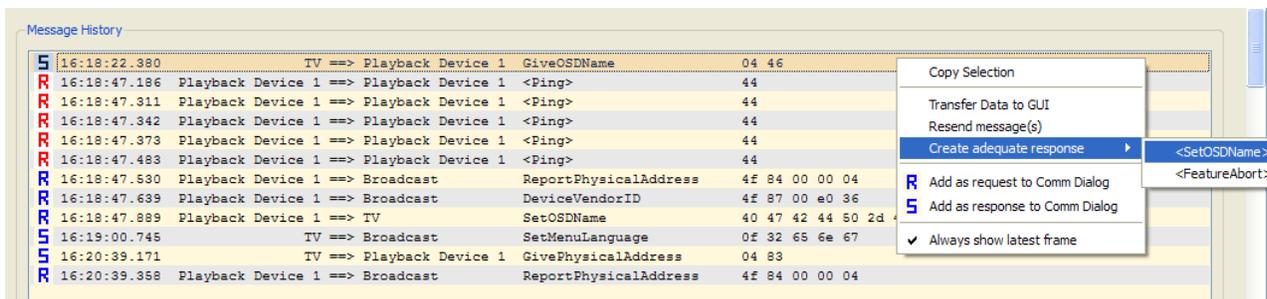


Figure 35: Message History List Box

For each sent or received frame, the list contains one row. Each row starts with an icon, indicating whether this frame was sent or received. More information is provided by the

color of the icon. Red-colored icons indicate that the frame was not acknowledged by the receiver, whereas blue-colored icons indicate that the frame was acknowledged. In the latter case the ACK-bit is 0 for directly addressed frames which is also shown in the corresponding *Pulse Graph*.

In case of broadcast messages, it is vice versa: the icon is blue, if the ACK-bit remains 1, indicating that none of the devices had problems to catch the frame. It gets red, if the ACK-signal was turned to 0 by any device. Since each byte is acknowledged separately, there may also be partly acknowledged messages, which are represented by special icons. Table 4 summarizes the possible icon types:

	Received/monitored frame that was completely acknowledged by the receiver, or received broadcast message that was not (partly) rejected by any device
	Sent frame that was completely acknowledged by the receiver
	Received/monitored frame with corrupt data (below nominal sample time)
	Sent frame with corrupt data (below nominal sample time)
	Received/monitored frame that was partly not acknowledged (blocks ACK signals differ)
	Sent frame that was partly not acknowledged (blocks ACK signals differ)
	Received/monitored frame that was completely not acknowledged
	Sent pulse that was subject to data collision on the CEC line
	Sent frame that was completely not acknowledged by the receiver
	Received/monitored frame with undefined state (e.g. ACK bit is missing since 1 <sup>st</sup> block is incomplete)
	Sent frame with undefined state (e.g. ACK bit is missing since 1 <sup>st</sup> block is incomplete)
	Unsent frame (offline frame)

**Table 4: Possible icons in the Message History List**

The second column of a row entry shows the time when the frame has been sent/received by the application. The format is hh:mm:ss, followed by milliseconds.

The third column shows source and destination of the message, the fourth column shows the opcode or <Ping> if that was a ping message only. Further possible entries here are:

- INCOMPLETE FRAME      missing EOM-1 flag
- ABORTED FRAME          incomplete CEC block
- ERROR PULSE            Pulse containing neither source, destination, opcode or even start bit.

The last column shows the complete hex-encoded CEC frame again, including header- and opcode data.

In case, the single selection of the list changes, the *Pulse Sequence Graph* (section 3.7) is updated to show the corresponding graph. Multi-Selection of entries is also possible, e.g. for pulse deletion or for copying the data to clipboard. This service is provided by the list item's context menu which is also reflected in Figure 35. The selected entries are **copied to the clipboard** and might be pasted to an arbitrary application that accepts text data.

The context menu (available via right mouse button) also includes the possibilities to **transfer the data to the GUI** (only single selection), to **resend a message** (one or more selected frames), and to **create an adequate response** (only single selection). Furthermore, a line entry can either be **added to the request- or the response list** as already mentioned above (3.1.4.2 and 3.1.4.3).

The adequate response is based on the CEC specification [1] and clicking this item includes exchange of source and destination (unless the response is expected broadcast). On checking '**Always show latest pulse**', the 'Pulse Sequence Graph' is always updated when a new entry is added to the *Message History List*. In addition, the list scrolls to the last entry. Unchecking this item prevents this. This flag is effectless when frame data is loaded from file (to increase performance).

Erroneous entries can be immediately identified by a red colored entry. The degree of error can reach from a time warning to a complete error pulse.

### **3.9 The Operands Edit Field**

This edit field at the bottom of the dialog (Figure 36) shows more detailed information about the operands of the currently selected message in the *Message History List* (section 3.8). For example, the operands of a '<Set OSD String>' opcode are listed below. Erroneous, excessive, or missing data would cause entries in red.

A context menu enables the user to copy the contents to the clipboard.

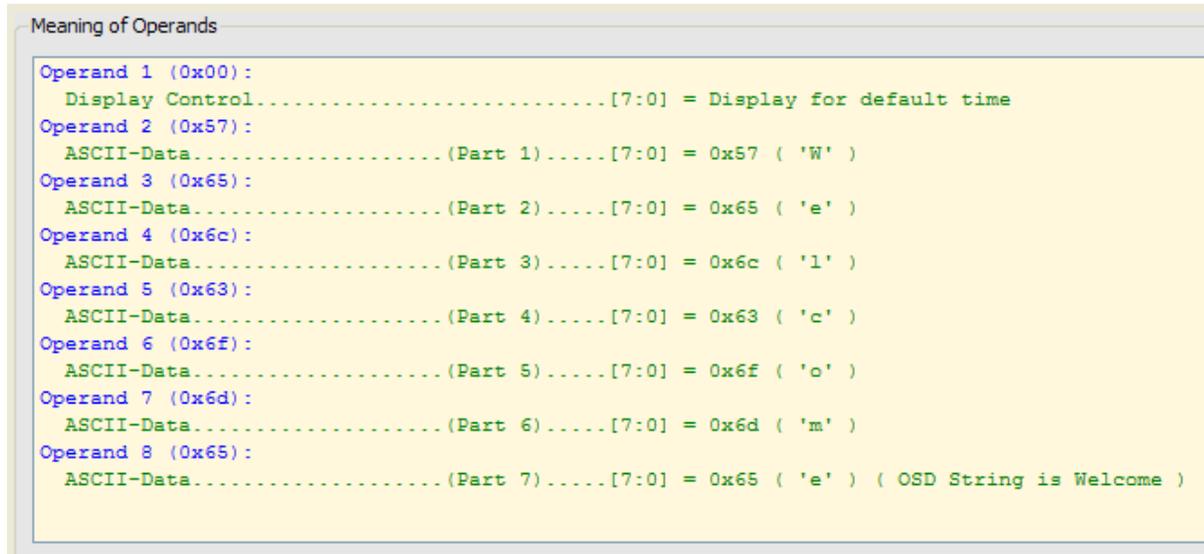


Figure 36: Operands Edit Field

## 4 Scripting Interface

### 4.1 Introduction

The scripting interface enables the user to setup a program flow for performing more complex communication. It is based on XML language where tag names, tag attributes and tag data reflect a proprietary language to control the flow. The language contains several elements of typical programming languages and some custom tags. Key words are case-sensitive. When opening a test case, the file is checked for well-formed XML content. When starting a test case, the script is executed according to the encoded tag types. Opening, starting and handling of script files has already been described (3.1.3). Therefore, this chapter mainly describes the syntax of the language.

### 4.2 Script Structure

The following text box reflects the basic structure of a script:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Testcase SYSTEM "../.../doctypedefs/cec_main.dtd">
<?xml-stylesheet type="text/xsl" href="../.../stylesheets/cec_main.xsl" ?>

<!-- Test Script 0.0.0 -->
<Testcase>
  <!-- Testcase ID -->
  <TestID>0.0.0</TestID>

  <!-- Testcase Objective -->
  <TestObjective>Demonstration of main structure</TestObjective>

  <!-- Testcase Constraints -->
  <TestConstraint>None</TestConstraint>

  <!-- Testcase Required Methods -->
  <TestReqMethod>None</TestReqMethod>

  <!-- Testcase Setup -->
  <TestSetup>
    <Device type="TV" logAddr="0" mode="real">
      <Device type="RD" logAddr="1" mode="virtual" />
    </Device>
  </TestSetup>

  <!-- Testcase Execution -->
  <TestExecution statistics="on" logFile="tc_0_0_0.log">
    <!-- main routine -->
    <Routine name="main">
      ...
    </Routine>
  </TestExecution>
</Testcase>
```

The first three lines point to the XML version and the associated dtd- and stylesheet files. The dtd file defines the rules for tag- and attribute structures whereas the stylesheet file converts the XML file to HTML format to be viewed in a browser.

The complete test case is then included in the `<Testcase>` root tag where `<TestID>`, `<TestObjective>`, `<TestConstraint>` and `<TestReqMethod>` have only informal character for the stylesheet file.

The `<TestSetup>` tag describes the connections between the devices. It contains an arbitrary number of `<Device>` tags which are hierarchically ordered to reflect parent- and child device relationships. The following rules must be met:

- There must be at least one `<Device>` child tag below `<TestSetup>`.
- There must be exactly one root device which must be of `type="TV"`.
- There must be only `<Device>` tags below `<TestSetup>`, nothing else.
- Each `<Device>` tag must contain at least the `type`-attribute and the `logAddr`-attribute which must fit according to the CEC specification [1]. Valid values for the type are TV, RD (recording dev.), TD (tuner dev.), PD (playback dev.), AS (audio system), FU (Free Use), RS (Reserved), and UR (Unregistered). The logical address must be in range  $0 \leq x \leq 15$  (decimal) and unique among the devices.
- The device list must contain at least one virtual device (`mode="virtual"`, `mode="dummy"` or mode attribute omitted).
- The real DUT must be assigned `mode="real"` but its presence is not necessary.
- To avoid setting the mask for a listed device, set `mode="phantom"`.
- There are lots of other attributes which have only descriptive character for the stylesheet file. For a list of all attributes, refer to the `cec_main.dtd` file.

If these initial tests have passed, the mask is set correspondingly in the *Device Emulation Identifier* (see 3.5). Example:

```
<!-- Testcase Setup -->
<TestSetup>
  <Device type="TV" logAddr="0" osdName="Simplay Demo Device">
    <Device type="RD" logAddr="1">
      <Device type="TD" logAddr="3" mode="real" />
    </Device>
    <Device type="PD" logAddr="8" mode="virtual" />
  </Device>
</TestSetup>
```

Figure 37 is an excerpt of the HTML page that was created by the stylesheet file for this xml file:

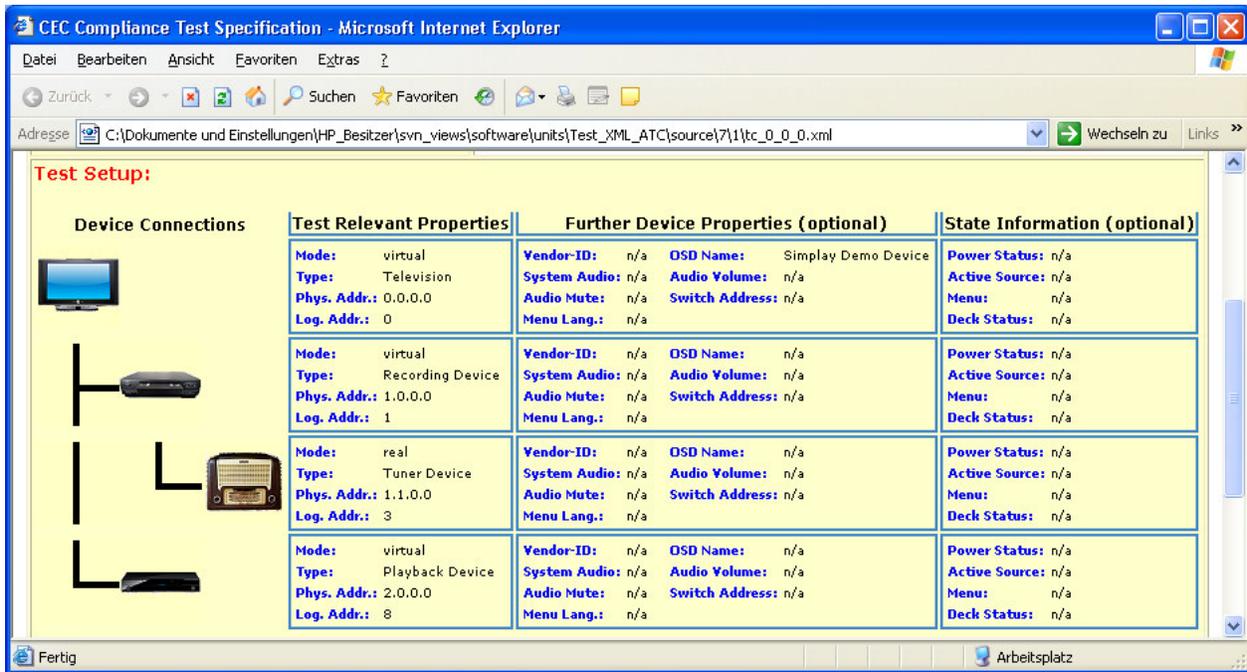


Figure 37: HTML converted XML script file

### 4.3 The <TestExecution> tag

There must be exactly one <TestExecution> tag, the child tags of which must be either of type <DefineVar> (to declare a global variable), <UserList> (to declare a global list) or of type <Routine>. Each <Routine> tag must contain a *name* attribute the value of which must be unique among the <Routine> tags. There must be exactly one main routine (<Routine name="main" ...> after which no more <DefineVar> tags must follow. The main routine is the entry point for the test case.

Note that a test case is setup similarly to a C-program. The <Routine> tag is the function body which may contain several commands as child tags. There can be an arbitrary number of <Routine> tags in a test case. Routines can be invoked from inside other routines (also recursively) and parameters can be passed to the routines (as a copy, i.e. call by value).

Variables can be defined and have a validity according to the scope in which they were defined. They are deleted when the scope is left. There can be operations on variables, there can be loops (with a loop variable, a loop operation, and a loop condition), there can be conditional statements (if, else if, else). But there can also be specific CEC actions (for user requests or logging of errors/warnings, or for sending of and waiting for frames).

The test case is finished when the main routine has finished or the program has encountered a run time error.

## 4.4 Script Execution Mode

Note that during script execution, most of the GUI elements are disabled. That is, one cannot send any additional frames or change the device mask while the test is running. Consider also that the script does not automatically change any hardware settings (apart from the device mask: see 4.2). That is, test path, HPD- and Power signals must be set manually in advance. However, there are certain tag types that perform hardware settings. These are presented in section 4.5 of this document.

There is also no limitation of the *Message History List* during script execution. This is due to certain tag commands which are able to access former CEC frames. Note that the complete script execution is logged (if configured). As a result, a log file and a corresponding psq file is created after the test case has finished.

## 4.5 Tag types and their attributes

The following paragraphs will describe all the tag types and the tag type specific attributes in more detail.

### 4.5.1 <TestExecution> tag

<b>SYNTAX:</b>		
<pre>&lt;TestExecution logFile="..." statistics="..." skipBoardReset="..."&gt;     ... &lt;/TestExecution&gt;</pre>		
<b>DESCRIPTION:</b>		
Root tag of executable test case part. It comprises all <Routine> tags (including main) and must be unique in a test case. Other direct child tags must be of type <DefineVar> or <UserList> where they must precede the main routine.		
<b>PROPERTIES:</b>		
Must not contain any tag data but only tags. Must not be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
logFile (optional)	at least five chars, ending with .log	Log file name (including path) where the script output is dumped to. The Explorer creates also a psq file with the same name. If omitted, no log files are created.
statistics (optional)	on off (default)	Enables final dump of statistics (errors/warnings) Disables final dump of statistics
skipBoardReset (optional)	0 (default) 1	Sets the mask according to <TestSetup> in advance Prevents mask setting prior to test start
<b>EXAMPLE:</b>		
<pre>&lt;TestExecution logFile="tc_0_0_0.log" statistics="on" skipBoardReset="0"&gt;</pre>		

## 4.5.2 <Routine> tag

<b>SYNTAX:</b>		
<pre>&lt;Routine name="..." params="..."&gt;     ... &lt;/Routine&gt;</pre>		
<b>DESCRIPTION:</b>		
<p>All &lt;Routine&gt; tags must be direct children of the &lt;TestExecution&gt; tag. There name must be unique and there must be exactly one main routine (the test case entry point). Routines are executable units which can be invoked from anywhere inside the script. However, the main routine must not be invoked manually. One can pass parameters to a routine on invocation.</p>		
<b>PROPERTIES:</b>		
<p>Must not contain any tag data but only tags. Must not be an empty tag.</p>		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
name (mandatory)	at least one char, must be unique in test case	Name of routine that is used for invocation in a <DoSub> tag
params (optional)	List of comma separated new variables	The variables in the list are assigned the values of the params attribute in a <DoSub> tag (in the order of appearance). The variable names must be unique. The main routine must not have any parameter. Parameters can be in cambered brackets (see 4.7).
<b>EXAMPLE:</b>		
<pre>&lt;Routine name="requestUser" params="V DEVICE, V REQUEST"&gt;</pre>		

## 4.5.3 <DefineVar> tag

<b>SYNTAX:</b>		
<pre>&lt;DefineVar name="..." value="..." /&gt;</pre>		
<b>DESCRIPTION:</b>		
<p>Defines a variable that can be used throughout the scope in which it was defined. The variable name must start with 'V_' and must be followed by at least one character. One must not use a system variable name (section 4.6), and one must only use letters (upper and lower case), digits and the underscore character ('_') for name definition. The name must be unique within the test case. The value can be arbitrary. If the value is a variable, its value is assigned.</p>		
<b>PROPERTIES:</b>		
<p>Must not contain any tag data or further tags. Must be an empty tag.</p>		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
name (mandatory)	V_, followed by at least one char, unique in test case	Variable name that can be used in almost all other tag- and attribute types as a reference.
value (optional)	arbitrary characters	Value to be assigned to the variable. If the value is a variable, it is replaced by its value in advance.
<b>EXAMPLE:</b>		
<pre>&lt;DefineVar name="V DEVICE" value="Playback Device 1" /&gt;</pre>		

#### 4.5.4 <DoOperation> tag

<b>SYNTAX:</b>		
<code>&lt;DoOperation operation="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Defines an operation to be applied to a variable. The operation consists of two operands and an operator in between them. Since all operators are 'Assign' operators, the result of the operation is assigned the first operand which must be a valid variable. If the operation fails, the test case is stopped.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
operation (mandatory)	Oper1 Operator Oper2	Please refer to section 4.8 for a detailed description of operations.
<b>EXAMPLE:</b>		
<code>&lt;DoOperation operation="V COUNTER += 1" /&gt;</code>		

#### 4.5.5 <UserList> tag

<b>SYNTAX:</b>		
<code>&lt;UserList id="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Defines a list of comma separated values. The script can access these values during execution by means of the <GetUserListEntry> tag. The <UserList> is valid within the scope it was defined and identified via its id attribute which must be unique among lists. Data is defined as tag data, and the user list must contain at least one value. If the list contains variables, they are already replaced at the time of definition, not at the time of access.		
<b>PROPERTIES:</b>		
Contains values as tag data. Must not contain further tags. Must not be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
id (mandatory)	arbitrary characters	List id used for later access of values.
<b>EXAMPLE:</b>		
<code>&lt;UserList id="destinations"&gt;0, 2, 4, 6, 10, 13&lt;/UserList&gt;</code>		

### 4.5.6 <GetUserListEntry> tag

<b>SYNTAX:</b>		
<code>&lt;GetUserListEntry id="..." index="..." result="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Retrieves the value from the specified list at the specified index. The value is assigned the variable of the result attribute. The variable must have been defined in advance. If the index is greater than the list size, the test case is stopped.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
id (mandatory)	arbitrary characters	Specifies the id of the list from which the value should be retrieved. Can be a variable.
index (mandatory)	(hexa)decimal number	Specifies the index within the list to access. Can be a variable. If the index starts with 'h:', the index base is hexadecimal. The first entry has index 0.
result (mandatory)	V_, followed by at least one char	Reference variable that retrieves the value of the list. The variable must exist at time of usage.
<b>EXAMPLE:</b>		
<code>&lt;GetUserListEntry id="destination" index="V COUNTER" result="V DEST" /&gt;</code>		

### 4.5.7 <DoSub> tag

<b>SYNTAX:</b>		
<code>&lt;DoSub name="..." params="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Invokes the routine specified by the name attribute. The routine must exist within the test case and it must not be the main routine. A list of comma-separated parameters can be specified that is passed to the routine (the list can contain variables). The <Routine> tag must contain a parameter list of equal size containing the variable names that are assigned the values of the <DoSub> parameter list. If variables are passed, it is a 'call by value' style, not 'call by reference' (i.e. a copy).		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
name (mandatory)	at least one char, must exist in test case	Name of the routine to invoke
params (optional)	List of comma separated value or existing variables	Parameters to pass to the routine. The <Routine> tag must contain a parameter list of equal size. Parameters can be in cambered brackets (see 4.7).
<b>EXAMPLE:</b>		
<code>&lt;DoSub name="analysePulse" params="V FRAME NR" /&gt;</code>		

### 4.5.8 <Return> tag

<b>SYNTAX:</b> <Return subRes="..." />		
<b>DESCRIPTION:</b> Returns from the routine that is currently executed. Can be invoked from anywhere within the routine, even from within the main routine. The latter would stop the test case. The test case continues to be executed after the <DoSub> tag. The tag can return a value defined by the subRes attribute. The value is assigned the system variable V_SUB (section 4.6).		
<b>PROPERTIES:</b> Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
subRes (optional)	arbitrary characters	Return value. Can be any character sequence. A variable value is replaced in advance. The value is assigned to V_SUB. If no return value is given, V_SUB is assigned "-1"
<b>EXAMPLE:</b> <Return subRes="V COUNTER" />		

### 4.5.9 <Loop> tag

<b>SYNTAX:</b> <Loop loopVar="..." loopStartValue="..." loopCondition="..." loopOperation="..." > ... </Loop>		
<b>DESCRIPTION:</b> Loops are handled as in the C-language. The loop starts with the initialization of a loop variable, checks the loop condition, performs the loop statements if the condition is true and performs the loop operation after having returned from all the statements. The <Loop> tag may have no attributes (endless loop). However, if a loop variable is defined, there must also be a loop value. Loops can occur within other loops. All attributes can contain variables.		
<b>PROPERTIES:</b> Must not contain any tag data but only tags. Must not be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
loopVar (optional)	V_, followed by at least one char	Loop variable which must be unique and which is only valid within the scope of the loop
loopStartValue (optional)	arbitrary characters	Initialization value of loop. Can be a variable which is replaced in advance.
loopCondition (optional)	Oper1 Operator Oper2	Please refer to section 4.9 for a detailed description of conditional statements.
loopOperation (optional)	Oper1 Operator Oper2	Please refer to section 4.8 for a detailed description of operations.
<b>EXAMPLE:</b> <Loop loopVar="V_CTR" loopStartValue="0" loopCondition=" V_CTR LT 5" loopOperation=" V_CTR += 1">		

#### 4.5.10 <Break> tag

<b>SYNTAX:</b> <Break />
<b>DESCRIPTION:</b> Leaves a loop immediately. The <Break> tag can only occur within a loop statement. There are no attributes.
<b>PROPERTIES:</b> Must not contain any tag data or further tags. Must be an empty tag.
<b>EXAMPLE:</b> <Break />

#### 4.5.11 <Continue> tag

<b>SYNTAX:</b> <Continue />
<b>DESCRIPTION:</b> Skips the residual loop statements and continues with executing the next loop operation. The <Continue> tag can only occur within a loop statement. There are no attributes.
<b>PROPERTIES:</b> Must not contain any tag data or further tags. Must be an empty tag.
<b>EXAMPLE:</b> <Continue />

#### 4.5.12 <Exit> tag

<b>SYNTAX:</b> <Exit message="..." />						
<b>DESCRIPTION:</b> Immediately stops the test case no matter where the program pointer is. If a message is defined, it is considered as error message and counted as an error (if statistics are enabled).						
<b>PROPERTIES:</b> Must not contain any tag data or further tags. Must be an empty tag.						
<table border="1"><thead><tr><th>ATTRIBUTE</th><th>VALUE</th><th>DESCRIPTION</th></tr></thead><tbody><tr><td>message (optional)</td><td>arbitrary string</td><td>Error message</td></tr></tbody></table>	ATTRIBUTE	VALUE	DESCRIPTION	message (optional)	arbitrary string	Error message
ATTRIBUTE	VALUE	DESCRIPTION				
message (optional)	arbitrary string	Error message				
<b>EXAMPLE:</b> <Exit message="Division by Zero" />						

### 4.5.13 <If/Elseif/Else> tag

**SYNTAX:**

```
<If condition/frame="...">
...
</If>
<ElseIf condition/frame="...">
...
</ElseIf>
<Else>
...
</Else>
```

**DESCRIPTION:**

Handles conditional statements. The tag might occur at any place within a routine. <ElseIf> or <Else> must be preceded by <If> or <ElseIf>. If/ElseIf must define a condition or a frame which is to be checked. An <Else> tag must not contain a condition. If a condition is true, the address pointer is set to the first child tag which must be present. All following <ElseIf> or <Else> tags are then skipped after a condition has been found true and its statements have been executed.

The condition to be checked either consists of two operands and one operator (see 4.9) or it is a frame that is compared to the last received frame. Thus, the user may check for an expected response after a CEC request. Similarly to the C-language, the initial conditional check must be an <If> tag, followed by an arbitrary number of <ElseIf> tags, and followed by an optional <Else>. There can be any type and any number of statements downstream of a condition. The attributes *condition* and *frame* are mutual exclusive.

**PROPERTIES:**

Must not contain any tag data but only tags. Must not be an empty tag.

ATTRIBUTE	VALUE	DESCRIPTION
condition (optional)	Oper1 Operator Oper2	Please refer to section 4.9 for a detailed description of conditional statements.
frame (optional)	regular frame expression	Compares the last received CEC frame, represented by the system variable V_FRAME (see 4.6) with the specified value. Can contain a variable which is replaced in advance.

**EXAMPLE:**

```
<If condition="V_CTR GT 5">
...
</If>
<ElseIf frame="V_Source V_Dest V_Opcode">
...
</ElseIf>
<Else>
...
</Else>
```

#### 4.5.14 <AutoResponse> tag

<b>SYNTAX:</b>		
<code>&lt;AutoResponse request="..." response="..." event="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Defines an automatic response for an incoming request which is handled as regular expression. Auto responses are valid from the time of definition until test case end. When the request is received, the response is automatically sent out. Moreover, one can define if an event is thrown on receiving the corresponding request. This is only relevant for previously defined <Wait> tags. Auto responses can be prevented by omitting the response attribute (event must be 0 then).		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
request (mandatory)	regular frame expression	Frame to automatically respond to. Can contain a variable which is replaced in advance.
response (optional)	CEC frame	Frame that is responded. Can contain a variable which is replaced in advance. The final response must be valid.
event (mandatory)	0 1	Prevents response to timer events of <Wait> tag Enables response to timer events of <Wait> tag
<b>EXAMPLE:</b>		
<code>&lt;AutoResponse request="01 97*" response="10 35 01" event="0" /&gt;</code>		

#### 4.5.15 <Extract> tag

<b>SYNTAX:</b>		
<code>&lt;Extract source="..." target="..." from="..." chars="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Extracts a sub string from a source string. The result is stored in the specified target. The extraction starts at position 'from' and concerns 'chars' characters. If the start position is outside the source's length, the target is assigned -1. If there are too less characters, the residual string is assigned.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
source (mandatory)	arbitrary string	Source to extract from. Can contain a variable which is replaced in advance.
target (mandatory)	V_, followed by at least one char	Existing variable which is assigned the result
from (mandatory)	decimal number	Start position of extraction. First character is at position 0. Can be a variable which is replaced in advance.
chars (mandatory)	decimal number	Number of characters to extract. Can be a variable which is replaced in advance.
<b>EXAMPLE:</b>		
<code>&lt;Extract source="V Frame" target="V Opcode" from="2" chars="2" /&gt;</code>		

### 4.5.16 <ChangeDeviceMask> tag

<b>SYNTAX:</b>		
<code>&lt;ChangeDeviceMask logAddr="..." set="..." eType="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Is used to change the device mask in the CEC Explorer hardware during script execution. Mask changes can refer to both addition or removal of a device or several devices. If the 'Free Use' address is added, the eType attribute must be specified since the auto response list is adapted to the new settings and <Report Physical Address> requires a device type.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
logAddr (mandatory)	1- or 4-digit hexadecimal value	If the set attribute is 0 or 1, a single address is added or removed, i.e. the logAddr value is a one-digit value. If set is 2, the value is a four-digit value. Can contain a variable which is replaced in advance.
set (mandatory)	0 1 2	indicates that a single device shall be removed Indicates that a single device shall be added Indicates that the complete mask shall be changed Can be a variable which is replaced in advance.
eType (optional)	TV RD TD PD AS	Type of device at 'Free Use' address. Can be a variable which is replaced in advance.
<b>EXAMPLE:</b>		
<code>&lt;ChangeDeviceMask logAddr="403d" set="2" eType="TV"&gt;</code>		

### 4.5.17 <LockPulseList> tag

<b>SYNTAX:</b>		
<code>&lt;LockPulseList /&gt;</code>		
<b>DESCRIPTION:</b>		
Locks the Message History List for sent and received frames and for acknowledge values. The XML interpreter adds a message but the frame is not added to the internal message list.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>EXAMPLE:</b>		
<code>&lt;LockPulseList /&gt;</code>		

#### 4.5.18 <GetBitTime> tag

<b>SYNTAX:</b>		
<code>&lt;GetBitTime frame="..." bit="..." result="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Extracts an impedance time span (in microseconds) from a specified bit of a specified frame.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
frame (mandatory)	decimal number	Frame index of the test case's Message History List. If the index starts with A:, index 0 is the first frame and the list is searched forwards. If not, index 0 is the latest frame and the list is searched backwards. If the frame does not exist, the assigned result is -1. Can contain a variable which is replaced in advance.
bit (mandatory)	decNr:decNr:decNr	Bit pattern, represented by a colon-separated list of three values. The first value specifies the block within the frame (where 0 is the start bit), the second value stands for the bit in the block (or the absolute bit number in the frame if block is -1 or the last bit if bit is -1) and the third number is either 0 (low impedance value), 1 (high impedance value) or 2 (sum of both values). Can contain a variable which is replaced in advance.
result (mandatory)	V_, followed by at least one char	Existing variable which is assigned the result
<b>EXAMPLE:</b>		
<code>&lt;GetBitTime frame="A:V_FrameIx" bit="0:0:0" result="V_TIME"&gt;</code>		

#### 4.5.19 <UnlockPulseList> tag

<b>SYNTAX:</b>		
<code>&lt;UnlockPulseList /&gt;</code>		
<b>DESCRIPTION:</b>		
Unlocks the Message History List after it was locked (see 4.5.17)		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>EXAMPLE:</b>		
<code>&lt;UnlockPulseList /&gt;</code>		

### 4.5.20 <GetFrame> tag

<b>SYNTAX:</b>		
<code>&lt;GetFrame frame="..." property="..." result="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Enables access to a former CEC frame of the Message History List. Depending on the <i>property</i> attribute, one can either retrieve the hex encoded frame, the acknowledge value or the time stamp in the Message History List.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
frame (mandatory)	decimal number	Frame index of the test case's Message History List. If the index starts with A:, index 0 is the first frame and the list is searched forwards. If not, index 0 is the latest frame and the list is searched backwards. If the frame does not exist, the assigned result is -1. Can contain a variable which is replaced in advance.
property (optional)	hexcode (default)	CEC frame, hex encoded without whitespaces
	timestamp	Timestamp in milliseconds
	ack	Acknowledge value of frame (ACK or NACK)
result (mandatory)	V_, followed by at least one char	Existing variable which is assigned the result
<b>EXAMPLE:</b>		
<code>&lt;GetFrame frame="0" property="hexcode" result="V Frame" /&gt;</code>		

### 4.5.21 <LogError> tag

<b>SYNTAX:</b>		
<code>&lt;LogError error="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Adds an error message to the log file and increments the internal error counter for the final statistics.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
error (mandatory)	arbitrary string	Error message. Can contain a variable which is replaced in advance.
<b>EXAMPLE:</b>		
<code>&lt;LogError error="The DUT did not respond" /&gt;</code>		

#### 4.5.22 <LogWarning> tag

<b>SYNTAX:</b> <code>&lt;LogWarning warning="..." /&gt;</code>		
<b>DESCRIPTION:</b> Adds a warning message to the log file and increments the internal warning counter for the final statistics.		
<b>PROPERTIES:</b> Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
warning (mandatory)	arbitrary string	Warning message. Can contain a variable which is replaced in advance.
<b>EXAMPLE:</b> <code>&lt;LogWarning warning="The DUT did not respond in time" /&gt;</code>		

#### 4.5.23 <LogMessage> tag

<b>SYNTAX:</b> <code>&lt;LogMessage message="..." /&gt;</code>		
<b>DESCRIPTION:</b> Adds a message to the log file.		
<b>PROPERTIES:</b> Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
message (mandatory)	arbitrary string	Message. Can contain a variable which is replaced in advance.
<b>EXAMPLE:</b> <code>&lt;LogMessage message="The DUT sent the correct response" /&gt;</code>		

#### 4.5.24 <SendCecFrame> tag

<b>SYNTAX:</b>		
<pre>&lt;SendCecFrame source="..." dest="..." type="..." wait="..." retransmit="..." &gt; ... &lt;/SendCecFrame&gt;</pre>		
<b>DESCRIPTION:</b>		
<p>Sends a CEC frame which is specified as hex encoded tag data, starting with the opcode. If the frame is not acknowledged, it can be defined to be retransmitted up to five times. A "wait" attribute suspends the test until the frame is acknowledged, responded or the time has elapsed. If data is omitted, a &lt;Ping&gt; is sent.</p>		
<b>PROPERTIES:</b>		
Can contain tag data if a non-Ping frame is sent. Must not contain any further tags.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
source (mandatory)	hex encoded number	Source of frame to be sent. Can be a variable which is replaced in advance.
dest (mandatory)	hex encoded number	Destination of frame to be sent. Can be a variable which is replaced in advance.
type (mandatory)	hex	Indicates that the frame is encoded by hex data within the tag data
wait (optional)	decimal number	Suspension time after frame was sent. Can contain a variable which is replaced in advance. If the time starts with 'A:', the test case waits for the acknowledge signal, otherwise it waits for the response. On timeout, however, the test case continues and V TIMEOUT is set to 1 (see 4.6).
retransmit (optional)	decimal number	Retransmission value which must be between 0 and 5. Can be a variable which is replaced in advance. If not specified, the Explorer's configuration value is used instead (see 3.1.1.1.4).
<b>EXAMPLE:</b>		
<pre>&lt;SendCecFrame source="V_Source" dest="V_Dest" type="hex" wait="A1000" retransmit="3" /&gt; 9f &lt;/SendCecFrame&gt;</pre>		

#### 4.5.25 <SetHPD> tag

<b>SYNTAX:</b> <code>&lt;SetHPD state="..." duration="..." /&gt;</code>		
<b>DESCRIPTION:</b> Sets the HPD signal to the HDMI In port. If a duration is specified, the HPD line is toggled for that time to be enabled afterwards. If no duration is specified, the state attribute indicates if the signal should be set or reset.		
<b>PROPERTIES:</b> Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
state	0	Resets HPD signal
(mandatory)	1	Sets HPD signal
		Can be a variable which is replaced in advance.
duration	decimal number	Time (in milliseconds) to set the HPD signal to 0 before setting it back to 1. If duration is specified, state must be 0.
(optional)		
<b>EXAMPLE:</b> <code>&lt;SetHPD state="0" duration="110" /&gt;</code>		

#### 4.5.26 <SetPhysAddr> tag

<b>SYNTAX:</b> <code>&lt;SetPhysAddr physAddr="..." /&gt;</code>		
<b>DESCRIPTION:</b> Sets the physical address to the HDMI In port which is then assigned to source devices connecting to this port.		
<b>PROPERTIES:</b> Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
physAddr	four hex digits, separated by dots	Physical address to set to the HDMI In port. Can contain a variable which is replaced in advance.
(mandatory)		
<b>EXAMPLE:</b> <code>&lt;SetPhysAddr physAddr="1.0.0.0" /&gt;</code>		

### 4.5.27 <SetPower> tag

<b>SYNTAX:</b>		
<code>&lt;SetPower state="..." port="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Sets or resets the Power signal to the specified HDMI Out port.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
state	0	Reset power signal at specified port.
	1	Set Power signal at specified port.
		Can be a variable which is replaced in advance.
port	1	Change signal at HDMI Out port 1
	2	Change signal at HDMI Out port 2
	3	Change signal at HDMI Out port 3
		Can be a variable which is replaced in advance.
<b>EXAMPLE:</b>		
<code>&lt;SetPower state="1" port="1" /&gt;</code>		

### 4.5.28 <UserAction> tag

<b>SYNTAX:</b>		
<code>&lt;UserAction cancelIsAbort="..."&gt;</code> ... <code>&lt;/UserAction&gt;</code>		
<b>DESCRIPTION:</b>		
Pops up a non-modal message box (Figure 38) containing the tag data as message. The message box contains an OK- and a CANCEL button. Pressing CANCEL or closing the box via the system menu, aborts the test case unless the <i>cancelIsAbort</i> attribute is set to 0. The test case is suspended as long as the message box exists. The tag data may contain variables which are replaced in advance.		
<b>PROPERTIES:</b>		
Must contain tag data. Must not contain any further tags.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
cancelIsAbort	0	Continues after <UserAction> when CANCEL is pressed
(optional)	1 (default)	Stops the test case when CANCEL is pressed
<b>EXAMPLE:</b>		
<code>&lt;UserAction&gt;</code> Switch off the DUT <code>&lt;/UserAction&gt;</code>		

### 4.5.29 <UserInput> tag

<b>SYNTAX:</b>		
<pre>&lt;UserInput cancelIsAbort="..." input="..." default="..."&gt; ... &lt;/UserInput&gt;</pre>		
<b>DESCRIPTION:</b>		
<p>Pops up a non-modal message box (Figure 39) with an editable line edit field. The tag data is used as message to prompt the user with. The edit field can be assigned a default value on pop up. The final input after confirmation is assigned the input reference value. The message box contains an OK- and a CANCEL button. Pressing CANCEL or closing the box via the system menu, aborts the test case unless the <i>cancelIsAbort</i> attribute is set to 0. The test case is suspended as long as the message box exists. The tag data may contain variables which are replaced in advance.</p>		
<b>PROPERTIES:</b>		
Must contain tag data. Must not contain any further tags.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
cancelIsAbort	0	Continues after <UserInput> when CANCEL is pressed
(optional)	1 (default)	Stops the test case when CANCEL is pressed
input	V_, followed by at least one char	Existing variable which is assigned the result
(mandatory)		
default	arbitrary string	Initial value in line edit field. Can contain a variable which is replaced in advance.
(optional)		
<b>EXAMPLE:</b>		
<pre>&lt;UserInput cancelIsAbort="1" input="V_Input" default="12"&gt;     Enter day of month &lt;/UserInput&gt;</pre>		



Figure 38: <UserAction> Message Box

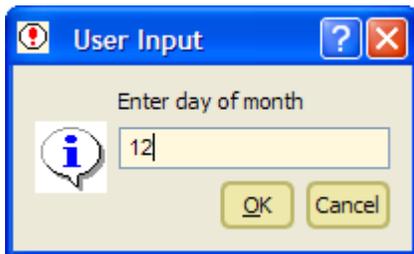


Figure 39: <UserInput> Message Box

### 4.5.30 <UserDecision> tag

<p><b>SYNTAX:</b></p> <pre>&lt;UserDecision cancelIsAbort="..." label="..."&gt;   &lt;Option request="..."&gt;     ...   &lt;/Option&gt;   &lt;Option request="..."&gt;     ...   &lt;/Option&gt;   ... &lt;/UserDecision&gt;</pre>		
<p><b>DESCRIPTION:</b></p> <p>Pops up a non-modal message box (Figure 42) with an arbitrary number of radio buttons. Each radio button is defined by an &lt;Option&gt; tag which must be a direct child tag of the &lt;UserDecision&gt;. The request attribute in the &lt;Option&gt; tag represents the label of the radio button. The message box itself does also have a label which is set on top of the radio buttons.</p> <p>Depending on the checked radio button, the statements below the corresponding &lt;Option&gt; tag are executed. The message box contains an OK- and a CANCEL button. Pressing CANCEL or closing the box via the system menu, aborts the test case unless the <i>cancelIsAbort</i> attribute is set to 0. The test case is suspended as long as the message box exists. The tag data, the top label and the single request label may contain variables which are replaced in advance.</p>		
<p><b>PROPERTIES:</b></p> <p>Must contain &lt;Option&gt; tags which in turn must contain arbitrary tag types.</p>		
ATTRIBUTE	VALUE	DESCRIPTION
label (mandatory)	arbitrary string	Prompt text of <UserDecision> at top of message box. Can contain a variable which is replaced in advance.
request (mandatory)	arbitrary string	Label of radio button. Can contain a variable which is replaced in advance.
<p><b>EXAMPLE:</b></p> <pre>&lt;UserDecision label="Did the Dut change to standby mode?"&gt;   &lt;Option request="yes"&gt;     &lt;LogMessage message="The DUT changed to standby mode" /&gt;   &lt;/Option&gt;   &lt;Option request="no"&gt;     &lt;LogError error="The DUT did not change to standby mode" /&gt;   &lt;/Option&gt; &lt;/UserDecision&gt;</pre>		

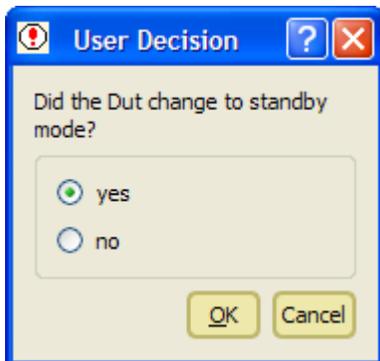


Figure 40: <UserDecision> Message Box

### 4.5.31 <Wait> tag

<b>SYNTAX:</b>		
<code>&lt;Wait event="..." duration="..." frame="..." /&gt;</code>		
<b>DESCRIPTION:</b>		
Suspends the test case until the defined time has elapsed or the described event has occurred.		
<b>PROPERTIES:</b>		
Must not contain any tag data or further tags. Must be an empty tag.		
<b>ATTRIBUTE</b>	<b>VALUE</b>	<b>DESCRIPTION</b>
event	frame	continues test on next received frame
	expframe	continues test on specified received frame
	ack	continues test on acknowledge signal of sent frame
	hpd1	continues test on HPD signal at HDMI Out port 1
	hpd2	continues test on HPD signal at HDMI Out port 2
	hpd3	continues test on HPD signal at HDMI Out port 3
	power	continues test on Power signal at HDMI In port
	physAddr	continues test on physical address assignment
	timeout	continues test after defined time has elapsed. Can be a variable which is replaced in advance.
duration (optional)	decimal number	Wait time in milliseconds. If omitted, the test case waits for the event without timeout. If the event is set to <i>timeout</i> , however, the duration is mandatory. Can be a variable which is replaced in advance.
frame (optional)	regular frame expression	In case the event is set to <i>expframe</i> , the frame attribute is mandatory. It specifies the frame to wait for as a regular expression. Can contain a variable which is replaced in advance.
<b>EXAMPLE:</b>		
<code>&lt;Wait event="expframe" frame="?f 84 1000" duration="10000" /&gt;</code>		

## **4.6 System Variables**

System variables are reserved variables which are exclusively used by the system. They are read-only variables and cannot be changed or overwritten by any tag. They are changed on certain events. These are:

V_SRC	last received source address in hexadecimal (0x0 <= x <= 0xf)
V_DEST	last received destination address in hexadecimal (0x0 <= x <= 0xf)
V_OPCODE	last received opcode in hexadecimal (0x00 <= x <= 0xff)
V_CORRUPT	corruption level of last received frame. Following levels are possible: 0 = Frame is OK 1 = Frame is incomplete (EOM flag is missing) 2 = Invalid bit timings exist 3 = Parse errors exist 4 = Corrupt pulses exist 5 = Frame was aborted
V_FRAME	last received complete frame (hex-encoded)
V_ACK	last received acknowledge value of sent frames: ACK (completely acknowledged) or NACK (partly or not acknowledged at all)
V_TIMEOUT	indicates result of last started timeout event. Event has either occurred before timeout (0), timer has elapsed before event (1) or timer was not set at all (-1).
V_SUB	return value of routine ( <i>subRes</i> attribute in <Return> tag) or -1 if there was no return value.
V_HPD1	HPD was set (1) at HDMI Out port 1 or reset (0)
V_HPD2	HPD was set (1) at HDMI Out port 2 or reset (0)
V_HPD3	HPD was set (1) at HDMI Out port 3 or reset (0)
V_POWER	Power was set (1) at HDMI In port or reset (0)
V_TEPHYSADDR	Physical Address of Test Environment as a four-digit hex number
V_FRAME_CTR	number of frames in <i>Message History List</i> since test case start (sent and received frames)
V_CANCEL	indicates if last user message box was canceled (1) or not (0).
V_DUTLOGADDR	Logical Address of DUT
V_VIRLOGADDR	Logical Address of TE if there is only one virtual device type. If you need more than one virtual device type, assign <i>mode="dummy"</i> in the <Device> tag
V_DUTPHYSADDR	Physical Address of DUT as a four-digit hex number
V_TYPE	DUT's type: TV, RD, TD, AS, PD, or SW.

## **4.7 Cambered Brackets**

Cambered brackets are used to define composed expressions containing whitespaces. These terms are considered as one operand and can be useful when working with operational (see 4.8) or conditional (see 4.9) statements. Note that both operations and

conditions must contain exactly two whitespace characters to separate operands and operator. If one of the operands also contains a whitespace character, cambered brackets must be used. An example for a conditional statement would be:

```
<If condition="V_Frame EQ {2 V_Dest 8c}">
```

The XML interpreter would first evaluate the term in the cambered brackets, i.e. the variable is replaced and all whitespaces are removed. Assume, `V_Dest` is 4, then the result would be:

```
<If condition="V_Frame EQ 248c">
```

Now there is an expression with two whitespaces, two operands and one operator which is considered valid. Note that one cannot write:

```
<If condition="V_Frame EQ 2V_Dest8c">
```

The XML interpreter would detect the start of the variable but it would not detect the end which's why cambered brackets are required here.

## **4.8 Operations**

Operations follow the general syntax "*Operand\_1 Operator Operand\_2*". That is, they consist of two operands and one operator, separated by a whitespace character. Operations are used in `<DoOperation>`- and `<Loop>` tags.

`Operand_1` must be an existing variable which is assigned the result of the operation.

`Operand_2` is an arbitrary value (can be a variable which is replaced in advance).

The `Operator` must be among the following ones:

- = simple assignment
- .= string concatenation
- += addition (of two decimal numbers)
- \*= multiplication (of two decimal numbers)
- = subtraction (of two decimal numbers)
- /= division (of two decimal numbers)

Note that the operation can contain cambered brackets (see 4.7).

## **4.9 Conditional Statements**

Conditional statements follow the general syntax "*Operand\_1 Operator Operand\_2*". That is, they consist of two operands and one operator, separated by a whitespace character. Conditional statements are used in `<If/ElseIf>`- and `<Loop>` tags.

Operand\_1 is an arbitrary value (can be a variable which is replaced in advance).

Operand\_2 is an arbitrary value (can be a variable which is replaced in advance).

The Operator must be among the following ones:

- EQ checks both operands for equality where operand 2 is handled as a regular expression.
- NE checks both operands for inequality where operand 2 is handled as a regular expression.
- LT checks if operand 1 is less than operand 2 (compares 2 numbers)
- LE checks if operand 1 is less than or equal to operand 2 (compares 2 numbers)
- GT checks if operand 1 is greater than operand 2 (compares 2 numbers)
- GE checks if operand 1 is greater than or equal to operand 2 (compares 2 numbers)

Note that the conditional statement can contain cambered brackets (see 4.7).

## 5 Simplay ATC CEC Test Tool

### 5.1 Introduction

Based on the scripting interface (section 4), the full featured version of the CEC Explorer can also be used to perform a complete CEC/CDC ATC test as specified in the HDMI CTS [2]. The complete test is handled by a sub dialog of the main GUI, named CDF/CTS dialog (section 3.1.3.1).

Prior to test start, the user must walk through the CDF dialog to fill in the device specific 'Capabilities Declaration Form' (CDF). That is, the device's capabilities are entered by means of checking the supported opcodes and other test relevant properties. Based on these settings, the CEC Explorer then determines the list of test cases the device must execute to pass an ATC test. Once created, the settings and the resulting test case list can be saved to a file (CDF files) for later usage or changes.

When the ATC test is started, the user is automatically guided through the complete test. This refers to requests for arranging test case specific setups but also includes a number of user interactions which are either instructions or assessments. During test execution, the results are logged enabling the user to subsequently recheck the results of individual test cases.

In case of errors or warnings, single tests can also be repeated without overwriting the results of other tests. A 'Summary Log File' stores the latest results for a device where each single test is provided an execution date, its main log output and its own test statistics. In addition, each single test case creates its own detailed log file and its own Pulse Sequence file. Therefore, it is quite easy to check the reasons for potential test failures.

The following sections present the CDF dialog in more detail and describe how to perform an ATC test with the CEC Explorer application.

### 5.2 The CEC ATC Test Dialog

The Simplay ATC Test Tool is started by selecting the item "Open CDF/CTS CEC 1.4" in the "Test Scripting" menu. This opens the CEC ATC Test Dialog reflected in (Figure 41). It consists of a control- and information unit on the left side and a tab widget on the right side. The dialog is not resizable and can be closed and minimized without losing its data.

#### 5.2.1 Control- and Information Unit

This unit (on the CDF dialog's left side) is always visible and provides some tool buttons at its top, the current test case list in the middle, and an information box at its bottom.

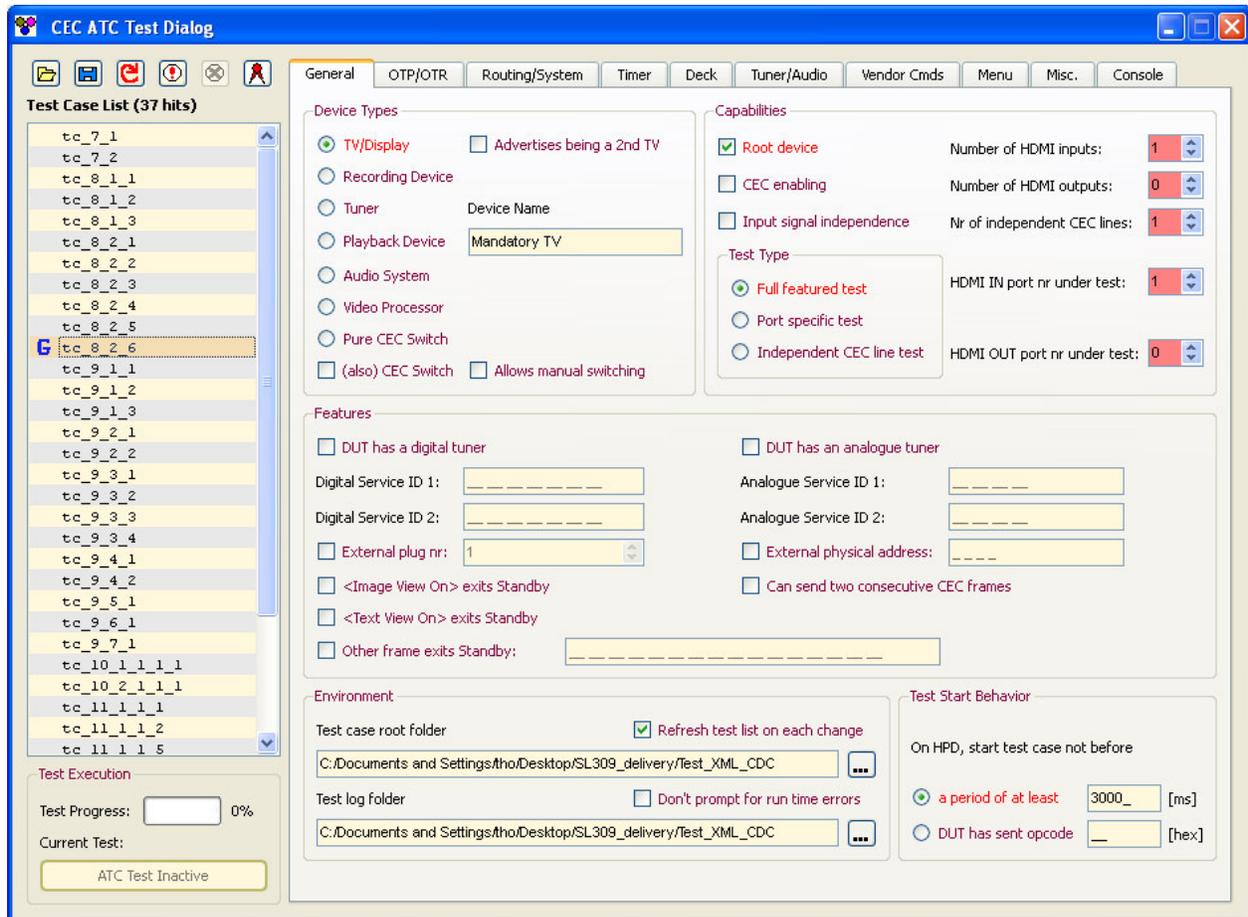


Figure 41: CEC ATC Test Dialog. Control unit (left) and Tab Widget (right)

### 5.2.1.1 The Tool Buttons of the CDF dialog

The CDF dialog contains six tool buttons the key shortcuts of which are presented in brackets. Note that the keys are active only if the CDF dialog gains focus.

 **Load CDF Data (Key F4).** This button loads CDF data that was previously prepared. Usually, a device under test (DUT) is associated with a specific CDF file, containing all the capabilities of the device. Once prepared, it can be saved and reloaded for later test sessions. In case a device has more than one HDMI port, there should be one CDF file for each port to execute port-specific tests only. During test execution, this button is disabled.

 **Save CDF Data (Key F5).** This button saves CDF data. Prior to test start, one fills in all the properties of the device under test in the tab widget (section 5.2.2). The data can then be saved for later usage. The file format is XML. During test execution, this button is disabled.

 **Refresh test case list (Key F6).** This button updates the list of test cases that currently results from the settings in the tab widget. Checking the check box

'Refresh test list on each change' in the *Environment* group box automatically updates the list after each change. During test execution, this button is disabled.

-  **Run selected test cases (Key F7):** This button starts the test session and executes all test cases of the test case list that are currently selected. Note that you must select at least one test case to enable this button. Before the test session starts, a number of checks are performed which are listed in section 5.6.1. To run the complete list of test cases, just select the complete list (via Ctrl + a, for example) in advance. Once started, the CDF dialog and many functionalities of the main GUI are disabled to avoid undesired disruptions.
-  **Stop ATC test (Key F8):** Stops the execution of the ATC test immediately. The currently running test is stopped, the last log output is dumped out, and the CDF dialog is completely enabled again (apart from this button).
-  **Create Test Certificate:** When all the ATC tests are done, click this button to generate a test certificate which contains the summary of all tests.

### 5.2.1.2 The Test Case List

The test case list just below the tool buttons is the reflection of the currently selected elements in the tab widget. Its content is either adapted just in time (if the check box "Refresh test list on each change" is checked) or as soon as the "Refresh test case list" tool button is pressed.

For the purpose of execution, the user can select a single item, multiple items or the complete list. The execution is then started by means of the corresponding tool button (section 5.2.1.1), the shortcut key (F7) or by the list's context menu (Figure 42).

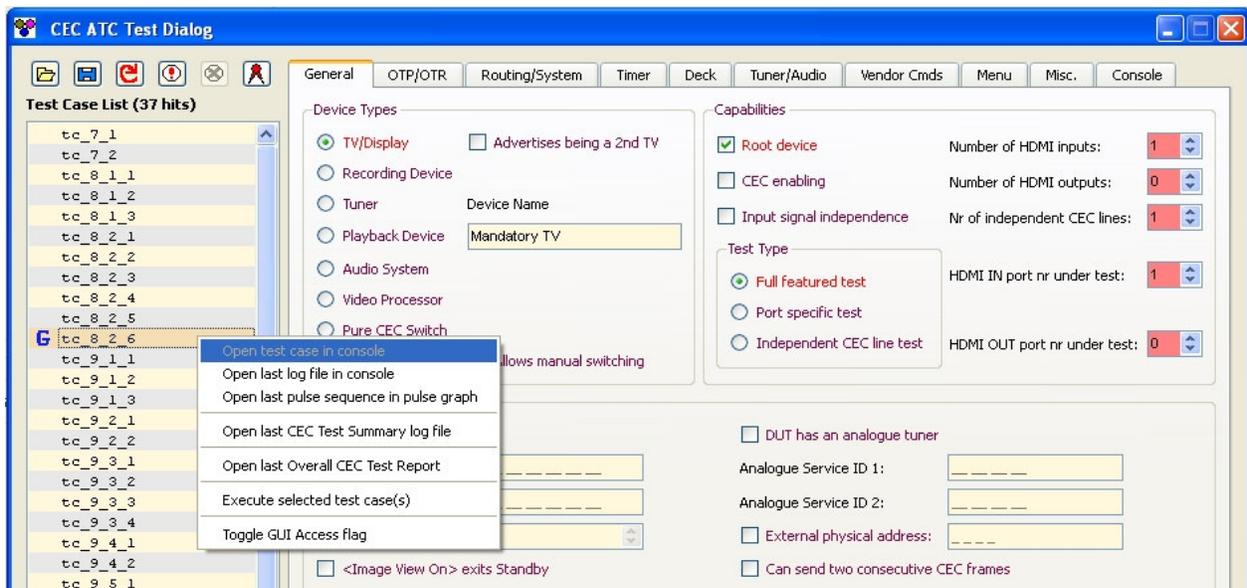


Figure 42: Context menu of test case list

The context menu (achievable via the right mouse button) is not available during test executions. Depending on single or multiple selections, it provides the following features:

- **Open test case in console:** Opens the selected test case in the Console View. The Console View is the last Tab of the tab widget. On top of the Console View, the current content is reflected (in Figure 42, it is test case 8.2-1). This item is only available on single list selections. It might also be disabled if the CEC Explorer has not created the test case map, yet.
- **Open last log file in console:** During test executions, each test case creates an individual log file which is located in the *Test Log Folder* (section 5.2.2.1). This context menu item tries to open this log file in the Console View. This item is only available on single list selections. The *Test Log Folder* must have been entered before.
- **Open last pulse sequence in pulse graph:** During test executions, each test creates its own *Pulse Sequence File* which is stored in the same folder as the log file. This context menu item tries to open the psq file in the *Message History List* of the main GUI. This item is only available on single list selections. The *Test Log Folder* must have been entered before.
- **Open last CEC Test Summary log file:** During an ATC test, a summary log file is created, containing the results of the individual tests. It is stored in an XML format, located in the *Test Log Folder*, and can be viewed in a browser if a corresponding style sheet is used (Figure 43). Alternatively, the summary log file can be viewed in a simple text editor or in the Console View of the CDF dialog. This item is available on single and multiple selections. The *Test Log Folder* must have been entered before.
- **Execute selected test case(s):** This item executes the currently selected test case(s) as already described for the corresponding tool button (section 5.2.1.1).
- **Toggle GUI Access flag:** Test cases do normally run autonomously with well defined user interactions. However, in some cases DUTs may need different circumstances to pass a specific test case, e.g. an additional device or a vendor command. For that purpose, single test cases may be marked for GUI access which is granted at test case start in most cases (controlled by a specific XML tag) but also later or repeatedly in some cases.  
During the GUI access period, the test case is suspended and the GUI elements to add more devices or to send additional CEC frames are enabled. However, access is restricted, i.e. the user may not remove any predefined virtual devices, and he must not send any frames in the name of the DUT. All actions are logged to be reproducible later. Checking this toggleable item enables GUI access for a given test case which is visible by a "G" icon as reflected for test case 8.2-6 in Figure 41.

<b>SUMMARY LOG FILE</b>		
<b>Device</b>  Date of last run: 2009/04/03 Start Time of last run: 00:51:45 End Time of last run: 00:52:00 Duration of last run: 14 seconds	<b>Port Number(s) Under Test:</b> HDMI OUT Port Nr: 1 HDMI IN Port Nr: 0	<b>Statistics:</b> Total Nr of Errors: 12 Total Nr of Warnings: 12  Number of passed Testcases: 0 Number of failed Testcases: 4 Number of aborted Testcases: 0
	<b>Used Versions:</b> Cec Explorer: 2.5 Firmware: undetected Test Suite: 1195	<b>Overall Test Result:</b> <b>FAILED</b>
<a href="#">Test case tc 8 1 1</a> Last execution date: 2009/04/03 00:51:45,	3 error(s) (Previous results: failed aborted failed failed )	3 warning(s)
<a href="#">Test case tc 8 1 2</a> Last execution date: 2009/04/02 21:10:00,	3 error(s) (Previous results: failed aborted failed failed failed )	3 warning(s)
<pre> WARNING: Sent frame was not acknowledged ERROR: The DUT did not respond the &lt;Abort&gt; message with a correct &lt;FeatureAbort&gt; in time WARNING: Sent frame was not acknowledged ERROR: The DUT did not respond the &lt;Abort&gt; message with a correct &lt;FeatureAbort&gt; in time WARNING: Sent frame was not acknowledged ERROR: The DUT did not respond the &lt;Abort&gt; message with a correct &lt;FeatureAbort&gt; in time                     </pre>		
<a href="#">Test case tc 8 1 3</a> Last execution date: 2009/04/03 00:51:54,	3 error(s) (Previous results: failed failed failed failed failed )	3 warning(s)
<a href="#">Test case tc 8 2 1</a>	3 error(s)	3 warning(s)

Figure 43: Output of Summary Log File in a Browser

In the Test Case List, there will be a small icon that shows the status of each test case after the test is completed. There are total of four types of icon and each one represents different status of the test case. Following is the detail of each icon:

- Pass:** The DUT has passed the test.
- Warning:** The DUT has passed the test without any error, but there are some warning messages.
- Fail:** The DUT has failed the test.
- Abort:** The user aborted the test prior to completion.

### 5.2.1.3 The Information Box

The *Information Box* is enabled during test case execution. It shows the currently executed test case, the complete test progress and the *GUI Access* button. Besides toggling the GUI Access flag via the list's context menu (see 5.2.1.2), one may also decide at short notice to access the GUI. For a period of several seconds, the button is enabled prior to each test enabling the user to toggle it when desired.

## 5.2.2 The Tab Widget

The *Tab Widget* represents the actual CDF and contains some more information for the ATC test configuration. All CDF settings are described in the HDMI CEC CTS [2], Appendix 1 (CEC Capabilities Declaration Form). Therefore, this manual only provides a short overview of the tabs. Note that many GUI elements in the CDF dialog are linked to tool tip texts giving more detailed information about their functionality. Also note that some elements must be triggered by other ones to be enabled.

### 5.2.2.1 The "General Tab"

This tab shows the general settings regarding the capabilities of the DUT (Figure 44). It is divided into several group boxes dealing with different aspects of the DUT:

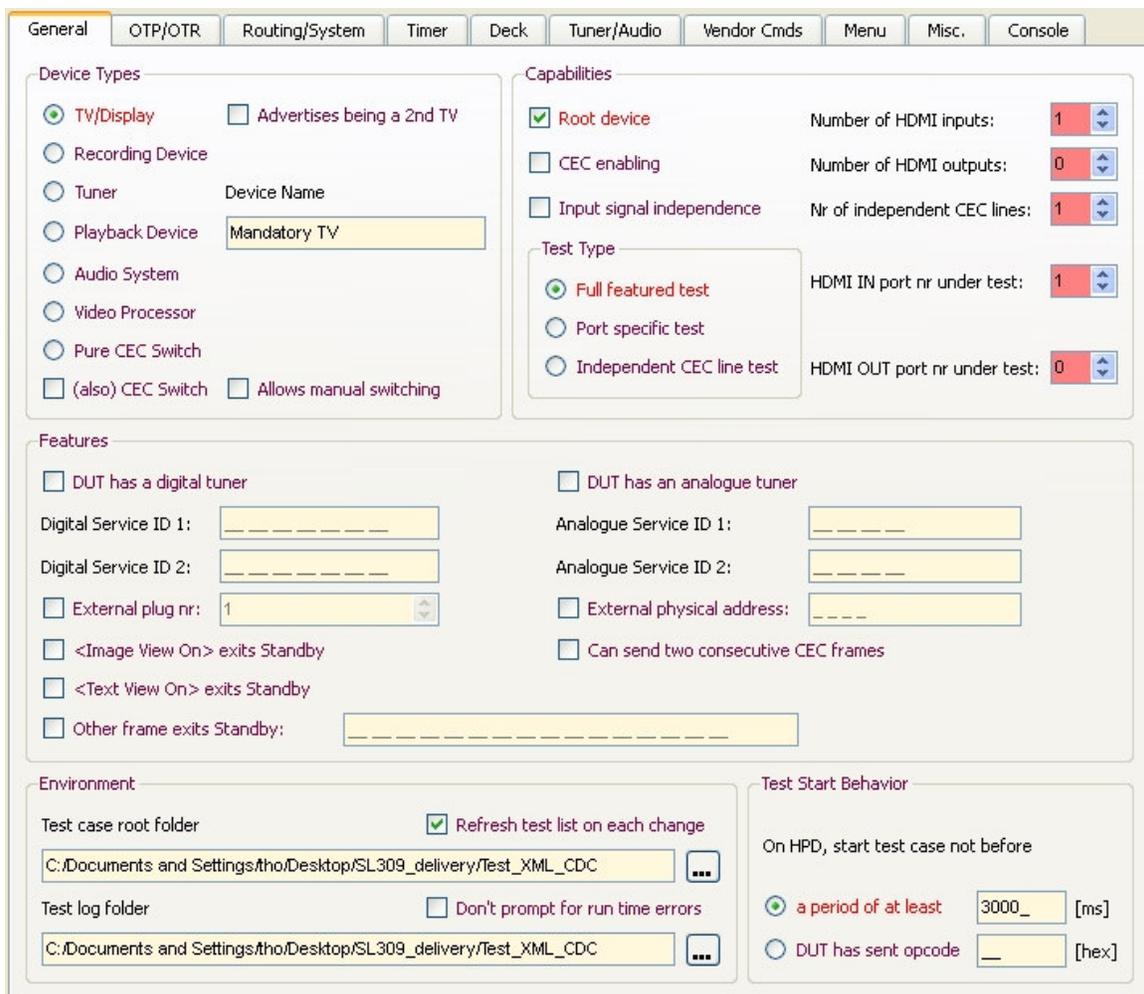


Figure 44: The "General Tab"

- **Device Types:** Here, the device type of the DUT is set. If a DUT represents more than one type, more than one CDF file has to be created. The switch device is an

exception. It can be additionally checked to each device type. If the DUT is a pure switch, the "*None of the above*" radio button must be checked. The *Device Name* field is to enter the name of the device. When this name is changed, a new summary file must be created by the CEC Explorer (for a new device). Therefore, the old summary file must be deleted manually in advance.

- **Capabilities:** Besides information of the CDF, the user must also specify here which port is currently under test. This information is logged and used for port specific test (when checked). Note that some of the ATC tests are required to be performed for each port of the DUT. In that case, a new CDF is actually created and the port numbers point to the ports under test. With multiple ports on the DUT, at least one port needs to run with 'full feature' test and the rest can run 'port specific' test, or choose 'independent CEC line' test if there is more than one CEC logic inside the DUT. For port specific tests, the CEC Explorer also demands to create a new summary file, i.e. the old one must be deleted by the user.
- **Features:** All elements here are derived from the CDF and are described elsewhere [2].
- **Environment:** Here, the user must define where to find the test cases and where to write the test output to (log- and psq files as well as the Summary file). There are '*Browse*' buttons to simplify the folder definitions. Note that the test case root folder must be named '*Test\_XML\_CEC*'. Both fields must point to a valid folder, otherwise the test is not executed. The two additional check boxes care for a permanent update of the test case list after each change (as already described in 5.2.1.1) and the second check box prevents popping up message boxes on run time errors.
- **Test Start Behavior:** Here, the user can define when a test case with a source under test shall start. In this context, note that each source test starts with a HPD toggle of the CEC Explorer hardware. Here, one can define that a test either starts after a certain period (of at least 1000ms) or after a defined opcode which must be entered as a two digit hex code.

### 5.2.2.2 The "OTP/OTR Tab"

This tab deals with all aspects of the '*One Touch Play / One Touch Record*' features (Figure 45). All elements here are derived from the CDF and are described elsewhere [2].

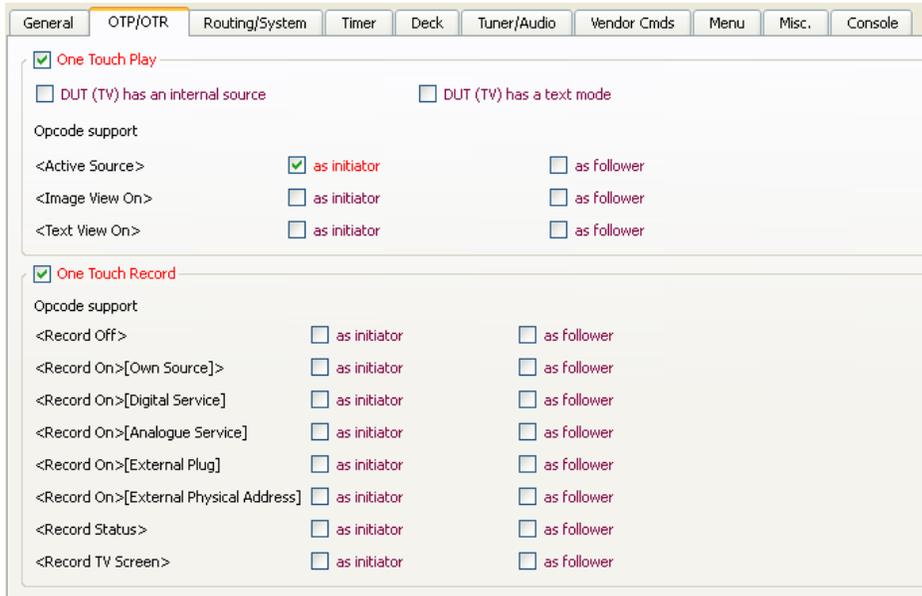


Figure 45: The "OTP/OTR Tab"

### 5.2.2.3 The "Routing/System Tab"

This tab deals with all aspects of the 'Routing Information / System Information' features (Figure 46). All elements here are derived from the CDF and are described elsewhere [2]. Note that supported and not-supported language codes must be encoded in hexadecimal. Also refer to the tool tip texts of the elements for further information.

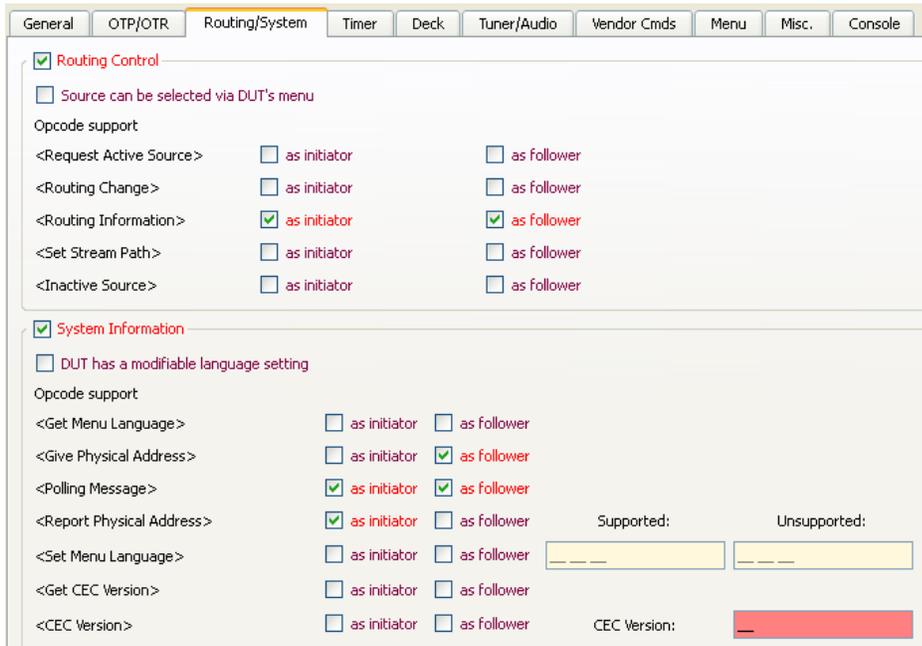


Figure 46: The "Routing/System Tab"

### 5.2.2.4 The "Timer Tab"

This tab deals with all aspects of the 'Timer Programming' feature (Figure 47). All elements here are derived from the CDF and are described elsewhere [2].

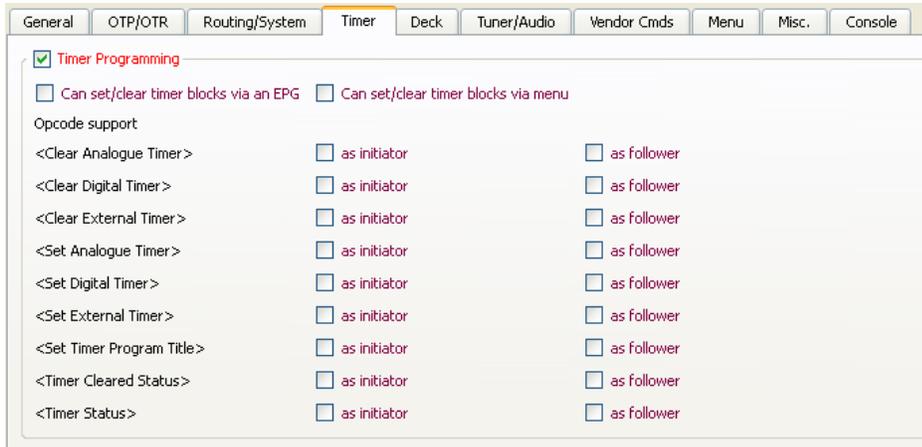


Figure 47: The "Timer Tab"

### 5.2.2.5 The "Deck Tab"

This tab deals with all aspects of the 'Deck Control' feature (Figure 48). All elements here are derived from the CDF and are described elsewhere [2].

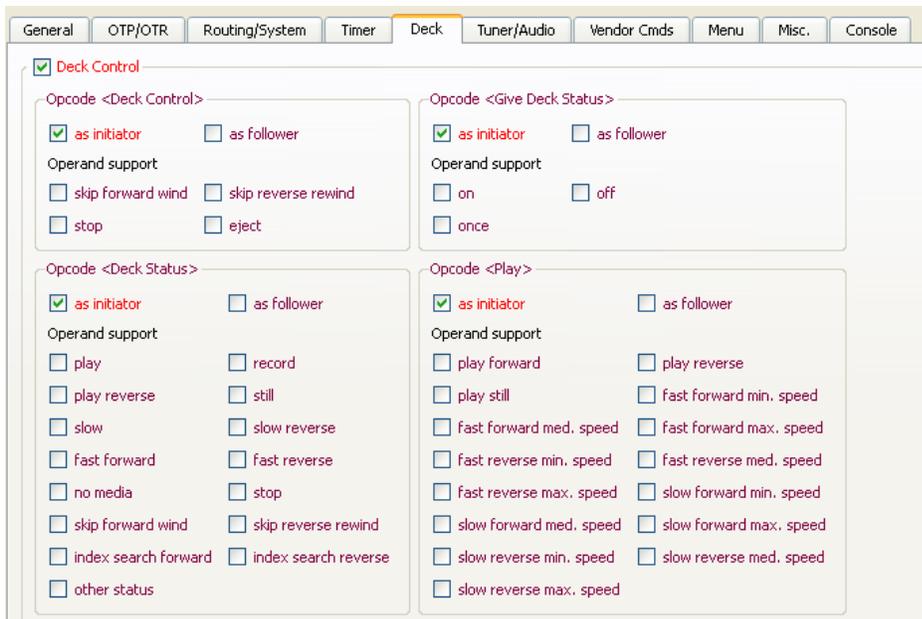


Figure 48: The "Deck Tab"

### 5.2.2.6 The "Tuner/Audio Tab"

This tab deals with all aspects of the 'Tuner Control / Audio Control' features (Figure 49). All elements here are derived from the CDF and are described elsewhere [2].

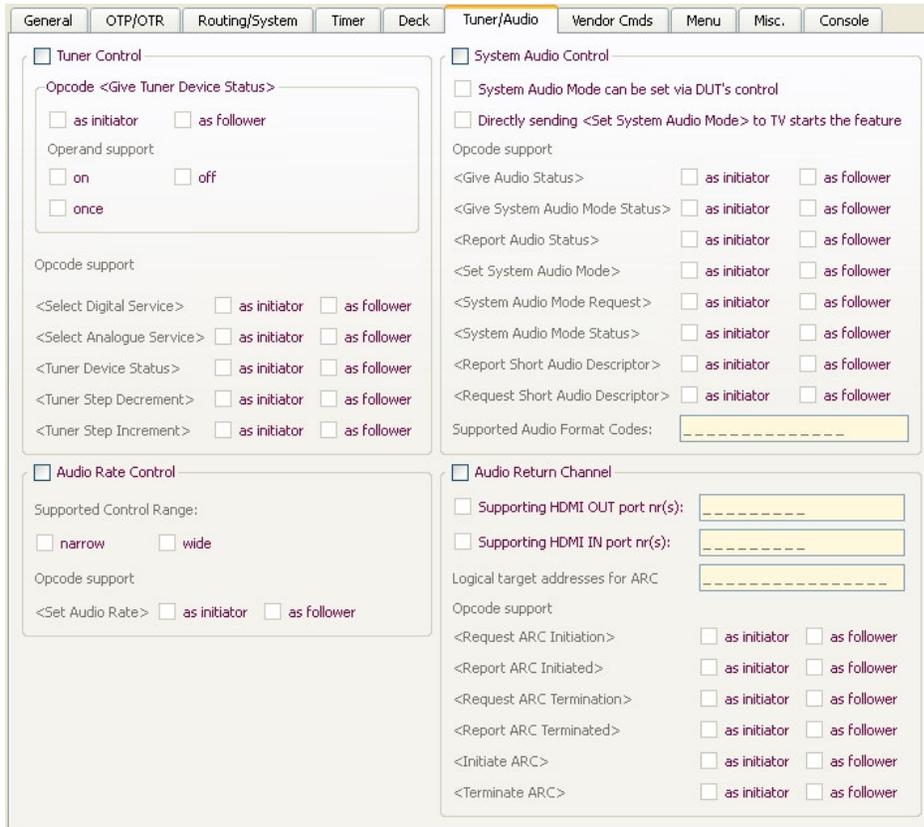


Figure 49: The "Tuner/Audio Tab"

### 5.2.2.7 The "Vendor Cmds Tab"

This tab deals with all aspects of the 'Vendor Commands' feature (Figure 50). All elements here are derived from the CDF and are described elsewhere [2]. Note that Vendor IDs must be encoded in hexadecimal.

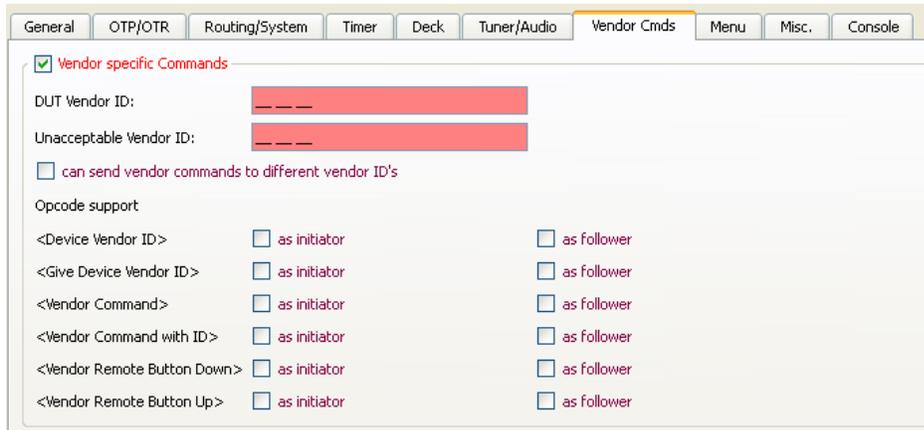


Figure 50: The "Vendor Cmts Tab"

### 5.2.2.8 The "Menu Tab"

This tab deals with all aspects of the 'Device Menu Control / Remote Control Pass Through' features (**Error! Reference source not found.**). All elements here are derived from the CDF and are described elsewhere [2]. Note that all *User Control Codes* are encoded in hexadecimal.

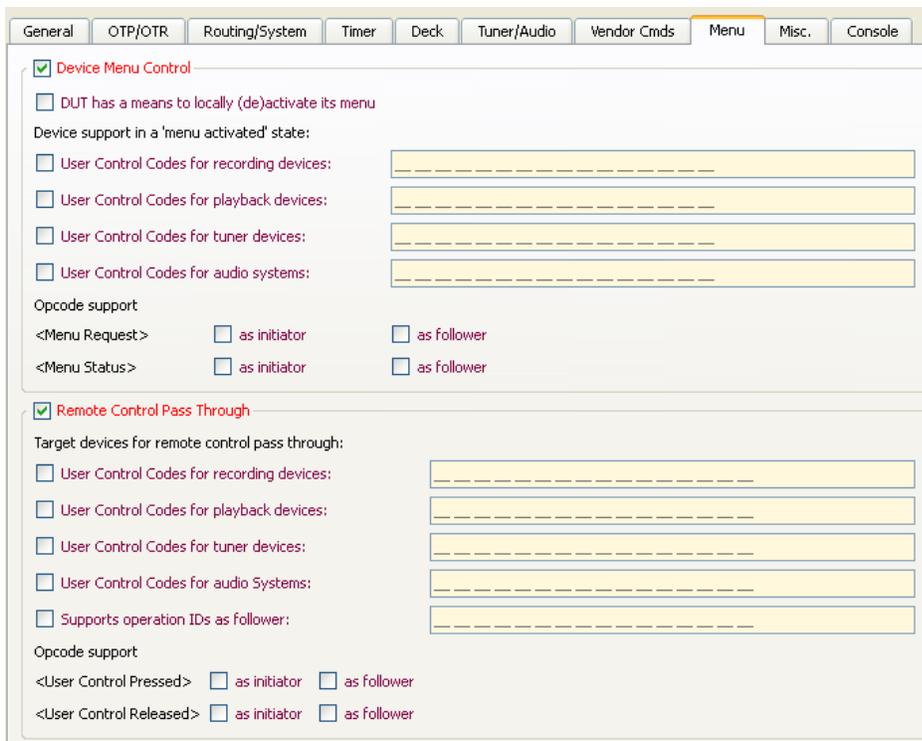


Figure 51: The "Menu Tab"

### 5.2.2.9 The "Misc Tab"

This tab deals with all aspects of the 'OSD Display / Device OSD Transfer / Standby / Power Status / Feature Abort' features (Figure 52). All elements here are derived from the CDF and are described elsewhere [2]. Note that all OSD strings are encoded in hexadecimal.

The "Enable Auto Response Settings" group box enables the user to determine whether or not and how the CEC Explorer shall automatically respond to certain CEC requests. In this context, note that the CEC Explorer automatically responds to certain opcodes which is not configurable (section 5.6.2).

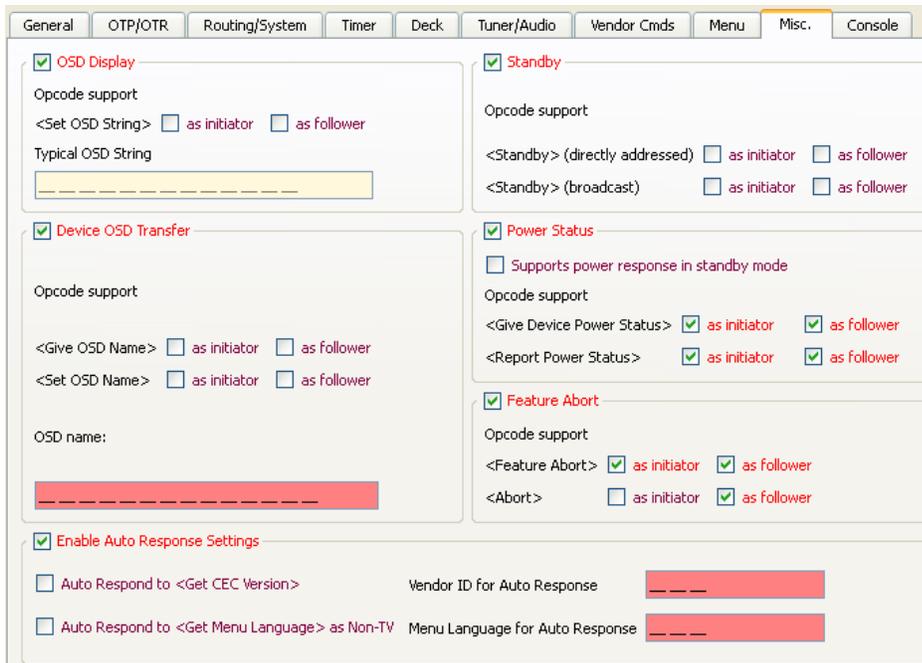
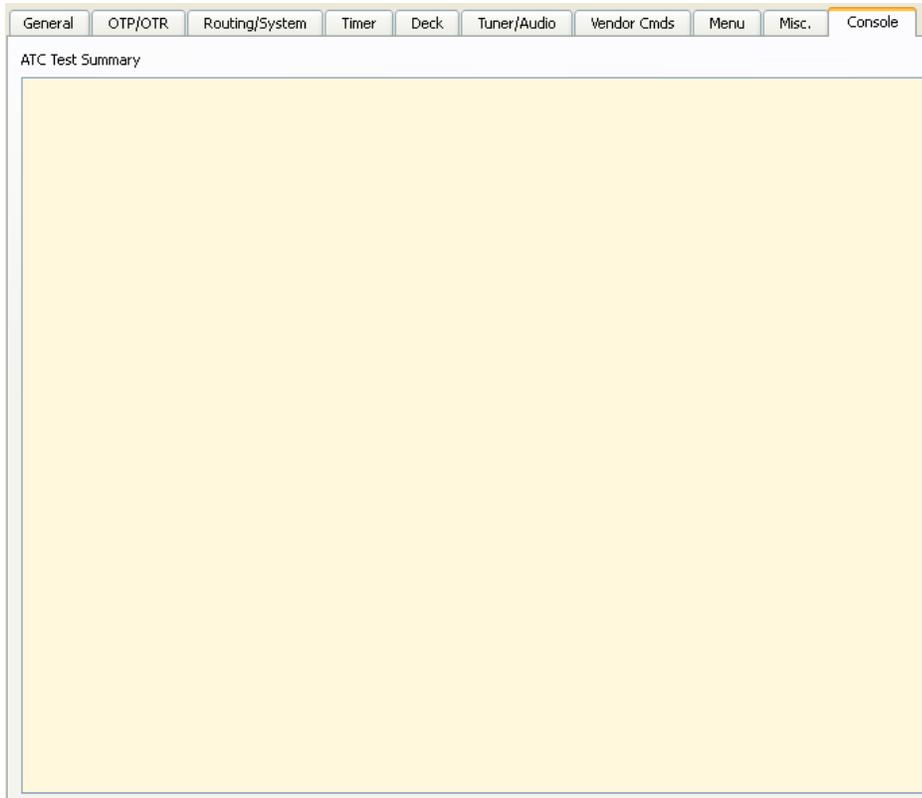


Figure 52: The "Misc Tab"

### 5.2.2.10 The "Console Tab"

The Console Tab is a multi-purpose, read-only console view (Figure 53). It can be used to view XML test cases, their last log files and the Summary Log file. During ATC tests, the user is prompted with the log output to the *CEC Summary Log* file. The console is exclusively loaded via the test case list's context menu. The label above the console points to its content.



**Figure 53: The "Console Tab"**

Note that due to the XML format, test cases can also be viewed in a web browser if the corresponding stylesheet file has been installed. An excerpt, representing test case 9.3-3 is reflected in Figure 54.

**Testcase 9.3.3 (HDMI-CTS)**  
( Revision: 1301 )

**Test Objective:**  
Ensure that the DUT will accept a negatively acknowledged response to a broadcast message and tries to re-transmit the message up to 5 times.

**Constraints:**  
No constraints defined

**Required Method:**  
Invoke the DUT to broadcast a message as described below: For all devices except CEC Switches: Send a <Give Physical Address> message to the DUT. For CEC Switches: Ensure that the DUT has been allocated a physical address of 1.1.0.0. Broadcast a <Routing Information> [1.1.0.0] message. Negatively acknowledge the header block within the message that the DUT broadcasts. Negatively acknowledge a message block within all retransmission attempts.

**Test Setup:**

Device Connections	Test Relevant Properties	Further Device Properties (optional)	State Information (optional)
	<b>Mode:</b> open <b>Type:</b> Virtual TV if DUT isn't TV <b>Phys. Addr.:</b> 0.0.0.0 <b>Log. Addr.:</b> (CDF dependent or 0 if virtual)	<b>Vendor-ID:</b> n/a <b>OSD Name:</b> n/a <b>System Audio:</b> n/a <b>Audio Volume:</b> n/a <b>Audio Mute:</b> n/a <b>Switch Address:</b> n/a <b>Menu Lang.:</b> n/a	<b>Power Status:</b> n/a <b>Active Source:</b> n/a <b>Menu:</b> n/a <b>Deck Status:</b> n/a
	<b>Mode:</b> open <b>Type:</b> Virtual RD if DUT is TV <b>Phys. Addr.:</b> 1.0.0.0 <b>Log. Addr.:</b> (CDF dependent or 1 if virtual)	<b>Vendor-ID:</b> n/a <b>OSD Name:</b> n/a <b>System Audio:</b> n/a <b>Audio Volume:</b> n/a <b>Audio Mute:</b> n/a <b>Switch Address:</b> n/a <b>Menu Lang.:</b> n/a	<b>Power Status:</b> n/a <b>Active Source:</b> n/a <b>Menu:</b> n/a <b>Deck Status:</b> n/a

Bring it home. Plug it in.

**Test Execution:**

*General Information: Statistics are enabled, log file = tc\_9\_3\_3.log, initial board reset is performed*

```

*****
ROUTINE main( void ) {
    DefineVar: V_Ctr = 0

    If ( V_DUTLOGADDR NE f ) {
        SetAckSignal: Acknowledge mask is set to 0xffffffff (for the next 2 frame(s))
        SendCecFrame: V_VIRTUALADDR V_DUTLOGADDR 83 (wait = All11)
        SetAckSignal: Acknowledge mask is set to 0xffffffff (for the next 5 frame(s))

        Loop ( (no loop variable); Condition: V_Ctr LT 8; (no loop operation) ) {
            Wait: Event = exprframe(V_DUTLOGADDR f*) (duration = 1000 ms)
        }
    }
}
            
```

Figure 54: Test case presentation in web browser

### 5.3 The CDC ATC Test Dialog

Similar to CEC ATC Test, the Simplay CDC ATC Test Tool is started by selecting the item “Open CDF/CTS CDC 1.4” in the “Test Scripting” menu. This opens the CDC ATC Test Dialog reflected in (Figure 41). In consists of a control- and information unit on the left side and a tab widget on the right side. The dialog is not resizable and can be closed and minimized without losing its data.

#### 5.3.1 Control- and Information Unit

This unit (on the CDF dialog's left side) is always visible and provides some tool buttons at its top, the current test case list in the middle, and an information box at its bottom.

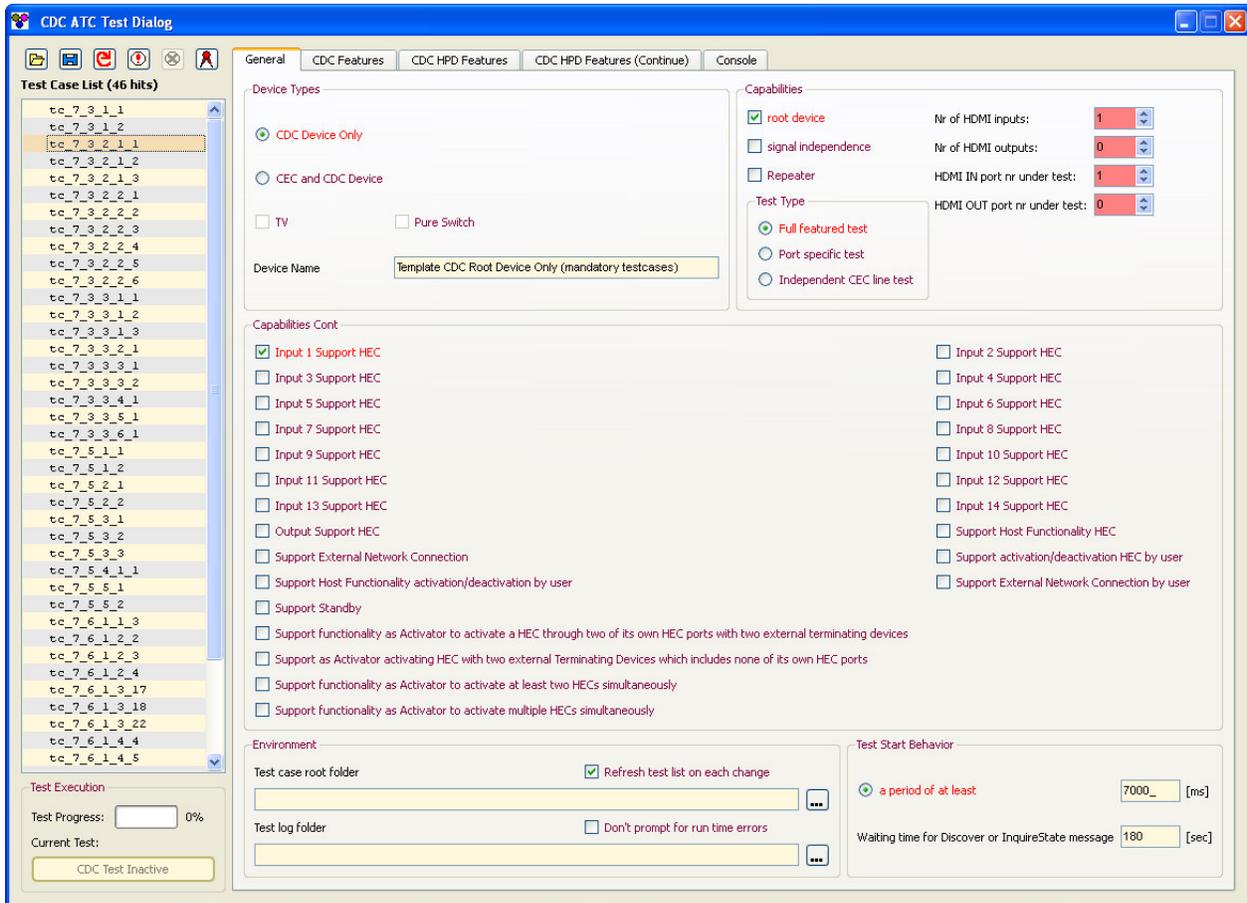


Figure 55: CDC ATC Test Dialog. Control unit (left) and Tab Widget (right)

### 5.3.1.1 The Tool Buttons of the CDF dialog

The CDF dialog contains six tool buttons the key shortcuts of which are presented in brackets. Note that the keys are active only if the CDF dialog gains focus.

-  **Load CDF Data (Key F4).** This button loads CDF data that was previously prepared. Usually, a device under test (DUT) is associated with a specific CDF file, containing all the capabilities of the device. Once prepared, it can be saved and reloaded for later test sessions. In case a device has more than one HDMI port, there should be one CDF file for each port to execute port-specific tests only. During test execution, this button is disabled.
-  **Save CDF Data (Key F5).** This button saves CDF data. Prior to test start, one fills in all the properties of the device under test in the tab widget (section 5.2.2). The data can then be saved for later usage. The file format is XML. During test execution, this button is disabled.
-  **Refresh test case list (Key F6).** This button updates the list of test cases that currently results from the settings in the tab widget. Checking the check box

'Refresh test list on each change' in the *Environment* group box automatically updates the list after each change. During test execution, this button is disabled.

-  **Run selected test cases (Key F7):** This button starts the test session and executes all test cases of the test case list that are currently selected. Note that you must select at least one test case to enable this button. Before the test session starts, a number of checks are performed which are listed in section 5.6.1. To run the complete list of test cases, just select the complete list (via Ctrl + a, for example) in advance. Once started, the CDF dialog and many functionalities of the main GUI are disabled to avoid undesired disruptions.
-  **Stop ATC test (Key F8):** Stops the execution of the ATC test immediately. The currently running test is stopped, the last log output is dumped out, and the CDF dialog is completely enabled again (apart from this button).
-  **Create Test Certificate:** When all the ATC tests are done, click this button to generate a test certificate which contains the summary of all tests.

### 5.3.1.2 The Test Case List

The test case list just below the tool buttons is the reflection of the currently selected elements in the tab widget. It's content is either adapted just in time (if the check box "Refresh test list on each change" is checked) or as soon as the "Refresh test case list" tool button is pressed.

For the purpose of execution, the user can select a single item, multiple items or the complete list. The execution is then started by means of the corresponding tool button (section 5.2.1.1), the shortcut key (F7) or by the list's context menu (Figure 42).

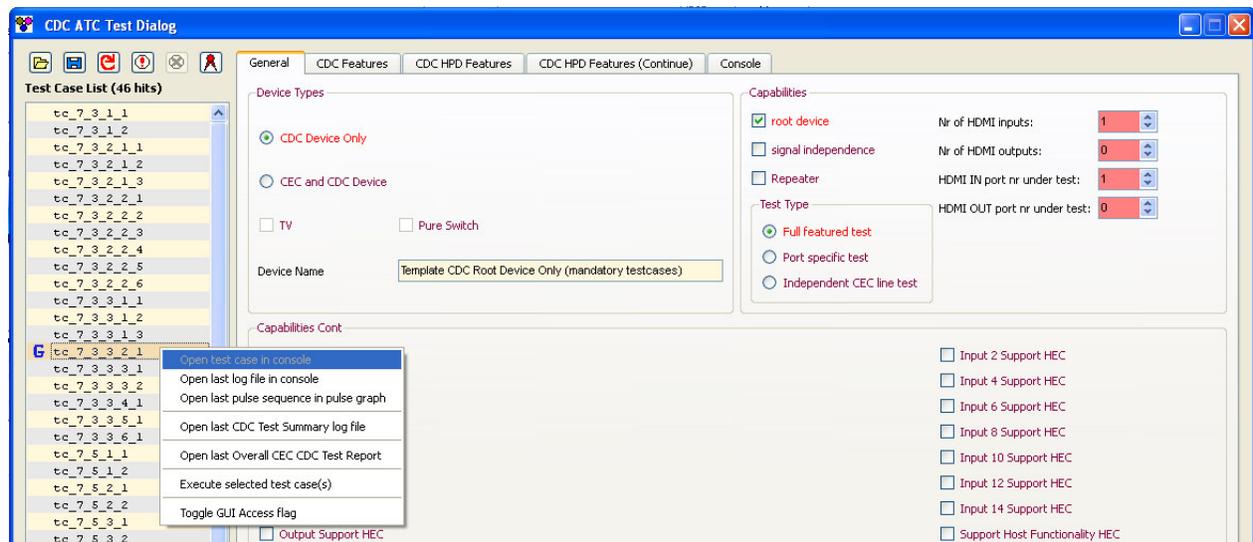


Figure 56: Context menu of test case list

The context menu (achievable via the right mouse button) is not available during test executions. Depending on single or multiple selections, it provides the following features:

- **Open test case in console:** Opens the selected test case in the Console View. The Console View is the last Tab of the tab widget. On top of the Console View, the current content is reflected. This item is only available on single list selections. It might also be disabled if the CEC Explorer has not created the test case map, yet.
- **Open last log file in console:** During test executions, each test case creates an individual log file which is located in the *Test Log Folder* (section 5.2.2.1). This context menu item tries to open this log file in the Console View. This item is only available on single list selections. The *Test Log Folder* must have been entered before.
- **Open last pulse sequence in pulse graph:** During test executions, each test creates its own *Pulse Sequence File* which is stored in the same folder as the log file. This context menu item tries to open the psq file in the *Message History List* of the main GUI. This item is only available on single list selections. The *Test Log Folder* must have been entered before.
- **Open last CDC Test Summary log file:** During an ATC test, a summary log file is created, containing the results of the individual tests. It is stored in an XML format, located in the *Test Log Folder*, and can be viewed in a browser if a corresponding style sheet is used. Alternatively, the summary log file can be viewed in a simple text editor or in the Console View of the CDF dialog. This item is available on single and multiple selections. The *Test Log Folder* must have been entered before.
- **Execute selected test case(s):** This item executes the currently selected test case(s) as already described for the corresponding tool button (section 5.2.1.1).
- **Toggle GUI Access flag:** Test cases do normally run autonomously with well defined user interactions. However, in some cases DUTs may need different circumstances to pass a specific test case, e.g. an additional device or a vendor command. For that purpose, single test cases may be marked for GUI access which is granted at test case start in most cases (controlled by a specific XML tag) but also later or repeatedly in some cases.  
During the GUI access period, the test case is suspended and the GUI elements to add more devices or to send additional CEC frames are enabled. However, access is restricted, i.e. the user may not remove any predefined virtual devices, and he must not send any frames in the name of the DUT. All actions are logged to be reproducible later. Checking this toggleable item enables GUI access for a given test case which is visible by a "G" icon.

In the Test Case List, there will be a small icon that shows the status of each test case after the test is completed. There are total of four types of icon and each one represents different status of the test case. Following is the detail of each icon:

-  **Pass:** The DUT has passed the test.
-  **Warning:** The DUT has passed the test without any error, but there are some warning messages.
-  **Fail:** The DUT has failed the test.
-  **Abort:** The user aborted the test prior to completion.

### 5.3.1.3 The Information Box

The *Information Box* is enabled during test case execution. It shows the currently executed test case, the complete test progress and the *GUI Access* button. Besides toggling the GUI Access flag via the list's context menu (see 5.2.1.2), one may also decide at short notice to access the GUI. For a period of several seconds, the button is enabled prior to each test enabling the user to toggle it when desired.

### 5.3.2 The Tab Widget

The *Tab Widget* represents the actual CDF and contains some more information for the ATC test configuration. All CDF settings are described in the HDMI CEC CTS [2], Appendix 1 (CEC Capabilities Declaration Form). Therefore, this manual only provides a short overview of the tabs. Note that many GUI elements in the CDF dialog are linked to tool tip texts giving more detailed information about their functionality. Also note that some elements must be triggered by other ones to be enabled.

#### 5.3.2.1 The "General Tab"

This tab shows the general settings regarding the capabilities of the DUT (Figure 44). It is divided into several group boxes dealing with different aspects of the DUT:

Figure 57: The "General Tab"

- Device Types:** Here, the device type of the DUT is set. If a DUT represents more than one type, more than one CDF file has to be created. The *Device Name* field is to enter the name of the device. When this name is changed, a new summary file must be created by the CEC Explorer (for a new device). Therefore, the old summary file must be deleted manually in advance.
- Capabilities:** Besides information of the CDF, the user must also specify here which port is currently under test. This information is logged and used for port specific test (when checked). Note that some of the ATC tests are required to be performed for each port of the DUT. In that case, a new CDF is actually created and the port numbers point to the ports under test. With multiple ports on the DUT, at least one port needs to run with 'full feature' test and the rest can run 'port specific' test, or choose 'independent CEC line' test if there is more than one CEC

logic inside the DUT. For port specific tests, the CEC Explorer also demands to create a new summary file, i.e. the old one must be deleted by the user.

- **Capabilities Cont(inued):** All elements here are derived from the CDF.
- **Environment:** Here, the user must define where to find the test cases and where to write the test output to (log- and psq files as well as the Summary file). There are '*Browse*' buttons to simplify the folder definitions. Note that the test case root folder must be named '*Test\_XML\_CDC*'. Both fields must point to a valid folder, otherwise the test is not executed. The two additional check boxes care for a permanent update of the test case list after each change (as already described in 5.2.1.1) and the second check box prevents popping up message boxes on run time errors.
- **Test Start Behavior:** Here, the user can define when a test case with a source under test shall start. In this context, note that each source test starts with a HPD toggle of the CEC Explorer hardware. Here, one can define that a test either starts after a certain period (of at least 1000ms) or after a defined opcode which must be entered as a two digit hex code.

### 5.3.2.2 The "CDC Features Tab"

This tab deals with all aspects of the '*CDC HEC Features*' (Figure 45). All elements here are derived from the CDF and are described elsewhere [2].

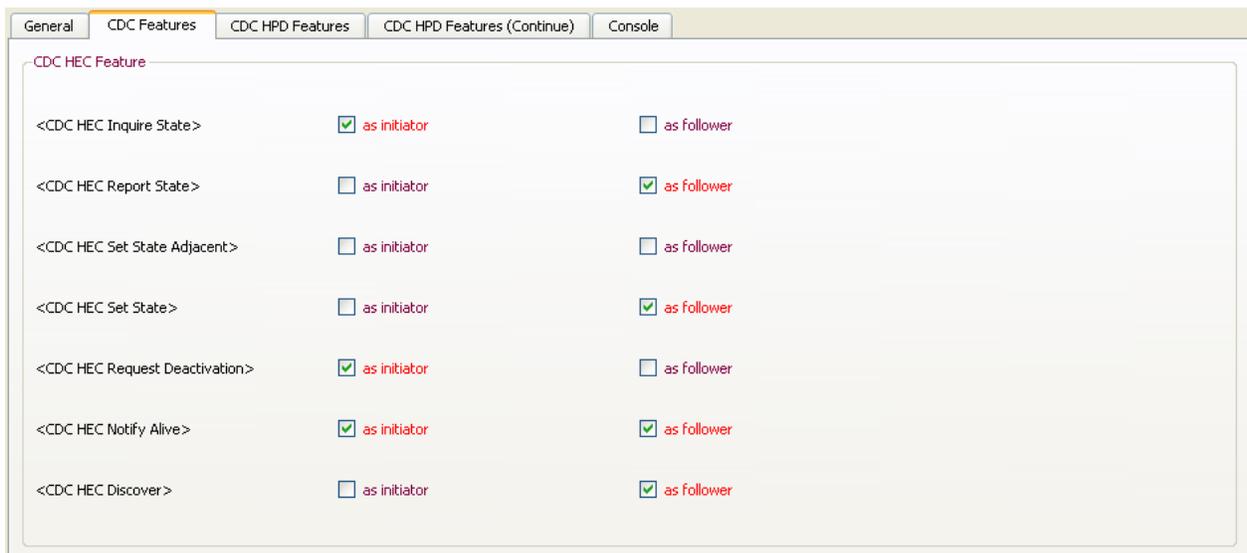


Figure 58: The "CDC Features Tab"

### 5.3.2.3 The "CDC HPD Features Tab"

This tab deals with all aspects of the 'CDC HPD Features' (Figure 46, Figure 46). All elements here are derived from the CDF and are described elsewhere [2].

The screenshot shows the 'CDC HPD Features' tab in the Simplay CEC Explorer. The interface is divided into three main sections:

- CDC HPD Opcode:** Contains two rows of configuration options. The first row is for '<CDC HPD Set State>' and the second is for '<CDC HPD Report State>'. Each row has a checked checkbox for 'as initiator' and an unchecked checkbox for 'as follower'.
- CDC HPD Source:** Contains two input fields. The first is labeled 'How many HDMI output ports support CDC\_HPDP?' with a value of 0. The second is labeled 'Which HDMI output ports support CDC\_HPDP?' with a value of 0.
- CDC HPD Sink:** Contains two input fields for the number of HDMI input ports (both set to 0) and several checkboxes for different EDID modes: 'Support CP\_EDID Mode', 'Support EDID Mode', 'CP\_EDID\_DISABLE\_ENABLE', '[EDID\_DISABLE] and [EDID\_ENABLE]', '[CP\_EDID\_DISABLE] and [CP\_EDID\_ENABLE]', and 'EDID\_DISABLE\_ENABLE'. All these checkboxes are currently unchecked.

Figure 59: The "CDC HPD Features Tab"

Figure 60: The "CDC HPD Features Tab" (Continued)

### 5.3.2.4 The "Console Tab"

The Console Tab is a multi-purpose, read-only console view (Figure 53). It can be used to view XML test cases, their last log files and the Summary Log file. During ATC tests, the user is prompted with the log output to the *CDC Summary Log* file. The console is exclusively loaded via the test case list's context menu. The label above the console points to its content.

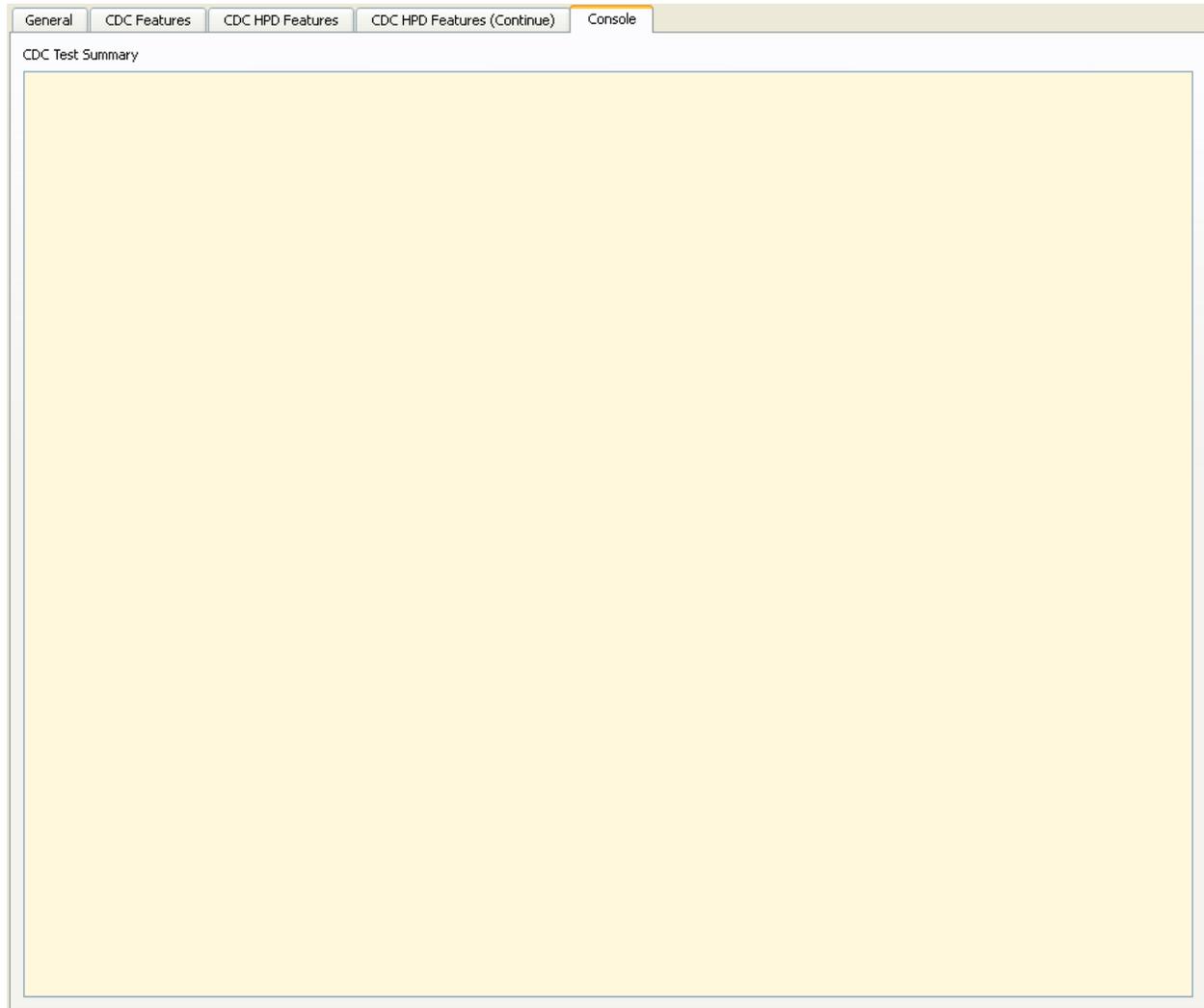


Figure 61: The "Console Tab"

## **5.4 Test Setup**

Prior to each test, the user is prompted with the necessary setup to execute a test unless it hasn't changed in comparison to the last test. The following setups are possible (the mentioned ports refer to HDMI ports of the CEC Explorer hardware unless otherwise stated):

- A source DUT shall be connected to the HDMI IN port. In this case the TV's HDMI IN port 1 has to be connected to the OUT1 port (Figure 63).
- A source DUT with an IN port under test shall be connected to the OUT1 port. In this case, a HDMI signal source has to be connected to the IN port (Figure 65).

- A sink DUT shall be connected to the OUT1 port. In this case, a HDMI signal source has to be connected to the IN port (Figure 54).
- A sink DUT with an OUT port under test shall be connected to the IN port. In this case, a TV's HDMI IN port 1 has to be connected to the OUT1 port (Figure 64).
- An oscilloscope (relevant for electrical test cases only) is connected to the OUT2 port (CEC signal to be measured) and the COM2 port (trigger signal for oscilloscope to capture the waveform from OUT2) (Figure 66).

For all connections, only use the cables provided with the Simplay test tool.

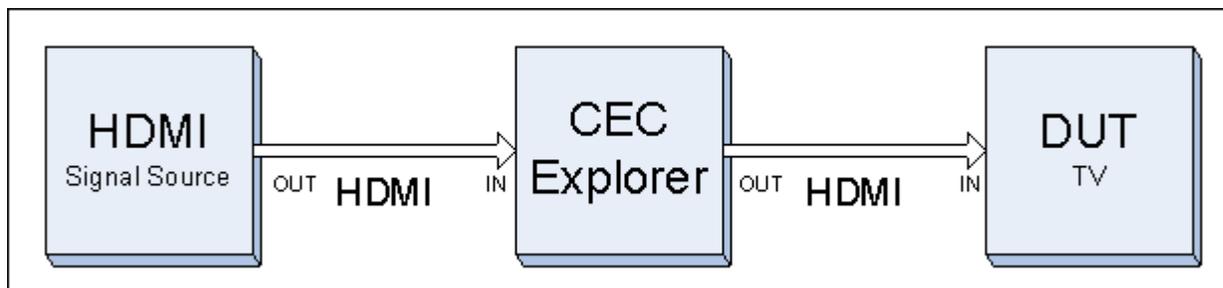


Figure 62: HDMI Signal Configuration

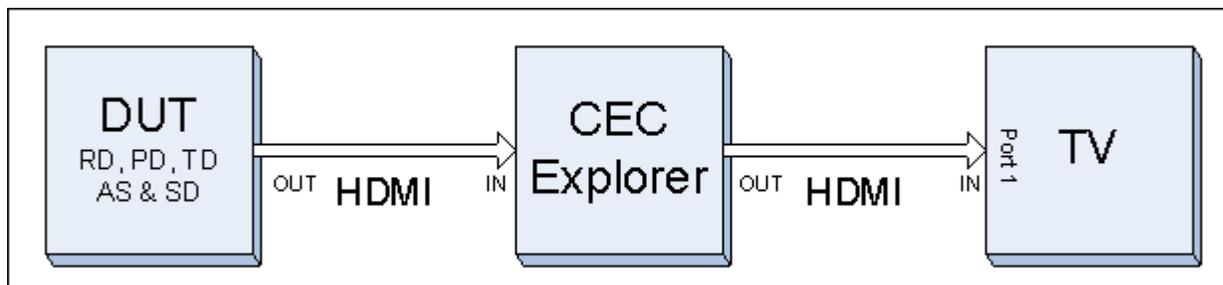


Figure 63: Source Device to TV Configuration

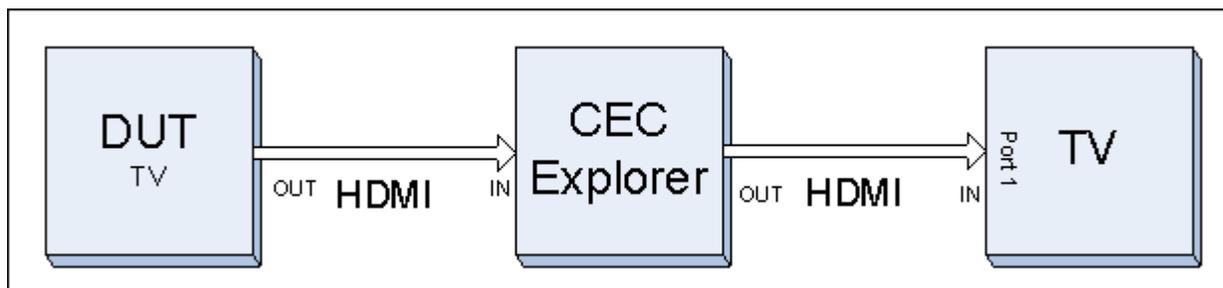


Figure 64: Configuration for a TV with HDMI Out Port

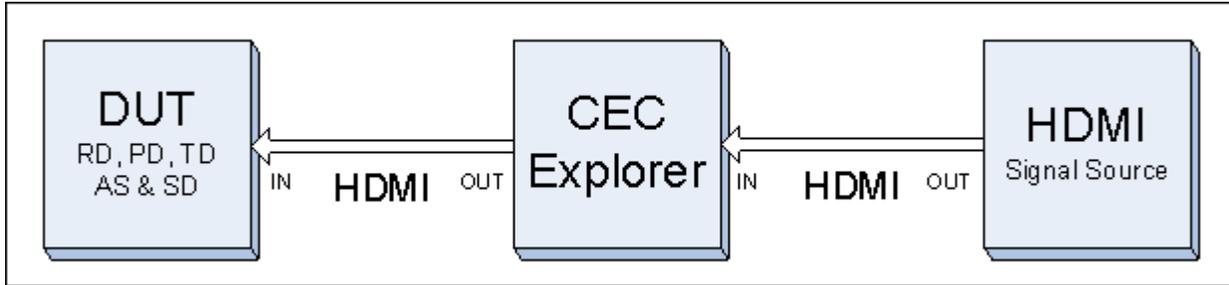


Figure 65: Configuration for a Source with HDMI In Port

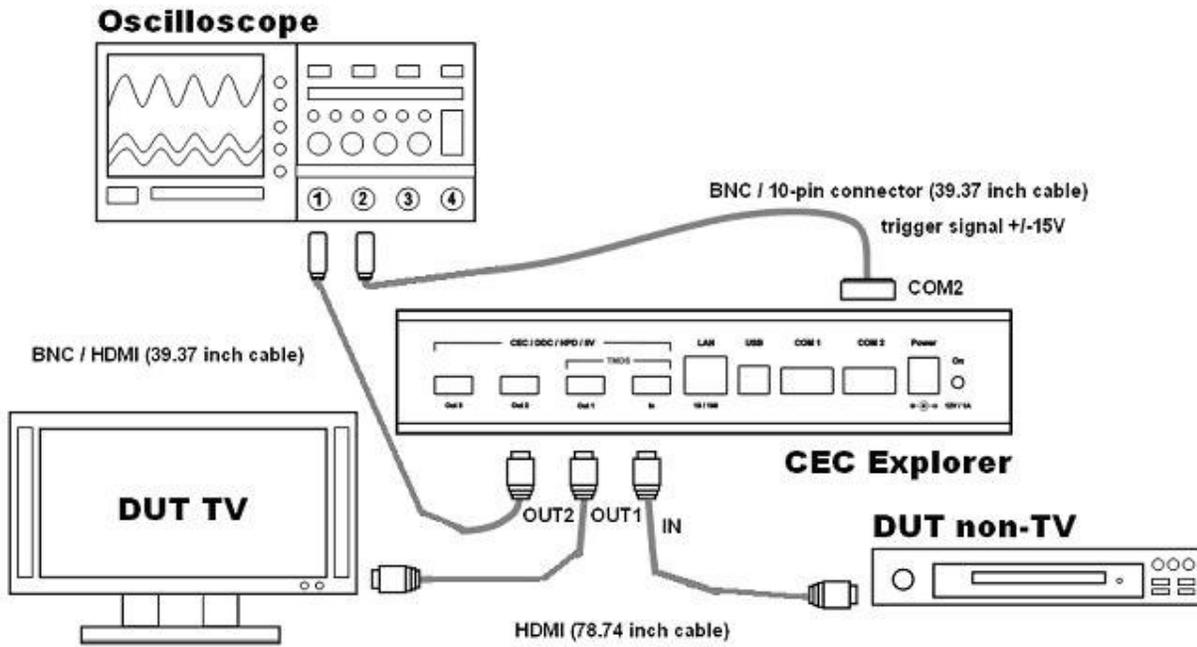


Figure 66: Basic Oscilloscope Configuration

### 5.5 Blue Boxes

The Original CDF enables the user to comment all the device's capabilities, e.g. to describe how to invoke the *One Touch Record* feature of a DUT.

The CEC Explorer handles such comments via *Blue Boxes*. These are little dialogs (with a blue background) enabling the user to enter a text. In many cases, the text is even mandatory since the test case itself requests the comment from the CDF dialog to prompt the tester with it.

Tool tip texts of GUI elements notify the user about mandatory comments to enter (Figure 67). If a comment was forgotten at test start, the CEC Explorer points to the item and opens the *Blue Box* automatically with a corresponding hint.

Blue Boxes are opened by the middle mouse button. Just click on the element you want to comment and enter the required text. Note that the Blue Box does not pop up if the item is not activated (checked).

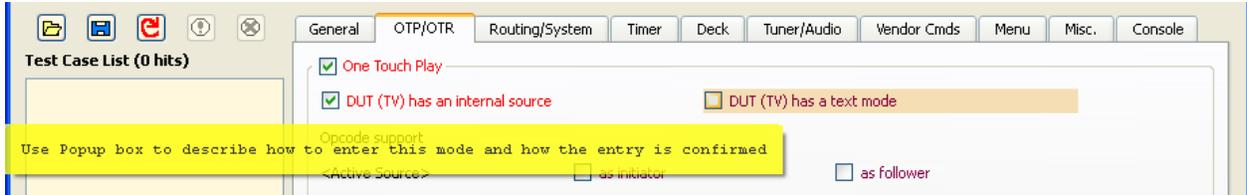


Figure 67: Tool tip text of a check box

The item's label can appear in three different colors:

- dark red when not activated ==>  DUT (TV) has a text mode
- red, when activated but not commented ==>  DUT (TV) has a text mode
- blue, when activated and commented ==>  DUT (TV) has a text mode

Figure 68 shows an example of the Blue Box. It opens directly next to the associated element. If you lack a middle mouse button, you might also press "Ctrl + Space" when the item has key focus. To close the box, either press "Ctrl + Space" or the middle mouse button again. One might also close the box via the green hook in the top right corner or by simply clicking on another item with the middle mouse button. Pressing Escape discards the message before closing the box. All comments are stored in the CDF file on saving the settings.

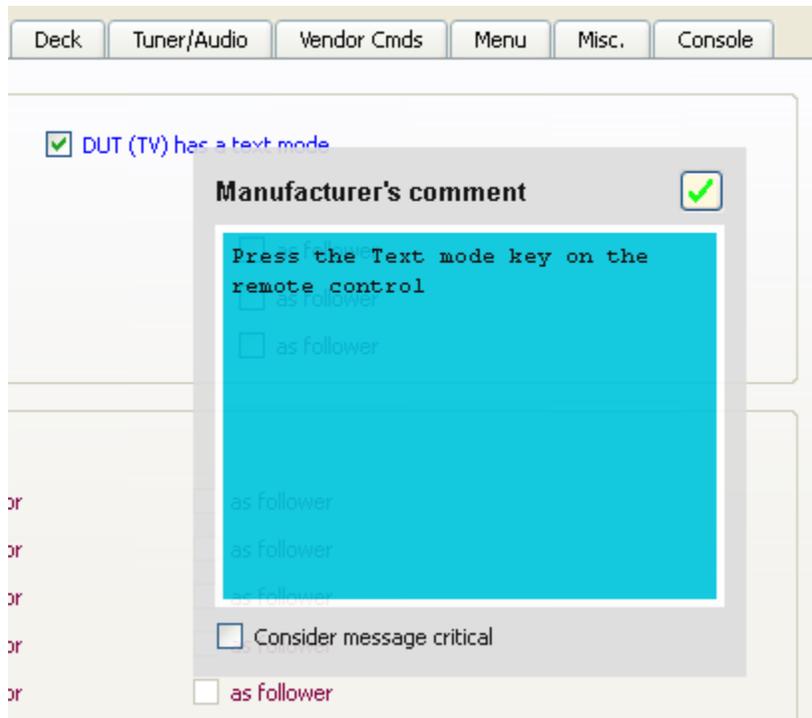


Figure 68: Blue Box of "text mode" Check Box

### 5.5.1 The Critical Flag

There is a check box at the bottom of each Blue Box. Activating this check box notifies the application that the user considers the message critical. This causes the CEC Explorer to prompt the tester with the comment prior to test start, and enables the user to take additional measures that might be necessary for passing the test.

If the check box is not activated, the tester does not see the comment during test run unless the test case itself recalls it. If the critical message box is opened, the tester has the option to skip the test case if the action mentioned in the comment is not possible. In that case the test case is not considered in the final statistics.

## 5.6 Test Execution

When all settings are done, and the desired test cases have been selected, the test session may start by pressing the corresponding tool button or the shortcut key (section 5.2.1.1). However, there are a number of preconditions that are checked in advance. If these are met, the GUI is disabled and the tester is automatically guided through the complete test.

Note that a potentially configured restricted capacity of the *Message History List* is switched off during ATC tests. Likewise, the test path settings of the *Configuration Dialog* are ignored since test paths are automatically set prior to each test case.

### 5.6.1 Preconditions

The following preconditions are checked before the test session starts:

- No test script must be open at time of start, i.e. the XML parser must not be busy.
- The hash value file must have been parsed successfully. All test cases are write-protected in terms of hash value checks. These are stored in file "version.xml" which must be located in the test case root folder. The hash value of each test case must fit the entry in the "version.xml" file.
- The *Log File Folder* must be defined and it must exist.
- The selected test case files and the test case root folder must exist.
- A number of CDF entries are checked for plausibility, validity and for mandatory comments.
- CDF entries are checked for mandatory opcodes according to the device type
- The former Summary Log File must have been parsed successfully.

If any of these checks fail, the test start is aborted and the user is notified to change or add the corresponding settings.

### 5.6.2 Auto Responses

During test execution, the CEC Explorer hardware emulates one or several device types (depending on the test case). As such, it automatically responds to potential CEC requests of the DUT. The responses are handled by the CEC Explorer application but can be partly configured as summarized and commented in Table 5.

CEC Request	CEC Response of Explorer hardware	Comment
<Get CEC Version>	<CEC Version> [Version 1.3a]	Response is directly addressed, optional, and must be configured (section 5.2.2.9).
<Get Menu Language>	<Set Menu Language> [English]	Response is broadcast and TV always responds. Non-TV devices can be configured to respond in the CDF dialog (section 5.2.2.9). The language can be configured for every device ('English' is default)
<Give Device Vendor Id>	<Device Vendor Id> [Simplay Vendor Id]	Response is broadcast. The vendor id is configurable (section 5.2.2.9). Default is the Simplay vendor id.
<Give Device Power Status>	<Report Power Status> [On]	Response is directly addressed
<Give OSD Name>	<Set OSD Name> [Name]	Response is directly addressed.

		<p>Each emulated device type with the corresponding logical address has its own name. These are:</p> <pre> TV      (0)  ==&gt;  TV RD 1   (1)  ==&gt;  Rec 1 RD 2   (2)  ==&gt;  Rec 2 TD 1   (3)  ==&gt;  Tun 1 PD 1   (4)  ==&gt;  Pbd 1 AS     (5)  ==&gt;  AudSy TD 2   (6)  ==&gt;  Tun 2 TD 3   (7)  ==&gt;  Tun 3 PD 2   (8)  ==&gt;  Pbd 2 RD 3   (9)  ==&gt;  Rec 3 TD 4   (a)  ==&gt;  Tun 4 PD 3   (b)  ==&gt;  Pbd 3 RS 1   (c)  ==&gt;  Res 1 RS 2   (d)  ==&gt;  Res 2 FU     (e)  ==&gt;  FreUs UR     (f)  ==&gt;  Unreg </pre>
<Give Physical Address>	<Report Physical Address> [Address]	<p>Response is broadcast. TV's address is always 0.0.0.0. Source devices respond with the assigned physical address of the DUT's EDID. The device type is according to the logical address</p>

**Table 5: Auto Responses during ATC tests**

## 6 Appendix

### 6.1 Support

In case of problems or questions please check at first the provided information on [www.simplaylabs.com](http://www.simplaylabs.com) (Manufacturers – Product Support) where you will find software updates, known issues, and other information.

If you need direct support please [go](#) to [http://www.simplaylabs.com/manufacturers/product\\_support.aspx](http://www.simplaylabs.com/manufacturers/product_support.aspx) and submit tool issues in our database system.

### 6.2 Terms and Conditions

#### 6.2.1 FreeRTOS

The Simplay CEC Explorer incorporates the FreeRTOS version 4.8.0 operating system as provided by [www.freertos.org](http://www.freertos.org) under the modified “GNU GENERAL PUBLIC LICENSE Version 2, June 1991”, for details see <http://www.freertos.org>. The license information is also available on the CD (see Licenses\Terms&Conditions\_FreeRTOS.pdf).

#### 6.2.2 STR91x Firmware Library

The Simplay CEC Explorer incorporates the STR91x Firmware Library as provided by STMicroelectronics, for details see <http://www.st.com/stonline/products/support/micro/files/um0233.zip>. The license information is also available on the CD (see Licenses\STR91x.txt).

#### 6.2.3 LWIP Ethernet Stack

The Simplay CEC Explorer incorporates the LWIP Ethernet Stack version 1.3.0, for details see <http://download.savannah.gnu.org/releases/lwip>

```
/*
 * Copyright (c) 2001, 2002 Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 * 3. The name of the author may not be used to endorse or promote products
 *    derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
 * SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
```

```

* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
* OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
* IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
* OF SUCH DAMAGE.
*
* This file is part of the lwIP TCP/IP stack.
*
* Author: Adam Dunkels <adam@sics.se>
*
*/

```

The license information is also available on the CD (see Licenses\Terms&Conditions\_LWIP.txt).

### **6.3 Abbreviations**

ACK	Acknowledge
ASCII	American Standard Code for Information Interchange
audio system	Audio System
ATC	Authorized Test Center
BNC	Bayonet Neill Concelman
CD	Compact Disk
CDF	Capabilities Declaration Form
CEC	Consumer Electronics Control
COM	Component Object Model
CTS	Compliance Test Specification
DDC	Display Data Channel
DHCP	Dynamic Host Configuration Protocol
DUT	Device Under Test
DVD	Digital Video Disc
EDID	Extended Display Identification Data
EOM	End Of Message
EU	European Union
FW	Firmware
GUI	Graphical User Interface
HD	High Definition
HDMI	High Definition Multimedia Interface

HPD	Hot Plug Detect
HTML	Hyper Text Markup Language
IF	Interface
IP	Internet Protocol
IR	Infrared
IT	Information Technology
LAN	Local Area Network
LED	Light Emitting Diode
LLC	Limited Liability Consortium
MAC	Media Access Control
MDI(X)	Medium Dependent Interface (crossover)
OS	Operating System
OSD	On Screen Display
PC	Personal Computer
PD	Playback Device
RD	Recording Device
RS	Reserved (for reserved device or reserved address)
RFC	Request for Comments
SW	Switch Device
TCP	Transmission Control Protocol
TMDS	Transition Minimized Differential Signaling
TD	Tuner Device
TV	Television
UDP	User Datagram Protocol
US	United States
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
WYSIWYS	What You See Is What You Send
XML	Extensible Markup Language

## **6.4 List of Figures**

Figure 1: Front Panel of CEC Explorer Hardware .....	6
Figure 2: Back Panel of CEC Explorer Hardware .....	6
Figure 3: Schematic presentation of CEC Explorer's Front Panel .....	7
Figure 4: Schematic presentation of CEC Explorer's Back View .....	8
Figure 5: Application Configuration for usage of AutoIP .....	11
Figure 6: Network Connections settings in the Control Panel.....	12
Figure 7: Properties of "Internet Protocol (TCP/IP)" .....	12
Figure 8: Application Configuration for usage of a fixed IP address .....	13
Figure 9: LAN Update Application, main dialog .....	16
Figure 10: LAN Update Application, file dialog for firmware file .....	17
Figure 11: LAN Update Application, configuration dialog.....	18
Figure 12: LAN Update Application, file dialog for feature bits.....	19
Figure 13: GUI of CEC Explorer application.....	21
Figure 14: Application menu.....	22
Figure 15: Individual context menu of Combo Box elements.....	23
Figure 16: Configuration Dialog (GUI Settings tab).....	23
Figure 17: Configuration Dialog (Color Settings tab) .....	24
Figure 18: Configuration Dialog (Comm Settings tab) .....	25
Figure 19: Configuration Dialog (CEC Settings tab) .....	26
Figure 20: Action Menu .....	28
Figure 21: Plug Dialog .....	29
Figure 22: Auto Communication Dialog.....	29
Figure 23: Help Box.....	30
Figure 24: Frame Editor.....	32
Figure 25: Test Scripting menu .....	35
Figure 26: Auto Comm menu .....	37
Figure 27: Help menu .....	39
Figure 28: About Dialog.....	39
Figure 29: The Tool Bar.....	40
Figure 30: Header and Opcode Data Group Box .....	40
Figure 31: Operand Data Group Box .....	42
Figure 32: Tool Tip Text pointing to expected data .....	42
Figure 33: Send Box, inactive (top) and active (bottom).....	43

Figure 34: Pulse Sequence Graph .....	45
Figure 35: Message History List Box.....	45
Figure 36: Operands Edit Field .....	48
Figure 37: HTML converted XML script file .....	51
Figure 38: <UserAction> Message Box.....	67
Figure 39: <UserInput> Message Box.....	67
Figure 40: <UserDecision> Message Box .....	68
Figure 41: CEC ATC Test Dialog. Control unit (left) and Tab Widget (right) .....	74
Figure 42: Context menu of test case list .....	75
Figure 43: Output of Summary Log File in a Browser .....	77
Figure 44: The "General Tab" .....	78
Figure 45: The "OTP/OTR Tab" .....	80
Figure 46: The "Routing/System Tab" .....	80
Figure 47: The "Timer Tab" .....	81
Figure 48: The "Deck Tab" .....	81
Figure 49: The "Tuner/Audio Tab".....	82
Figure 50: The "Vendor Cmds Tab" .....	83
Figure 51: The "Menu Tab" .....	83
Figure 52: The "Misc Tab" .....	84
Figure 53: The "Console Tab" .....	85
Figure 54: Test case presentation in web browser .....	86
Figure 55: CDC ATC Test Dialog. Control unit (left) and Tab Widget (right) .....	87
Figure 56: Context menu of test case list .....	88
Figure 57: The "General Tab" .....	91
Figure 58: The "CDC Features Tab" .....	92
Figure 59: The "CDC HPD Features Tab" .....	93
Figure 60: The "CDC HPD Features Tab" (Continued).....	94
Figure 61: The "Console Tab" .....	95
Figure 62: HDMI Signal Configuration.....	96
Figure 63: Source Device to TV Configuration.....	96
Figure 64: Configuration for a TV with HDMI Out Port.....	96
Figure 65: Configuration for a Source with HDMI In Port.....	97
Figure 66: Basic Oscilloscope Configuration.....	97
Figure 67: Tool tip text of a check box .....	98
Figure 68: Blue Box of "text mode" Check Box .....	99

## **6.5 List of Tables**

Table 1: LED status at the CEC Explorer's front panel while powering up .....	15
Table 2: LED status at the CEC Explorer's front panel while updating firmware .....	16
Table 3: Shortcut keys for the Frame Editor.....	34
Table 4: Possible icons in the Message History List .....	46
Table 5: Auto Responses during ATC tests .....	100

## **6.6 References**

- [1] HDMI Specification Version 1.3a
- [2] HDMI Compliance Test Specification, Version 1.3c
- [3] [http://en.wikipedia.org/wiki/Medium\\_dependent\\_interface](http://en.wikipedia.org/wiki/Medium_dependent_interface)