Boxee for Intel CE4100 Plumbing

1. Document History

0.1	06/15/10	• Initial revision
0.2	07/04/10	• Add AudioDAC to system • For wireless, get/set the authentication for SSID which are not broadcasted

2. Overview

This document describes the system services. This includes: init scripts, HAL, network configuration, disk mounting, etc. The description here is a recommendation which is accompanied by a reference implementation. It is the responsibility and ownership of the ODM to create and deliver a robust system.

2.1. Boxee HAL

Boxee HAL provides an abstraction to low level hardware and networking notifications and configuration. There are other tools available (such as hald and DeviceKit) but it is probably the best to have our own implementation to give us maximum flexibility and reduce the complexity (of using DBus for example). Also these tools rely on many software packages which will be a pain to cross compile.

Boxee HAL will run as a daemon under user root and expose its interface via TCP/IP to allow other non-root processes (such as Boxee) to get / set hardware information, notification and configuration.

The configuration of the Boxee HAL is persistent so that at the next reboot it will restore the system to it's last known configuration.

2.2. Boxee Launcher

The Boxee Launcher (BL) is a daemon that's started at boot time:

1. Started last at boot time, assuming that the Boxee HAL has already started and that all the system services are up.
2. Runs under the user “root”
3. Launches Boxee under the user “boxee”.
4. If boxee exits, it relaunches “boxee” (TBD: what happens if we want to reboot, we don't boxee to relaunch?)

3. System Specification

3.1. Root File System

Current distribution of the CE4100 is not based on a standard distribution and does not have any packaging tools, therefore there is no clear upgrade path of existing components. Due to this, for this platform, Boxee provides a single tarball which has all the required binaries to run boxee, with additional instructions how to configure the system manually. All the files provided by boxee are stored under /opt: the boxee application

resides under /opt/boxee and 3rd parties reside under /opt/local. It could be that in some in cases, the 3rd parties that are provided with boxee are newer versions of existing 3rd parties which are provided by Intel and reside under /usr.

3.2. Boot

The boot scripts are stored in /etc/init.d and their boot order is defined in /etc/rc3.d. Most of the items there should remain as is with the following exceptions:

1. Samba server needs to be started
2. Telnet should be disabled in production
3. Network startup script should be disabled and instead the Boxee Hal needs to be started. Boxee Hal will take care of all the network configuration setup and will then launch boxee.
4. Boxee startup script should be called at the end of the boot sequence.

4. Boxee HAL Specification

This section goes into much more details about Boxee HAL.

4.1. Operation

4.1.1. Network

Boxee HAL manages network connections. It uses NETLINK to get network interfaces add/removal events, it uses WPA supplicant to establish and maintain wireless communications and udhcpd to acquire/renew DHCP leases. When Boxee HAL is started, it brings back the network to its last configured state for wired or wireless networks.

A connection has the following run-time properties:

1. Type – Wired, Wireless
2. Configured – true/false. If a connetion is not configured, it cannot be enabled.
3. Enabled – true/false. Either Wired or Wireless can be enabled, but not both of them at the same time.
4. State. The following states exist:
    1. Disabled
    2. Interface Down
    3. Interface Up
    4. Associating (Wireless)
    5. Assigning IP Address
    6. Defining Route
    7. Running
5. Enabled – true/false. Either Wired or Wireless can be enabled, but not both of them at the same time.

4.1.2. Storage

Boxee HAL manages storage mounts. Udev processes device insertion/removal requests and calls a script that mounts them and notifies Boxee HAL about it.

4.1.3. Input

Boxee HAL listens on kernel input device events via NETLINK.

4.2. Objects

The protocol will be object oriented in the sense that messages are sent to specific object instance and results returned. All requests, responses and notification relate to a specific instance of an object. Some classes will be singletons so no object id will be required.

The following classes are supported:

Class	Description
ethernet	Ethernet interface in the system
wireless	Wireless interface in the system
clock	System clock
power	Power management
input	USB keyboard / mouse

storage	USB storage
host	Host information
led	LEDs
thermal	Thermals read and fan control
system	Retrieve general information about the system

4.3. Protocol

HTTP will be used as the transport layer over TCP port 570 for both request/response and for notifications. The body of the HTTP will be JSON.

4.3.1. Request

A request is sent to a specific object, calling a specific method. Parameters could have either a single value or multiple values.

```
GET /<class>.<method>?p1=v1&p2=v2... HTTP/1.1
Content-Length: 0
User-Agent: <Application>/<version>
```

Note: the parameters must be URL encoded

4.3.2. Response

A response is sent to a specific request. The result will have an attached response code. Response code 0 means success. Parameters could have either a single value or multiple values.

Example of a successful response:

```
HTTP/1.1 200 OK
Content-Length: <length-in-bytes>
Server: BOXEE HAL/<boxee-hal-version>

{ class : "name", instance: "id", method: "methodId", resultCode: "0",
params: { id1: "value", id2: "value" } }
```

For failed response there will be an error message associated. There could be multiple messages for different languages:

```
HTTP/1.1 400 BAD REQUEST
Content-Length: <length-in-bytes>
Server: BOXEE HAL/<boxee-hal-version>

{ class : "name", instance: "id", method: "methodId", resultCode: "cod",
resultLanguage: "en_US", resultMessage: "a message" }
```

4.3.3. Notification

A notification is sent by the server asynchronously on a separate socket. Parameters could have either a single value or multiple values. The transport layer will HTTP with chunked transfer.

To open notification channel:

```
GET /notifications HTTP/1.1
User-Agent: <Application>/<version>
```

The server will response:

```
HTTP/1.1 200 OK
Server: BOXEE HAL/<boxee-hal-version>
Transfer-Encoding: chunked
```

```
<length of chunk in bytes (hex)>
<chunk>

<length of next chunk in bytes (hex)>
<chunk>
```
For example:

```
25
This is the data in the first chunk

1C
and this is the second one

0
```

Each chunk will be a json.

4.4. Classes Reference

4.4.1. Common

1. All requests and responses will include in their input and output parameters: { class : string, method : string }
2. All notifications will include in their in their parameters: { class : string, notification : string }
3. All requests have their input parameters specified as JSON. However, since we are using HTTP GET, those parameters should not be passed as JSON but rather as regular encoded GET parameters.

4.4.2. Specifics

Class: ethernet

Method:	GetConfig
Input:	{ instance : number }
Output:	For disabled: { instance : number, config : "none" } For dynamic { instance : number, config : "dhcp" } For static: { instance : number, config : "static", ipAddress: ipaddress, netmask : ipaddress, gateway : ipaddress, dns : ipaddress }

Method:	SetConfig
Input:	To disable: { instance : number, config : "none" } For dhcp: { instance : number, config : "dhcp" } For static: { instance : number, config : "static", ipAddress: ipaddress, netmask : ipaddress, gateway : ipaddress, dns : ipaddress }
Output:	None

Method:	GetInfo
Input:	{ instance : number }
Output:	{ instance : number, config : “none” | "dhcp" | "static", link-up: boolean, running: boolean, ipAddress: ipaddress, netmask : ipaddress, gateway :

	ipaddress, dns : ipaddress, macAddress: string }

Notification:	LinkUp / LinkDown
Description:	Sent whenever the link state of the ethernet adapter has changed. This happens when the cable is connected / disconnected.
Parameters:	{ instance : number, link-up: boolean }

Notification:	AddressChanged
Description:	Sent whenever the IP address have changed in the following cases: 1. After LinkUp for static addresses 2. After new DHCP lease was acquired
Parameters:	Same as GetInfo

Class: wireless

Method:	GetConfig
Input:	None
Output:	For disabled: { config : "none" } For dhcp: { ssid : string, auth : “none” | “wep-passphrase” | “wep-key” | “wpa-psk” | “wpa2-psk”, config : "dhcp" } For static: { ssid : string, auth : “none” | “wep-passphrase” | “wep-key” | “wpa-psk” | “wpa2-psk”, config : "static", ipAddress: ipaddress, netmask : ipaddress, gateway : ipaddress, dns : ipaddress }

Method:	SetConfig
Input:	To disable: { config : "none" } For dhcp: { ssid : string, auth : “dontcare” | “none” | “wep-open” | “wep-shared” | “wpa-psk” | “wpa2-psk” , key : string, config : "dhcp" } For static: { ssid : string, auth : “dontcare” | “none” | “wep-passphrase” | “wep-key” | “wpa-psk” | “wpa2-psk”, password : string, config : "static", ipAddress: ipaddress, netmask : ipaddress, gateway : ipaddress, dns : ipaddress }
Output:	None

Method:	GetInfo
Input:	None
Output:	If disabled: { config : "none" }

	If not connected: { ssid : string, associated: "false" , macAddress: string, config : "dhcp" | "static" } If connected: { ssid : string, associated: "true", config : "dhcp" | "static", running: boolean, ipAddress: ipaddress, netmask : ipaddress, gateway : ipaddress, dns : ipaddress, macAddress: string }

Method:	Scan
Input:	None
Output:	{ [{ ssid : string, secure: boolean, signal: number(0-100) }] }

Notification:	LinkUp / LinkDown
Description:	Sent whenever the link state of the wire;ess adapter has changed. This happens when the adapter got connected / disconnected from the access point.
Parameters:	None

Notification:	AddressChanged
Description:	Sent whenever the IP address have changed in the following cases: 1. After LinkUp for static addresses 2. After new DHCP lease was acquired
Parameters:	Same as GetInfo

Class: clock

Method:	SetTimezone
Input:	{ timezone : string }
Output:	None

Method:	GetTimezone
Input:	None
Output:	{ timezone : string }

Class: power

Method:	Standby
Output:	None
Description:	In case the kernel does not support suspend / resume, and in fact there is only implementation for "soft" suspend / resume, the client may request resume due to an external event. In this case the "Resume" notification will not be sent from the server.

Method:	Shutdown
Input:	None
Output:	None

Method:	Reboot
Input:	None
Output:	None

Notification	Resume
Description:	Occurs when the machine returns from standby back to running state. This is only used in case the kernel supports suspend / resume and the server needs to notify the client about this state change.

Class: input

Method:	GetAllDevices
Input:	None
Output:	{ instance: number, label : string, path : string }, ...

Notification:	Connect / Disconnect
Description:	Sent when a keyboard/mouse is connected / disconnected
Parameters:	{ instance: number, label : string, path : string }

Class: storage

Method:	GetAllDevices
Input:	None
Output:	{ instance: number, label : string, path : string }, ...

Method:	EnableSambaShares
Input:	{ workgroup : string, username : string (optional), password : string (optional) }
Output	None

Method:	DisableSambaShares
Input:	None
Output:	None

Method:	Eject
Input:	{ path : string }
Output:	None

Notification:	Mount / Unmount
Description:	Sent whenever a storage device is mounted / unmounted.
Parameters:	{ path : string }

Class: host

Method:	SetHostName
Input:	{ hostname : string }
Output:	None
Description:	Sets the host name. This is what people will see when browsing SMB or

	when searching for the host with the iPhone remote.

Method:	GetHostName
Input:	None
Output:	{ hostname : string }
Description:	Gets the host name.

Class: led

Method:	SetBrightness
Input:	{ instance : number, startBrightness : number(0-100), endBrightness : number(0-100) }
Output:	None
Description:	Sets the brightness level of an LED. 0 - is off, 100 - is full brightness. The brightness will change from the start value to the end value gradually. The instance number is the actual LED number in the system.

Class: thermals

Method:	GetCPUTemperature
Input:	None
Output:	{ temperature : number }
Description:	Returns the CPU temperature in celsius

Method:	GetFanSpeed
Input:	None
Output:	{ speed : number(0-100) }
Description:	Returns the CPU fan speed.

Class: system

Method:	GetHardwareInfo
Input:	None
Output:	{ vendor : string, model : string, revision : string, serialNumber : string, uniqueId : string}
Description:	Returns general information regarding the device hardware

Method:	GetSoftwareInfo
Input:	None
Output:	{ version : string, regionSKU : string, language : string }
Description:	Returns general information regarding the device software

Method:	RequestUpgrade
Input:	None
Output:	None
Description:	This is called whenever an upgrade process is requested. This is called after the new image has been downloaded.

Method:	SetAudioDACState
Input:	{ mute : boolean }
Output:	None
Description:	Mutes or unmutes the audio DAC

Class: vpn

Method:	Config
Input:	{ host : string, username : string, password : string, encryption : string }
Output:	none
Description:	Creates vpn configuration but doesn't actually setup one. Encryption is either “force” or “auto” and it's in respect to MPPE encryption.

Method:	GetConfig
Input:	none
Output:	{ host : string, username : string, password : string, encryption : string }
Description:	Returns vpn configuration. Encryption is either “force” or “auto” and it's in respect to MPPE encryption.

Method:	Dial
Input:	none
Output:	none
Description:	Setup a vpn using the parameters given in the vpn.Config

Method:	Hangup
Input:	none
Output:	none
Description:	Close the vpn tunnel.

Method:	GetInfo
Input:	none
Output:	{ host : string, username : string, password : string, encryption : string, vpnStatus : string }
Description:	Returns vpn information. Encryption is either “force” or “auto” and it's in respect to MPPE encryption. vpnStatus is either - “disabled”,“connected” or “disoconnected”. “disconnected” - vpn was enabled but wasn't able to setup a vpn.

Notification:	Status
Description:	Sent whenever the vpn state changes from connected to disconnected and vice versa. If vpn is connected 'status' is 'true' and 'false' otherwise.
Parameters:	{ status : boolean}