



Connectivity

# 1 Contents

2	Network management . . . . .	2
3	Switching to a different connection . . . . .	3
4	Application requirements and expressing preferences . . . . .	4
5	Binding to the appropriate network interface . . . . .	5
6	Connections policies and store applications manifests . . . . .	5
7	Network-related events and how applications need to behave . . . . .	6
8	Connectivity policies on Android . . . . .	7
9	Real-time communications . . . . .	8
10	Traditional Telephony (GSM, SMS) . . . . .	8
11	Tethering from mobile devices . . . . .	9
12	Counting bytes and getting information about bandwidth . . . . .	10
13	Providing Internet connectivity to other devices . . . . .	10
14	A web server to provide information . . . . .	11
15	Bluetooth support . . . . .	11
16	Bluetooth 3.0, 4.0, High Speed, L2CAP, SDP, RFCOMM . . . . .	11
17	SPP . . . . .	12
18	PAN and DUN . . . . .	12
19	GOEP, OBEX . . . . .	12
20	PBAP, MAP, OPP, SYNCH . . . . .	12
21	AVRCP, A2DP, VDP . . . . .	13
22	HFP . . . . .	13
23	HSP . . . . .	14
24	GSM 07.07 AT-commands . . . . .	15
25	GNSS . . . . .	15
26	Global Positioning System (GPS) . . . . .	15
27	Media Downloading . . . . .	16
28	UPnP . . . . .	17

29 Network management is the task of managing the access to networks. In other  
30 words, deciding when and through which means to connect to the internet. In  
31 an IVI context this task is affected by several conflicting requirements. Connec-  
32 tivity may be spotty at times, with tunnels, high speed causing WiFi networks  
33 to come and go quickly, low cell phone signal strength, and so on. On the other  
34 hand, potentially good connectivity while parked, since the user might have high  
35 quality WiFi at the office and at home. Network Management will be discussed  
36 in [Network management](#).

37 Online and cellular-based real-time communications, including chatting, voice  
38 calls, VoIP and video calls are covered in [Real-time communications](#).

39 It is very common these days to have people carrying one or more smart devices  
40 with them. People want those smart devices to connect to their in-vehicle  
41 infotainment system for playing audio, importing contacts and also use or share  
42 Internet connections. This is discussed in [Tethering from mobile devices](#).

43 The main medium used for inter-device communication, Bluetooth, and its var-

44 ious profiles are discussed in [Bluetooth support](#). A brief discussion of using  
45 GPS to enhance network management and about the GeoClue framework are  
46 the subject of [Global Positioning System \(GPS\)](#).

47 Contacts management is covered by a separate document. Integration with  
48 other devices by means other than Bluetooth and USB mass-storage, such as  
49 reading songs off of an iPod is the topic discussed in [Media downloading](#).

## 50 **Network management**

51 The main goals of network management in an IVI system are to make sure the  
52 best connection is being used at all times while providing enough information  
53 to applications so that they can apply reasonable policies. For example, the  
54 IVI system should be able to fall-back to a metered 3G connection when an  
55 active WiFi connection is lost (because, say, the user drives their car out of  
56 their garage). In addition, big downloads should be paused in such a case; these  
57 would only be resumed when on an unmetered connection, to avoid significant  
58 charges on the user's phone bill.

59 [ConnMan](#)<sup>1</sup> is the central piece of the network management system being consid-  
60 ered for Apertis. It is focused on mobile use cases, provides good flexibility and  
61 features that allow implementation of the use cases mentioned above. [oFono](#)<sup>2</sup> is  
62 the de-facto standard when it comes to cellular connections and related features,  
63 and it is able to work in cooperation with ConnMan.

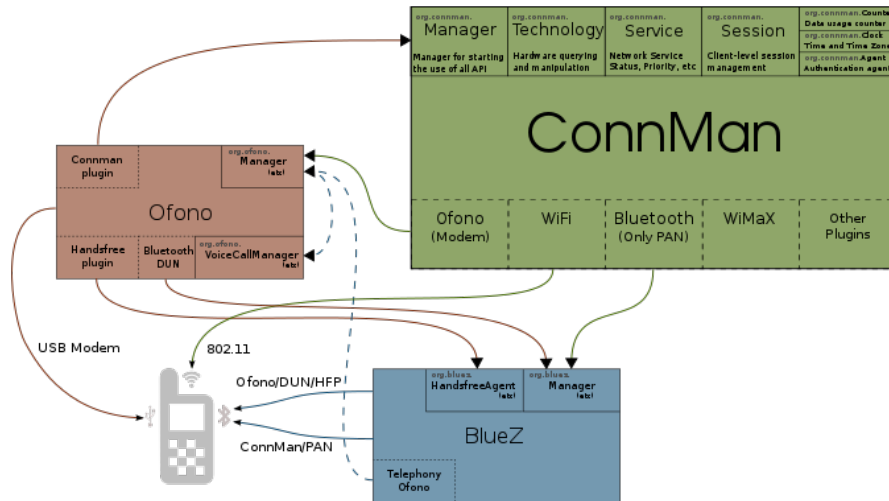
64 To complete the functionality expected from a modern network management  
65 framework, Bluetooth integration is also important. [BlueZ](#)<sup>3</sup> is used to provide  
66 that integration, allowing ConnMan to use Bluetooth devices to go online. Il-  
67 lustration has a schematic view of the interactions among these frameworks.

---

<sup>1</sup><https://git.kernel.org/pub/scm/network/connman/connman.git/>

<sup>2</sup><http://oFono.org/>

<sup>3</sup><http://www.bluez.org/>



68

## 69 Switching to a different connection

70 There are very specific requirements about how and when the system should  
 71 switch from an active Internet connection to a newly-available one. ConnMan  
 72 developers are working in a infrastructure for policy plugins. Collabora believes  
 73 this new infrastructure can be used to implement the policies required to satisfy  
 74 the requirements.

75 The two main concerns are that the system should not switch to a WiFi network  
 76 that just became available, since that may just be an open network in a café  
 77 the car is passing by, but that it should also take advantage of good known  
 78 connections when they are available.

79 ConnMan provides several facilities to gather information useful for such policy  
 80 decisions. For instance, a network that has been manually selected by the user  
 81 will have the Favorite property set to true.

82 That can be used to implement a policy of never automatically migrating to  
 83 open WiFi networks that are detected, unless it has been successfully used  
 84 before. This would guarantee that the system is able to switch automatically to  
 85 relevant networks without running into the problem of trying to associate with  
 86 the every open network it passes by.

87 Because connections may be lost or replaced by a better connection by Conn-  
 88 Man, applications need to be aware that their session may go away at any time,  
 89 and be able to recover from that. When a connection change happens, Conn-  
 90 Man will emit a D-Bus signal, and applications may need to drop connections  
 91 they have started and restart whatever they were doing.

92 A concrete example would be the email application that is connected to an  
 93 IMAPx server; when a connection change happens, the application gets notified

94 the connection it was using has gone away, so it drops all connections it had with  
95 the IMAPx server. If the new connection satisfies the requirements specified by  
96 the email client on its ConnMan session, it gets a “now online” notification and  
97 reconnects to the server.

## 98 **Application requirements and expressing preferences**

99 One very important characteristic of Apertis is that its Internet connectivity  
100 will vary a lot – going from a high speed WiFi network to a slow, unreliable,  
101 metered GSM connection and back is a common scenario, as discussed in the  
102 previous section. It may also be required that a particular type of connection  
103 be established to accommodate the needs of some applications.

104 ConnMan has a feature called Sessions. What that feature provides is a way for  
105 applications to tell ConnMan what they expect from an Internet connection,  
106 and get from ConnMan a network connection status that is relative to those  
107 requirements.

108 For instance, an application that downloads podcasts may have a policy that  
109 it would only perform the downloads when on WiFi. This application would  
110 create a session with ConnMan, and specify settings that ConnMan uses to  
111 decide whether that session is to be considered online or not.

112 The main settings are the **AllowedBearers** and **ConnectionType**. The first  
113 of these specifies which kinds of connections are allowed for the type of traffic  
114 this application intends to do. It is a simple list that specifies the preferred  
115 connection methods, such as, [**cellular, wifi, bluetooth**], which would specify  
116 a preference of cellular connection over both WiFi and Bluetooth .

117 A special “\*” bearer can be used to specify all bearers, which makes it easy  
118 to specify preference for one over all others, which will be treated as equiva-  
119 lent. When one of the connections allowed for an application comes online, the  
120 sessions is declared to be online. When a change happens on a Session setting  
121 ConnMan updates the application with the new values for the changed settings.

122 The second, **ConnectionType**, is used by the application to tell ConnMan if  
123 a connection wants to be online or if local connectivity is enough. Local connec-  
124 tivity means only connectivity in the internal network is needed, for example,  
125 an application may want to exchange data with other devices inside the same  
126 network. There is not much use for this setting in Apertis.

127 An application can have more than one ConnMan session at the same time,  
128 allowing applications to specify multiple policies and preferences, and perform  
129 work according to what is actually available. In addition to these three settings  
130 discussed here, ConnMan also provides several settings that can be used to  
131 customize how both sessions and the system deal with [networking](#)<sup>4</sup>.

---

<sup>4</sup><https://git.kernel.org/pub/scm/network/connman/connman.git/tree/doc/session-api.txt?id=HEAD>

132 Note that the Session API is still in a experimental state and its implementation  
133 and API are changing rapidly. This means both that it cannot be considered  
134 a stable part of the API supported by Apertis and that very few existing appli-  
135 cations use its current form. This should not be a problem for Apertis since  
136 applications are not intended to use ConnMan directly, so a wrapper API can  
137 be specified for the SDK.

### 138 **Binding to the appropriate network interface**

139 ConnMan allows multiple connections to exist at the same time. This might be  
140 useful for various reasons but it also brings some complications with it. First  
141 of all, if an application wants to use a specific connection it needs to explicitly  
142 bind its network usage to the desired network interface.

143 However, binding to a specific interface requires the `NET_RAW` capability<sup>5</sup>  
144 that is not something that regular applications should be allowed to have. A  
145 possible solution would be to also delegate this binding to special application  
146 that has the privileges to do such binding. The viability of such a solution needs  
147 to be properly investigated during the initial development of the feature.

148 Also, keeping in mind the desire to take complexity and control away from  
149 applications it seems desirable to abstract this complexity away to the SDK. The  
150 SDK can provide APIs that wrap ConnMan functionality and handle binding  
151 for the application. This means more Apertis-specific code, however, meaning  
152 less code reuse for existing applications.

### 153 **Connections policies and store applications manifests**

154 As discussed above, some control can be exerted on how ConnMan ranks and  
155 chooses connections by having applications (or a system service on their behalf)  
156 provide ConnMan with a list of their requirements using the Session APIs.

157 It has been made clear that applications from the store should be specifying  
158 their needs as much as possible through the manifest file that will be distributed  
159 along with applications on the app store. For network management this means  
160 specifying the allowed bearers, mainly.

161 The policy plugin mentioned above could use information provided by the appli-  
162 cation manager and application manifest files to decide on what the best policy  
163 to implement is. This would not require changing applications, but limits the  
164 flexibility the developer has to work with.

### 165 **Network-related events and how applications need to behave**

166 For applications that are written to work with ConnMan, two signals are essen-  
167 tial: connection is up, connection is down. When a connection comes up the

---

<sup>5</sup><http://git.kernel.org/?p=linux/kernel/git/next/linux-next.git;a=blob;f=net/core/sock.c;h=b374899aecb6ea3a8590ae9ccdbb3e60225561d4;hb=HEAD#1470>

168 application takes the appropriate steps to start whatever its functionality is.  
169 An IMAP mail client would at this point connect to the IMAP server, and look  
170 for new messages, a podcast downloader would look for new podcasts to start  
171 downloading or resume any downloads that had previously been started, and so  
172 on.

173 When the connection goes down – even if it’s just being switched from one  
174 connection method to another – any existing IP connections would not work  
175 any more, since the IP address will have changed. The application needs to  
176 close any connections. This means an IMAP mail client would close the sockets  
177 it had open with the IMAP server, a podcast downloader will close the HTTP  
178 or FTP connections, and so on. The connections can be re-established/resumed  
179 in case a new notification comes in that the system is online once more.

180 The following is a potential list of applications and events they will be interested  
181 in handling. As will be seen the events an application needs to handle are  
182 essentially limited to having a connection and not having a connection any  
183 more.

#### 184 **Email client**

- 185 • Connected event
  - 186 – Connect to IMAP server and check for new mail; note that IMAP
  - 187 connections are usually kept alive to receive notifications of new email
  - 188 from the server
  - 189 – Connect to the SMTP server to send any emails stored in the outgoing
  - 190 mail box
- 191 • Disconnected event
  - 192 – Drop IMAP connections
  - 193 – Cancel ongoing loading of email messages
  - 194 – Cancel ongoing sending of email messages, making sure they stay in
  - 195 the outgoing mail box

#### 196 **Media player**

- 197 • Connected event
  - 198 – In case multiple connections are supported, when a faster connection
  - 199 appears switch to it if needed.
  - 200 – If media is being played and previously disconnected, resume buffer-
  - 201 ing
- 202 • Disconnected event
  - 203 – Drop connections

#### 204 **Feed reader**

- 205 • Connected event
  - 206 – Begin download of the latest entries
  - 207 – If it's a fast connection, begin pre-caching of images and other big
  - 208 feed attachments
- 209 • Disconnected event
  - 210 – Drop connections

## 211 Continuing downloads

212 The HTTP protocol provides clients and servers with the ability of picking up a  
213 transfer from a given point, so that partially downloaded content does not need  
214 to be re-downloaded in full when a connection is dropped and reconnected.  
215 Details about how the protocol supports partial downloads can be found in  
216 [RFC2616](#)<sup>6</sup>.

217 In summary, when picking up a download the client should send a *Range* header  
218 specifying the bytes it wants to download. If the server supports continuing  
219 downloads and the range is acceptable a **206 Partial Content** response will  
220 be sent instead of the usual **200 OK** one. The client can then append the  
221 data to the partially downloaded file. If the server does not reply with a **206**  
222 response, then the file needs to be truncated, since the download will be starting  
223 from scratch.

## 224 Connectivity policies on Android

225 Connectivity policies on Android are a way more simpler. The Android system  
226 does not implement any per application configuration on how application should  
227 access the internet (wifi, 3g, etc.).

228 Also Android does not have any mechanism to notify the applications that the  
229 system is online, the applications just get a notification about a Network State  
230 Change and then they have to figure out by themselves if the system is online  
231 by requesting a “route to host”.

232 One of the few network configurations android has is to enable/disable Wi-Fi,  
233 mobile data and roaming allowance globally. Apart from that the user can also  
234 restrict background data usage for each applications in the Global Settings.

## 235 Real-time communications

236 The Apertis needs to be well-connected with Internet communication services  
237 such as Google Talk and Skype. [Telepathy](#)<sup>7</sup> provides a framework for enabling

---

<sup>6</sup><http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35>

<sup>7</sup><http://telepathy.freedesktop.org/wiki/>



238 messaging, video and audio calling through many of the existing services, and  
239 more can be supported by developing custom [connection managers](#)<sup>8</sup>.

240 Telepathy provides a D-Bus API which abstracts away the specific connection  
241 managers allowing a UI to seamlessly support various protocols while only hav-  
242 ing little or no protocol specific knowledge. The fact that Telepathy is imple-  
243 mented as several D-Bus services makes it possible to integrate messaging and  
244 voice features throughout the system.

245 A good example of these are the chat, dialler and address book applications  
246 present on the Nokia N900 and [N9 devices](#)<sup>9</sup>, which use Telepathy to support  
247 messaging, GSM and VoIP Calls using one single user interface, while at the  
248 same time providing ways for the user to choose which protocol they want for  
249 a given conversation.

250 Existing open-source connection managers support messaging through Jabber,  
251 GTalk, Facebook, Live Messenger, and more. Audio and video calls are also  
252 supported over Jabber, GTalk and SIP. See the [Telepathy site](#)<sup>10</sup> for more de-  
253 tails. Before deciding on shipping any of these, however, it's important to verify  
254 whether any legal issues may arise, mainly related to trademarks.

255 As discussed before, Telepathy is pluggable, enabling a mix of closed and open-  
256 source connection managers to coexist. This enables OEMs to enable as many  
257 third-party services as desired, requiring for the technical side at most the cre-  
258 ation of a new connection manager.

259 Collabora has been involved in consultancy projects to integrate various propri-  
260 etary backends in the past and is ready to do so again if it is decided to include  
261 support for more protocols. More details about this will be included in reference  
262 produced during the development phase by the documentation team.

## 263 **Traditional Telephony (GSM, SMS)**

264 The system shall support making and receiving calls and sending/receiving text  
265 messages through a paired cell phone. Telepathy has a backend on top of oFono  
266 to make calls and send text messages, which makes it possible to easily have a  
267 single, integrated user interface for both regular phone calls and messages along  
268 with those of online services.

269 Telepathy is focused on messaging and calling; as such, it does not include  
270 GSM/UMTS-specific functionality like signal-strength, data connections, and  
271 so on. Those features are accessible through oFono directly.

<sup>8</sup><http://telepathy.freedesktop.org/doc/book/sect.connection.connection-manager.html>

<sup>9</sup><http://techprolonged.com/index.php/2011/11/12/nokia-n9-a-complete-walk-through-meego-harmattan-software-and-user-interface-experience/#contacts-calling>

<sup>10</sup><https://telepathy.freedesktop.org/components/>

## 272 Tethering from mobile devices

273 There are six main ways to hop on to a mobile device's Internet connection:

- 274 • WiFi connection for devices that support mobile hotspot
- 275 • Using the DUN Bluetooth profile, through oFono
- 276 • Using the PAN Bluetooth profile
- 277 • Using Ethernet over USB
- 278 • Using 3G USB modem
- 279 • Using device-specific proprietary protocols

280 A mobile hotspot feature is becoming more common on mobile devices and, in  
281 a way, taking the place once occupied by Bluetooth for tethering. It is a good  
282 connection method because it is very simple to set up.

283 The PAN profile is supported by ConnMan through BlueZ and DUN also needs  
284 these two components to works plus a extra one, which is oFono. This difference  
285 is due to the fact that the DUN profiles behaves like a modem and thus needs  
286 oFono to handle it. All interactions between BlueZ and the two other daemons,  
287 ConnMan and oFono, are performed using BlueZ's D-Bus interface, and as such  
288 should not cause problems in case is planned to replace BlueZ with a proprietary  
289 counterpart that implements the same interfaces.

290 Note that connecting a phone to the car for tethering over Bluetooth is a process  
291 that requires user intervention: the user needs to first pair the two devices. In  
292 most systems that support connections over the cell phone network the user  
293 is also asked to choose the plan they acquired from their provider from a list,  
294 which will also need to be done for the Apertis; only then the connection will  
295 be made available through ConnMan.

296 Ethernet over USB is supported by Linux using the usbnet driver. Among  
297 device-specific protocols, Apple devices in particular are important. Linux in-  
298 cludes, since version 2.6.34, the [ipheth](#)<sup>11</sup> driver, which enables using Ethernet  
299 over the USB connection for Apple devices. In addition to the driver, pairing  
300 of the device by a user-space program is required. That pairing can be per-  
301 formed either by using the standalone tool provided at the project's web page  
302 or through the tools distributed by the **libimobiledevice** project, discussed in  
303 [Media downloading](#).

304 Collabora believes the three main components discussed here, BlueZ, oFono and  
305 ConnMan are capable of supporting tethering to most mobile devices. Provided  
306 appropriate user interfaces are implemented, ConnMan is able to provide all  
307 requirements regarding having several different phones in the car, including  
308 prioritizing and selecting which one should be used.

---

<sup>11</sup><https://github.com/torvalds/linux/blob/master/drivers/net/usb/ipheth.c>

## 309 **Counting bytes and getting information about bandwidth**

310 ConnMan provides an API called **Counters** that is used for tracking how much  
311 traffic has gone through a given connection, and can be used by the network  
312 connections management UI to inform the user about the quantity of data that  
313 has been transmitted. The counters are per-connection and are automatically  
314 updated with the information by ConnMan.

315 oFono also provides the **CallMeter** API for tracking how much conversation  
316 time is still available for a GSM phone, using data from the SIM. oFono is able  
317 to emit a warning when the limits are close to be reached.

318 For measuring bandwidth there is no convenient API at the moment. Clients can  
319 register a counter and specify an update interval, but ConnMan advises against  
320 using that API for tracking time. A more robust and correct implementation  
321 would be to have applications and services that care about that information  
322 track the RX/TX bytes and run a timer of their own to estimate how much  
323 bandwidth is being used at a given point in time.

324 It is important to note that tracking connection quality taking in account used  
325 bandwidth (and possible other variables as connection latency and available  
326 bandwidth) is not a easy task. Usually those variables doesn't give enough  
327 information to decide which connection has the better quality.

## 328 **Providing Internet connectivity to other devices**

329 In case the Apertis has Internet connectivity itself, it should be able to share it  
330 with other devices through either Bluetooth or WiFi.

331 ConnMan supports sharing the current Internet connection by using the WiFi  
332 interface in master mode or via Bluetooth PAN profile, becoming an access  
333 point that other devices can connect to. This is done by turning on tethering  
334 mode on WiFi or Bluetooth.

335 See the tethering properties at the bottom of [https://git.kernel.org/  
336 pub/scm/network/connman/connman.git/tree/doc/technology-  
337 api.txt?id=HEAD](https://git.kernel.org/pub/scm/network/connman/connman.git/tree/doc/technology-api.txt?id=HEAD)

338 As is the case with other features, this needs proper UI to be created to let the  
339 user turn the tethering on as well as specify the desired SSID and pass-phrase for  
340 WiFi, or to pair the Bluetooth devices. In order for this feature to be provided,  
341 the driver for the wireless chip used in the development board needs to support  
342 the master mode.

## 343 **A web server to provide information**

344 Apertis will have a web server running internally to provide information about  
345 the system and the car for access by smart phones. Collabora's working as-  
346 sumption is the server will be available to devices that connect to the WiFi

347 or Bluetooth hotspot provided by the system, regardless of whether it is being  
348 used to provide Internet connectivity to the devices or not. Libwebsockets can  
349 be used to write a solution for web server.

350 For users to access the web server, the manual of the device will contain a  
351 specific URI, and the DNS server provided by the device will resolve the name  
352 to the address the system has in the address space used by its DHCP.

## 353 **Bluetooth support**

354 The automotive space is by far the biggest user of Bluetooth for communications  
355 between the car and external devices, such as phones, tablets, notebooks, and  
356 so on. Plans have been stated to acquire a proprietary solution and supplement  
357 BlueZ in the apertis.

358 That solution sits in between applications that use BlueZ and the BlueZ daemon,  
359 and adapts requests to make sure specific device quirks are satisfied.

360 BlueZ is currently a fairly complete Bluetooth stack, and has support for all of  
361 the major [Bluetooth profiles](#)<sup>12</sup>. It's important to note, however, that applica-  
362 tions need to be written to use the Bluetooth infrastructure for connecting to  
363 mobile devices for music playing, remote control, file transfer, downloading of  
364 contacts and tethering.

365 For other Bluetooth profiles, the ones supported by BlueZ will be provided,  
366 development of support for more profiles is out of scope for this project. The  
367 list of Bluetooth profiles bellow was extracted from the Apertis Feature List  
368 document, and information is provided on the general level of support provided  
369 by BlueZ. For a more detailed list of existing support and gaps, Collabora would  
370 require a more detailed list of requirements.

371 When it comes to pairing support BlueZ supports both Legacy Pairing (PIN  
372 entry, many old devices only support this type of pairing) and Secure Simple  
373 Pairing (Numeric Comparison).

## 374 **Bluetooth 3.0, 4.0, High Speed, L2CAP, SDP, RFCOMM**

375 BlueZ currently supports the both Bluetooth 3.0 and 4.0 core specification.  
376 However, High Speed support through 802.11 AMP is still under development,  
377 so it is not currently supported. The Bluetooth core support provided by BlueZ  
378 includes Logical Link and Control Adaptation Protocol (**L2CAP**), Service Dis-  
379 covery Protocol (**SDP**) and Radio Frequency Communications (**RFCOMM**).

## 380 **SPP**

381 The Serial Port Profile (**SPP**) 1.1 is supported by BlueZ. The Serial Port Profile  
382 allows emulation of serial ports over a Bluetooth link.

---

<sup>12</sup><http://www.bluez.org/profiles/>

383 **PAN and DUN**

384 As discussed in sections [Network management](#) and [Tethering from mobile de-](#)  
385 [vices](#), BlueZ provides support for the Personal Area Networking (**PAN**) profile  
386 both in the NAP role (BlueZ acting as connection provider) or PANU role  
387 (BlueZ using a internet connection over Bluetooth).

388 There is support for the DUN profile. The Client role is implemented by an  
389 extra oFono's daemon and can be used to connect to devices providing internet  
390 connection.

391 There is also support for the server role of the Dial-up Networking (**DUN**)  
392 profiles, which can be used with oFono and ConnMan to provide Internet con-  
393 nection to an external device. However current implementation only supports  
394 sharing a DUN connection only if the device has a GPRS data connection ac-  
395 tive. Collabora thinks that lack the support for DUN server won't be a problem.  
396 DUN is rapidly being replaces by the PAN profile.

397 **GOEP, OBEX**

398 The Generic Object Exchange Profile (**GOEP**) and Object Exchange Proto-  
399 col (**OBEX**) are also supported by BlueZ. They enable file exchange between  
400 Bluetooth-capable devices and the Apertis system.

401 **PBAP, MAP, OPP, SYNCH**

402 The Phone Book Access Profile (**PBAP**) is supported by BlueZ, and can be used  
403 for downloading contacts from the external devices. There is also support for  
404 Object Push Profile (**OPP**), used for transferring vCards and vCalendars. Fi-  
405 nally, BlueZ also supports the 1.0 version of the Message Access Profile (**MAP**)  
406 version 1.0 in the client role\*\*, \*\* which can be used to download SMS messages  
407 and email from a phone onto the Apertis system, however support to upload  
408 delete and mark messages as read/unread is lacking at the moment.

409 Note that, although BlueZ includes support for these profiles, it's up to ap-  
410 plications on the system to make use of the framework to provide the actual  
411 features. For instance, the contacts application needs to talk to BlueZ to per-  
412 form the phone book download.

413 There is currently no support for the Synchronization Profile (**SYNCH**) profile.  
414 Collabora's current understanding is this does not pose a problem for the use  
415 cases planned, since only support for download of the contacts is required.

416 For more information on the specific use-cases, problems and proposed solutions  
417 regarding contacts in the Apertis system refer to the Contacts design document  
418 prepared by Collabora.

#### 419 **AVRCP, A2DP, VDP**

420 These profiles are used to communicate with devices that are able to reproduce  
421 multimedia content and/or control media playing remotely.

422 BlueZ supports the Audio/Video Remote Control Profile 1.0 (**AVRCP**) in the  
423 controller role, but it only supports the two commands at the moment: Volume  
424 Down and Volume Up. Collabora recommends the development of the missing  
425 features for AVRCP 1.0 version and also support for the 1.4 version, which is not  
426 yet supported by upstream. This would provide metadata information about  
427 the media and folder browsing support.

428 When acting as a controller, an application needs to provide the user with an  
429 interface for inputting commands. Collabora will provide sample code for an  
430 application acting on the controller role.

431 Also included is support for acting as sink for the Advanced Audio Distribu-  
432 tion Profile (**A2DP**) version 1.2, using PulseAudio to provide audio routing.  
433 When the device starts to send an A2DP stream to Apertis PulseAudio will  
434 automatically make it available as an output device.

435 PulseAudio has a module that can automatically redirect streams to new output  
436 devices. However, for systems with complex requirements for audio routing it's  
437 probably a better idea to have a system daemon or application managing that;  
438 the car system interface D-Bus daemon is one viable candidate.

439 The Video Distribution Profile (**VDP**) is not yet supported

#### 440 **HFP**

441 The Hands-Free Profile (**HFP**) version 1.6 is supported by BlueZ. Hands-free is  
442 the technology that allows making phone calls with voice commands, and having  
443 audio routed from the phone to a different device, such as the Apertis system  
444 which can then play it to the car speakers, for instance. The BlueZ framework,  
445 along with oFono, can be used to add hands-free support to the system. Wide  
446 Band Speech – high quality audio for calls, though, is not yet supported by  
447 BlueZ.

448 After a SIM-enabled device in Audio Gateway mode has been paired with BlueZ,  
449 PulseAudio will be able to use it as source and sink through its Bluetooth module  
450 and route streams from the car's microphone to the phone and the audio from  
451 the call to the car's speakers. In this case BlueZ acts as in the Hands-free role.

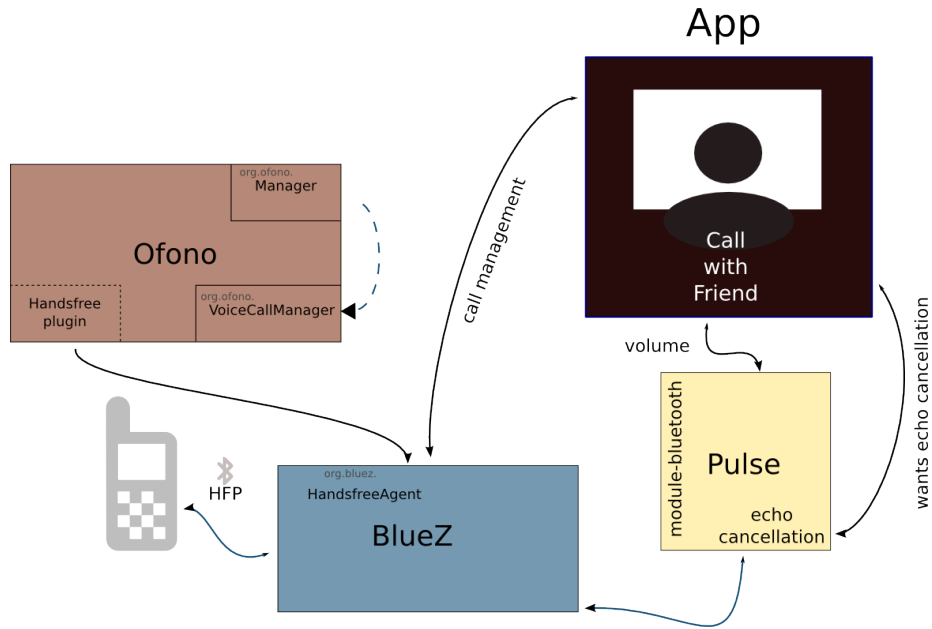
452 The application which handles the calls can use PulseAudio APIs to control the  
453 volume of the source and sink streams, and should set the **filter.want** property  
454 of the [PulseAudio streams](#)<sup>13</sup> to let PulseAudio know echo cancellation should  
455 be used.

---

<sup>13</sup>[http://freedesktop.org/software/pulseaudio/doxygen/proplist\\_8h.html#a87c586045175fa05e28e6ee1cbaac4de](http://freedesktop.org/software/pulseaudio/doxygen/proplist_8h.html#a87c586045175fa05e28e6ee1cbaac4de)

456 This will cause PulseAudio to automatically load the echo cancellation module.  
 457 The echo cancellation module can also contain a noise cancellation sub-module.  
 458 PulseAudio ships with an Open Source sub-module based on speex for echo  
 459 cancellation, but it can be replaced by custom or proprietary modules if required,  
 460 which was the course chosen by Nokia for its phones, for instance. The same goes  
 461 for the noise cancellation sub-module, it can be easily replaced by an proprietary  
 462 noise cancellation solution just by rewriting the sub-module.

463 The diagram in Illustration shows how the various pieces of such a set-up are  
 464 related.



465

## 466 HSP

467 Currently BlueZ has no support for the Headset Profile (**HSP**). Collabora rec-  
 468 ommends it be supported, but that would require development of the feature.  
 469 VoIP applications rely on HSP to operate calls over Bluetooth, It is also im-  
 470 portant to mention that older phones only support the HSP profile for phone  
 471 calls.

## 472 GSM 07.07 AT-commands

473 oFono has support for most of the GSM 07.07 AT command set. It is through  
 474 the AT command set that we control the phone in the HFP profile.

## 475 GNSS

476 The Global Navigation Satellite System Profile is currently not supported by  
477 BlueZ, nonetheless adding support would be simple in the BlueZ side. For this  
478 profile, BlueZ would only provide the Bluetooth connection handling, all the  
479 navigation specific data would be passed to the GPS specific application to  
480 handle it.

## 481 Global Positioning System (GPS)

482 The Apertis platform provided by Collabora will include the GeoClue geoloca-  
483 tion framework. [GeoClue](#)<sup>14</sup> provides a D-Bus service that can be queried to  
484 establish the current location of the system with configurable accuracy. Geo-  
485 Clue is able to use the GPS from the system to provide very accurate location  
486 information.

487 This technology will be used, for instance, to power the existing GeoLocation  
488 implementation of the WebKit-Clutter library. The service can be made avail-  
489 able for use by store apps, potentially including selective restrictions on the  
490 accuracy each app can query. This should be discussed and specified during the  
491 development phase.

492 The connectivity considerations document discusses using GPS data for predict-  
493 ing connectivity conditions, and pre-emptively switching to a different connec-  
494 tion before entering an area with bad coverage, for instance. This seems to be  
495 a risky strategy unless very up-to-date and very extensive data are accessible  
496 to the system at all times. In any case, the GeoClue framework could serve the  
497 purpose of providing the location information from the GPS.

498 One additional advantage of using GeoClue is it supports using different  
499 providers for its information like cell towers through oFono-based gsmloc, WiFi  
500 networks through integration with ConnMan, IP addresses through HostIP,  
501 and so on.

502 Note that HostIP is essentially useless for mobile use cases, since it tries to use  
503 the IP address as an indication of the location, but that is not very accurate in  
504 general and for mobile specifically

505       A somewhat outdated list: [https://gitlab.freedesktop.org/geoclue/  
506       geoclue/-/wikis/home](https://gitlab.freedesktop.org/geoclue/geoclue/-/wikis/home)

507 Those could be useful to provide location information with coarse accuracy for  
508 applications such as the web browser with no need for turning the GPS on or  
509 for systems with no GPS hardware. If only GPS matters, and tighter control is  
510 required, Collabora can support using the GPS service directly through [gpsd](#)<sup>15</sup>  
511 or [gypsy](#)<sup>16</sup>.

---

<sup>14</sup><http://www.freedesktop.org/wiki/Software/GeoClue>

<sup>15</sup><https://savannah.nongnu.org/projects/gpsd>

<sup>16</sup><http://gypsy.freedesktop.org/>



512 If different accuracy levels want to be defined that the store application can use,  
513 different permissions for GPS access can be created representing the different  
514 levels of accuracy. These permissions would be specified in the application's  
515 manifest and prompted to the user at the moment the user authorizes the in-  
516 stallation of an application. Refer to the Applications design for more details  
517 on this.

## 518 **Media Downloading**

519 This chapter discusses how communication with various devices is made to pro-  
520 vide the Apertis system with ways of downloading media from them. For more  
521 detailed information on use cases, requirements, problems and solutions refer  
522 to the Media Management design document (sometimes called the Media and  
523 Indexing design) prepared by Collabora.

524 In general media will be brought into the system through USB sticks, mobile  
525 devices and online sources. USB sticks are mounted using the USB mass-storage  
526 support. Most USB sticks use the VFAT file system, which is, unfortunately,  
527 patent-encumbered and has been used in the past by Microsoft to promote  
528 law suites against companies shipping devices that support the file system, see  
529 <http://www.groklaw.net/article.php?story=20090401152339514>.

530 When considering communication with specific devices the ones that stand out  
531 are the Apple devices, which are both very well-known and have widespread  
532 usage. This document discusses the Open Source tools that are currently the  
533 state of the art for communicating with Apple devices but does not specifically  
534 recommend their usage.

535 The [libimobiledevice](#)<sup>17</sup> suite is the state of the art on Open Source libraries and  
536 tools for accessing Apple devices. It implements the protocols used for com-  
537 munication with Apple's iPhone, iPad, iPod Touch and TV products, covering  
538 almost all available functionality, including downloading of music and video,  
539 when used in conjunction with libgpod of the gtkpod project<sup>22</sup>. Its pairing tool  
540 is also a requirement for using the ipheth driver mentioned before.

541 The project is a community effort and although it does not require the devices  
542 to be jail-broken, it's not supported by Apple, which means the protocol is  
543 reverse-engineered and often lags behind recent Apple releases. As an example,  
544 iOS 5 support has only recently (22/03/2012) seen the light of day in a release.  
545 Despite these shortcomings, the suite would provide the technical means for  
546 writing the applications that interact with Apple products.

547 Microsoft has also developed a protocol for media exchange called Media Trans-  
548 fer Protocol (MTP). This protocol has been standardized and is currently pub-  
549 lished by the [USB implementers forum](#)<sup>18</sup>. A LGPL-licensed library exists that  
550 supports the *Initiator* side of the communication, meaning it is able to access

---

<sup>17</sup><http://www.libimobiledevice.org/>

<sup>18</sup>[https://usb.org/sites/default/files/MTPv1\\_1.zip](https://usb.org/sites/default/files/MTPv1_1.zip)

551 media on devices that support the MTP *Responder* side: [libmtp](http://libmtp.sourceforge.net/)<sup>19</sup>. The libmtp  
552 library is currently shipped as a part of Ubuntu and can be provided in the  
553 Apertis middleware platform to be used to implement applications. Note that  
554 as is the case for Apple devices, Collabora is unable to provide legal counselling  
555 about the use of libmtp.

## 556 **UPnP**

557 Universal Plug and Play ([UPnP](http://www.upnp.org/)<sup>20</sup>) is a protocol used for discovering and brows-  
558 ing multimedia content made available by media centers. This protocol will  
559 be supported by the Apertis middleware platform using the gupnp library. For  
560 more information about this please see the Media Management design document  
561 (sometimes referred to as Media/Indexing design).

---

<sup>19</sup><http://libmtp.sourceforge.net/>

<sup>20</sup><http://www.upnp.org/>