



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

Linux-based Mobile Phone Middleware Application Programming Interface Reference Architecture

Document: CELF_MPP_RA_FR3_20060706

WARNING : This is a working draft for review only, it is NOT a published specification of the CE Linux Forum. It is likely that further substantial changes will be made in the course of review and issue resolution. Send comments on this version to:

MppApiComments@tree.celinuxforum.org

Revision History

Revision	Comment	Reviewer	Editor	Date
1.0	Initial draft for discussion at San Francisco face-to-face		NEC, Panasonic	July 2005
1.0	Minor revisions made at the San Francisco meeting		NEC, Panasonic	July 2005
2.0	Revision for Kawasaki meeting;		Scott Preece	September 2005
2.2.1	Changed to new format		Scott Preece	2005.10.21
2.2.2	Work with figures		Scott Preece	2005.10.24
2.2.3	Revisions based on Tamagawa discussions and San Francisco comments, including NEC comments. Resolution of issues raised previously and first work on harmonization with LiPS Reference Model.		Scott Preece	2005.12.22
2.2.4	Revisions to resolve Andre's issues		Scott Preece	2006.1.11
FR1	First version for Formal Review		Scott Preece	2006.3.1
FR2	Revised to resolve reviewer comments		Scott Preece	2006.6.2
FR3	Revised based on work at Editors' meeting; title updated; copyright updated		Scott Preece	2006.7.6

26	0. Introduction.....	5
27	0.1 Scope	5
28	0.2 Vocabulary and Abbreviations.....	5
29	0.3 References.....	5
30	1. The Mobile Phone Domain.....	6
31	2. Architecture.....	8
32	2.1 Applications Domain	9
33	2.1.1 Application layer	9
34	2.1.2 Middleware layer.....	9
35	2.1.3 Kernel/Driver layer.....	10
36	2.2 Communications Domain.....	11
37	3. Description of functional entities.....	12
38	3.1 Linux Kernel	12
39	3.2 Common API.....	12
40	3.3 Application Framework	12
41	3.3.1 Window Manager	13
42	3.3.2 Application Manager	13
43	3.3.3 UI Toolkit	13
44	3.3.4 UI Renderer	13
45	3.3.5 Event Bus.....	13
46	3.3.6 Others	14
47	3.4 Telephony Service	14
48	3.4.1 Circuit-Switched (Voice) Communication service	14
49	3.4.2 Packet-Switched (Data) Communication service	14
50	3.4.3 SMS Communication service	15
51	3.4.4 Equipment service	15
52	3.4.5 Personal Information Manager service	15
53	3.4.6 Data Exchange service.....	15
54	3.4.7 Record and Playback service	15
55	3.4.8 Light Management service	15
56	3.4.9 Sound service.....	15
57	3.4.10 User Profile library	15
58	3.5 Multimedia Service	15
59	3.5.1 Multimedia Manager	16
60	3.5.2 Multimedia Library.....	16
61	3.5.3 Multimedia Driver API.....	16
62	3.6 Data Processing.....	16
63	3.6.1 Bar Code Library	16
64	3.6.2 OCR Library	16
65	3.6.3 Location Services	16
66	3.7 Connectivity Service	17
67	3.8 Platform Management Service	17
68	3.9 Terminal Adaptation Function (TAF)	17

69 **3.10 Driver API..... 17**

70 **4. Data Flows 18**

71 **4.1 Voice communication..... 18**

72 **4.2 Video phone 18**

73 **4.3 Internet Application..... 19**

74 **4.4 Dial-up Networking with External Devices 20**

75 **4.5 SMS communication..... 21**

76

DRAFT

77 0. Introduction

78 This document defines a Reference Architecture – a commonly-understood organization of components for
79 implementing mobile handsets. The purpose of a reference architecture is to define a standard vocabulary
80 for talking about products in a domain. A practitioner should recognize the components and the
81 organization of the components of typical handsets as variations on such a reference architecture. The
82 purpose of this Reference Architecture is explanatory rather than constraining. Typical products will have
83 implementation architectures that modify or extend this architecture in various ways.

84 The Architecture presented in this document is based on an architecture that was originally the
85 collaborative work NEC Corporation, Panasonic Mobile Communication Ltd., and NTT DoCoMo, Inc.

86 Chapter 1 is an overview of the domain, characterizing different product tiers.

87 The basic architecture is described in chapter 2

88 The functions of each component of the architecture are described in chapter 3.

89 The data and control flows between components during typical use cases are described in chapter 4.

90 0.1 Scope

91 This document defines the reference architecture of Linux-based mobile phone. This is a non-normative
92 part of the Specification. The goal of the Reference Architecture is to provide the context for the
93 descriptions in the normative parts of the Specification.

94 The Reference Architecture does not describe the internal architecture of the communication protocol stack
95 or the Application Framework.

96 0.2 Vocabulary and Abbreviations

97 See the corresponding section in the Preface document.

98 0.3 References

99 [CS] CE Linux Forum, *Linux-based Mobile Phone Middleware Application Programming Interface –*
100 *Circuit-Switched Communication Service*

101 [Preface] CE Linux Forum, *Linux-based Mobile Phone Middleware Application Programming Interface –*
102 *Preface and Common Types*

103 [PS] CE Linux Forum, *Linux-based Mobile Phone Middleware Application Programming Interface –*
104 *Packet-Switched Communication Service*

105 [TAF] Technical Specification Group Core Network; Terminal Adaptation Functions (TAF) for services
106 using synchronous bearer capabilities (Release 4); 3GPP TS 27.003 V4.1.0 (2001-03), 3rd Generation
107 Partnership Project

108

1. The Mobile Phone Domain

109

It is customary in the mobile handset industry to describe phones as falling into different “tiers,” broad classes of phones with common characteristics – feature set, price, display size, etc. Because different tiers have different performance, price, and feature requirements, they will often be implemented by different software and hardware architectures.

113

Table 1 gives working definitions of a set of Reference Tiers. Note that these represent a particular point in time (end-of 2004) and some of the details of specific areas will change as technology changes. Many products do not fit neatly into one niche and will blend characteristics of different tiers. Also, many products will add features not covered by this table.

117

The Reference Architecture corresponds broadly to the “Smart Phone” and “Multimedia Phone” tiers in this table. The differences between those tiers are typically in the application of the architecture (the way it is used) and in the details of the components and the physical characteristics of the device. This architecture could be used for the lower tiers (“Feature Phone” and “Plain-Old Mobile”), but would include capabilities and performance enablers that would probably make such application economically inappropriate.

118

119

120

121

122

123

124

However, as component costs decrease and their performance increases, it might become practical to use this architecture more broadly; the economies of scale resulting from reusable components might offset the hardware costs.

	Tier			
Aspect	Smart Phone	Multimedia Phone	Feature Phone	Plain-Old Mobile
Focus	business focus	Personal/Entertainment Focus	Lifestyle Focus (voice plus social networking support features)	Voice
Primary Functionality	Full PDA functionality (Calendaring, address book)	Strong PIM support, personal content management features	Minimal PIM functionality (phonebook, datebook)	Phonebook and call logs
Extensibility	Extensible (downloadable features)	Limited extensibility (MIDlets or BREW)	Limited extensibility (MIDlets/BREW)	No extensibility
Multimedia	Optional	Vido capture support, Media/content players, stereo	Limited multimedia support (pictures, MP3, MIDI, Simple, low-frame-rate animations)	None
DRM	Optional	Multiple DRM schemes	Hard DRM (limits on copying any media of given types)	None
Camera	Optional	2-3 megapixel camera	VGA camera or no camera	No camera
Browser	XHTML Browser	XHTML Browser	WAP Browser (text-centric)	Embedded access to

	Tier			
Aspect	Smart Phone	Multimedia Phone	Feature Phone	Plain-Old Mobile
				specific URLs
Display	QVGA or larger color display	QSIF or larger color display	QSIF or smaller color display	Small display (64x96), non-color
Interaction	Touchscreen UI or QWERTY keyboard plus pointing device	Specialized keypad for media/game interaction	Standard keypad plus carrier-specific keys	Standard keypad
Connectivity	3G connectivity, possibly WLAN, Bluetooth, IrDA	2.5G or 3G connectivity, possibly WLAN; High-speed USB; Bluetooth	2G connectivity; USB or serial cable	2G connectivity; proprietary accessory cable
Memory	32M RAM, 64M ROM, removable storage	64M RAM, 64M ROM, Hard Disk or large removable storage	16M RAM, 16M ROM, no removable storage	8M RAM, 8M ROM or less
Processor	120MHz	200MHz	30MHz	15MHz

125

2. Architecture

126

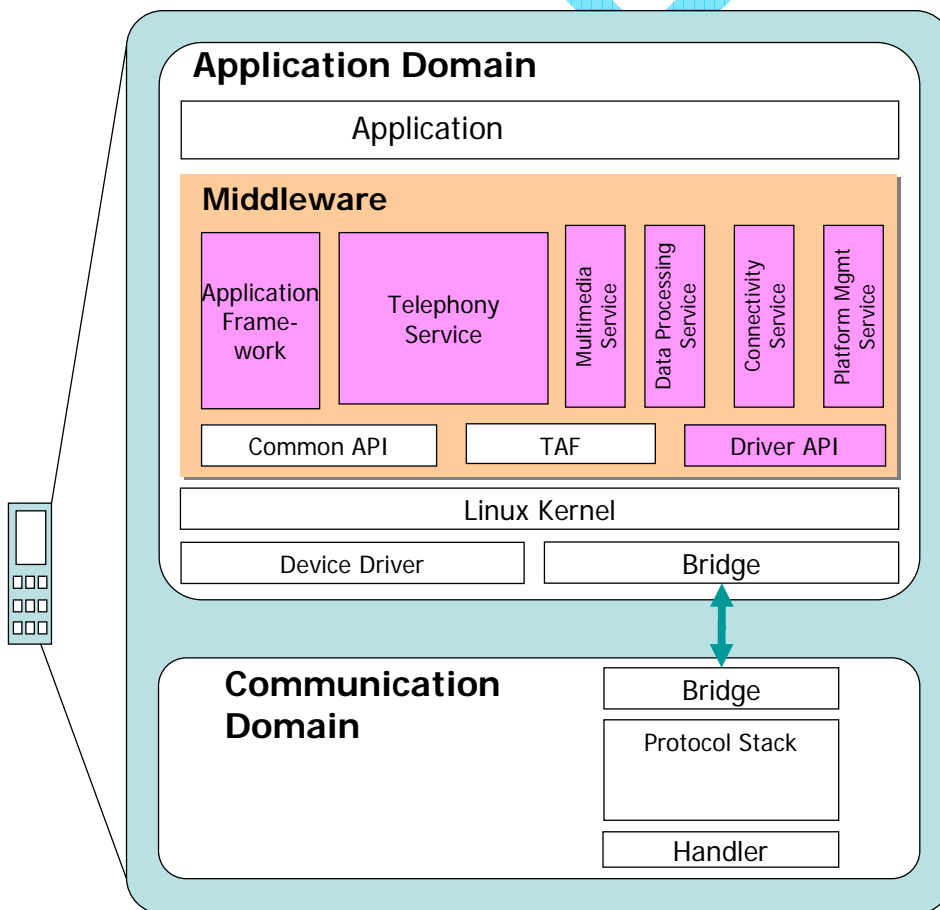
127 This document describes a reference architecture for mobile phones based on the Linux operating system.
 128 Any real product would be likely to have an implementation architecture that modifies or extends this
 129 reference architecture.

130 The architecture separates processing between two domains: the application domain and the
 131 communications domain. The two domains might run on separate processors, as separate processes on a
 132 single architecture, as separate virtual processors running over a micro-kernel, or other physical
 133 implementation. The Communications domain would include all activities requiring hard real-time
 134 behaviour.

135 Figure 1 is a top-level view of the architecture. Note that while the layering between the large boxes is
 136 meant to be strict (for instance, Applications interact with the Kernel only through Middleware
 137 components, not directly). The horizontal arrangement is arbitrary.

138 The Reference Architecture identifies this partitioning of processing domains but does not dictate the
 139 physical realization used. In Figure 1, the Bridge represents whatever mechanism is used in the particular
 140 realization to support interaction between the domains. The implementation of the Bridge is not in the
 141 scope of the Reference Architecture.

142 The blocks shown with a white background in Figure 1 are not part of the scope of the Mobile Phone API.
 143 They are shown as part of the common understanding of the structuring of a Linux-based mobile phone.



144

145

146

Figure 1 Overall Architecture

147 **2.1 Applications Domain**

148 The software running in the Application Domain contains the following 4 layers:

149 Application

150 Middleware

151 Linux kernel

152 Driver & Bridge

153 **2.1.1 Application layer**

154 The application layer contains various applications. They are classified into the following 8 categories;
155 samples are listed to illustrate the categorization, but are not meant either to be requirements or to be a
156 complete list of the applications in a given category:

157 **2.1.1.1 Telephony applications**

158 Telephony applications include Standby Screen (Idle), main menu, videophone application, phone
159 applications, phonebook and other Personal Information Management (PIM) applications, and phone
160 personalization settings.

161 **2.1.1.2 System applications**

162 System applications include Air download, Generic LCD display, Backside LCD display, PIN
163 authentication and monitor mode, other function setup, Equipment alarm, etc.

164 **2.1.1.3 Multimedia applications**

165 Multimedia applications include still image viewer, video viewer, camera app, vector graphics viewer,
166 avatar and ring tone management.

167 **2.1.1.4 Data-processing applications**

168 Data-processing applications include OCR, barcode, SD-PIM, data transfer, memory transfer, external I/F
169 communication, user data, IR, schedule, voice memo, schedule alarm, and data folder.

170 **2.1.1.5 Internet-applications**

171 Internet applications are those that use TCP/IP to access resources on the internet, including e-mail,
172 Browser, HTML mailer, etc.

173 **2.1.1.6 Internet Application Engine**

174 Internet application engines are application-level services that would be used by applications that include
175 internet-enabled functions; examples include engines for HTTP, SSL, embedded languages, etc.

176 **2.1.1.7 Java Application Engine**

177 The Java applications engine includes a Java Virtual Machine (JVM), Java Application Manager (JAM),
178 and class libraries.

179 **2.1.1.8 Others**

180 Other applications could include the Accessory menu, productivity applications e.g., text memo,
181 calculator), etc.

182 **2.1.2 Middleware layer**

183 Middleware layer contains the following components.

184 **2.1.2.1 Applications framework**

185 The Applications framework provides application developers with a common framework of services
186 commonly used by mobile-phone applications.

187 **2.1.2.2 Telephony service**

188 The telephony service provides application developers with a framework of services for communications
189 and handset management.

190 **2.1.2.3 Multimedia service**

191 The multimedia service provides video phone service (H324, for example), and multimedia decoding,
192 encoding, and rendering facilities.

193 **2.1.2.4 Data processing service**

194 The data-processing service supports processing the data from various devices, e.g., bar-code reader,
195 optical character reader, etc.

196 **2.1.2.5 Platform Management service**

197 The Platform Management Service provides the functions of system management, including installation of
198 software and control of system processes.

199 **2.1.2.6 Connectivity Service**

200 The Connectivity Service handles inter-device functions, such as synchronization and OBEX data
201 exchange.

202 **2.1.2.7 TAF (Terminal Adaptation Function)**

203 The TAF provides the back-end services needed by the Telephony Service APIs [TAF]. It mediates
204 between the phone software's model of network programming and the specifics needed for the particular
205 networks supported by a specific handset. Note that the TAF could equally well be located in the
206 Communications Domain or in the device driver layer; the placement shown in the diagram is common, but
207 not universal.

208 **2.1.2.8 Common API**

209 The Common API provides application developers with standard C-language functions, including standard
210 libraries and system calls.

211 **2.1.2.9 Driver API**

212 The device-driver API provides middleware and application programs access to devices and to services
213 modeled as devices.

214 **2.1.3 Kernel/Driver layer**

215 **2.1.3.1 Kernel and Device Drivers**

216 The Kernel / Driver layer contains the Linux Kernel and device drivers and the Application-Domain end of
217 the Bridge.

218 **2.1.3.2 Bridge**

219 The Bridge supports communication between the Application and Communication domains, which may be
220 implemented as separate processors or not, but will minimally have different scheduling regimes.

221

2.2 Communications Domain

222

The Communications Domain performs all processing that requires hard real-time behaviour, including executing the lower levels of the protocol between the device and the network. Its protocol stack contains the network stack's L1, L2, L3 layers. The Bridge supports communication with the Applications Domain.

223

224

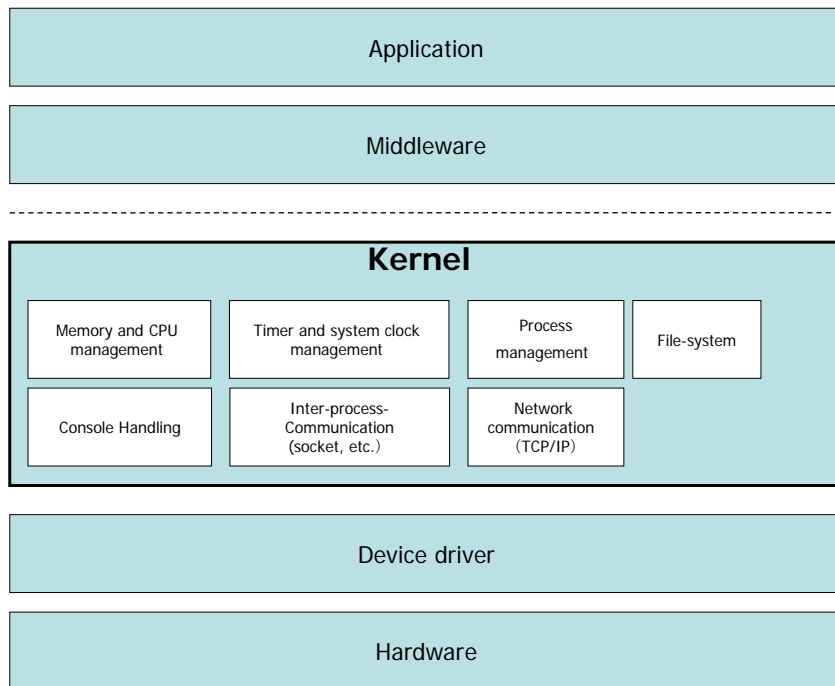
DRAFT

225 3. Description of functional entities

226 3.1 Linux Kernel

227 The Linux Kernel provides:

- 228 • Memory and CPU management
- 229 • Timer and system clock management
- 230 • Process management: create, destroy and dispatch
- 231 • File-systems: files, directories, and space management
- 232 • Console handling
- 233 • Inter-process-communication: sockets, message queues, shared memory, etc.
- 234 • Network communication: TCP/IP



235

236

Figure 2 Linux Kernel

237 3.2 Common API

238 The Common API contains various functions for applications to use, including the standard C libraries. The
239 Common APIs conform to the POSIX standard.

240 3.3 Application Framework

241 Fig-3 is an overview of the application framework. This is an area where variation is common in the
242 domain, with the specific requirements of the application set and the UI framework chosen for specific
243 products potentially differing from this reference architecture, Usually, however, the functional separation
244 is similar to this.

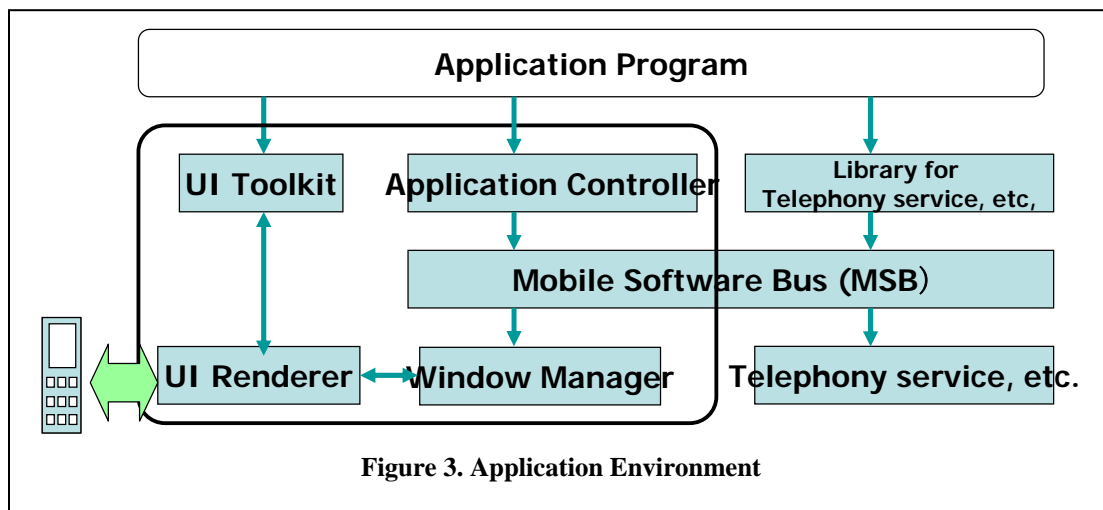


Figure 3. Application Environment

245

246 3.3.1 Window Manager

247 The Window Manager provides unified operation and decoration for windows and controls overlap of
248 windows (clipping and layering).

249 In the Application framework for mobile-phone, window manager provides:

250 Foreground and background control of application window

251 Tracking the state of applications (active, inactive, idle, not-started)

252 3.3.2 Application Manager

253 The Application Manager (AM) controls the start and end of applications. That is, starts applications,
254 switches between applications, and shuts down applications at power-off. AM also manages application
255 start status, and controls application switching operation (e.g., selection of an application to be next used as
256 a foreground application at application switching).

257 The Window Manager controls window stacking, focusing, and properties for each group during the period
258 from window generation to deletion. The Window Manager receives requests from the AM through the
259 Event Bus and resolves contention between applications, according to the priority table based on the
260 application status. It also control windows overlapping and key focusing.

261 The Application Manager uses the Event Bus to communicate between applications and the Window
262 Manager.

263 3.3.3 UI Toolkit

264 The UI Toolkit is a set of functions and facilities for the application to use to present and manage
265 interaction with the user.

266 3.3.4 UI Renderer

267 The UI renderer controls presentation of the interaction elements on the display(s).

268 3.3.5 Event Bus

269 The Event Bus is a framework for passing messages between entities. It supplies communication services
270 (synchronous and asynchronous communication) between applications and services. Different
271 implementations of the reference architecture may use different kinds of inter-process communication for
272 this role. See [Preface] for more information about the programming model and the portion of the Event
273 Bus semantics that are visible in the APIs.

274 **3.3.6 Others**

275 **3.3.6.1 PICT (PICTograph) display library**

276 It controls turning on/off of the upper pictograph elements such as antenna and battery.

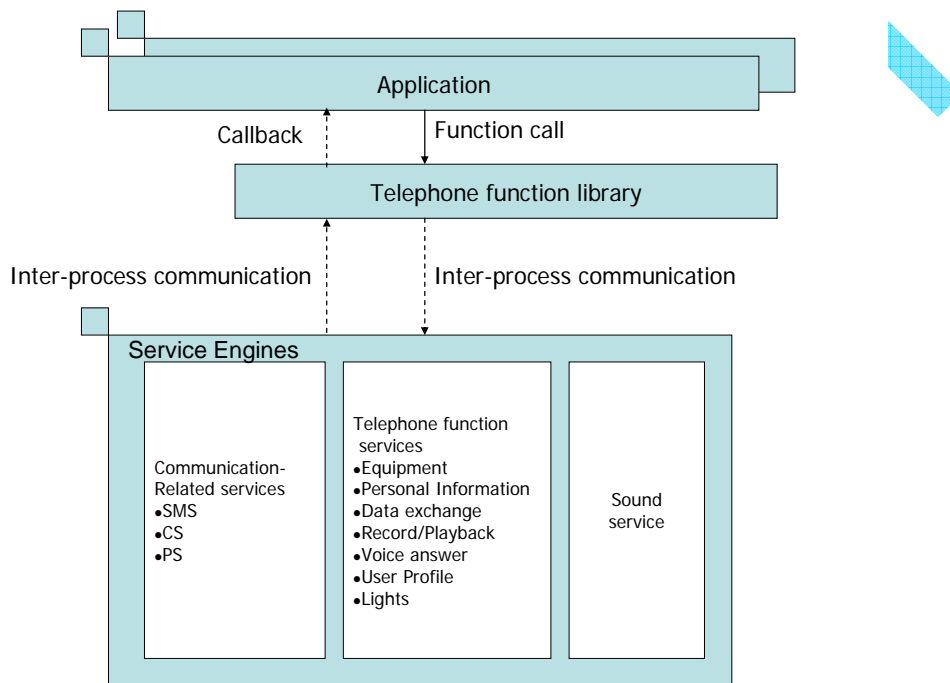
277 **3.3.6.2 Image library**

278 It Converts, extracts, or compresses image data, aloes used to set or acquire image information.

279 **3.4 Telephony Service**

280 Fig-4 describes the telephony processing framework. The Telephony Service provides abstract (not specific
 281 to the network technology or the handset hardware) APIs for applications to use to access the functions of
 282 the handset, including both communications services and control of the typical range of equipment features.

283 The architecture separates the interfaces to the various function-specific services from their
 284 implementations. A specific implementation architecture might choose to make this a deployment
 285 separation, as shown in Figure 4, or might choose to bundle them within a library implementation.



286
 287 **Figure 5 Telephony Processing**

288 **3.4.1 Circuit-Switched (Voice) Communication service**

289 The Circuit-Switched Communication (CS) service provides application programs with abstract APIs for
 290 dialing, call disconnection, rejecting incoming calls, etc., for voice and video communications. This API is
 291 available as [CS].

292 **3.4.2 Packet-Switched (Data) Communication service**

293 The Packet-Switched Communication (PS) service provides application programs with abstract APIs for
 294 initiating and terminating sessions and connections, rejecting incoming calls, etc., for data communications.
 295 This API is available as [PS].

296 **3.4.3 SMS Communication service**

297 The SMS Communication service provides application programs with abstract APIs for sending and
298 receiving SMS messages, receiving notification of events, and checking the status of the SMS service, etc.

299 **3.4.4 Equipment service**

300 The Equipment service provides APIs for programs to set up, control, and read the status of various handset
301 hardware elements (batteries, headsets, etc.).

302 **3.4.5 Personal Information Manager service**

303 PIM provides APIs for programs to register a schedule (calendar), sort the schedule data, read to-do data,
304 etc.

305 **3.4.6 Data Exchange service**

306 The Data Exchange service provides APIs for programs to handle phone-book, memo, image, and video,
307 etc. on internal and removable memory stores.

308 **3.4.7 Record and Playback service**

309 Record and Playback provides application programs with record and playback of voice memo.

310 **3.4.8 Light Management service**

311 The Light Management service provides APIs for programs to for control various lights.

312 **3.4.9 Sound service**

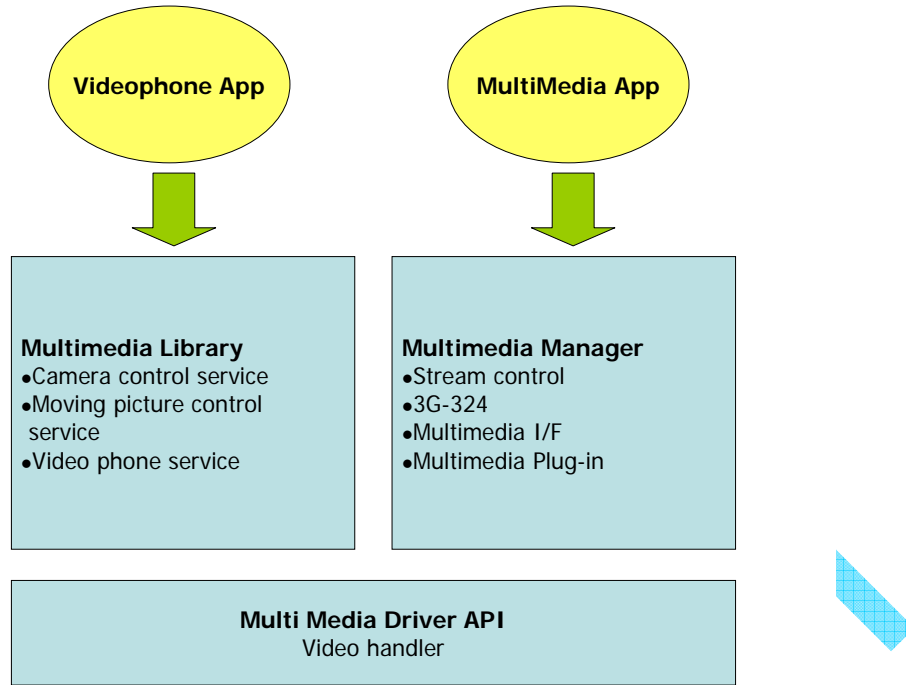
313 The Sound service provides abstract APIs for functions to play and manage ring-tones and other sounds.

314 **3.4.10 User Profile library**

315 The User Profile library provides application programs with the ability to read and set various attributes of
316 the user's profile, such as registered phone numbers, owner name, and e-mail addresses.

317 **3.5 Multimedia Service**

318 Fig-5 describes the multimedia service.



319

320

Figure 6 Multimedia Framework

321 3.5.1 Multimedia Manager

322 The Multimedia Manager provides interfaces for interconnecting multimedia functions, such as video
323 phone library, 3G-H324M protocol support, camera control library, and moving picture control library.

324 3.5.2 Multimedia Library

325 The Multimedia Library provides interfaces for camera control, moving picture control, and video phone
326 services.

327 3.5.3 Multimedia Driver API

328 Multimedia Driver API provides video handler interfaces.

329 3.6 Data Processing

330 This block covers various kinds of add-on functionality related to dealing with external data

331 3.6.1 Bar Code Library

332 It provides the bar code reader functions.

333 3.6.2 OCR Library

334 It provides the OCR access reader functions.

335 3.6.3 Location Services

336 This provides access to the geographical location of the phone and events related to recognizing specific
337 locations, for phones equipped with the appropriate capabilities.

338 **3.7 Connectivity Service**

339 The OBEX (object exchange) module supports synchronization and sharing of information between
340 devices by exchange of data objects. It provides an interface that is used by OBEX to perform
341 communication processing based on request messages from application programs and to return processing
342 results to the application programs. It also provides an interface that is used by OBEX to convert objects to
343 canonical format for interchange.

344 **3.8 Platform Management Service**

345 The Platform Management monitors the activation of each task at the time of power on and the deactivation
346 of each task at the time of power off, as well as the charging and other statuses of the mobile terminal.

347 **3.9 Terminal Adaptation Function (TAF)**

348 The TAF is the interface between the phone software's networking functionality (voice and data) and the
349 network protocols that are used over the connected network(s); it is the entity that actually constructs
350 sessions and connections in response to abstract requests from clients, adapting a generic view of
351 connections to the specific interactions needed for a particular network.

352 Applications do not use the TAF API directly; they use it only through the APIs provided by the Telephony
353 Service.

354 **3.10 Driver API**

355 The Driver API provides upper layer components (middleware and application programs) present an
356 abstract interface to device drivers, so that device-independent components do not need to be aware of the
357 particular device drivers available on a specific handset, hardware-specific ioctls, etc. This avoids hardware
358 dependencies, enabling development of portable middleware and application programs for mobile phones.
359 The Driver API sits between the client applications and the actual device drivers.

360

4. Data Flows

361

This section describes data and control flow between components of the architecture in various domains.

362

4.1 Voice communication

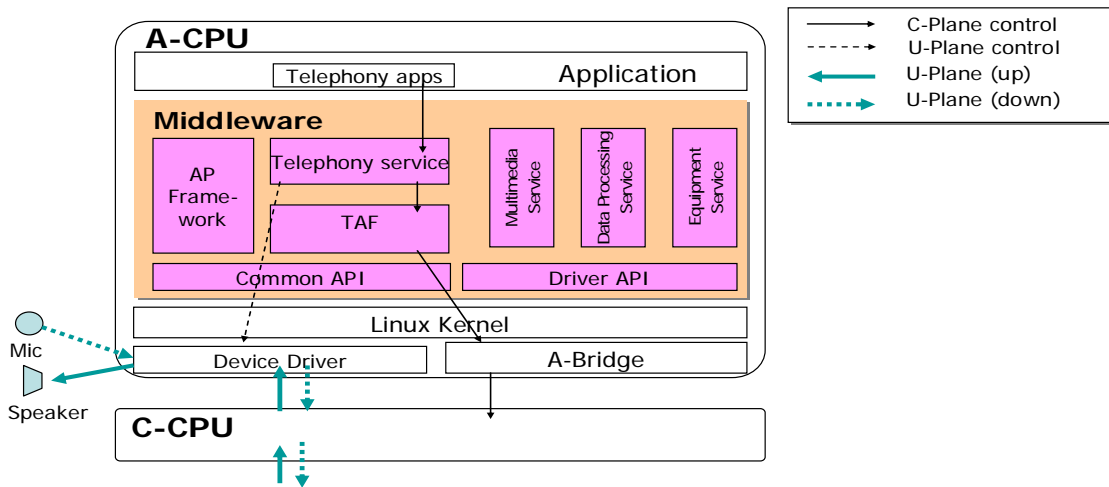
363

Telephony application controls the C-plane using the telephony service. This figure shows the flows for a mobile-originated call; for a mobile-terminated call there would also be C-plane flows upward into the telephony service to announce the incoming call.

364

365

Voice data is decoded by a protocol-specific codec and sent to the appropriate audio output.



367

368

369

Figure 7 Voice Communication

370

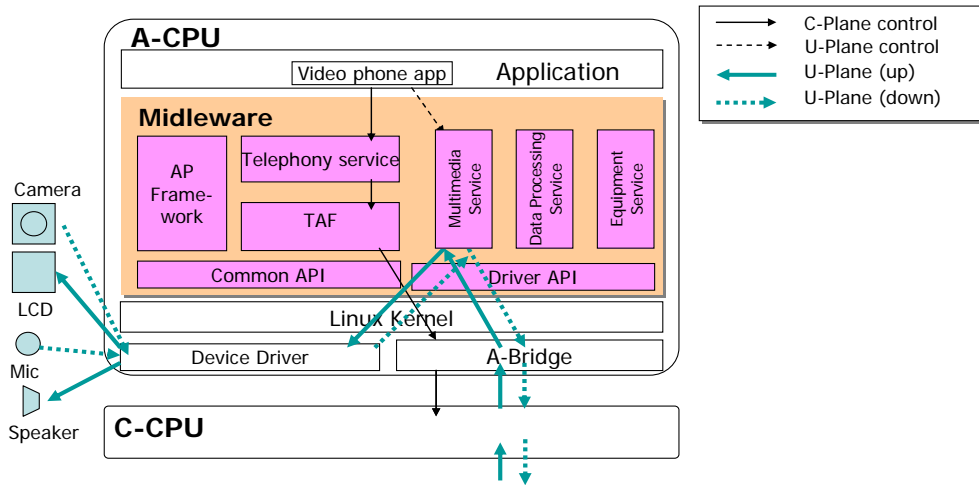
4.2 Video phone

371

Video phone application program controls C-Plane by telephony service.

372

Multimedia service provide decoding and encoding facilities to support video telephony, such as H.324.



373

374

Figure 8 Video Phone

375

4.3 Internet Application

376

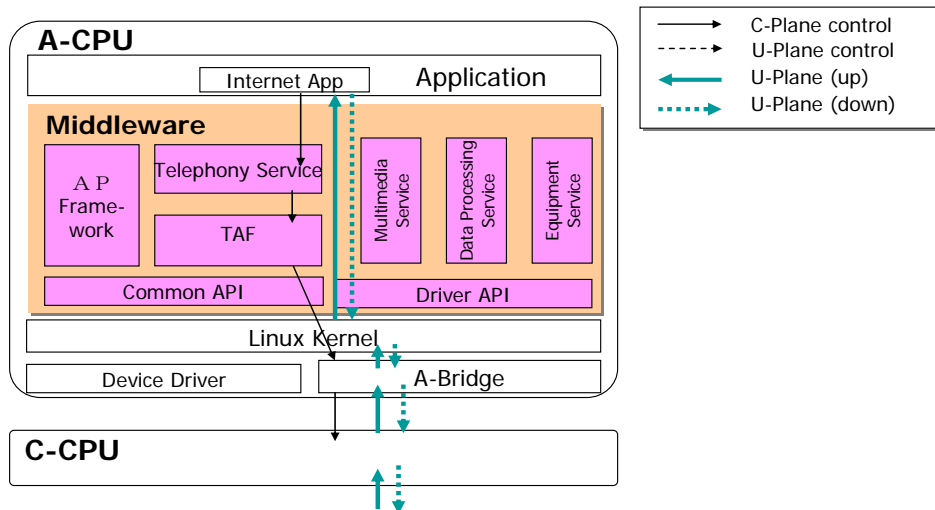
Internet Application program controls C-Plane by telephony service.

377

Data from the Internet is transported by a protocol stacked based on TCP/IP using the Linux Kernel

378

networking capabilities.



379

380

Figure 9 Internet Application

381

4.4 Dial-up Networking with External Devices

382

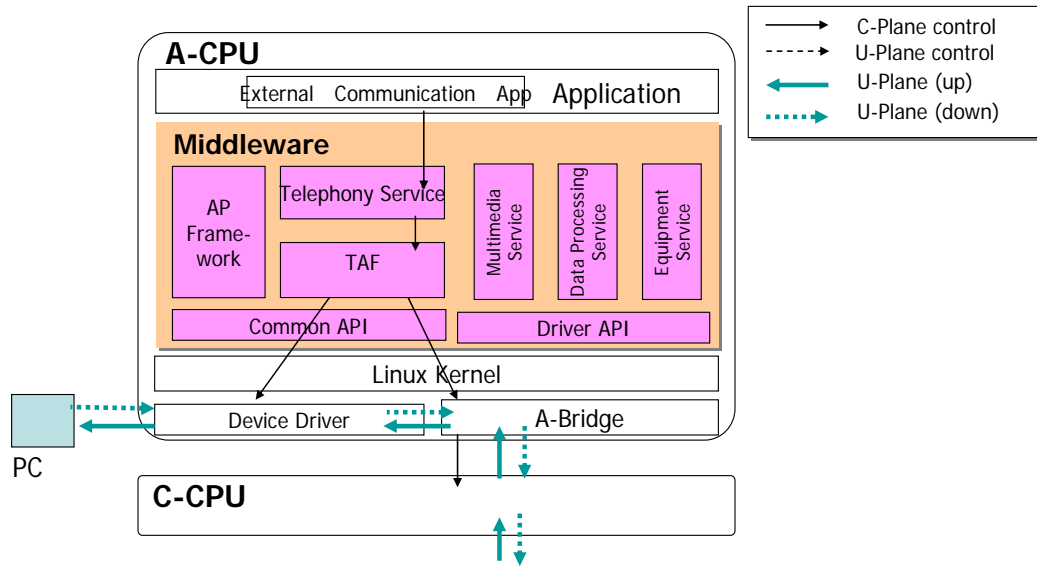
Applications program for dial-up networking controls C-Plane by telephony service.

383

Data are received and transmitted to an attached external device through USB or other communication bus

384

and device driver and routed back to the internet through the communications stack.



385

386

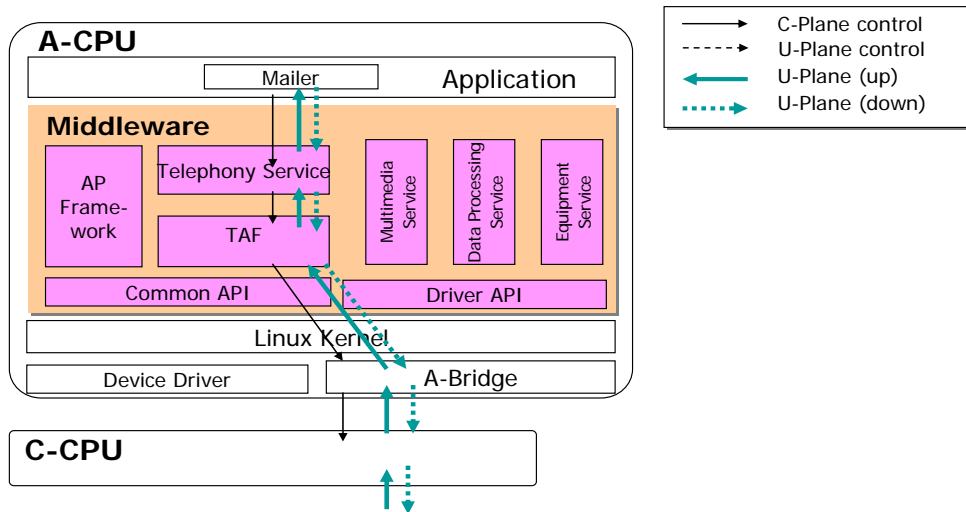
Figure 10 PPP Communication

387

4.5 SMS communication

388

Telephony Service SMS library controls C-Plane and U-Plane to send and receive short messages.



389
390
391
392
393

Figure 11 SMS Communication

DRAFT