
# Linux-based Mobile Phone Middleware Application Programming Interface

# Reference Architecture

Document: CELF_MPP_RA_FR3_20060706

Deleted: 3G Specification

Deleted: Multimedia Mobile Phone API

Deleted: FR2_20060602

Formatted: English (U.S.)



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 |



 

Formatted: Italian (Italy)
Field Code Changed
Formatted: Italian (Italy)
Field Code Changed
Formatted: Italian (Italy)
Field Code Changed
Formatted: Italian (Italy)
Field Code Changed
Formatted: Italian (Italy)
Field Code Changed
Formatted: Italian (Italy)
Field Code Changed
Formatted: Italian (Italy)

Deleted: ¶
¶
¶
¶
0. Introduction 5¶
0.1 Scope 5¶
0.2 Vocabulary and Abbreviations 5¶
0.3 Reference 5¶
1. The Mobile Phone Domain 6¶
2. Architecture 8¶
2.1 Applications Domain 9¶
2.1.1 Application layer 9¶
2.1.2 Middleware layer 10¶
2.1.3 Kernel/Driver layer 11¶
2.2 Communications Domain 11¶
3. Description of functional entities 12¶
3.1 Linux Kernel 12¶
3.2 Common API 12¶
3.3 AP Framework 12¶
3.3.1 Window Manager 13¶
3.3.2 Application Controller (APC) 13¶
3.3.3 UI Toolkit 13¶
3.3.4 UI Renderer 13¶
3.3.5 Mobile Software Bus (MSB) 13¶
3.3.6 Others 14¶
3.4 Telephony processing 14¶
3.4.1 Circuit-Switched (Voice) Communication service 14¶
3.4.2 Packet-Switched (Data) Communication service 14¶
3.4.3 SMS Communication service 15¶
3.4.4 Equipment service 15¶
3.4.5 Schedule Error! Bookmark not defined.¶
3.4.6 Data Exchange 15¶
3.4.7 Record and Playback 15¶
3.4.8 Light Management 15¶
3.4.9 Sound System 15¶
3.4.10 User Profile Library 15¶
3.4.11 Removable Media Management 15¶
3.5 Multimedia Framework 15¶
3.5.1 Multimedia Manager 16¶
3.5.2 Multimedia Library 16¶
3.5.3 Multimedia Driver API 16¶
3.6 Data Processing 16¶
3.6.1 Bar Code Library 16¶
3.6.2 OCR Library 16¶
3.7 Connectivity Service 17¶
3.8 Platform Management Service 17¶
3.9 Driver API 17¶
4. Data Flows 18¶
4.1 Voice communication 18¶
4.2 Video phone 18¶
4.3 Internet Application 19¶
4.4 Dial-up Networking with External Devices 20¶
4.5 SMS communication 21¶

# 0. Introduction

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

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

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

The basic architecture is described in chapter 2

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

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

## 0.1 Scope

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

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

## 0.2 Vocabulary and Abbreviations

See the corresponding section in the Preface document.

## 0.3 References

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

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

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

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

**Deleted:** v

**Comment [sep1]:** 4 "Linux based mobile phone " scope is not same with scope in document title; TITLE CHANGED

**Deleted:**

**Comment [sep2]:** 26 Point out, that the architecture should be extendible DONE

**Comment [sep3]:** 2 "Reference " Syntax error DONE

**Comment [sep4]:** 11 Add a reference to the API Preface document and both the CS and PS APIs and add references to the CS and PS documents to sections 3.4.1 and 3.4.2, respectively. DONE

**Deleted:** [Preface] <provide reference to the Preface document>¶
[CS] <provide reference to the Circuit-Switched Communication Service section>¶
[PS] <provide reference to the Packet-Switched Communication Service section>

# 1. The Mobile Phone Domain

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.

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.

Comment [sep5]: 6 (and-of 2004) not easy to understand CLARIFIED

Comment [sep6]: 5 tables title rows have same background with content rows DONE

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. 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.

Formatted Table

| | 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 |



 

Formatted Table

| | 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

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

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

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

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

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

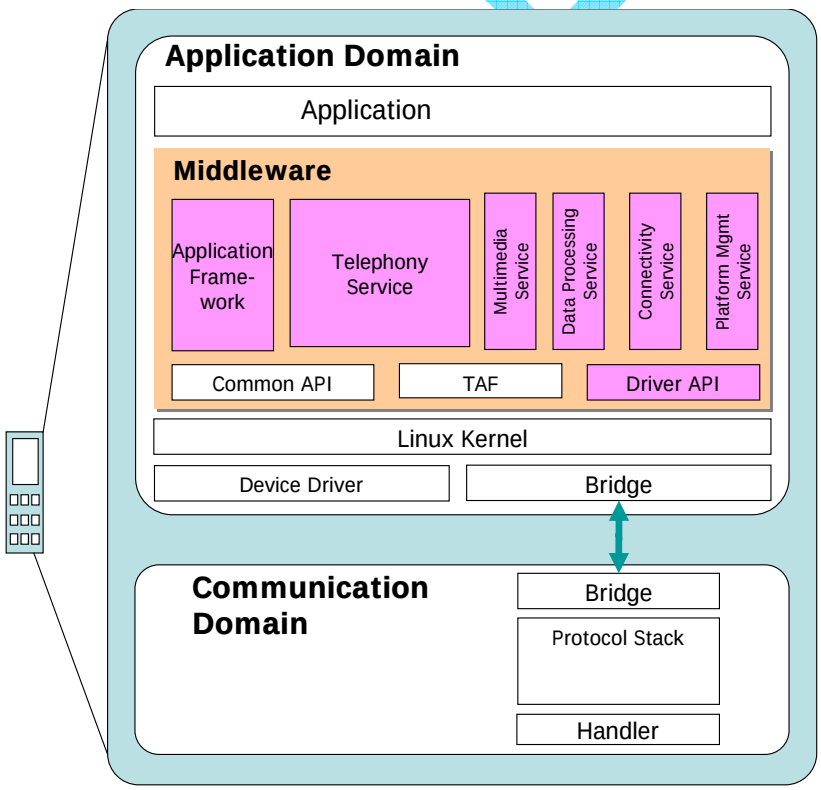


**Figure 1 Overall Architecture**

**Comment [sep7]:** 27 Point out, that the architecture should be extendible DONE

**Comment [sep8]:** 28 Typo: processors FIXED

**Comment [sep9]:** 29 Typo: 'The implementation of the bridge isn't …' FIXED

**Comment [sep10]:** 44 The picture, as drawn, with the protocol stack on the right hand side implies that the components above it use the protocol stack, when actually it is the components on the left hand side (telephony service, TAF) which use the protocol stack. DIAGRAMS TO BE UPDATED

**Comment [sep11]:** 45 The diagram includes components L3-A and L3-C which are never described. REMOVED

**Deleted:** ¶

**Comment [sep12]:** 15 The layering and positioning of the TAF component below the Telephony Service implies the TAF component's interface is not directly visible to the Applications Layer. The text in Section 2 should make this implicit dependency explicit. DIAGRAMS TO BE UPDATED IN FUTURE.

**Comment [sep13]:** 17 Add L3-A to the Preface document's terminology section REMOVED REFERENCE

## 2.1 Applications Domain

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

Application

Middleware

Linux kernel

Driver & Bridge

### 2.1.1 Application layer

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

#### 2.1.1.1 Telephony applications

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

#### 2.1.1.2 System applications

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

#### 2.1.1.3 Multimedia applications

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

#### 2.1.1.4 Data-processing applications

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

#### 2.1.1.5 Internet-applications

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

#### 2.1.1.6 Internet Application Engine

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

#### 2.1.1.7 Java Application Engine

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

#### 2.1.1.8 Others

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

### 2.1.2 Middleware layer

Middleware layer contains the following components.

**Comment [sep14]:** 30 Term 'Internet Application Engine' not clear. CLARIFIED

**Comment [sep15]:** 31 PIM (Personal Information Management) is missing. ADDED

**Deleted:** s includes

**Deleted:** Accessories

**Deleted:** (

**Deleted:** etc.

**Deleted:** .



 

#### 2.1.2.1 Applications framework

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

#### 2.1.2.2 Telephony service

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

#### 2.1.2.3 Multimedia service

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

**Comment [sep16]:** 19 Add H234 and H324M to the Preface document's terminology section and add their references to Section 0.3. REWORKED

#### 2.1.2.4 Data processing service

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

#### 2.1.2.5 Platform Management service

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

#### 2.1.2.6 Connectivity Service

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

#### 2.1.2.7 TAF (Terminal Adaptation Function)

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

**Comment [sep17]:** 43 Some more information on what the TAF is would be really helpful – stating that the TAF component consists of other TAF components for voice, packet, etc. doesn't help me picture what scope of responsibility it has. CLARIFIED

#### 2.1.2.8 Common API

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

#### 2.1.2.9 Driver API

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

### 2.1.3 Kernel/Driver layer

#### 2.1.3.1 Kernel and Device Drivers

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

#### 2.1.3.2 Bridge

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



 

## 2.2 Communications Domain

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.

**Comment [sep18]:** 21 Add L1, L2 and L3 to the Terminology section of the API Preface as well as C-Plane and U-Plane (referenced in Sections 4.). A a reference for L1-L3 to Section 0.3 of the Preface. DONE

# 3. Description of functional entities

## 3.1 Linux Kernel

The Linux Kernel provides:

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

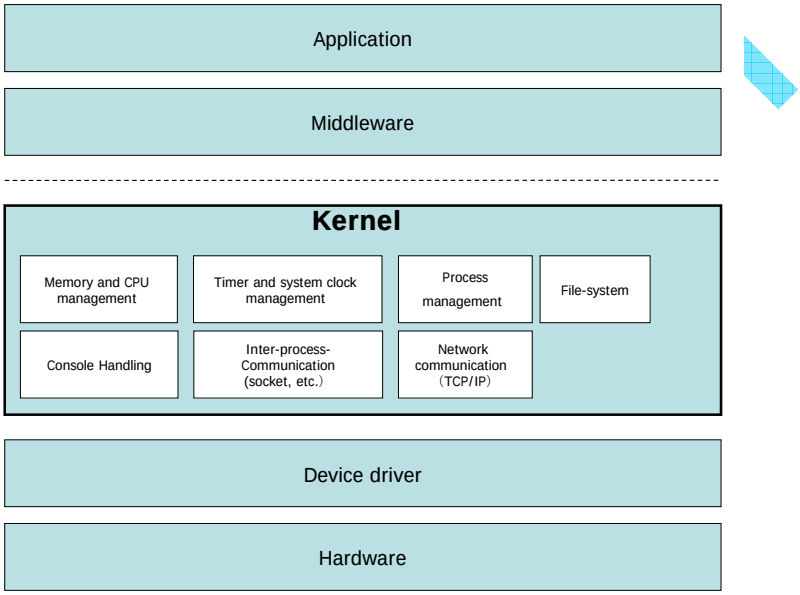


**Figure 2 Linux Kernel**

## 3.2 Common API

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

## 3.3 Application Framework

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

**Comment [sep19]:** 32 Chapter about 'TAF' is missing No, it's discussed above

**Comment [sep20]:** AB: Do we really need any explanation/opening up of the kernel here? It's not really part of our scope and could probably just be an opaque box. NOT CHANGED FOR NOW.

 

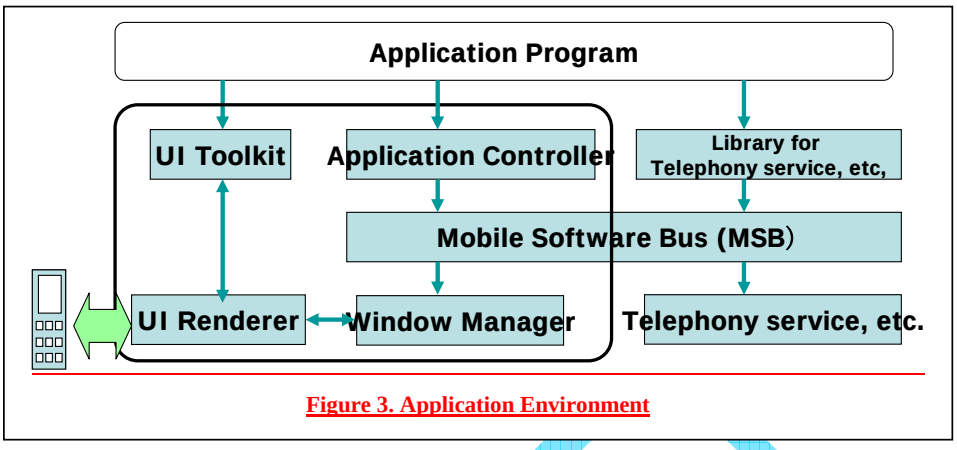


Figure 3. Application Environment

### 3.3.1 Window Manager

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

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

Foreground and background control of application window

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

### 3.3.2 Application Manager

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

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

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

### 3.3.3 UI Toolkit

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

### 3.3.4 UI Renderer

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

### 3.3.5 Event Bus

The Event Bus is a framework for passing messages between entities. It supplies communication services (synchronous and asynchronous communication) between applications and services. Different implementations of the reference architecture may use different kinds of inter-process communication for

Deleted: <sp>

Formatted: Centered, Keep with next

Formatted: Centered

Deleted: Controller

Deleted: (APC)

Deleted: APC

Deleted: APC

Deleted: APC

Deleted: library

Deleted: Controller



 

this role. See [Preface] for more information about the programming model and the portion of the Event Bus semantics that are visible in the APIs.

### 3.3.6 Others

#### 3.3.6.1 PICT (PICTograph) display library

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

#### 3.3.6.2 Image library

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

## 3.4 Telephony Service

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

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

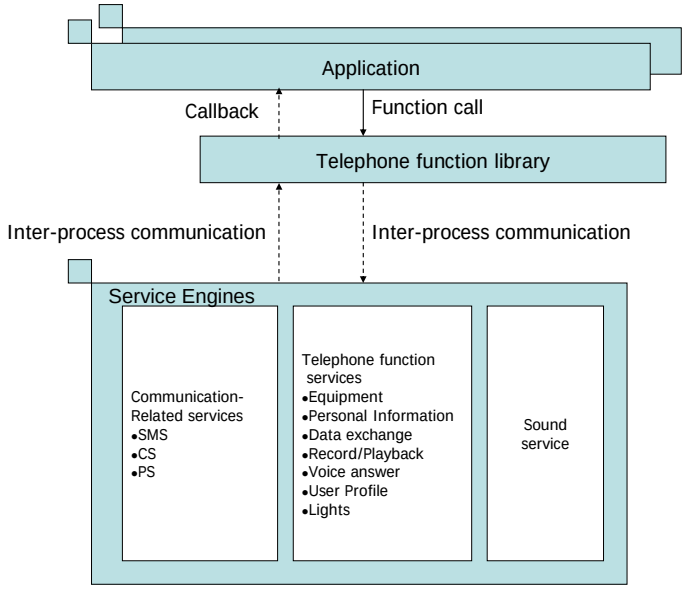


**Figure 5 Telephony Processing**

### 3.4.1 Circuit-Switched (Voice) Communication service

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

### 3.4.2 Packet-Switched (Data) Communication service

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

**Comment [sep21]:** 34 Description of 'Key' is missing. REF DELETED

**Deleted:** 4

**Comment [sep22]:** 23 Need to clarify the functionality of the CS and PS services versus the "voice communication" and "packet communication" functionality that is part of the TAF component described in Section 2.1.2.7. CLARIFIED

### 3.4.3 SMS Communication service

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

Deleted: .

### 3.4.4 Equipment service

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

### 3.4.5 Personal Information Manager service

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

### 3.4.6 Data Exchange service

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

### 3.4.7 Record and Playback service

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

### 3.4.8 Light Management service

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

### 3.4.9 Sound service

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

### 3.4.10 User Profile library

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

## 3.5 Multimedia Service

Fig-5 describes the multimedia service.

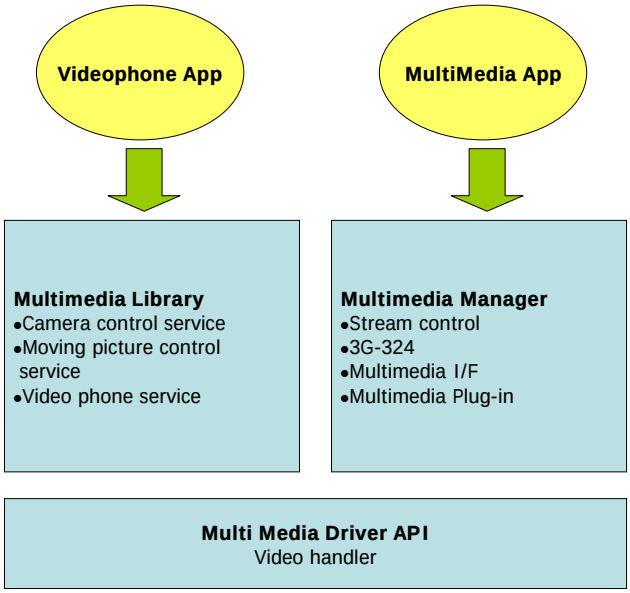


Figure 6 Multimedia Framework

### 3.5.1 Multimedia Manager

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

### 3.5.2 Multimedia Library

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

### 3.5.3 Multimedia Driver API

Multimedia Driver API provides video handler interfaces.

## 3.6 Data Processing

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

### 3.6.1 Bar Code Library

It provides the bar code reader functions.

### 3.6.2 OCR Library

It provides the OCR access reader functions.

### 3.6.3 Location Services

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

**Comment [sep23]:** 7 No the relationship between Multimedia library/Multimedia Manager/ Multimedia driver blocks NOT SURE WHAT IS WANTED

**Deleted:** 5

**Comment [sep24]:** 19 Add H234 and H324M to the Preface document's terminology section and add their references to Section 0.3. DONE - Added definitions, not references; references not needed, since they're just examples.



 

## 3.7 Connectivity Service

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

## 3.8 Platform Management Service

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

## 3.9 Terminal Adaptation Function (TAF)

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

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

## 3.10 Driver API

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

**Comment [sep25]:** 37 Is this on top of the Linux driver interface? YES



 

# 4. Data Flows

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

## 4.1 Voice communication

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.

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

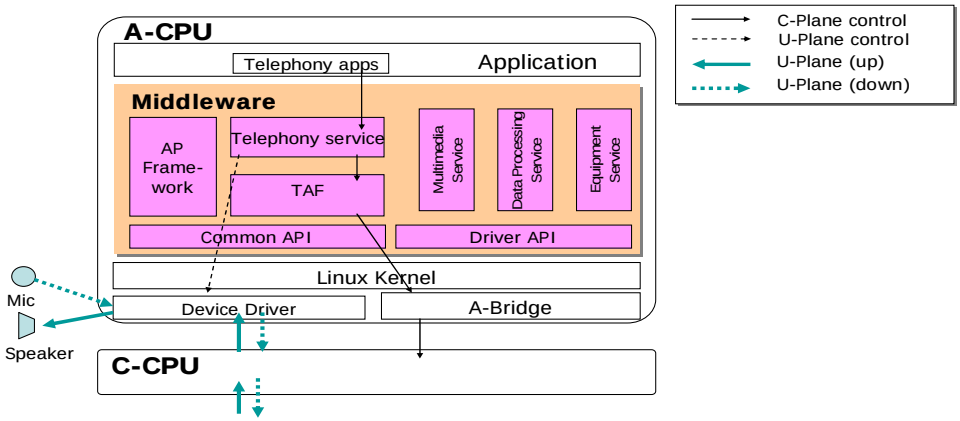


**Figure 7 Voice Communication**

## 4.2 Video phone

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

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

**Comment [sep26]:** 42 More information would be useful about the assumed partitioning of responsibilities between the application, telephony services and TAF modules, for all telephony use cases. REWORKED A LITTLE

**Comment [sep27]:** 46 All of section 4 – It would be really useful to explain, at some point, what the assumed processing is for each component on each message. Showing that there is a flow of data is not sufficient – there is an implied transformation of the data as it passed through components which is never discussed here. We have no idea just what these components do, other than that some messages pass through them WILL REWORK WHEN DIAGRAMS UPDATED

**Comment [sep28]:** 41 C-Plane is not defined in the glossary or previous in this section NOW IN GLOSSARY

**Deleted:** a

**Comment [sep29]:** 38 Update with figure 1. This is true for alle the following figures DIAGRAMS TO BE UPDATED

**Comment [sep30]:** 39 An incoming call would be signaled from the C-CPU and goes up to the telephony apps CLARIFIED

**Comment [sep31]:** 25 figs 6-10 These figures should be instances of Figure 1 thus all references to A-CPU and C-CPU should be replaced with Application Domain and Communication Domain, respectively. DIAGRAMS TO BE UPDATED

**Deleted:** 6

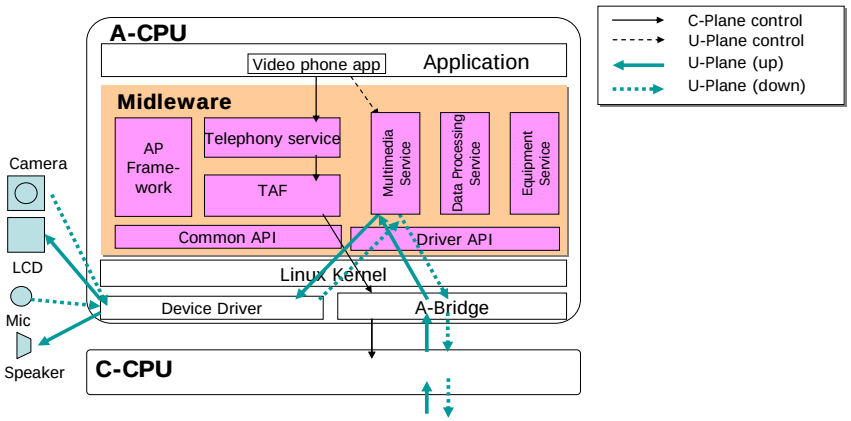


**Figure 8 Video Phone**

Deleted: 7

## 4.3 Internet Application

Internet Application program controls C-Plane by telephony service.

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



 

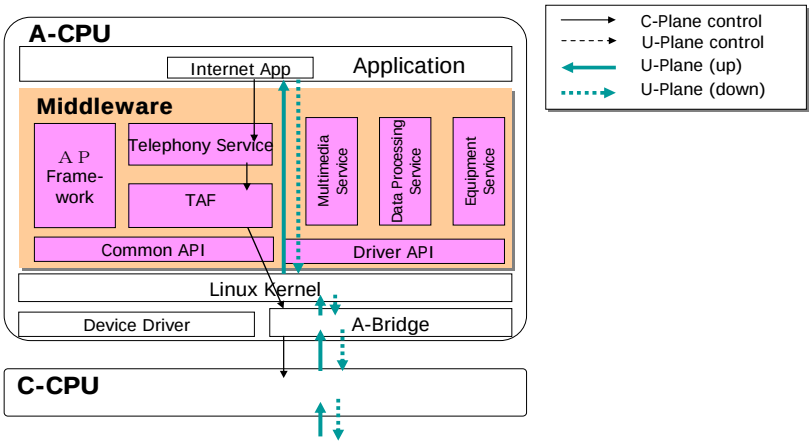


**Figure 9 Internet Application**

Deleted: 8

## 4.4 Dial-up Networking with External Devices

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

Data are received and transmitted to an attached external device through USB or other communication bus and device driver and routed back to the internet through the communications stack.

**Comment [sep32]:** 40 There shouldn't be direct communication between device driver and A-Bridge WHY?

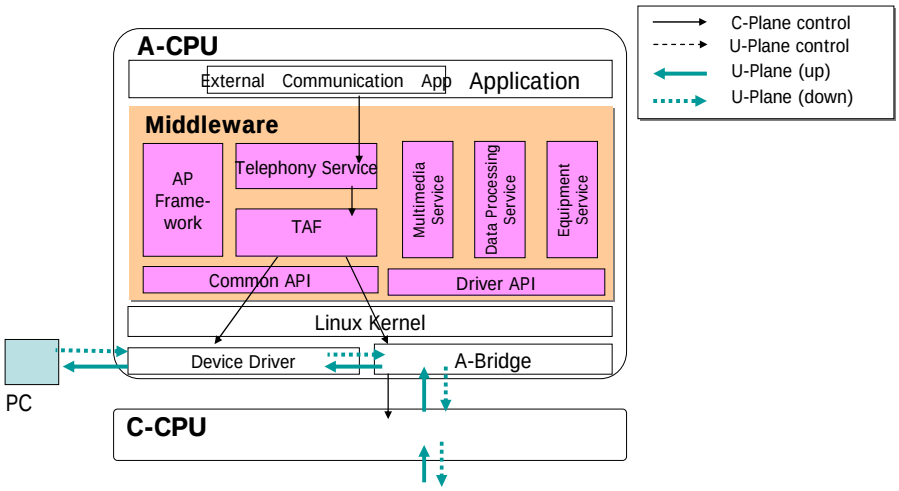


**Figure 10 PPP Communication**

Deleted: 9

## 4.5 SMS communication

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



 

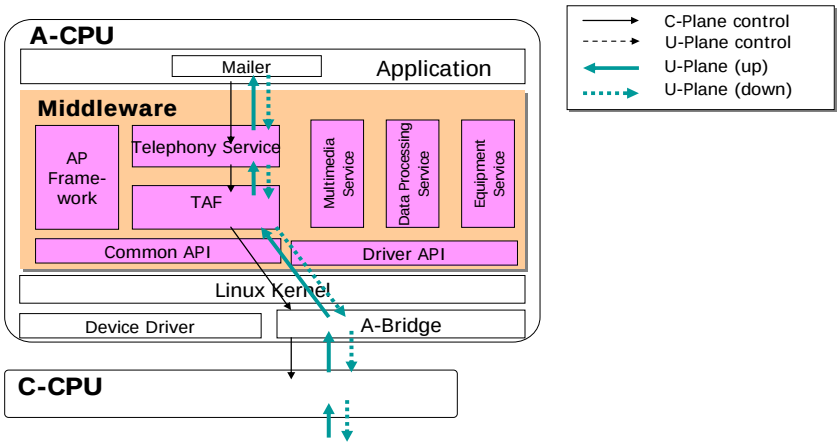


**Figure 11 SMS Communication**

Deleted: 10



 