Linux-based Mobile Phone Middleware

Application Programming Interface

Preface and Common Types

Document: CELF_MPP_Preface_FR4_20061103

Deleted: Linux based 3G Specification

Deleted: Multimedia Mobile Phone API

Deleted: FR2_20060606

Formatted: English (U.S.)

DRAFT

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

© 2005, 2006 CE Linux Forum. All rights reserved.

Revision History

Revision	Comment	Reviewer	Editor	Date
2.2	Prepared for Tamagawa meeting		NEC, Panasonic	2005.9.28
2.2.1	Prepared for San Francisco meeting, new format		Scott Preece	2005.10.31
2.2.3	Integrated issue resolutions from Reference Architecture and added Common Types and Programming Model sections.		Scott Preece	2005.12.20
2.2.4	Updated to clarify that Application ID and Client ID are the same thing.		Scott Preece	2006.2.1
FR1	First version for Formal Review		Scott Preece	2006.3.1
FR2	Revised to resolve first group of Formal Review comments		Scott Preece	2006.6.6
FR3	Revisions from editors' meeting		Scott Preece	2006.7.6
FR4	Revisions from July meeting, new AppID API		Scott Preece	2006.11.3

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

Formatted: French (France)
Field Code Changed
Formatted: French (France)
Field Code Changed
Formatted: French (France)
Field Code Changed
Formatted: French (France)
Field Code Changed
Formatted: French (France)

Deleted: ¶
¶
0. Introduction 4¶
0.1.1 Circuit Switched Communication Service 4¶
0.1.2 Packet Switched Communication Service 4¶
0.1.3 Short Message Service 4¶
0.1.4 Equipment Service 4¶
0.2 Structure of API Documents 4¶
0.2.1 Introduction Section 4¶
0.2.2 Primitives Section 4¶
0.2.3 Functions Section 4¶
0.3 Terminology and abbreviations 5¶
0.4 References 9¶
0.4.1 Normative 9¶
0.4.2 Informative 9¶
1. Programming Model 10¶
1.1 Events and Notifications 10¶
1.1.1 Spontaneous Events 10¶
1.1.2 Result Events 10¶
1.1.3 Application IDs 10¶
1.1.4 Callback Notification Functions 10¶
1.1.5 Registering 11¶
1.2 Synchronous Service Interfaces 11¶
1.3 Asynchronous Service Interfaces 11¶
2. Common Primitives 12¶
2.1 Constants 12¶
2.2 Enums 12¶
2.2.1 CelfMpStatus 12¶
2.3 Data Types and Structures 12¶
2.3.1 CelfMpCallRef 12¶
2.3.2 CelfMpEvent 12¶
2.3.3 CelfMpCallback 13¶
2.3.4 CelfMpAppId 13¶

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

0. Introduction

This Preface to the CELF Mobile Phone API describes the overall structure of the API specification of the Telephony Service for 3G multimedia mobile telephone based on Linux. It also provides an introduction to some common concepts and terminology used in the Specifications and defines some common datatypes.

This document is the work of the CE Linux Forum's Mobile Phone Profile Working Group [MPPWG].

The major sections of the API in the current release are described below. This Preface and the individual Service chapters are considered normative; a conforming implementation is required to satisfy statements written in "SHALL" form. However, conformance may be determined on a service-by-service basis.

0.1.1 Circuit Switched Communication Service

The Circuit Switched Communication Service (CS Service) API [CS] provides access to functionality for call control, call state management, tone control, and log processing. This chapter includes the Voice communication service, the Video communication service, and the Unrestricted Digital data Communication service.

0.1.2 Packet Switched Communication Service

The Packet Switched Communication Service (PS Service) API [PS] provides access to functionality for packet call control and for sending and receiving data packets. This chapter includes the PPP dial-up communication service and the IP connection data transfer service.

0.1.3 Reference Architecture

The Reference Architecture [RefArch] is an illustrative, non-normative description of a commonly understood way of implementing mobile handsets using a Linux-based application environment. A conforming implementation is not required to conform to the reference architecture.

0.2 Structure of API Documents

Each specification chapter defines the API for a major sub-area of functionality. The content of each chapter is divided into:

1. Introduction – An overview of the service, placing it in context.
2. Primitives – Definitions of the data types, constants, and enumerations used in the API definitions.
3. Functions – Definitions of the individual functional interfaces provided by the service.

0.2.1 Introduction Section

An introduction to the functionality available through the API of the service described by the chapter.

0.2.2 Primitives Section

This section is subdivided into sub-sections for Data Types and Structures and for Constants. In each case, the primitive is named, its use is described, and its formal definition (as would appear in a header file) is given.

Note that this is a source-level specification. In many cases the value of constants and enum elements is not defined by the specification. In these cases it is expected that applications would need to be compiled with header files specific to a particular implementation, which would define those values.

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

0.2.3 Functions Section

Each function appears as a separate section. The information given for each function includes:

Symbol The formal (programming) name of the function.

Syntax Syntax used in programming in C language

Argument Arguments of API function in C language

Return value Return value of API function in C language

Include file File name to be included in Programming

Functional description Definition and detail explanation of API function

0.3 Terminology and abbreviations

The following words, phrases, and acronyms have specific meanings within the context of the API.

word	explanation
32K AV communication	Communication mode with AV at the speed of 32Kbps
32K data communication	Data communication mode at a stable communication speed of 32Kbps. Unlimited digital 32K communication.
64K AV communication	Communication mode with AV at the speed of 64Kbps
64K data communication	Data communication mode at a stable communication speed of 64Kbps. Unlimited digital 64K communication.
accumulated reset	Resetting of the accumulated duration data. The handset stores data on the total duration of all calls .
API	Application Program Interface
APN	Access Point Name
App or Application	Application program; a program run in user space.
ASF	Advanced Streaming Format
automatic incoming call	Operating mode in which the handset automatically accepts incoming calls, without the user accepting each call by a manual operation..
automatic transmission	Placing a call by keying in all the digits and then initiating the connection. Same as “on-hook originating”.
call duration	The duration of a voice call.
call quality alarm	The indication that radio reception from the network has deteriorated and the call is likely to be dropped.

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

call reference	An identifier for a particular call. This identifier is assigned by the network or mobile phone, and used in the call-management APIs to operate on a particular.
C Plane	Control Plane – the subset of components in a network architecture that are responsible for controlling connections.
CS	Circuit Switched operation; a mode of communication in which a dedicated channel is maintained between the handset and the remote party and the call content is routed over that identified channel.
DCF	Device Control Function. The module that provides the following functions: ・Mobile phone control via AT commands. ・Monitoring S-IF eassage and notice status change event to service. ・To notice MTF (block which exchange message between TAF-NW) when sets up receive denial.
DTMF	Dual Tone Multi Frequency. The tones generated to correspond to key presses while a CS connection is open (off-hook originating). On digital connections, the tones may be represented by designated codes rather than encoded audio.
Earphone (external option)	Controls whether audio is routed to an attached earphone (headset) or to a built-in loudspeaker.
emergency originating restriction	A network condition in which call from handsets are not accepted because an emergency requires all of the available network capacity..
Engine	Application Engine; a software module providing “backend” processing to support a service interface.
external AV communication	Videophone communication using a USB connection cable, etc., to connect terminal and external equipment (such as a PC㊍personal computer) to the handset).
FLASH	Macromedia Flash Player; the engine that execute Flash programs.
H234 and H324M	3G multimedia services – H324M is video telephony, H234 is encryption key management
high priority communication mode	The display mode in which an alert or icon is displayed in case of (a) an incoming packet switched communication when circuit switched communication is active, or (b) an incoming circuit switched communication when packet switched communication is active.
hold tone	A tone or melody that sounds when a voice call or AV call is

Deleted: m

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

	changed to hold status.
HTTP	Hyper Text Transfer Protocol
I/F	Interface
IMEI	IMEI (International Mobile Station Equipment Identity). A unique number allocated to each individual mobile station (handset).
internal AV communication	Videophone communication between terminals.
IR	Infra-Red
JAM	Java Application Manager
JVM	Java Virtual Machine
Kernel	Linux Kernel
keypad dial lock	When this function is set, the handset does not allow voice or videophone calls by dialing phone numbers, extension number, or SIP. Dialing from previously stored "Phonebook" entries and from the "Dialed calls" or "Redial" entries remains possible.
LCD	Liquid Crystal Display
Low-voltage alarm	The alarm sounded to indicate that the battery is is about to run out of power.
manner mode	Manner mode provides a quick and convenient way of muting the terminal's ring tones and keypad sound to avoid disturbing people around you.
manual transmission	Same as off-hook originating
MAW	Monitoring and Watching
MPPWG	The CE Linux Forum's Mobile Phone Profile Working Group, which defined this specification and the related Service specifications.
MSB	Mobile Software Bus
multiple calls	It is the combination of maximum of three calls. The conversation, hold and incoming call is at most one call.
noise canceller	A function that reduces ambient transmitted over a connection so that the other party can hear the voice more clearly.
normal originating restriction	When this mode is set, outgoing calls are permitted only to designated special numbers.

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

number notification	On option that determines whether the handset's telephone number is sent to the other party when a call is initiated.
OBEX	Object Exchange protocol
OCR	Optical Character Recognition
off-hook originating	Placing a call by keying the digits after pressing the start button; when five seconds have elapsed since the last input digit the call is initiated.
on-hook originating	Placing a call by pressing the start button after inputting all dial digits.
out-of-communication area	The mode of operation when the handset is unable to establish communication with the network because it is out of the service area or the signal is too weak or there is no network with which the handset is allowed to register.
phone-answering message	A message sent to the calling party when the handset can not respond to an incoming call.
phone-answering message service	A network-side service that provides for recording messages from callers when the handset is not in service..
PIM lock	A handset mode in which the user has indicated that no access is allowed to personal-information resources, such as "Phonebook", "Schedule", "Mail", "Messenger", and "Presence".
PIN	Personal Identification Number
PS	Packet Switched network
receive level	The receive level is the strength of the radio signal received from the network.
reconnection tone	The tone that sounds when the handset reconnects to the network after being out of service.
SCA	Stream Control API
SD	SD memory card
SDFS	SD File System
secret mode	A handset mode that controls whether personal information resource display or hide those entries that have been marked by the user as secret.
SMS	Short Message Service
special number	A number to connect with a service center in the network.

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

SS	A Supplementary network service accessible using the SS protocol, which encodes a service code as a four-part data string starting with '*', '#' or "*#" and ending with '#'. The service code is either a standardized 3GPP code or a code defined by operator (USSD).
SSL	Secure Socket Layer
supplementary service	An optional service provided by the network and available to the handset through special signalling.
TAF	Terminal Adaptation Function. The module that connects handset functions to communication services.
U Plane	User Plane – the subset of components within a network architecture that are responsible for the transfer of user data.
UIM	Same as USIM (Universal Subscriber Identity Module).The removable hardware module that contains information identifying a network account plus various kinds of user-defined information (phone book entries, messages, service-specific information, applications, etc.).
USSD	Unstructured Supplementary Service Data a network- specific supplementary service code.
WDC	Watching Device Condition
within-communication area	The condition when the handset is in service area and able to communicate with the network.

0.4 References

0.4.1 Normative

The following documents are available at [MPPWG].

[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*

[RefArch] CE Linux Forum, *Linux-based Mobile Phone Middleware Application Programming Interface – Reference Architecture*

0.4.2 Informative

[MPPWG] CE Linux Forum Mobile Phone Profile Working Group, http://tree.celinuxforum.org/CelfPubWiki/MobilePhoneProfileWorkingGroup.

Formatted: Normal

Formatted: Font: Italic

Deleted: This document.

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

1. Programming Model

The MPP API defines both synchronous and asynchronous interfaces. Synchronous interfaces return a result directly to the calling program, whose execution is blocked until the function returns. Asynchronous interfaces return a result directly, but the result indicates only whether the request was successfully initiated. The actual result of an asynchronous service request is received as an event notification sometime after the request has been made. Asynchronous operations are used when the delay involved in processing a request is likely to be too long for the client to block and be unable to do other work.

1.1 Events and Notifications

MPP API clients can register with service providers to receive notification when specified events occur. The API implementation delivers notifications by calling a function (the callback function) specified by the client at the time the client registered the request for notification. Registration is persistent; a client remains registered until it explicitly unregisters or exits.

The event-delivery model is widely used throughout the interface, both for delivering the results of asynchronous service requests ("result events") and for notifying clients of events that occur in the system ("spontaneous events"). For instance, a client can register to receive notification when an incoming call arrives from the network.

1.1.1 Spontaneous Events

Spontaneous events are the means by which clients become aware of activities of the network or of anomalous situations in the device (such as low battery conditions). Spontaneous-event notifications are multicast: when a spontaneous event occurs, the server implementation calls the notification callback functions of all clients that have registered for notification of the given event. While multiple clients may register for notification of a given event, each may register only one callback for that event – each registration replaces any previous registration by that client.

1.1.2 Result Events

Result events are the means by which clients receive the results of asynchronous operations. When the server completes processing of an asynchronous service request, it calls the notification callback function most recently registered for that event by the client. The client may register only one callback for a given result; each registration replaces any previous registration. Result Events are delivered only to the application that requested the service.

1.1.3 Application IDs

Comment [sep1]: 26 Instead of an application ID, use an dynamically assigned handle. DEFERRED - this is an area where we will probably change the model.

In order for events to be delivered to the right client, each client provides an application ID to the server when it registers for notifications. The ID is a unique integer value associated with each application or server that needs to receive events. No special semantics are associated with the value, but it must be unique for each client.

1.1.4 Callback Notification Functions

When an event occurs, the service implementation calls the callback notification function(s) registered for that event. The function is called with one argument, a pointer to a CelfMpEvent structure, which contains a fixed part with members that identify the type and subtype of the event and an open part that contains data fields appropriate to the specific event type.

The function is called in the process context of the client, so the client's internal namespace is available in writing the function. The method by which the system arranges for the process to be called in the client's context is outside the scope of the API definition.

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

1.1.5 Registering

A client requests notification of particular events by calling a registration function (which usually has a name that starts with “start” and ends with “notification”), providing a application ID, event mask, and call back function pointer as arguments. The event mask indicates which of the events provided by the particular service the client is requesting notification for. There is a separate notification_..._start() function for each cluster of services in the MPP API; for instance, the SMS service has a registration service separate from the packet-switched communications service.

Comment [sep2]: What happens if app crashes and service then tries to deliver an event? Need more clarification of semantics of app IDs.

1.2 Synchronous Service Interfaces

The processing of a synchronous request looks to the client like any other normal function call. The implementation may do special processing to pass associated data between the client’s process context and the service implementation’s process context, but that is outside the definition of the API. The client process is blocked during the processing of the request and resumes execution with the assignment of the provided result into the given variable (if appropriate).

1.3 Asynchronous Service Interfaces

Comment [sep3]: 15 All of the asynchronous APIs specified in the various API specs must include a mechanism to cancel (outstanding) requests. DEFERRED – this is expected to change.

To use an asynchronous service, the client must first call an interface to register to receive the notifications associated with the service to be requested. The registration request would include a list of the events requested and the callback function the server should call when the given events occur. A client may register different callbacks for different events provided by the same service.

When the client makes an asynchronous request, it receives a result from that function call that indicates whether the server accepted the request successfully. The client can then continue doing whatever processing it has to do or can block waiting for the result to come back through a call to one of the callback notification functions that it has registered.

When an event occurs (either completion of a service request or a spontaneous event), the MPP server will check to see whether any clients are registered for that event and, if so, will arrange for the callback notification functions that those clients registered against the event to be called in the application process context.

1.4 Memory Management Model

Comment [sep4]: Need to define the ownership of memory TOOK A STAB – NEEDS TO BE VERIFIED.

Formatted: Heading 2

Applications are responsible for allocating memory for their needs and for releasing it when it is no longer needed. The API specification indicates, for each specific service request, whether memory for results is provided by the client or by the service and which is responsible for freeing it.

The service implementation allocates space for data carried with callback function calls (the data field of the event structure) and frees it when the callback function returns. The callback function must copy any data the application needs to retain into memory owned by the application, before the callback returns.

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

2. Common Primitives

This section documents data types and values used throughout the sections of the API specification.

2.1 Constants

2.2 Enums

2.2.1 CelfMpStatus

Description: Status returned by MPP API functions

Definition:

CELF_MP_STATUS_OK: Successful completion

CELF_MP_STATUS_APP_ID_ERR: Invalid Application ID

CELF_MP_STATUS_EVENT_SET_ERR: The set of event is invalid

The following status return is common to all call-related interfaces. It indicates that the call reference argument did not match an open call. See [CS] and [PS] for more details:

CELF_MP_STATUS_CALL_REF_ERR: Call reference argument is invalid

Deleted: Invalid Call reference

The following constants with PS in their names are Packet-Switched Communication Service status returns. See [PS] for more information

CELF_MP_STATUS_PS_PDP_TYPE_ERR: Unsupported PDP type

CELF_MP_STATUS_PS_DENIED: Request rejected by network due to no subscription to packet communication service

CELF_MP_STATUS_ERR: Other error

2.2.2 CelfMpEventCategory

Formatted: Bullets and Numbering

Description: Category associated with a particular event. The set of categories is the union of the categories defined by the different services.

Definition: An UINT32 enum.

CELF_MP_EVENT_CATEGORY_VOICE_NOTIFY Event defined by [CS]

Formatted: Normal, Indent: Left: 0.38"

CELF_MP_EVENT_CATEGORY_PACKET_NOTIFY Event defined by [PS]

Formatted: Indent: Left: 0"

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

2.2.3 CelfMpEventSubtype

Description: Subtype information associated with a particular event category. The set of values is the union of the values defined by the different services.

Definition: An UINT32 enum. Details of use of these values are in [CS] and [PS] as indicated by the name of the value.

CELF_MP_EVENT_SUBTYPE_CS_CONN_INFO

CELF_MP_EVENT_SUBTYPE_CS_TEL_CALL_TIME

CELF_MP_EVENT_SUBTYPE_CS_DISC_CAUSE

CELF_MP_EVENT_SUBTYPE_CS_FW_RESULT

CELF_MP_EVENT_SUBTYPE_CS_OFFHK_TRN

CELF_MP_EVENT_SUBTYPE_CS_DCF_EVENT_TYPE

CELF_MP_EVENT_SUBTYPE_CS_AREA_INFO

CELF_MP_EVENT_SUBTYPE_CS_RSSI_LEVEL

CELF_MP_EVENT_SUBTYPE_PS_CALL_STATE

CELF_MP_EVENT_SUBTYPE_PS_SERVICE_STATE

CELF_MP_EVENT_SUBTYPE_PS_CALL_DATA

CELF_MP_EVENT_SUBTYPE_PS_APN_INITIALIZATION

2.3 Data Types and Structures

2.3.1 CelfMpCallRef

Description: Call Reference for the current call. Service requests that establish calls return a unique Call Reference value identifying that call.

Definition: `typedef unsigned char CelfMpCallRef;`

2.3.2 CelfMpEventInfo

Description: Event information associated with a particular event category. The values in this field are data values specific the service sending the event.

Definition: `typedef INT32 CelfMpEventInfo;`

2.3.3 CelfMpEventSubinfo

Description: Event additional information associated with a particular event category. The values in this field are data values specific the service sending the event.

Definition: `typedef INT32 CelfMpEventSubinfo;`

Formatted: Bullets and Numbering

Formatted: Normal, Indent: Left: 0.38"

Deleted: the

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

2.3.4 CelfMpEvent

Description: MPP notification events structure. This defines the common structure of the data passed with an event. The details of the use of the fields of the structure are specific to the individual services and are not common. The data field is used to define the base of a memory area containing data specific to the event; its size is defined by the provider of the event.

Definition:

```
typedef struct {
        CelfMpEventCategory   category ;
        CelfMpEventSubtype    subtype ;
        CelfMpEventInfo       info ;
        CelfMpEventSubinfo    subinfo ;
        UINT8                 data[];
} CelfMpEvent;
```

2.3.5 CelfMpCallback

Description: Pointer to a callback function

Definition: typedef void (* CelfMpCallback)(CelfMpEvent *);

2.3.6 CelfMpAppId

Description: Application ID

Definition: typedef UINT32 CelfMpAppId;

Formatted: Bullets and Numbering

Formatted: Font: (Default) Arial

Deleted: int

Deleted: int

Deleted: int

Deleted: int

Deleted: union {¶
...¶
messages structures¶
...¶
} data ;

Formatted: Bullets and Numbering

Comment [sep5]: What is the signature for the callback functions?

Formatted: Font: (Default) Arial

Formatted: Bullets and Numbering

Formatted: Font: (Default) Arial

Deleted: int

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.

3. Functions

This section contains the definitions of the functions of the service interface to this service.

3.1 Symbol: celf_mp_get_app_id

3.1.1 Syntax

CelfMpAppId celf_mp_get_app_id()

3.1.2 Argument

None.

3.1.3 Return Value

Type: CelfMpAppId

I/O:

Description: An opaque value used to identify the client application in service calls.

3.1.4 Include File

/usr/include/celf/mp_common.h

3.1.5 Functional Description

Returns an application identifier to be used in calls to the functions of the Mobile Phone API. The identifier is an opaque value used by the services of the API.

An implementation may provide other mechanisms by which a client application may obtain such an identifier.

Formatted: Bullets and Numbering
Formatted: Bullets and Numbering
Formatted: Normal Indent
Formatted: Bullets and Numbering
Formatted: Normal
Formatted: Bullets and Numbering
Formatted: Font: Bold
Formatted: Normal Indent
Formatted: Font: Bold
Formatted: Bullets and Numbering
Formatted: Normal Indent, Line spacing: single
Formatted: Bullets and Numbering
Formatted: Normal Indent

DRAFT

© 2005, 2006 CE Linux Forum. All rights reserved.