|  |  |
| --- | --- |
|  | **2010** |
|  | Evandro Lima Filho |

|  |
| --- |
| **[arquitetura de um android™]** |
| Este documento tem como objetivo esclarecer a arquitetura do core do ANDROID™ |

Sumário

[Visão Geral 2](#_Toc272453734)

[Camada Drivers 2](#_Toc272453735)

[Multimedia 2](#_Toc272453736)

[Power Management 2](#_Toc272453737)

[Telephony 2](#_Toc272453738)

# Visão Geral

[Apresentar uma visão geral da arquitetura]

# Camada Drivers

[Apresentar a camada de Drivers: libhardware]

# ****Multimedia****

[Apresentar a camada Multimedia]

# Power Management

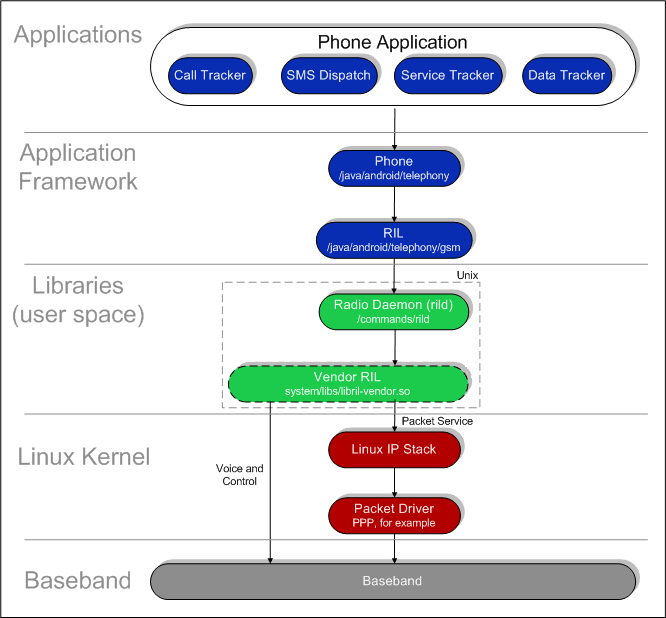
[Apresentar a camada de Gerenciamento de Energia]

# Telephony

## Introduction

Android's Radio Interface Layer (RIL) provides an abstraction layer between Android telephony services ([android.telephony](http://code.google.com/android/reference/android/telephony/package-descr.html)) and radio hardware. The RIL is radio agnostic, and includes support for Global System for Mobile communication (GSM)-based radios.

The diagram below illustrates the RIL in the context of Android's Telephony system architecture.



Solid elements represent Android blocks and dashed elements represent partner-specific proprietary blocks.

The RIL consists of two primary components:

* **RIL Daemon**: The RIL daemon initializes the Vendor RIL, processes all communication from Android telephony services, and dispatches calls to the Vendor RIL as solicited commands.
* **Vendor RIL**: The radio-specific Vendor RIL of ril.h that processes all communication with radio hardware and dispatches calls to the RIL Daemon (rild) through unsolicited commands.

## RIL Initialization

Android initializes the telephony stack and the Vendor RIL at startup as described in the sequence below:

1. RIL daemon reads rild.lib path and rild.libargs system properties to determine the Vendor RIL library to use and any initialization arguments to provide to the Vendor RIL
2. RIL daemon loads the Vendor RIL library and calls RIL\_Init to initialize the RIL and obtain a reference to RIL functions
3. RIL daemon calls RIL\_register on the Android telephony stack, providing a reference to the Vendor RIL functions

See the RIL Daemon source code at //device/commands/rild/rild.c for details.

## RIL Interaction

There are two forms of communication that the RIL handles:

* Solicited commands: Solicited commands originated by RIL lib, such as DIAL and HANGUP.
* Unsolicited responses: Unsolicited responses that originate from the baseband, such as CALL\_STATE\_CHANGED and NEW\_SMS.

### Solicited

The following snippet illustrates the interface for solicited commands:

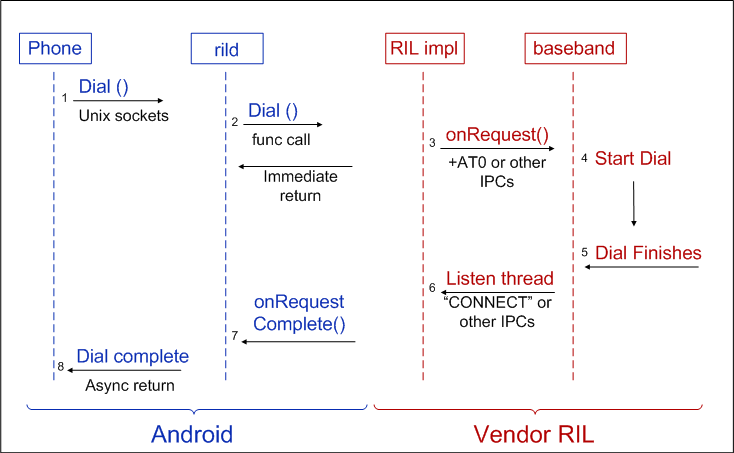
void OnRequest (int request\_id, void \*data, size\_t datalen, RIL\_Token t);

void OnRequestComplete (RIL\_Token t, RIL\_Error e, void \*response, size\_t responselen);

There are over sixty solicited commands grouped by the following families:

* SIM PIN, IO, and IMSI/IMEI (11)
* Call status and handling (dial, answer, mute…) (16)
* Network status query (4)
* Network setting (barring, forwarding, selection…) (12)
* SMS (3)
* PDP connection (4)
* Power and reset (2)
* Supplementary Services (5)
* Vendor defined and support (4)

The following diagram illustrates a solicited call in Android.



### Unsolicited

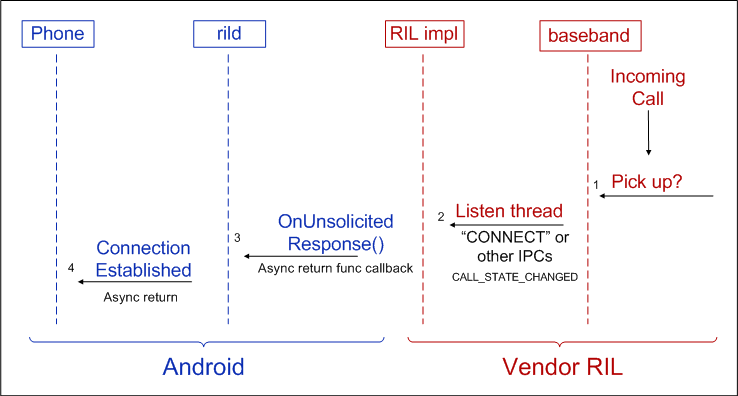
The following snippet illustrates the interface for unsolicited commands:

void OnUnsolicitedResponse (int unsolResponse, void \*data, size\_t datalen);

There are over ten unsolicited commands grouped by the following families:

* Network status changed (4)
* New SMS notify (3)
* New USSD notify (2)
* Signal strength or time changed (2)

The following diagram illustrates an unsolicited call in Android.



## Implementing the RIL

To implement a radio-specific RIL, create a shared library that implements a set of functions required by Android to process radio requests. The required functions are defined in the RIL header (/include/telephony/ril.h).

The Android radio interface is radio-agnostic and the Vendor RIL can use any protocol to communicate with the radio. Android provides a reference Vendor RIL, using the Hayes AT command set, that you can use as a quick start for telephony testing and a guide for commercial vendor RILs. The source code for the reference RIL is found at /commands/reference-ril/.

Compile your Vendor RIL as a shared library using the convention libril-<companyname>-<RIL version>.so, for example, libril-acme-124.so, where:

* **libril**: all vendor RIL implementations start with 'libril'
* **<companyname>**: a company-specific abbreviation
* **<RIL version>**: RIL version number
* **so**: file extension

### RIL\_Init

Your Vendor RIL must define a RIL\_Init function that provides a handle to the functions which will process all radio requests. RIL\_Init will be called by the Android RIL Daemon at boot time to initialize the RIL.

RIL\_RadioFunctions \*RIL\_Init (RIL\_Env\* env, int argc, char \*\*argv);

RIL\_Init should return a RIL\_RadioFunctions structure containing the handles to the radio functions:

type structure {

int RIL\_version;

RIL\_RequestFunc onRequest;

RIL\_RadioStateRequest onStateRequest;

RIL\_Supports supports;

RIL\_Cancel onCancel;

RIL\_GetVersion getVersion;

}

RIL\_RadioFunctions;

## RIL Functions

ril.h defines RIL states and variables, such as RIL\_UNSOL\_STK\_CALL\_SETUP, RIL\_SIM\_READY, RIL\_SIM\_NOT\_READY, as well as the functions described in the tables below. Skim the header file (/device/include/telephony/ril.h) for details.

### RIL Solicited Command Requests

The vendor RIL must provide the functions described in the table below to handle solicited commands. The RIL solicited command request types are defined in ril.h with the RIL\_REQUEST\_ prefix. Check the header file for details.

|  |  |
| --- | --- |
| **Name** | **Description** |
| void (\*RIL\_RequestFunc) (int request, void \*data, size\_t datalen, RIL\_Token t); | This is the RIL entry point for solicited commands and must be able to handle the various RIL solicited request types defined in ril.h with the RIL\_REQUEST\_ prefix.   * request is one of RIL\_REQUEST\_\* * data is pointer to data defined for that RIL\_REQUEST\_\* * t should be used in subsequent call to RIL\_onResponse * datalen is owned by caller, and should not be modified or freed by callee   Must be completed with a call to RIL\_onRequestComplete().  RIL\_onRequestComplete() may be called from any thread before or after this function returns. This will  always be called from the same thread, so returning here implies that the radio is ready to process another command (whether or not the previous command has completed). |
| RIL\_RadioState (\*RIL\_RadioStateRequest)(); | This function should return the current radio state synchronously. |
| int (\*RIL\_Supports)(int requestCode); | This function returns "1" if the specified RIL\_REQUEST code is supported and 0 if it is not. |
| void (\*RIL\_Cancel)(RIL\_Token t); | This function is used to indicate that a pending request should be canceled. This function is called from a separate thread--not the thread that calls RIL\_RequestFunc.  On cancel, the callee should do its best to abandon the request and call RIL\_onRequestComplete with RIL\_Errno CANCELLED at some later point.  Subsequent calls to RIL\_onRequestComplete for this request with other results will be tolerated but ignored (that is, it is valid to ignore the cancellation request).  RIL\_Cancel calls should return immediately and not wait for cancellation. |
| const char \* (\*RIL\_GetVersion) (void); | Return a version string for your Vendor RIL |

The vendor RIL uses the following callback methods to communicate back to the Android RIL daemon.

|  |  |
| --- | --- |
| **Name** | **Description** |
| void RIL\_onRequestComplete(RIL\_Token t, RIL\_Errno e, void \*response, size\_t responselen); | * t is parameter passed in on previous call to RIL\_Notification routine. * If e != SUCCESS, then response can be null and is ignored * response is owned by caller, and should not be modified or freed by callee * RIL\_onRequestComplete will return as soon as possible |
| void RIL\_requestTimedCallback (RIL\_TimedCallback callback, void \*param, const struct timeval \*relativeTime); | Call user-specified callback function on the same thread that RIL\_RequestFunc is called. If relativeTime is specified, then it specifies a relative time value at which the callback is invoked. If relativeTime is NULL or points to a 0-filled structure, the callback will be invoked as soon as possible. |

### RIL Unsolicited Commands

The functions listed in the table below are call-back functions used by the Vendor RIL to invoke unsolicited commands on the Android platform. See ril.h for details.

|  |  |
| --- | --- |
| **Name** | **Description** |
| void RIL\_onUnsolicitedResponse(int unsolResponse, const void \*data, size\_t datalen); | * unsolResponse is one of RIL\_UNSOL\_RESPONSE\_\* * data is pointer to data defined for that RIL\_UNSOL\_RESPONSE\_\* * data is owned by caller, and should not be modified or freed by callee |