


# Rapid Radio Interface Layer (RRIL) White Paper

**Intrinsyc Software, Inc.**  
**Version 1.0**

June 14<sup>th</sup>, 2011

**Created By:**

Intrinsyc Software International, Inc.  
700 West Pender Street  
10th Floor  
Vancouver, BC  
Canada V6C 1G8  
1-(800)-474-7644  
(Fax) 1 (604) 801-6417  
[www.intrinsyc.com](http://www.intrinsyc.com)


	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

## TABLE OF CONTENTS

---

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
<b>2</b>	<b>RIL Overview .....</b>	<b>1</b>
2.1	Android .....	3
2.2	Windows Mobile .....	3
<b>3</b>	<b>RapidRIL Overview .....</b>	<b>4</b>
<b>4</b>	<b>High-Level RapidRIL Architectural Overview .....</b>	<b>4</b>
4.1	OS Interface .....	5
4.2	Core Engine .....	6
4.3	OEM Extensions .....	7
4.4	RIL Utilities .....	8
<b>5</b>	<b>RapidRIL special features .....</b>	<b>9</b>
5.1	Soft Reboot .....	9
5.2	NITZ .....	9
5.3	Multiple PDP Context .....	9
5.4	SIM Application Toolkit .....	10
5.5	Power Management .....	10
5.6	Audio Management .....	10
<b>6</b>	<b>Advantages of RapidRIL .....</b>	<b>10</b>

---

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

## 1 Introduction

---

In today's marketplace there are an increasing number of connected devices, such as Smartphones, Tablets and eBook readers. By definition these connected devices have a large number of applications accessing the radio hardware, such as the dialer, contacts application, browser and messaging applications. Customizing this large set of applications to interface with multiple variants of radio hardware interfaces would be a daunting task. To address this problem, connected devices implement a virtual radio interface used to access the radio hardware; this virtual radio interface is usually called a Radio Interface Layer (RIL).

The primary responsibility of a RIL is to respond to OS API requests and translate those requests into modem specific commands to execute the requests. While doing so the RIL is responsible to handle events from the cellular telephony radio equipment and alert the OS if necessary. Additionally, a RIL will manage tertiary activities such as ensuring that the radio health is monitored or managing power requirements.

The RIL is a complex component, a typical sample RIL from Microsoft or the Open Handset Alliance (OHA) can contain up to 200,000 lines of code. For connected devices which require feature enhancements to support field testing, diagnostics, or carrier dictated requirements, the RIL may end up with more than half a million lines of code – a significant development effort that may require six to nine months to complete.

## 2 RIL Overview


---

The RIL is a key component of connected devices that enables wireless voice and data applications to seamlessly communicate with GSM/UMTS or CDMA modems. The RIL offers the telephony applications, like Dialer, Messaging, Browser and 3<sup>rd</sup> party applications, a common interface to access the radio hardware, thus isolating the radio hardware from the rest of the system, allowing for code re-use shorter time to market and product differentiation.

The RIL services system requests for radio functions, e.g. voice, data, SMS, etc. and provides asynchronous notifications of changes to the system, e.g. signal strength, coverage, incoming voice calls, incoming SMS messages.

The RIL is architected such that it provides a standard interface, API set and call back mechanisms which can be used across different platforms, with the specific details being hidden from the remainder of the system.

Access to the RIL is flow controlled by a thin layer sitting on top of the RIL, called RIL proxy or RIL daemon. This RIL proxy provides the APIs that telephony applications use to access the radio hardware through the RIL, and provides arbitration between the multiple clients for access to the single RIL driver. The RIL proxy is provided as part of the OS and does not require modification by OEMs.

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11


The diagram below shows a high level architectural view of how the RIL fits in the OS:



As can be seen from this diagram all interaction between the Telephony applications and the radio stack is via the RIL with the exception of a PPP connection, which initially uses the RIL to setup the connection and then bypasses the RIL path to allow direct connection to the virtual serial port connected to the modem. Please note that, although PPP connection is supported out of the box, RapidRIL can easily be configured to use raw IP or proprietary L2 drivers.

It can also be observed from the above diagram that the RIL does not communicate directly with the modem. The RIL usually interfaces with a Multiplexer - commonly following the GSM 27.010 specification, though some interfaces provide their own protocol to expose multiple logical channels. The multiplexer is a useful extension to the standard modem interface as it can provide multiple concurrent channels to the modem allowing the RIL to perform several functions simultaneously without the expensive and time consuming switch between command and data modes. The main advantage of the multiplexer is that it allows AT commands to be sent to the modem without interrupting the data stream during an active data connection, which simplifies the handling of voice calls and SMS during a data connection. The multiplexer essentially emulates a multiple serial port modem as it provides several virtual modem/COM port interfaces to the upper layers whilst using a single physical serial interface.

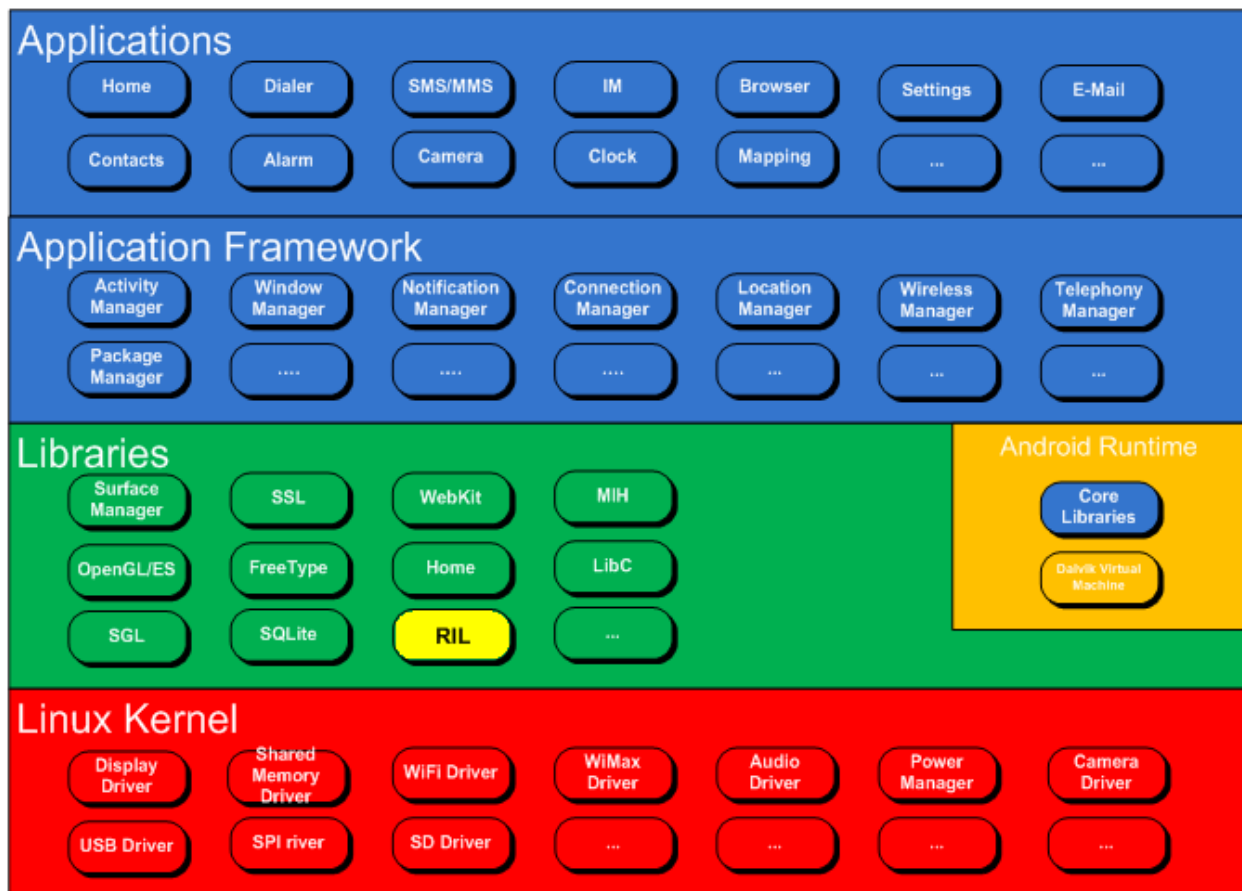
The final link to the modem is via a serial driver, the most common implementations being USB, SPI, UART and shared memory. In some implementations the USB driver offers multiple virtual end-points over a single physical end-point, in which case the multiplexer is no longer required, as the USB performs a similar role.

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

## 2.1 Android


In Android the RIL is a library (.so file). This library is loaded by the RIL Daemon, which fulfills the role of proxy as described in the previous section. The RIL provides a set of APIs as dictated by the RIL Daemon; the RIL Daemon uses these APIs to initialize the RIL and send requests to it – this forms the interface between the RIL Daemon and the RIL.

The following diagram shows how the RIL fits in Android OS (source: <http://developer.android.com/guide/basics/what-is-android.html>):

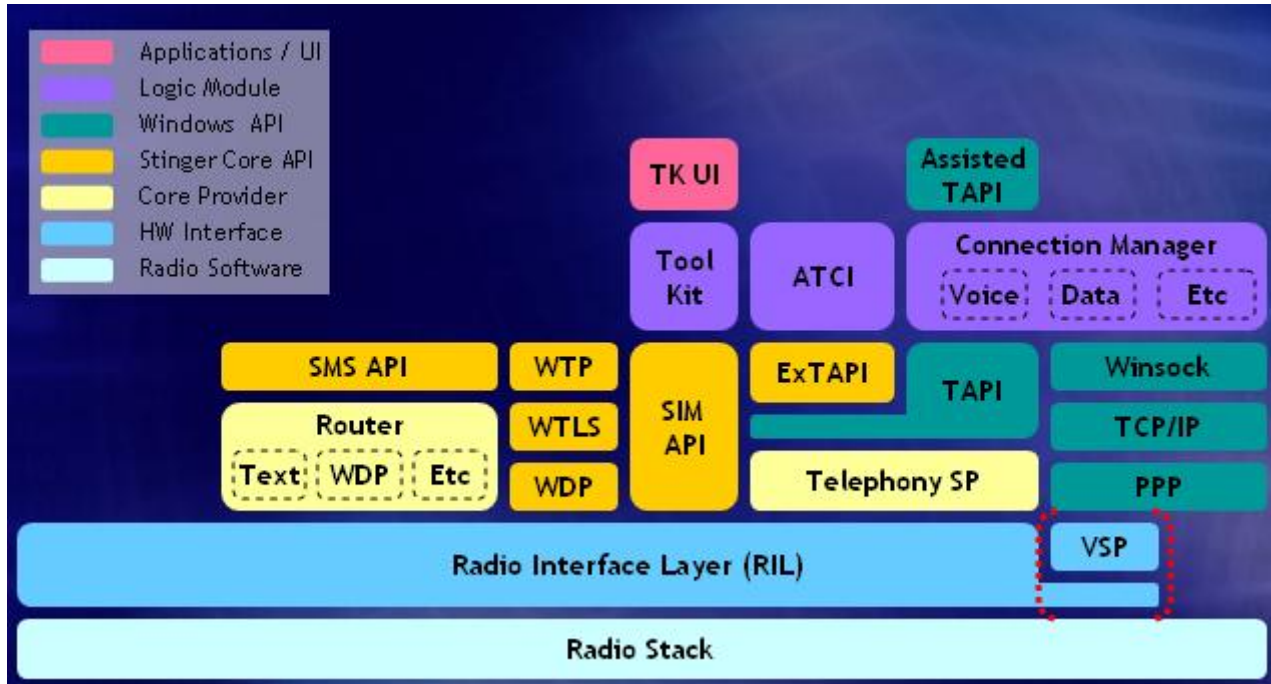


## 2.2 Windows Mobile

In Windows Mobile and Windows CE the RIL is a DLL. The RIL is loaded by the OS according to a registry key which indicates the DLL name, the DLL prefix and the loading priority. Similarly to Android, communication with the RIL in Windows Mobile and Windows CE is done through the RIL proxy. As indicated earlier, this proxy dictates the APIs to be implemented by the RIL, and which are used for the communication between the RIL proxy and the RIL.

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

The following diagram shows how the RIL fits in Windows Mobile / CE:




### 3 RapidRIL Overview

Intrinsyc’s Telephony team has successfully custom developed dozens of RIL drivers for baseband processors used in mobile phones and other connected devices. This expertise has been packed into the RapidRIL, a RIL implementation that allows developers and manufacturers of Android and Windows Mobile devices to shorten the time to market and lower development costs of their connected devices.

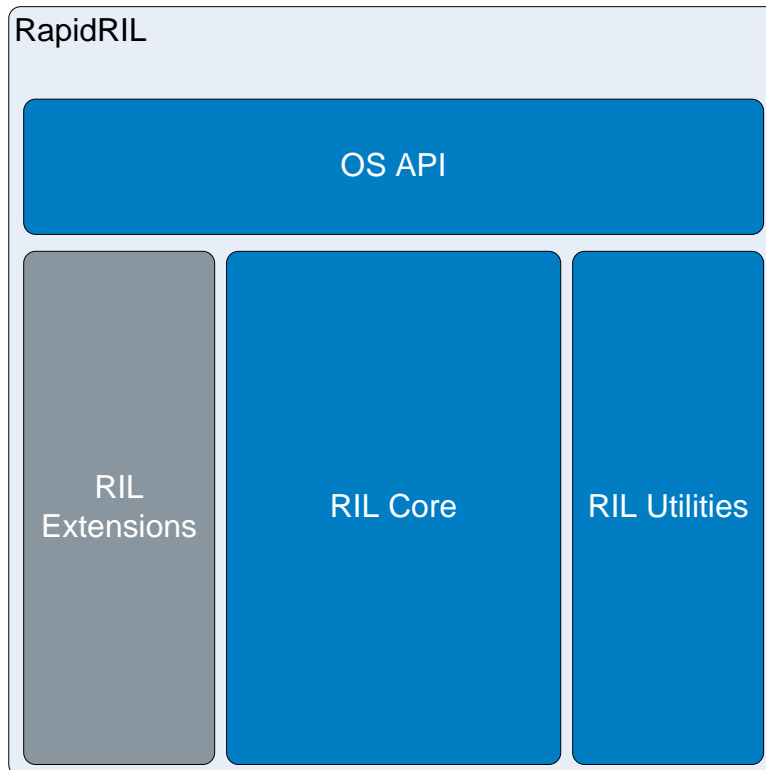
RapidRIL is available for both Android and Windows Mobile / CE OS, with some of the code being common to both platforms. RapidRIL packs a wealth of features, including multiple channel support, multi-threaded implementation, ease of customization, support for all GSM features and small footprint. All these features, and others, will be described in the following sections.

### 4 High-Level RapidRIL Architectural Overview

RapidRIL was designed from the ground-up to be multi-OS. The API set of every OS is different, and this has an impact on RapidRIL, RapidRIL’s interface to the OS will be different for every supported OS. In order to maximize code reuse, RapidRIL is divided into two main layers; the top layer is the interface to the RIL Proxy, and will largely differ across the different OS. The bottom layer of RapidRIL is the engine; the engine is shielded from the OS to a certain degree, and hence allows for parts of it to be shared across all OS.

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

Besides the proxy interface and the engine, RapidRIL comprises to additional blocks. One of those is the RIL Extensions, a means for OEMs to customize the RIL behaviour which will be described in more detail later. Another block, not truly a RIL component, is the Utilities; this block offers parsing functions and wrappers around OS APIs, to isolate the RIL from the OS and hence allow for RIL code to be OS agnostic. The diagram below shows the main blocks that compose RapidRIL, these will be described in more detail in the next sections.




## 4.1 OS Interface

The OS Interface is the top layer of RapidRIL, the entry point for the RIL requests. This layer implements the APIs as dictated by the RIL proxy.

In Android, RapidRIL is built as a library (.so file) and the proxy to interface with is the RIL Daemon. Communication between the RIL Daemon and the RIL is done through function calls. During the initialization phase, the RIL Daemon calls an initialization function in RapidRIL, which returns a set of pointers to functions; these functions are used in the RIL Daemon to send requests to the RapidRIL and to forward the responses from the modem to the upper layers.

In Windows Mobile and Windows CE RapidRIL is built as a DLL, and interfaces with the RIL proxy. Similar to Android, the communication between the RIL Proxy and RapidRIL is done through function calls. These function calls fulfill a similar role to the Android ones, i.e., allow the Proxy to send requests to the modem via the RIL, and return the response to those requests.

As can be expected, there is no correlation between the RIL Daemon in Android and the RIL Proxy in Windows Mobile / CE, therefore the OS Interface will largely differ across OS.

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

## 4.2 Core Engine

The Core Engine is the heart of RapidRIL. The Core Engine is responsible for sending commands to the radio hardware, receiving responses from the radio hardware, and maintaining the state machines.

The Core Engine has been designed to rely as little as possible on the OS that hosts it. On one side, the Core Engine is isolated from the OS by the RapidRIL OS Interface, which translates the functions required by the RIL Proxy into OS-agnostic versions implemented by the Core Engine. On the other side, the Core Engine is isolated from the OS APIs by the utilities, which effectively act as a wrapper around OS calls.


One of the key features of RapidRIL is the channels. The Core Engine in RapidRIL is composed of a series of channels, each channel being a physical or virtual interface to the modem. Each channel is responsible for a certain number of commands, usually grouped by functionality (a common split would assign one channel for each of these command groups: voice, SMS, data, SIM, Network, SIM Toolkit). Channels are loosely coupled, therefore additional channels can be easily added for specific purposes, for instance operator specific commands, or removed if not required.

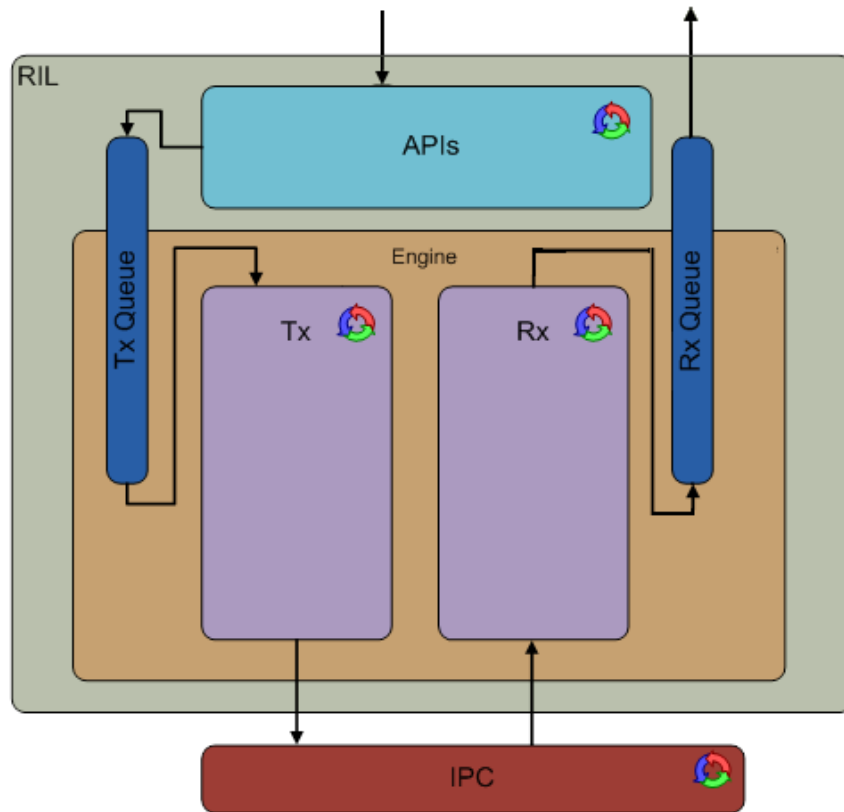
By taking advantage of the multiple channels exposed by the modem, the RapidRIL can send multiple commands to the modem simultaneously. For instance, a request to retrieve the available operator names, which may take upwards of a minute to complete, would not block further requests on other channels, meaning that voice, SMS or data requests can proceed while the long network operation is in course. This key feature of RapidRIL makes the whole system more responsive, providing a better user experience.

Furthermore, each channel in the Core Engine is multithreaded, thus decoupling the sending of a command from the reception and parsing of its response, for increased responsiveness in the RIL.

Each of RapidRIL's channels contains both a Tx and a Rx queue. As modems can have only 1 outstanding request per channel, the Core Engine is responsible for buffering the incoming requests until the outstanding one completes, and then sending the next request to the modem. The same applies on the response path, the Rx thread places the responses to the queue of the Tx thread, which processes them and pushes them to the RIL Proxy. The queues in RapidRIL are priority-based; each command placed on the queue will have a priority value assigned to it, and will jump ahead of commands already in the queue with a lower priority. This feature is important for situations where a request must be executed regardless of any previous request sent to the modem, for instance placing a call to an emergency number.

The following diagram shows a high level view of RapidRIL's Core Engine. Only one channel is represented in this diagram; for each additional channel there would be another set of Tx/Rx threads with their corresponding queues.


	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11



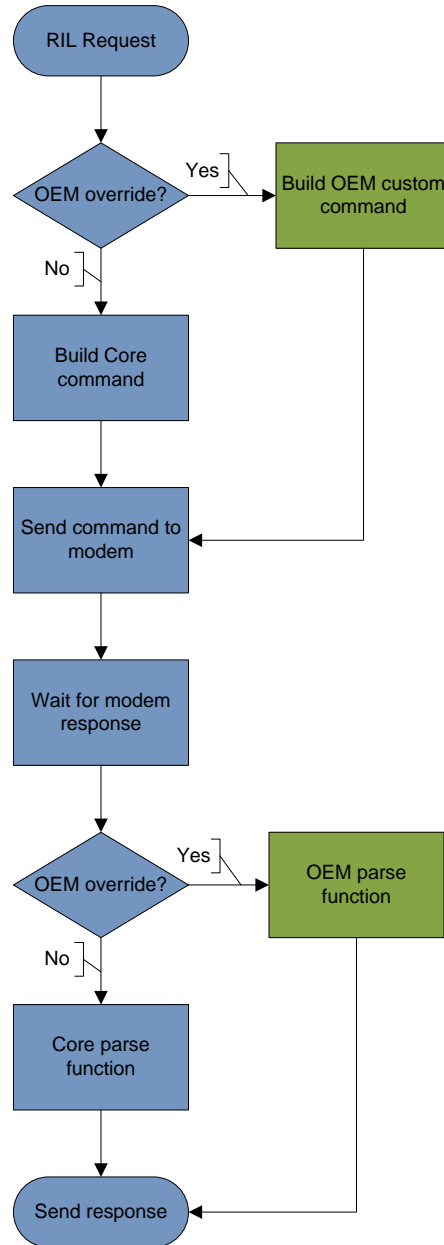
### 4.3 OEM Extensions

The OEM Extensions are another key feature of RapidRIL. This component is an extension to the RapidRIL that allows customers to modify the RIL behavior. The OEM Extensions are compiled as a separate library and are later linked against the other RapidRIL libraries. This means that it is possible to customize the RapidRIL behavior without modifying the RapidRIL core code; the core code remains common to all products, and any customizations are deferred to the OEM extensions.

For every request supported by the RapidRIL, RapidRIL defines a pair of functions, one is responsible for building the command to send to the modem to complete that request, and the other is responsible for parsing the modem response. This pair of functions can be overridden in the OEM Extensions, and will be called in place of the core ones if defined. Note that it is not necessary to override both functions, it is possible to customize only the building function or only the parsing function. An example where only the parsing function would be updated is when RapidRIL is required to execute a specific action upon reception of a modem response. The OEM override would parse the response and then execute the new action.


	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

The following chart exemplifies the role of the OEM Extensions in RapidRIL:



## 4.4 RIL Utilities

The RIL Utilities are the last major block in RapidRIL. As it was mentioned earlier, the RIL Utilities is a set of parsing functions, utilities and wrappers around OS APIs. The purpose of this component is to decouple the Core Engine as much as possible from the OS, to maximize the engine code that can be common to Android and Windows Mobile/CE.

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

## 5 RapidRIL special features

RapidRIL offers a series of additional features for added robustness and flexibility. These features are described in this section.

### 5.1 Soft Reboot

RapidRIL's Soft Reboot is a feature that monitors the radio hardware health and reacts accordingly. If a command sent to the radio hardware times out, RapidRIL notes the failure and tries the command again, transparently to the upper layers. If this command has failed a predetermined number of times (configurable), RapidRIL will reboot the radio hardware and then restore the previous state, including unlocking the SIM if it was previously locked.

To make the process as transparent as possible to the upper layers, RapidRIL queues requests received while the radio hardware is being rebooted, and sends them once the radio hardware has been restored to its previous state. Request that can complete using previously cached values are not queued, they are completed instead. In most circumstances the user would be unaware that the radio has been restarted.

### 5.2 NITZ

NITZ is a mechanism for provisioning local time and date, as well as network provider identity to mobile devices via a wireless network. As such, NITZ can be used to automatically update the system clock on connected devices.

RapidRIL supports reception of NITZ responses from the modem, and will broadcast the necessary events to notify the upper layers of the information retrieved from the NITZ response.


### 5.3 Multiple PDP Context

A PDP context is a data pipe between the mobile device and the network provider, allowing the mobile device user access to data services. PDP contexts are associated with an APN – the APN is akin to a server on the mobile network that fulfills requests from the mobile device. APNs may specify the type of service they support, for instance an APN could be used for general internet browsing, another one for MMS and a third one for video streaming.

All data-enabled devices will support at least one PDP context, but in some circumstances it is beneficial to support multiple PDP contexts. The main advantage of multiple PDP context support is that an existing connection does not need to be torn down if the user wants to momentarily connect to a different APN, providing a different kind of service. For instance, a mobile user browsing the internet on a single PDP context device would have to disconnect the existing PDP context to send an MMS; with a multiple PDP context device the MMS can be sent via a 2<sup>nd</sup> PDP context established on the MMS APN.

Windows Mobile / CE support multiple PDP contexts, and RapidRIL can be configured on Windows Mobile / CE to use up to 3 simultaneous PDP contexts.

Android, on the other hand, does not provide native support for multiple PDP contexts at the time of writing. However, RapidRIL has been architected with multiple PDP contexts in mind, and this feature can be enabled as soon as support for multiple PDP contexts is added to the Android framework.

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

## 5.4 SIM Application Toolkit

SIM Application Toolkit (STK) is a GSM standard which enables the SIM to initiate actions which can be used for various value-added services.

STK consists of a set of commands programmed into the SIM card which define how the SIM should interact directly with the outside world and initiates commands independently of the handset and the network. This enables the SIM to build up an interactive exchange between a network application and the end user and access or control access to the network. The SIM also gives commands to the handset, such as displaying menu and asking for user input.

STK has been deployed by many mobile operators around the world for many applications, often where a menu-based approach is required, such as Mobile Banking and content browsing. Designed as a single application environment, STK can be started at the initial power up of the SIM card and is especially suited to low level applications with simple user interfaces.

RapidRIL supports SIM Application Toolkit on Android and Windows Mobile / CE out of the box. Furthermore, a dedicated channel can be configured for STK commands, hence avoiding any impact on RapidRIL performance due to long running STK operations.

## 5.5 Power Management

In connected devices battery life is a premium, and it is imperative to maximize it for best user experience. In connected devices the modem is one of the heaviest battery users and the RIL, being responsible to control the modem, plays a significant role in battery life.

Both Android and Windows Mobile / CE provide APIs to control the modem, allowing the RIL to turn on and off the modem, as well as enabling and disabling power saving.

Furthermore, RapidRIL has additional features to improve battery life, including but not limited to disabling network notifications when the screen is off and placing the IPC bus into low power mode.


## 5.6 Audio Management

RapidRIL has full support for audio gains and audio muting. Although Android has a limited support for audio management and provides only the necessary APIs to query and set audio muting, RapidRIL supports also audio gains and these can be accessed via extension APIs. On Windows Mobile / CE, RapidRIL supports all the OS APIs to control audio gains and muting.

## 6 Advantages of RapidRIL

Intrinsyc's Telephony team has successfully custom developed dozens of RIL drivers for baseband processors used in mobile phones and other connected devices. This expertise has been packed into the RapidRIL, a product that allows developers and manufacturers of Android and Windows Mobile devices to

- Accelerated RIL development with out-of-the-box modem support. Today, RapidRIL has out-of-the-box support for the Infineon XMM™ Dual-Core and X-Gold and Sierra Wireless LS8XXX modems, resulting in a development cycle measured in just days. Developers can re-use one code base for multiple phones, modems and operating systems for even faster development and time to market.

	<b>Title:</b> Rapid Radio Interface Layer (RRIL) White Paper	<b>Date Originated:</b> 10-Jun-11
		<b>Revision:</b> 1.0
INTRINSYC Software, Inc.	<b>Doc #:</b> 01-RRIL-TN-001	<b>Date Revised:</b> 14-Jun-11

- Reduced development costs and fast time to market with any modem. Because RapidRIL is flexible and extensible, Intrinsic can support virtually any AT-command based or binary-interface wireless radio modem out-of-the-box without any non-recurring engineering charges.
- Support for multi-channel architecture. RapidRIL is architected with the concept of multiple channels. This encourages re-use of code to handle such regular tasks as multiple data channels – a common requirement for EDGE and UMTS technologies. As a result of this innovative design, RapidRIL is optimized for 2.5G and 3G handsets.
- Support for multiple radio devices. The same architecture that supports multiple data channels also supports multiple radio devices, or multiple SIMs. RapidRIL has been designed with the assumption that some devices will need multiple modems. This means that RapidRIL is ideal for any device manufacturer pursuing a dual-radio or dual-SIM solution.
- Improved quality and reliability. RapidRIL is a product with a common code base and specific modem extensions. Regression testing and improvements for a specific modem are reflected in common code, which improves the overall quality and reliability for all RapidRIL supported modems and across both Windows Mobile and Android.
- Innovative connected wireless devices. RapidRIL makes it easy for developers to extend RIL commands to enable innovative features, such as location-based services, advanced power management or advanced telephony features.

Compared to the reference RILs provided with the OS, RapidRIL has the following key advantages:

- The reference RIL does not support multiple channels, all commands and responses are sent through a single channel, with potential delays when a long running command is being executed.
- The reference RIL does not provide a soft reboot or any graceful error recovery mechanism; if the modem is unresponsive the user needs to restart the device to recover.
- There is no separation into a standard compliant “core” and OEM-specific functionality. The reference RIL is also polluted with TI specific commands and conditional compilation statements, resulting in increased maintenance efforts across the product lifecycle.
- The reference RIL does not provide an implementation for all the RIL APIs supported in Android, several of them are stubbed out.

RapidRIL is available for Windows Mobile / CE and Android OS. RapidRIL supports all Android OS versions, up to and including Gingerbread (2.3). For Windows Mobile / CE, RapidRIL is available for versions 6.x.