TIZEN™

面向 Tizen 平台开发人员和制造商的白皮书

目录

概述

本文档是一个指南，面向构建基于 Tizen 的设备的 Tizen 平台开发人员和制造商。本文内容基于 Tizen 2.3 Alpha 内部测试版。

本文档提供有关如何在新硬件上引导 Tizen 以及创建基于 Tizen 操作系统的产品的信息。本文通过详尽阐述 Tizen 体系结构、所需工具和开发环境设置，以及演示如何创建 Tizen 映像和执行跨不同功能领域所需的修改，详细地介绍了移植过程。

1. Tizen 简介

Tizen 是一种基于标准的平台，它为开发用于多种设备类别的应用程序提供 Web 和本机 API。Tizen 目前针对移动、可穿戴、电视、IVI 和摄像头设备，并且计划在将来扩展到更多的设备类别。

1.1 Tizen 体系结构

下图阐释了 Tizen 体系结构。

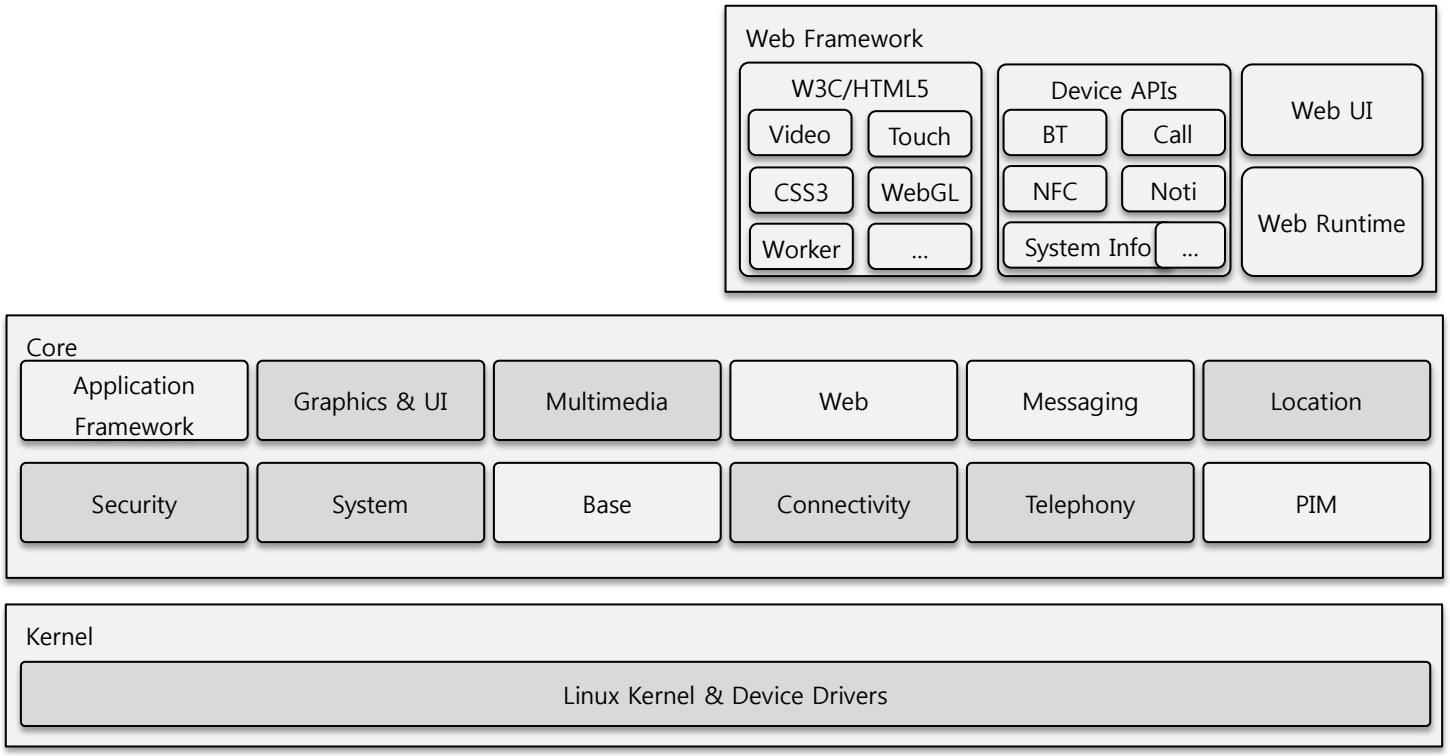

图 1. Tizen 体系结构

Application Framework（应用程序框架）

Application Framework 提供应用程序管理，包括使用程序包名称、URI 或 MIME 类型启动其他应用程序。它还启动预定义的设备，例如系统拨号器应用程序。Application Framework 向应用程序通知常见事件，例如内存不足、电池电量不足、屏幕方向变化和推送通知。

Base

Base 模块包含基于 Linux* 的基本系统库，它提供数据库支持、国际化和 XML 分析等关键功能。

Connectivity（连接性）

Connectivity 模块由所有与网络和连接性相关的功能构成，例如 3G、Wi-Fi、蓝牙、HTTP 和 NFC（近距离通信）。数据网络基于 ConnMan（连接管理器），它提供基于 3G 和 Wi-Fi 的网络连接管理。

图形和 UI

就移动方面而言，图形和 UI 模块由系统图形和 UI 堆栈构成，它们称作“本机框架”。该模块在内部利用 EFL（基本类库）来用于基于 X11 的窗口管理和电话指示器。还通过 EFL 的基本框架提供用于流畅动画的专门的控制。该本机框架具有不同的内置输入方法和 OpenGL® ES API。

该模块提供能够在完整浏览器 UI 或专用 Web Runtime（无浏览器窗口）内运行的基于 WebKit 的图形，全都基于 Tizen 自己的 HTML5 画布 WebKitEFL 实现方式。以后的支持将涵盖针对 UI 的基于 WebGL 和 Web 的框架，例如 jQuery Mobile（有助于移植现有 jQuery 代码）。

Location

Location 模块提供基于位置的服务 (LBS)，包括位置信息、地理编码、卫星信息和 GPS 状态。它还提供来自不同定位来源的位置信息，这些位置来源包括 GPS、WPS（Wi-Fi 定位系统）、小区 ID 和传感器等。

Messaging

Messaging 模块提供 SMS、MMS、电子邮件和 IM 功能。

Multimedia（多媒体）

Multimedia Framework（多媒体框架）提供不同的多媒体功能。它由播放器/流式传输框架、摄像头/记录框架、媒体内容框架和屏幕镜像框架构成。

PIM（个人信息管理）

通过 PIM 模块，可以在设备上实现用户数据管理。它使您能够管理日历、联系人和任务，并且检索与设备环境有关的数据（例如设置位置和缆线状态）。

Security

Security 模块负责跨系统的安全部署。它由支持平台安全性的功能构成，例如访问控制、证书管理和防病毒框架。

System

System 模块由系统和设备管理功能构成，包括：

- 用于访问设备（例如传感器、显示器或振动器）的接口。
- 电源管理，例如 LCD 显示器背光变化和应用程序处理器休眠。
- 在 USB、MMC、充电器和耳机插孔事件等情况下的设备监视和事件处理。

- 系统升级。
- 移动设备管理。

Telephony

Telephony 模块由与调制解调器进行通信的蜂窝功能构成：

- 管理针对 UMTS 和 CDMA 的呼叫相关的和非呼叫相关的信息和服务。
- 管理针对 UMTS 和 CDMA 的数据包服务和网络状态信息。
- 管理针对 UMTS 和 CDMA 的 SMS 相关服务。
- 管理 SIM 文件、电话簿和安全性。
- 管理针对 UMTS 的 SIM 应用程序工具包服务。

Web

Web 模块提供为低功率设备优化的 Tizen Web API 的完整实现。它包括 WebKit，这是精心设计的一种布局引擎，可允许 Web 浏览器呈现网页。它还为 Web 应用程序提供 Web Runtime。

2. 获取源代码和生成内部版本

本章介绍如何检索源代码以及生成平台、内核和映像内部版本。

2.1 获取源代码

要获取源代码：

1. 从 Tizen Gerrit 项目列表 (https://review.tizen.org/gerrit/#/admin/projects) 确认程序包名称。
   还可以使用以下命令：
   ```
   $ ssh review.tizen.org gerrit ls-projects
   ```

2. 使用以下命令克隆 Gerrit 项目：
   ```
   $ git clone [-b <Branch>] ssh://<Username>@review.tizen.org:29418/<Gerrit_Project> [<Local_Project>]
   ```

   例如：
   ```
   $ git clone ssh://<Username>@review.tizen.org:29418/platform/upstream/dbus
   ```

要克隆所有项目，请阅读 https://source.tizen.org/documentation/developer-guide/getting-started-guide/cloning-tizen-source 中的说明。

2.2 生成平台内部版本

要生成平台内部版本：

1. 要获取访问权限，请在 https://www.tizen.org/user/register 处注册一个帐户。

2. 安装下图中定义的 Tizen 平台开发工作流程执行。

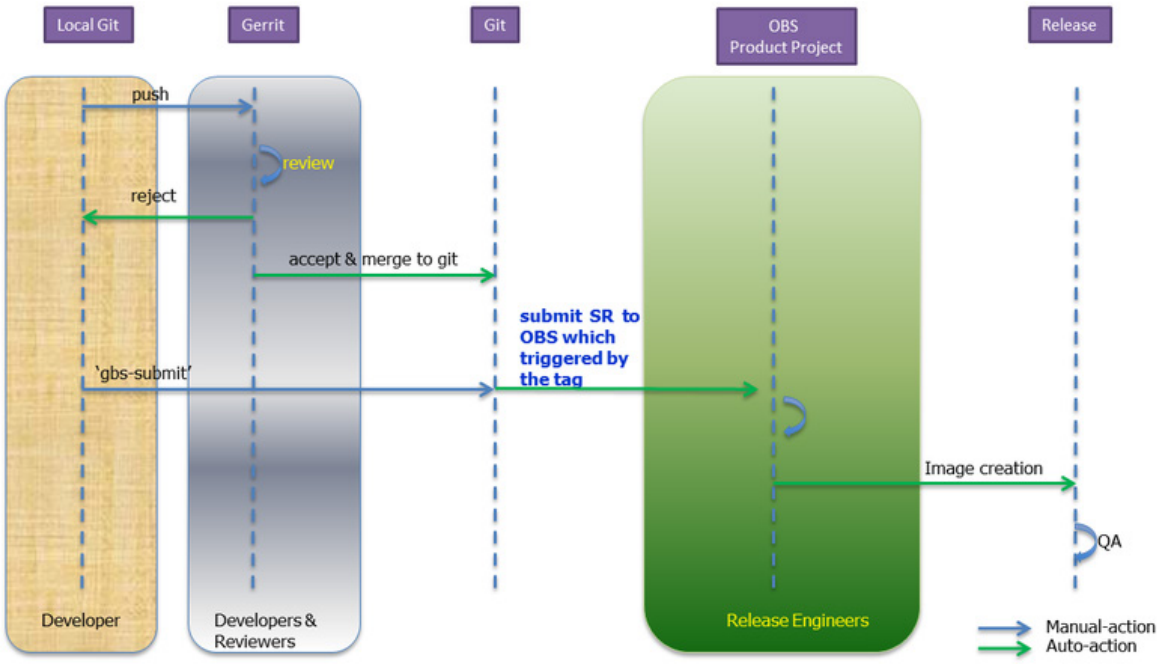

图 2. Tizen 平台开发工作流程

Git

Git 是一种特别强大、灵活和低系统开销的版本控制系统，能够实现高效、强健的协作开发。有关详细信息，请参阅：

- Git 社区手册：http://git-scm.com/documentation
- Git Wiki：https://git.wiki.kernel.org/index.php/Main_Page
- Git 手册页：https://git.wiki.kernel.org/index.php/Main_Page

Gerrit

Gerrit 是一种基于 Web 的代码检查系统，有助于使用 Git 版本控制系统进行针对项目的在线代码检查。通过在并排显示中显示更改和支持行内注释，Gerrit 可以优化代码检查过程，增强检查质量。此外，通过允许任何授权用户将更改提交到中央 Git 存储库，Gerrit 简化了基于 Git 的项目的维护，实现了对 Git 的更加集中的使用。
有关详细信息，请参阅：

- Gerrit 文档页：https://review.tizen.org/gerrit/Documentation/index.html

OBS

开放式生成服务 (OBS) 是一种开放且完备的分发开发平台，它为开发人员提供基础结构，以便为不同硬件体系结构上的不同 Linux 分发轻松地创建和发布开放源软件。此外，OBS 提供协作性环境，使开发人员团队能够生成和提交对其他项目的更改。有关详细信息，请参阅：

- 开放式生成服务：http://openbuildservice.org/
- OBS 门户：http://en.opensuse.org/openSUSE:Build_Service

GBS

Tizen 开发人员使用 git 和 gbs 命令行工具来进行其大多数工作。使用 Git 生成系统 (GBS)，您可以在本地生成您的 git 存储库。有关详细信息，请参阅：

- 使用 GBS 在本地生成：https://source.tizen.org/documentation/developer-guide/getting-started-guide/building-packages-locally-gbs

MIC

MIC Image Creator 可用于创建可下载的二进制映像：还可以使用 MIC 在您的本地创建自定义二进制映像。有关详细信息，请参阅：

- 使用 MIC 创建 Tizen 映像：https://source.tizen.org/documentation/developer-guide/getting-started-guide/creating-tizen-images-mic

2.3 生成内核内部版本

要生成 Tizen ARM 内核内部版本：

1. 如果目标和您的主机不同（例如 x86），则在您的系统上安装并且设置交叉编译工具。
   您可以使用其 Linarotoolchain 二进制文件或 Ubuntu 程序包对您的环境进行设置，以便用于交叉编译工具（例如，export CROSS_COMPILE=....）。

2. 为 RD-PQ 设置 `.config` 文件：
   ```
   $ make ARCH=arm trats2_defconfig
   ```

3. 在根据您的需要重新配置后（使用 `make ARCH=arm menuconfig` 命令）或者使用原有配置（无需修改）后，生成内核：
   ```
   $ make ARCH=arm uImage
   ```

4. 构建和生成 Kernel 模块映像。
   注意：如果需要，首先执行 sudo 以便让 sudo -n 在脚本中起作用。
   ```
   $ sudols
   $ scripts/mkmodimg.sh
   ```

5. 通过 lthor 将映像发送到目标：
   ```
   $ lthor arch/arm/boot/uImageusr/tmp-mod/modules.img
   ```

   或者从以下两个文件生成您自己的 tar 文件：
   ```
   $ tar cf FILENAME_YOU_WANT.tar -C arch/arm/boot uImage -C ../../../usr/tmp-mod
   modules.img
   ```

2.4 生成映像内部版本

生成二进制映像：

1. 将平台程序包安装到临时根系统目录中。

2. 运行自定义脚本。

3. 使用分区信息创建文件系统。

Tizen 2.x

要定义如何为配置文件生成二进制映像，可使用 2 个特殊的 gerrit 项目：

- tools/package-groups (https://review.tizen.org/gerrit/gitweb?p=tools/package-groups.git)
  定义程序包组。查看 `patterns` 目录：每个 `*.yaml` 文件都定义程序包组。

- tools/image-configurations (https://review.tizen.org/gerrit/gitweb?p=tools/image-configurations.git)
  定义如何构建二进制映像。可以定义多个配置（`configurations.yaml`）：
  - 具有大写字母的目录存储每个二进制配置。
  - “custom/part”承载分区。
  - “custom/scripts”在安装所有平台程序包后承载脚本。
  每个二进制配置都可以承载不同的程序包组。使用“@<程序包组的名称>”描述组。

3. Tizen System Layer（系统层）

本章详细介绍 Tizen System Layer。

3.1 Kernel 和 BSP

内核是驱动平台的操作系统。在本文中，“内核”指的是为 Tizen 平台自定义的开放源 Linux 内核。下面几节概要介绍 Tizen 内核配置以及用于自定义内核的环境。

内核配置

下表定义了要用于 Tizen 平台的推荐的内核配置。

表 1. 内核配置

配置	说明
CONFIG_CGROUPS/CONFIG_CGROUP_MEM_RES_CTRL	控制组
CONFIG_CMA/CONFIG_DMA_CMA	连续内存分配器
CONFIG_DMA_SHARED_BUFFER	在 dma 设备中共享缓冲区
CONFIG_DRM	指示呈现管理器图形基础结构
CONFIG_VIDEO_V4L2_SUBDEV_API	用于多媒体 API 的 Linux 的视频
CONFIG_USB_GADGET	USB gadget 支持
CONFIG_USB_G_SLP	Tizenusb_mode (sdb, ether) 组合驱动程序
CONFIG_ANDROID_LOGGER	Dlog 日志驱动程序
CONFIG_EXT4_FS	Ext4 文件系统
CONFIG_SECURITY_SMACK	SMACK 安全访问控制

Tizen 文件系统

Tizen 采用 ext4 文件系统作为默认的根文件系统。与 Android 不同，Tizen 不使用 ramdisk 和 initramfs。该 VFAT 文件系统可用于外部 SD 卡。

Tizen 分区布局

下图阐释了 Tizen 分区布局的一个示例。产品供应商可根据需要修改其设备的顺序或分区布局。

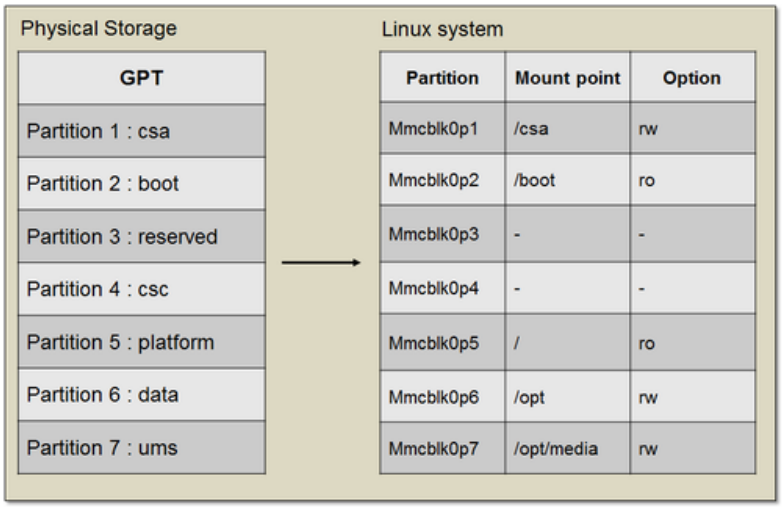

图 3. Tizen 分区布局

- 分区 1：CSA（配置保存区域）
  包含作为调制解调器的校准值的非易失性数据。
- 分区 2：根
  包含内核映像、调制解调器映像和 Device Driver 模块。
- 分区 3：平台
  包含根文件系统、基本的 Tizen 框架以及某些一般的 Linux 实用工具。
- 分区 4：数据
  包含应用程序、应用程序库和平台数据库。
- 分区 5：CSC（客户软件配置）
  存储客户的软件配置。
- 分区 6：UMS（USB 大量存储）
  包含默认（媒体）内容。

Tizen 中的文件系统体系结构标准

每个分区都具有下图中所阐释的体系结构。

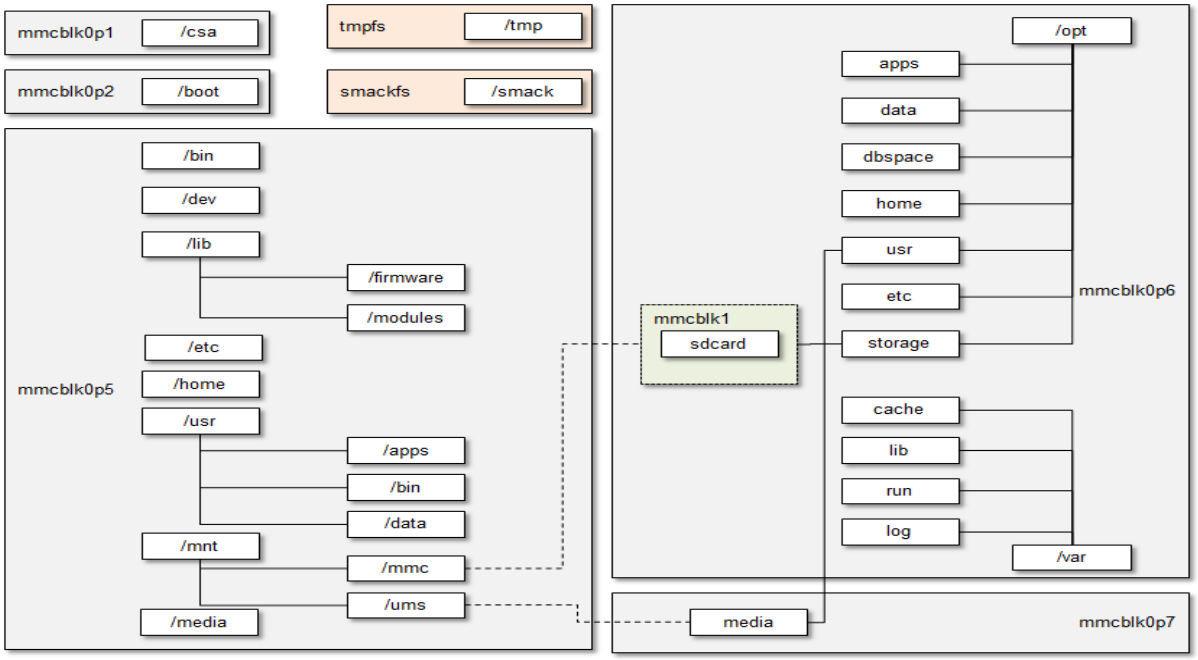

图 4. 文件系统体系结构

- `/usr/apps`：内部应用程序
- `/opt/usr/apps` (`/opt/apps`)：第三方和可下载应用程序
- `/opt/dbspace`：针对 Tizen 框架的数据库文件
- `/opt/storage/sdcard`：外部 sdcard 装入点
- `/smack`：用于加载和存储 smack 规则的 SMACK 文件系统

Tizen 中的内存管理

Tizen 使用 cgroup 维护进程及其资源。如果某一进程从后台移到前台，则进程组将移入“前台”cgroup。

根据前台组策略，将提升该流程的优先级，以便降低在内存不足的情况下终止该进程的可能性。如果系统内存达到了内存不足阈值，则内核将会通知资源。资源将回收后台组以便获取足够的可用内存。如果资源未能获取可用内存，它会尝试终止具有最高分数（“oom_score_adj”x“进程的内存大小”）的放弃的进程。

3.2 System

本节介绍系统和传感器框架。

3.2.1 System Framework（系统框架）

下图阐释 System Framework。

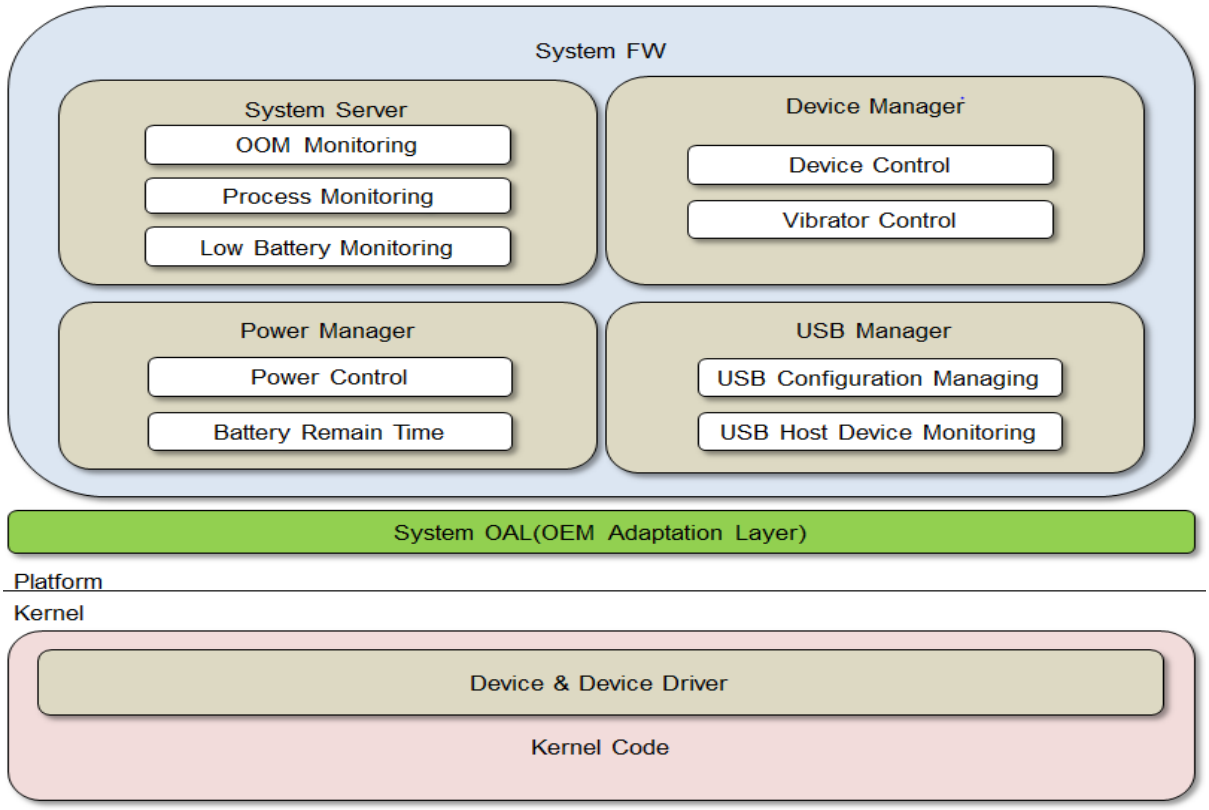

图 5. System Framework

System server（系统服务器）

System Server 处理系统事件，例如内存不足事件以及电池电量水平和即插即用设备状态中的变化，还可以管理进程“看门狗”功能。

Power manager（电源管理器）

Power Manager 是用于管理系统电源的会话守护程序。它提供条件状态转换。Tizen 电源管理器功能控制显示器背光变化和设备休眠。

Device manager（设备管理器）

Device manager 提供用于控制所有设备的接口，例如 LCD 背光变化和应用程序处理器休眠。

USB manager（USB 管理器）

USB manager 对 USB 配置进行设置以便连接到 PC，以及监控外部 USB 主机设备，例如键盘、鼠标、摄像头、USB 存储和 USB 打印机。

移植 OAL Interface（OAL 接口）（必需）

System Framework 具有针对应用程序的 libdeviced 库，以及作为 OAL（OEM 适配层）的包装程序库的 libdevice-node 库。OAL 接口通过根据设备类型组合 OAL API，为设备访问内核提供了一个轻松的方法。OAL API 支持供应商指定的所有设备。

OEM 开发人员必须实现在 `devman_plugin_intf.h` 中定义的 API 并且将其库编译为 `libslp_devman_plugin.so`。

Devman 库使用 sysfs 用于与 Device Driver 和 Kernel 的接口。sysfs 是 Linux 2.6 或更高版本提供的虚拟文件系统。

要配置 OAL Interface：

1. 将作为 `libslp_devman_plugin.so` 的 OEM 库安装到 `/usr/lib`。
2. OAL API 定义位于 `devman_plugin_intf.h` 头文件中，该文件位于 libdevice-node 中。

System OAL 内核配置（必需）

电源管理支持、CPU 空闲 PM 支持、CPU DVFS、CPU 热插拔、使用动态热插拔

systemd

systemd 是用于执行用户空间的第一个进程。它管理引导顺序，并且启动设备、资源和其他守护程序。

Tizen 的单元文件位于 `/usr/lib/systemd/system` 和 `/usr/lib/systemd/user` 中。您必须在那里添加您自己的单元文件。

您必须在 systemd 的内核配置中启用“cgroup”和“autofs”选项。systemd 还依赖于 dbus 和一些库，例如 libnotify 和 libudev。

有关系统的更详细信息，请参阅 systemd wiki (http://www.freedesktop.org/wiki/Software/systemd/)。

3.2.2 传感器框架

传感器框架向应用程序和系统组件提供传感器事件。传感器事件是来自基于传感器或虚拟传感器的硬件的测量信息。

Tizen 支持针对以下传感器的单独的插件框架：

- Accelerometer sensor（加速计传感器）
- Gyroscope sensor（回转器传感器）
- Proximity sensor（近距离传感器）
- Geomagnetic sensor（地磁传感器）
- Light sensor（光传感器）

下图阐释传感器框架。

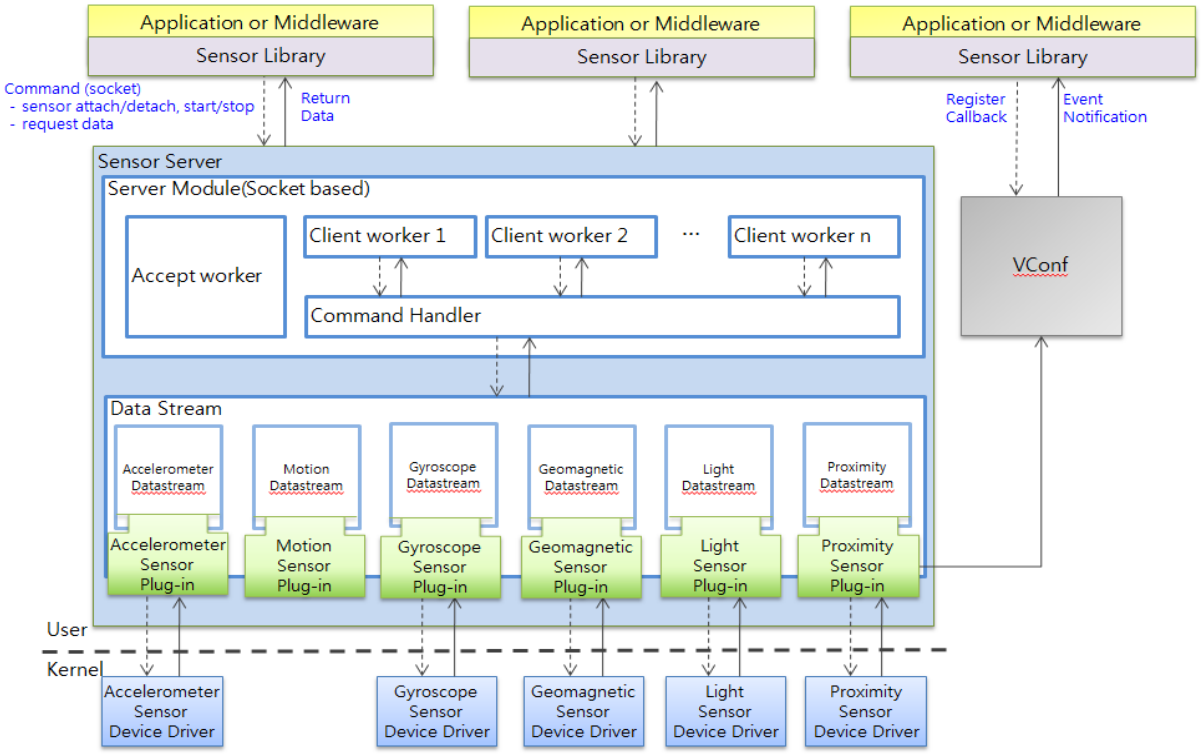

图 6. 传感器框架

移植 OAL Interface（必需）

传感器 OAL 包含处理器插件、筛选器插件和传感器插件。以 Accelerometer sensor (accel) 为例阐释各种插件原型的实现方式。您可以在 git 中使用常见的传感器插件（处理器、筛选器、传感器）：`sensor-framework/sensor-plugin-source`。

- 处理器插件：处理数据或从筛选器或传感器数据生成事件的主动组件（具有线程）
- 传感器插件：从内核节点获取原始数据的被动组件
- 筛选器插件：将传感器原始数据转换为其他类型的数据的被动组件。

配置

传感器框架加载 `sf_sensor.conf`、`sf_filter.conf`、`sf_processor.conf` 和 `sf_datastream.conf` 配置文件以便用于加载传感器、筛选器和处理器插件。所有配置文件都包括在传感器框架程序包中。

要向库中添加传感器 API 和枚举，请添加用于您的传感器的 API 并且为事件/数据类型分配您的专用编号。

有关传感器框架的更详细信息，请参阅 Tizen wiki 网站 (https://wiki.tizen.org/wiki/Porting_Guide#Description_2) 的“传感器框架”部分。

3.2 图形和 Window 系统（OpenGL、X Server）

下图阐释了 Tizen UI 和图形模块体系结构。

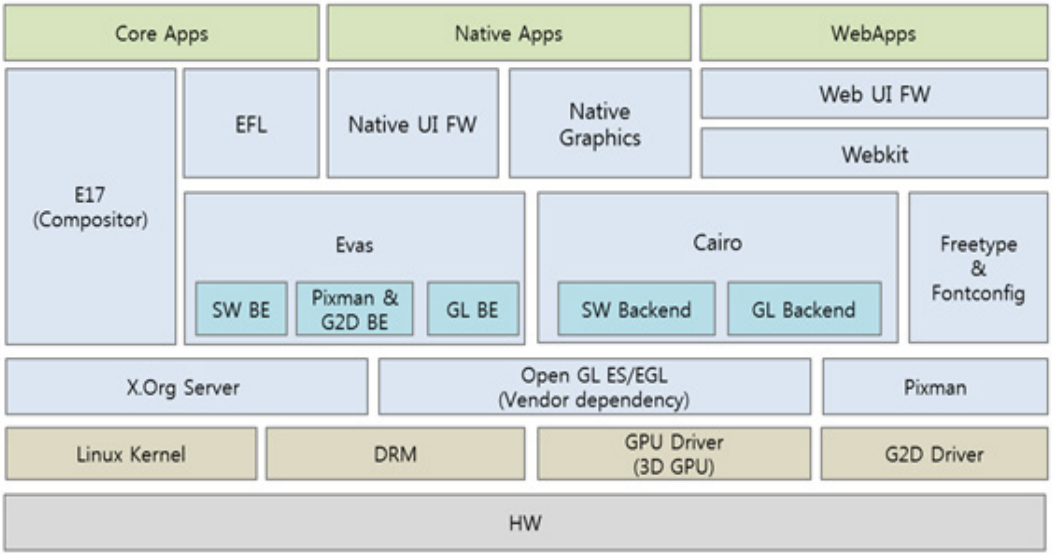

图 7.UI 和图形模块体系结构

图形

Tizen 提供高性能的 3D 图形作为 UI 和图形模块的组件。存在针对 3D 应用程序（例如 3D 游戏）的硬件加速 OpenGL® ES（开放图形库嵌入系统）和 EGL™（嵌入系统图形库）功能。

OpenGL® ES 是以手持和嵌入设备为目标的用于高级 3D 图形的应用程序编程接口 (API)。为了克服设备约束，例如受限的处理功能和内存可用性，它提供了 OpenGL® 中的一部分功能。

添加了嵌入系统特定的功能以便增强呈现效率，例如向来自 OpenGL® ES 2.0 的着色语言添加了精度限定器。

OpenGL 3D 图形库

下图阐释了用于 OpenGL® 的 Tizen 高性能 3D 图形库的接口。

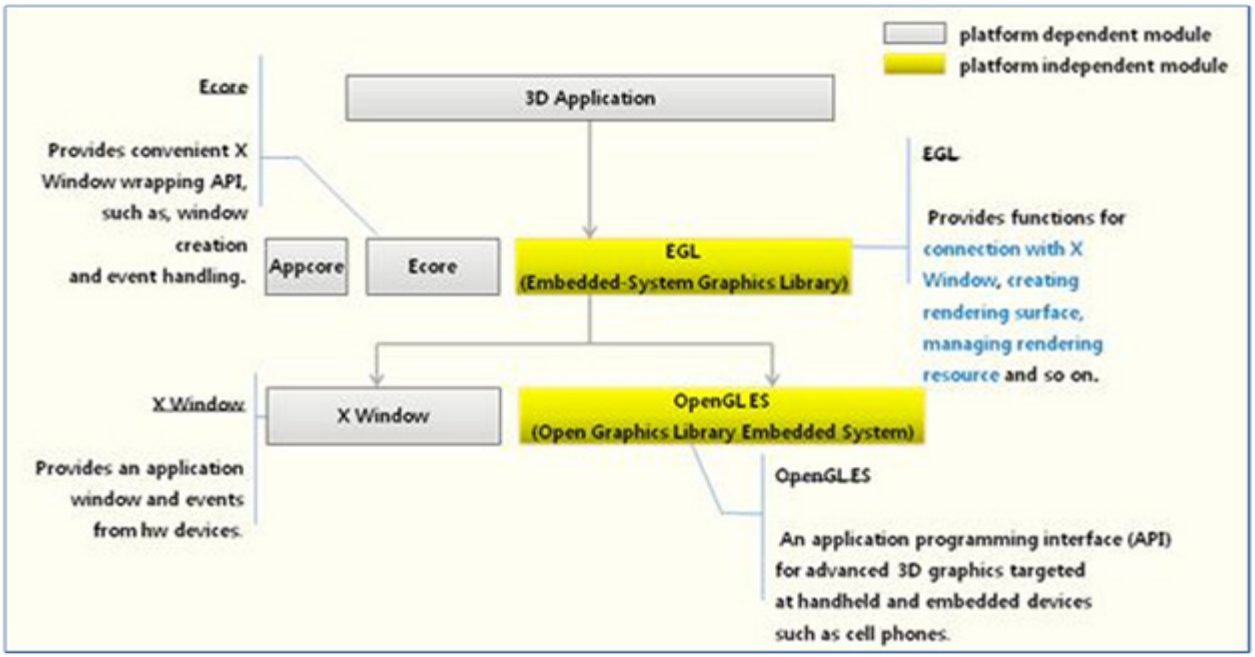

图 8. 3D 图形库接口

移植要求

3D 图形供应商必须满意以下移植要求以便 Tizen 正常行使功能：

- OpenGL® ES
    - 要求的 OpenGL® ES 版本：1.1 和 2.0
    - 驱动程序还必须支持对 OpenGL® ES 1.1 的以下扩展：

GL_OES_framebuffer_object	GL_OES_blend_subtract
GL_OES_blend_func_separate	GL_OES_matrix_palette
GL_OES_draw_texture	GL_OES_texture_cube_map
GL_OES_query_matrix	GL_OES_point_size_array

    - 驱动程序还必须支持对 OpenGL® ES 2.0 的以下扩展：

GL_OES_EGL_image	GL_EXT_texture_format_BGRA8888
GL_OES_get_program_binary	GL_OES_texture_npot
GL_OES_fragment_precision_high	GL_OES_rgb8_rgba8
GL_OES_depth24	GL_OES_vertex_half_float
GL_OES_texture_float	GL_OES_compressed_ETC1_RGB8_texture
GL_OES_packed_depth_stencil	GL_OES_standard_derivatives

GL_OES_element_index_uint	GL_OES_mapbuffer
GL_EXT_multi_draw_arrays	GL_OES_vertex_array_object
GL_IMG_texture_compression_pvrtc	GL_OES_read_format
GL_EXT_multisampled_render_to_texture	

- 供应商需要提供工具以便在 Linux 上生成二进制着色器。

- EGL™
  - EGL™ 所需的最低版本为 1.4。EGL™ 必须支持 DRI2。
  - 常用 3D 应用程序使用基于 DRI2 的 EGL™。
  - 3D 组合窗口管理器要求基于 DRI2 的 EGL™。
  - 驱动程序必须支持对 EGL™ 1.4 的以下扩展：

GL_KHR_image	EGL_KHR_image_base
EGL_KHR_gl_texture_2D_image	EGL_KHR_gl_texture_cubemap_image
EGL_KHR_gl_renderbuffer_image	EGL_KHR_image_pixmap
EGL_KHR_lock_surface	EGL_KHR_fence_sync
EGL_KHR_reusable_sync	EGL_IMG_context_priority
FBDEV based EGL （可选）	

  - FBDEV 必须基于三缓冲区机制。必须基于翻页功能而不是复制缓冲区对其进行操作。
  - 为了避免撕裂问题，应使用 LCD vsync 信号进行协调。通常，复合器使用 `eglSwapInterval()` API 启用 vsync。

- 基于 DRI2 的 EGL™（必需）
  - 非常重要！
  - 必须基于 X11 DRI2 协议实现 EGL™。
  - EGL™ 必须支持来自 pixmap 的纹理（使用 `EGL_KHR_image_pixmap` 和 `GL_OES_EGL_image`）。

- GPU 同步 API
  - 为避免闪烁，需要等待缓冲区复制命令完成的内部 EGL™ API。

- 标头和库名称
  - 必须根据 Khronos API 实现者指南在驱动程序内提供头文件和库。

注意：您可能需要实现一些内存管理技术，以便在您的 CPU 和图形处理单元之间共享内存。

Window 系统 (Xserver)

下图阐释了 XServer 的主要部分以及 Linux kernel 中的相反部分。

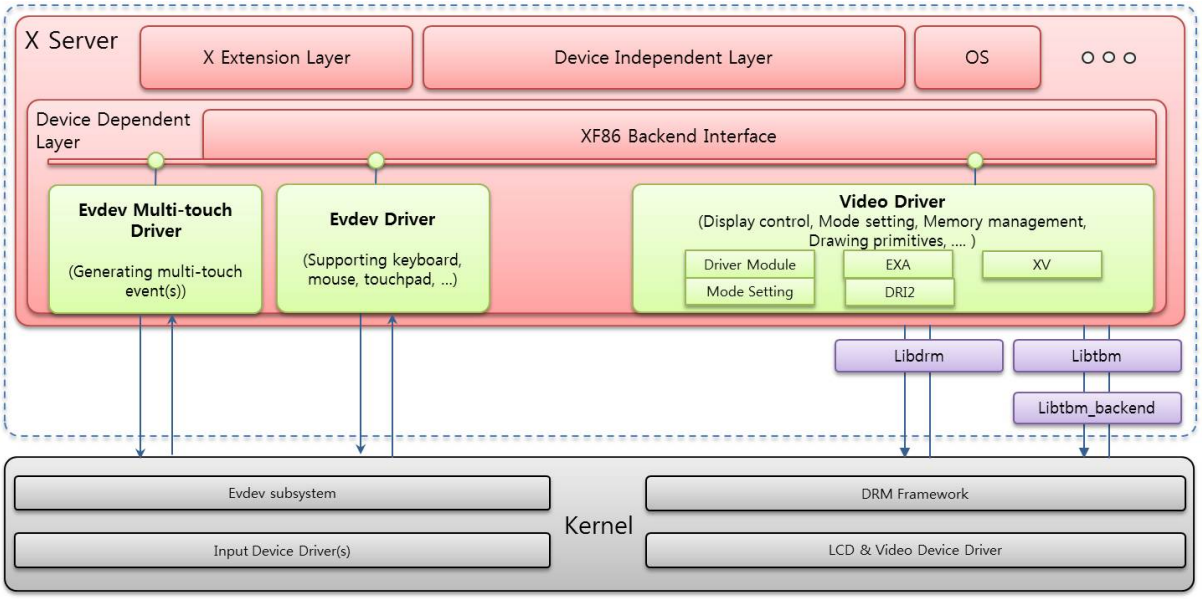

图 9. XServer 组件

与其他 Unix/Linux 系统相似，Tizen 包含基于 XServer 的窗口系统。X Window（X 窗口）系统提供用于操作资源（例如窗口、像素图和 gc）的界面。

X 客户端将请求发送到 X 服务器以便绘制内容或操作窗口。X 服务器接受客户端请求并且返回答复。此外，在来自设备的输入事件挂起时，或者在 X 服务器遇到 X 客户端感兴趣的事件（例如配置事件或公开事件）时，X 服务器将向客户端发送 X 事件。

Tizen 使用 X.org 实现 Xserver，这是 X Window 系统 (http://www.x.org/) 的开放源实现方式。

X.org server 可以划分为两个部分：

- DIX 层是执行独立于设备的任务的独立于设备的层。
- DDX 层是执行依赖于设备的任务的依赖于设备的层。

为使 X Window 系统能够在设备上工作，您必须在 DDX 层中移植 X 输入/视频驱动程序。

X 输入驱动程序

有 2 种 X 输入驱动程序：X Evdev driver（X Evdev 驱动程序）和 X Evdev-multitouch driver（X Evdev-multitouch 驱动程序）。

每种驱动程序都从各输入设备节点读取输入事件流，生成 X 内部事件，并且将这些事件放置于 X 服务器的内部事件队列中。为执行此操作，X 输入驱动程序主要使用在 X 服务器内的 xf86 DDX 层上实现的接口。

Evdev driver：

- 该驱动程序解释来自内核 Evdev subsystem（Evdev 子系统）的事件，以便用于键（键盘）设备、鼠标设备或触摸板等设备。
  注意：基本上，可以在不进行任何修改的情况下使用该驱动程序。

- 有关要为设备实现的数据结构和基本功能的信息，请参阅 http://www.x.org/wiki/Development/Documentation/XorgInputHOWTO/。

Evdev-multitouch 驱动程序：

- 该驱动程序解释来自内核 Evdev subsystem 的事件，以便用于触摸屏设备。
- 它支持 2 种类型的 MT 协议：
    - MT 协议 A（包括跟踪 ID）
    - MT 协议 B

  注意：基本上，如果设备内核支持 MT 协议 B，则可以使用该驱动程序。
- 您可以通过修改源代码支持旧式的单触点协议。建议使用 MT 协议 B。
- 要支持新的触摸协议，请执行以下操作之一：
    - 在驱动程序内为新的触摸协议实现一些内容。
    - 克隆 evdev multi-touch driver 并且在克隆中实现新的触摸协议。
    - 从新协议转换到现有协议，evdev multi-touch driver 默认支持此转换。

X 输入驱动程序配置：

- “ServerFlags”之下的以下选项是必不可少的：
    - AllowEmptyInput：“true”意味着启动 X 服务器且没有任何键盘/鼠标驱动程序。
    - AutoAddDevices：“false”意味着不支持热插拔输入设备。
    - AutoEnableDevices：“true”意味着“DevicePresenceNotify”事件默认发送给所有 X 客户端。
- 有关 X 配置的更详细信息，请参阅 http://www.x.org/releases/current/doc/man/man5/xorg.conf.5.xhtml。通过应用和修改配置文件（例如 `xorg.conf` 或 `xorg.conf.d` 目录下的文件），您可以应用可选功能。

X 视频驱动程序

X 视频驱动程序是一个驱动程序库，包含支持 xf86 DDX 层的实现，以便在屏幕上显示内容：

- Driver Module（驱动程序模块）
    - 针对 xf86 DDX 的 Driver Module 的实现。
- Mode Setting（模式设置）
    - 使设备在其上显示图像的实现。
    - 内核可以支持模式设置（例如，Drm 模式设置）。
- EXA
    - 针对图形加速体系结构的实现。内存分配和附加绘制基元。
- DRI2
    - 针对 DRI2（直接呈现基础结构版本 2）的实现。
    - 对于图形加速，必须通过全局内存管理分配和共享图形内存。
- XV
    - XFree86 提供 X 视频扩展 (XV)，这允许客户端将视频视作任何其他基元并且使视频处于可绘制模式。默认情况下，该扩展不会将任何视频适配器报告为可用，因为 DDX 层尚未初始化。

移植 X 视频驱动程序

X 视频驱动程序必须实现以下驱动程序模块和扩展，并且在 X 服务器中使用 xf86 DDX 和 X 扩展提供的接口。在 http://www.x.org/releases/X11R7.6/doc/xorg-server/DESIGN.txt 中描述了所有步骤。

3.3 Multimedia

本节介绍 Multimedia Framework 的各个组件。

3.3.1 编解码器

Tizen Multimedia Framework 提供基于 Gst-OpenMax IL Codec plug-in（Gst-OpenMax IL 编解码器插件）的 HW 编解码器。Gst-openmax IL 编解码器插件是允许与 OpenMAX IL 组件进行通信的 GStreamer 插件程序包。

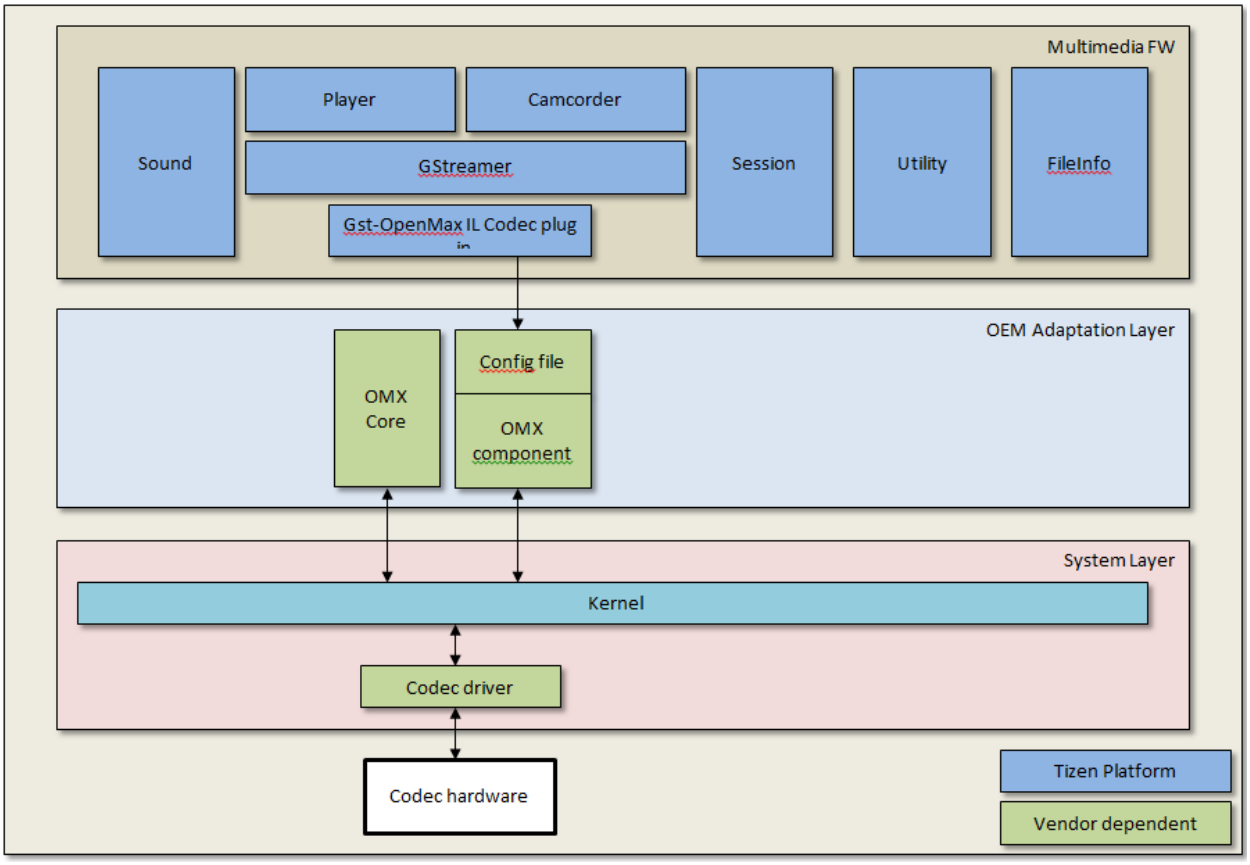

图 10. Multimedia Framework 中的编解码器

移植 OAL Interface

Gst-openmax IL 编解码器插件用作其他 GStreamer 插件。有关详细信息，请参阅 http://www.freedesktop.org/wiki/GstOpenMAX。

参考

有关 OpenMAX IL 组件的详细信息，请参阅：

- http://www.khronos.org/openmax/il/

3.3.2 Camcorder（摄像机）（包括摄像头）

多媒体 Camcorder 框架控制 GStreamer 的摄像头插件，以便捕获来自设备的摄像头数据。

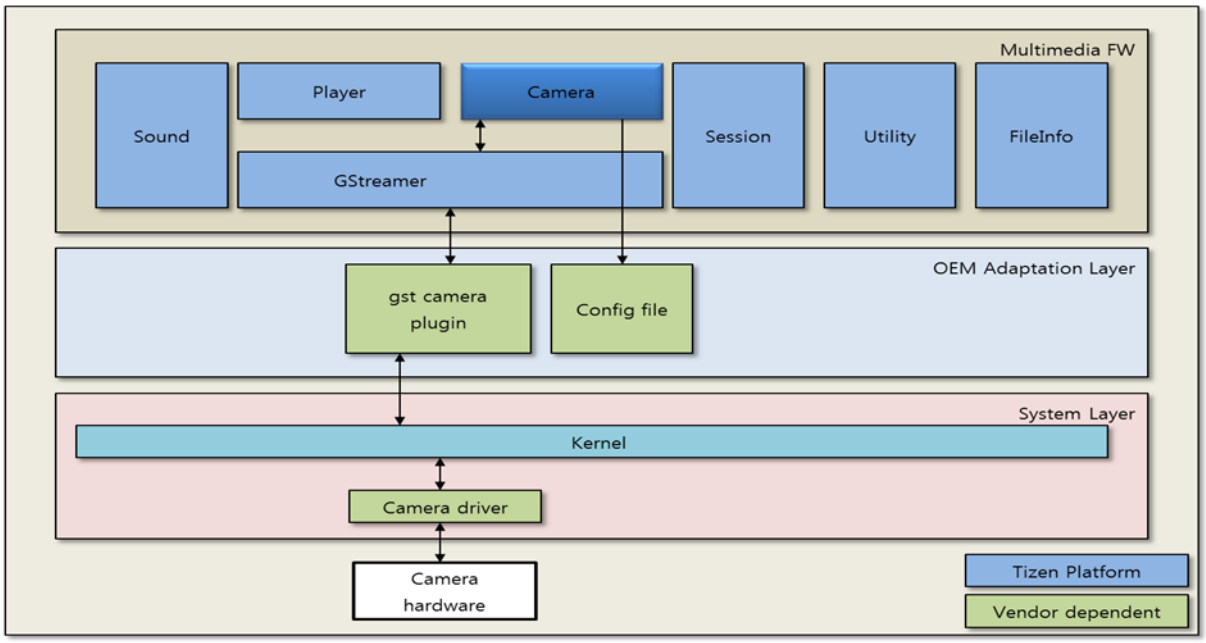

图 11. Multimedia Framework 中的摄像头

请注意，对于不同的芯片集，控制摄像头设备的内核接口可能不同，因此，必须针对各芯片集实现摄像头插件。每个配置文件都包含它自己的依赖于硬件的特定信息。

- 针对 GStreamer 的摄像头源插件
    - 从摄像头设备获取摄像头数据（预览图像或捕获的图像）
    - 继承“GstPushSrc”
- 用于多媒体 Camcorder 框架的配置文件
    - `mmfw_camcorder.ini`：Camcorder 设置文件。
    - `mmfw_camcorder_dev_video_pri.ini`：该文件包含高分辨率后摄像头的设置。
    - `mmfw_camcorder_dev_video_sec.ini`：该文件包含低分辨率前摄像头的设置。

移植 OAL Interface

GStreamer 摄像头插件：

- 因为对于不同的芯片集，控制摄像头设备的内核接口可能不同，所以，必须针对各芯片集实现摄像头插件。要开发新插件，请参阅 GSteamer 插件开发指南 (http://gstreamer.freedesktop.org/data/doc/gstreamer/head/pwg/html/index.html)。
- 摄像头的主要功能是由 GStreamer 摄像头控制接口和信号回调提供的，Tizen 已添加了它们。摄像头插件必须支持它们（`gst-plugins-base0.10/gst-libs/gst/interfaces/cameracontrol.h` 接口和信号回调：“静止捕获”）。

- 想要实现摄像头源插件的任何第三方开发人员都必须从 GstPushSrc 派生它。

参考

有关所有 GStreamer 文档，请参阅：

- http://gstreamer.freedesktop.org/documentation/

要开发 GStreamer 插件，请参阅：

- http://gstreamer.freedesktop.org/data/doc/gstreamer/head/pwg/html/index.html

3.3.3 无线电

Multimedia Framework 的收音机接口部分支持 API 以便使用 Linux V4L2 接口实现以下 FM 收音机功能：

- 调频
- 获取和设置频率
- 扫描所有可用频率
- 向上搜索和向下搜索
- 获取频率信号

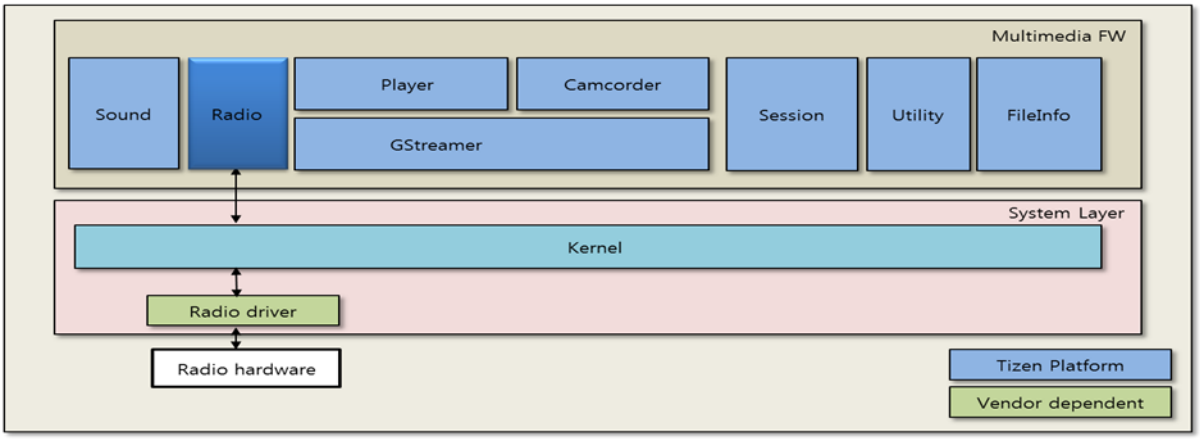

图 12. Multimedia Framework 中的收音机

移植 OAL Interface

针对 FM 收音机的 OAL Interface 是 Linux 内核 V4L2 接口。该收音机模块直接使用 V4L2 ioctls 执行不同的收音机硬件配置。

参考

V4L2 规范：

- http://v4l2spec.bytesex.org/spec-single/v4l2.html

3.3.4 音频

下图阐释 Multimedia Framework 的音频组件。

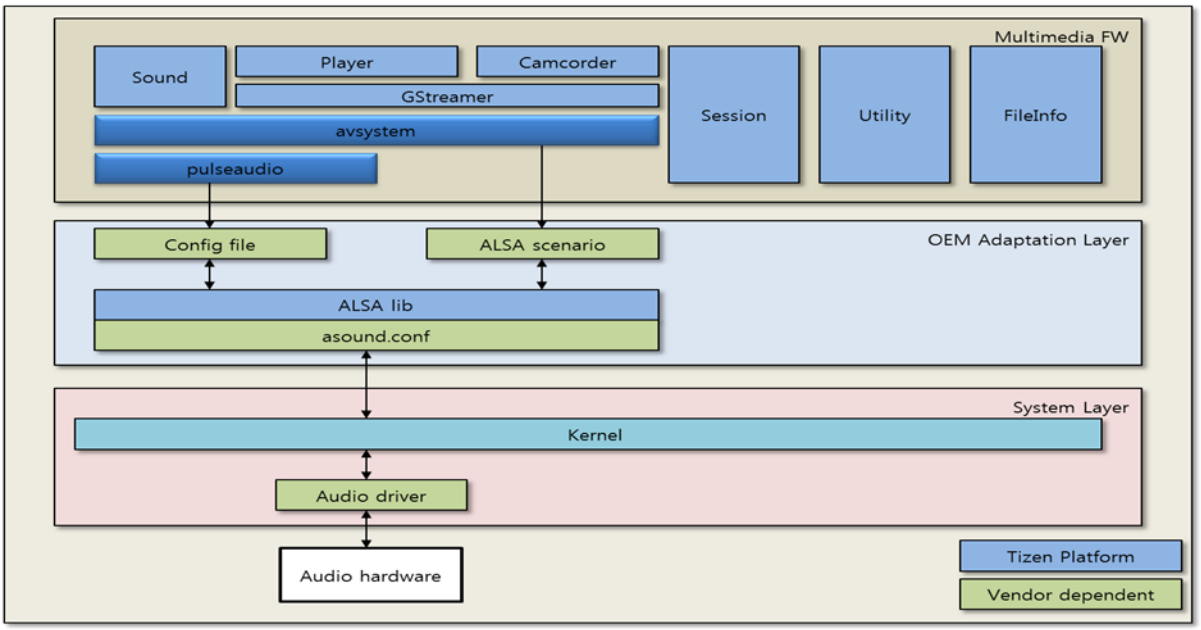

图 13. Multimedia Framework 中的音频

- Avsystem
    - Avsystem 是一个本机 Tizen 组件，它提供用于在上层的层中处理流和声音路径的 API。
- PulseAudio
    - PulseAudio 是声音服务器，它接受来自一个或多个源的声音输入并且将其重定向到一个或多个接收器中。
- `asound.conf`
    - 用于附加 ALSA DAI（数字音频接口）（例如 AIF1 和 AIF2）的配置文件。

移植 OAL Interface

PulseAudio：

- Tizen 中的 PulseAudio 不支持 udev。
- PulseAudio 必须以系统模式启动。
- 为支持多种设备，某些配置文件已从 PulseAudio 隔离到 mmfw-sysconf。有关详细信息，请参阅下面的配置详细信息。
- 不要更改 `/etc/pulse/system.pa` 中的默认接收器。某些模块使用该默认接收器。如果更改了该默认接收器，播放声音会遇到问题：
  `load-module module-remap-sink sink_name=mono_alsa master=alsa_output.0.analog-stereo channels=1`

`asound.conf`：

- 名称 AIF2 和 AIF3 不得更改。
- `/etc/asound.conf` 依赖于硬件。有关更详细信息，请参阅下面的配置详细信息。

配置

为支持多种设备，某些配置文件已从 PulseAudio 隔离到 mmfw-sysconf。PulseAudio 具有以下配置文件：

- `/etc/pulse/client.conf`
- `/etc/pulse/daemon.conf`
- `/etc/pulse/default.pa`
- `/etc/pulse/system.pa`
- `/usr/share/pulseaudio/alsa-mixer/profile-sets/default.conf`
- 此外，还有几个配置文件。目前，这几个配置文件没有太大影响，可根据需要进行修改。
  - `usr/share/pulseaudio/alsa-mixer/profile-sets/`
  - `usr/share/pulseaudio/alsa-mixer-paths/`

参考

PulseAudio：

- http://www.freedesktop.org/wiki/Software/PulseAudio

ALSA

- http://www.alsa-project.org/

3.4 Connectivity

本节介绍 Connectivity 模块的各个组件。

3.4.1 WLAN

主要 WLAN 功能是：

- WLAN (802.11 b/g/n)
- WPS PBC
- EAP（AKA、SIM、PEAP、TTLS）

下图阐释了 Tizen WLAN 体系结构。

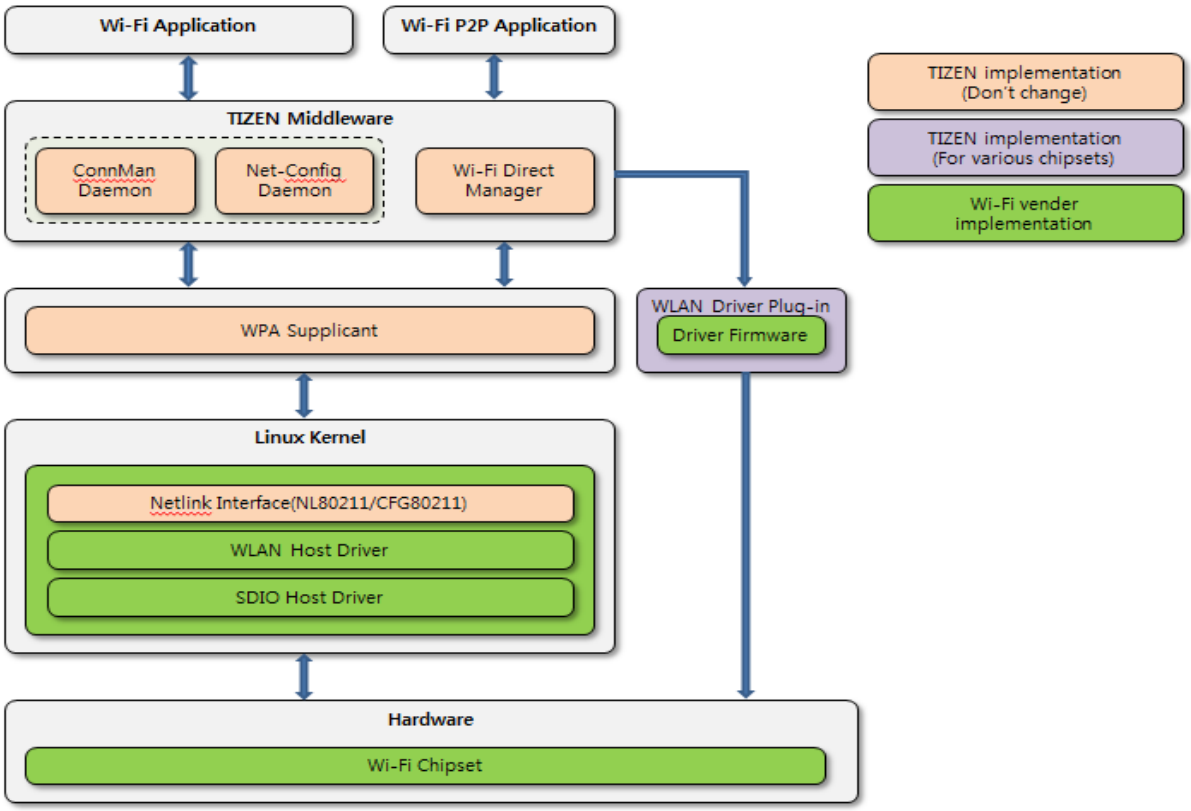

图 14. WLAN 体系结构

WLAN 体系结构集中处于 Linux 无线 (IEEE-802.11) 子系统上。Linux 无线 SW 堆栈定义在 Tizen 中需要使用的 WLAN HW 适配 SW 接口。

连接管理器 (ConnMan) 是用于管理在 Linux 操作系统上运行的嵌入设备内的 Internet 连接的守护程序。

WPA 客户端 (wpa_supplicant) 提供对 WPA 和 WPA2 (IEEE 802.11i / RSN) 的支持。该客户端是在客户端工作站中使用的 IEEE 802.1X/WPA 组件。

移植 OAL Interface

该 WLAN 驱动程序插件是特定于 Wi-Fi Chipset（Wi-Fi 芯片集）的。Wi-Fi Chipset 固件和工具文件必须复制到 WLAN 驱动程序插件目录中，并且必须在测试 Wi-Fi 功能前生成和安装。

Wi-Fi 驱动程序必须创建 `/opt/etc/.mac.info` 文件，该文件具有设备的 MAC 地址。

WLAN 驱动程序插件包含称作 `wlan.sh` (`/usr/bin/wlan.sh`) 的文件，该文件用于加载或卸载 Wi-Fi 驱动程序固件。所有其他 Wi-Fi 相关的功能均由 ConnMan daemon（ConnMan 守护程序）处理。

参考

- 连接管理器 (ConnMan) 项目网站：http://connman.net/
- Linux 无线 (IEEE-802.11) 子系统：http://linuxwireless.org/
- 有关 Linux WPA/WPA2/IEEE 802.1X 客户端的信息：http://hostap.epitest.fi/wpa_supplicant/
- 最新 ConnMan 版本：http://git.kernel.org/?p=network/connman/connman.git;a=summary
- WLAN 驱动程序插件 git 路径：`adaptation/devices/wlandrv-plugin-xxx`

参考内核配置

如果驱动程序支持 cfg802.11 配置 API，必须启用以下选项：

- CONFIG_CFG80211
- CONFIG_LIB80211
- CONFIG_MAC80211（如果驱动程序支持 softMAC 功能，则启用此标志。）

如果驱动程序支持无线扩展 API，则必须启用以下选项：

- CONFIG_WIRELESS_EXT=y
- CONFIG_WEXT_CORE=y
- CONFIG_WEXT_PROC=y
- CONFIG_WEXT_PRIV=y
- CONFIG_WEXT_SPY=y
- CONFIG_WIRELESS_EXT_SYSFS=y

3.4.2 蓝牙

Tizen 使用开放源蓝牙组件，例如 Bluez 和 Obexd。Bluez 和 Obexd 以守护程序的形式运行，并且应用程序使用蓝牙框架接口库通过 D-Bus 接口访问 Bluez 或 Obexd。

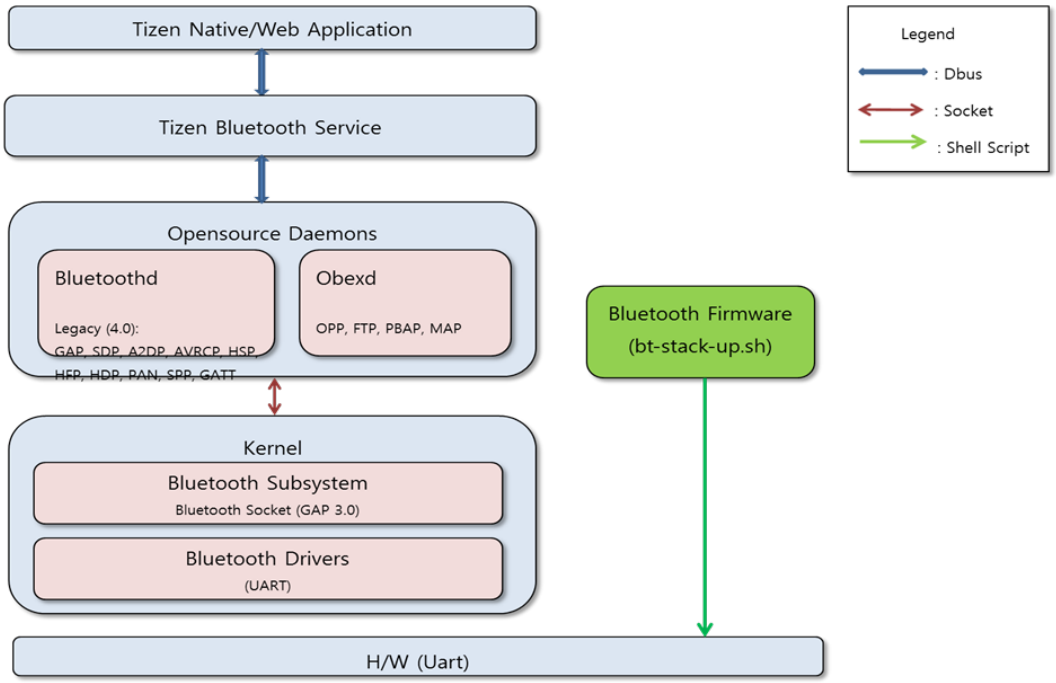

图 15. 蓝牙体系结构

- 支持的功能：GATT、FTP、OPP、MAP、PBAP、A2DP、AVRCP、HSP/HFP、RFCOMM、HID、HDP 和 PAN
- 应用程序定义：为用户提供对话框。控制 BlueZ/ObexD/PulseAudio 收集程序。

蓝牙低能耗功能是在 bluez&bluetooth-frwk 中实现的。但是，没有与 BLE 功能有关的 SDK API。目前计划是在 Tizen 3.0 版中包括它们。

移植 OAL Interface

以下 OAL 脚本在 BT 堆栈开始和结束序列期间运行。这些脚本调用 BT 芯片特定的（例如 Broadcom）脚本，芯片集供应商提供这些脚本以便执行芯片特定的配置。这些脚本连同 bluetooth-dev-tools 一起提供。在安装该程序包时，它会将这些脚本复制到 `/usr/etc/Bluetooth/` 下。

- `bt-stack-up.sh`：该脚本文件用于运行特定于硬件的脚本文件，以便开启或启动 BT 硬件以及后台进程，例如 bluez 和 obexd。
- `bt-stack-down.sh`：该脚本文件用于运行特定于硬件的脚本文件，以便关闭或停止 BT 硬件以及后台进程，例如 bluez 和 obexd。
- `bt-reset-env.sh`：该脚本文件用于通过运行 `bt-stack-down.sh` 重置 BT 芯片以及执行资源清理。

配置

必须进行某些配置更改以便使特定的芯片集、脚本和其他配置信息（例如 UART 速度和 UART 终端 (tty)）能够针对该芯片集打开。芯片集供应商必须提供所需更改。

以下设置是 Broadcomm 进行的 BCM4330 蓝牙芯片集的配置示例：

- hciattach
  对项目 `bluez/tools/hciattach.c` 进行了修补，以便支持 BCM4330 芯片集特定的 hciattach 工具。该服务以 3000000 的波特率将 BT UART HCI 接口附加到 BT 堆栈。它还负责在 BCM4330 上加载 BT 固件。

- 使用的蓝牙 UART 为 `/dev/ttySAC0`

- 使用的 Broadcom 固件为 BCM4330B1_002.001.003.0221.0265.hcd

- 针对 BCM4330B1 的 UART 速度配置为 3000000

- 使用的 bcmtool 为 bcmtool_4330b1

- .bd_addr 包含唯一蓝牙地址，这是在第一个蓝牙激活期间生成的

- 要注册蓝牙设备：

```
bcmtool_4330b1 /dev/ttySAC0 -FILE=BCM4330B1_002.001.003.0221.0265.hcd -
BAUD=3000000 -ADDR=/csa/bluetooth/.bd_addr -SETSCO=0,0,0,0,0,0,0,3,3,0 -LP
```

- 要使用 UART HCI 将串行设备连接到针对 broadcom 设备的蓝牙堆栈：

```
hciattach /dev/ttySAC0 -S 3000000 bcm2035 3000000 flow
```

- 要运行蓝牙守护程序版本 4.101：

```
bluetoothd
```

- 要启动设备、设置设备名称和启用 SSP 模式：

```
hciconfig hci0 up
hciconfig hci0 name “Tizen”
hciconfig hci0 sspmode 1
```

- 要开启蓝牙收音机：

```
rfkill unblock bluetooth
```

- 要关闭蓝牙收音机：

```
rfkill block bluetooth
```

3.4.3 NFC

下图阐释了 NFC 组件。

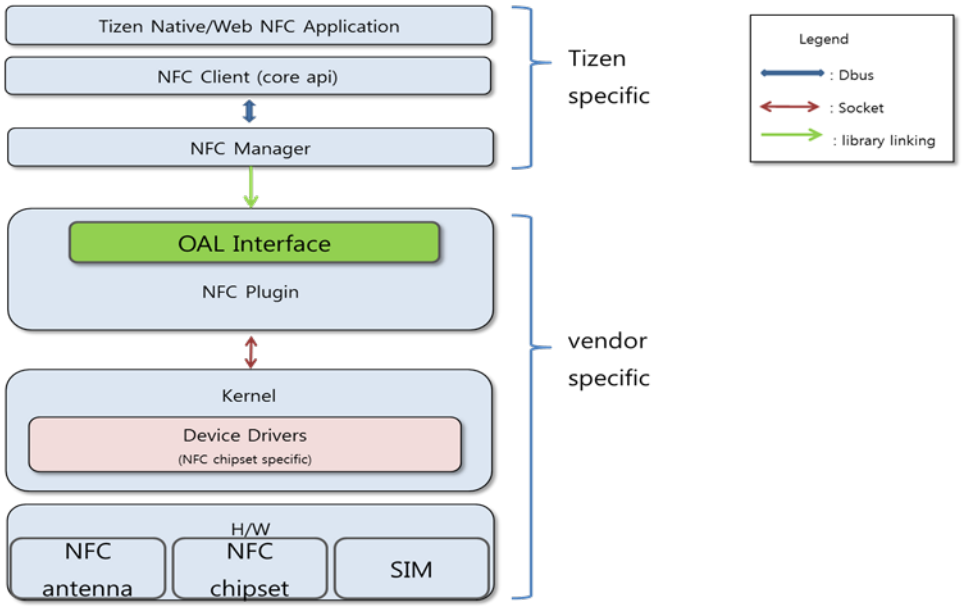

图 16. NFC 体系结构

- NFC 应用程序使用户能够读取和导入在 NFC 标记上写入的内容，以及在 NFC 标记中写入和保存数据。
- 在任何物理标记中写入或编辑标记信息时，NFC client（NFC 客户端）充当 NFC 应用程序和 NFC manager（NFC 管理器）之间的接口。
- NFC manager 是主接口，它实际上处理 NFC 物理标记、创建与标记的连接和检测标记。它是用于控制 NFC 芯片集（例如 nxp pn544 或 lsi）的守护程序进程。它提供标记读/写服务和基本的 P2P 通信服务。它向客户端应用程序提供基本的 API。
- NFC plugin 和 Device Driver 充当 NFC 芯片集和 NFC Manager 之间的接口。

如果您是芯片供应商（例如 nxp、Broadcom 或 lsi），则必须为 Tizen NFC 框架提供 NFC 芯片集、Device Driver 和 NFC plugin。有关可供参考的 NFC plugin 的实现方式，请转到 Tizen 公共代码网站并且下载 adaptation/devices/nfc-plugin-nxp.git。

要在 Tizen NFC 框架中使用任何特定的 NFC 芯片集，必须在芯片供应商提供的 NFC plugin 中创建 OAL Interface。

移植 OAL Interface

NFC plugin 作为共享库实现并且它连接 Tizen nfc-manager 和供应商 NFC 芯片。NFC manager 在运行时从 `{library_path}/libnfc-plugin.so` 加载 `libnfc-plugin.so` 库。任何供应商特定的插件都必须安装在相同路径中。该插件必须使用预定义的 OAL API 接口写入。

在初始化过程中，nfc-manager 加载 `libnfc-plugin.so`，搜索 `onload` API，并且使用接口结构实例作为映射所有 OAL Interface 的参数调用它。

OAL/OEM 接口是根据基本 NFC 芯片集实现的。在完成映射后，NFC manager 与 nfc-plugin 交互，这将实现供应商特定的 OAL Interface。

`nfc_oem_interface_s` 结构在 nfc-plugin 中导出。使用此接口结构，nfc-manager 在运行时与 OAL Interface 进行通信。

有关可供参考的 OAL Interface 的实现方式，请转到 Tizen 公共代码网站和下载 adaptation/devices/nfc-plugin-nxp.git，并且查看 src/oem/oem_nxp.c。

参考

- NFC 技术规范

3.5 Telephony

本节介绍 Telephony 模块。

Telephony 体系结构

Telephony 支持插件体系结构，它提供灵活性以便通过非常少的更改将不同类型的预定义插件包括在系统中。

下图阐释了 Telephony 插件体系结构。跨不同插件的函数调用执行 Tizen Telephony 库 (libtcore) API。

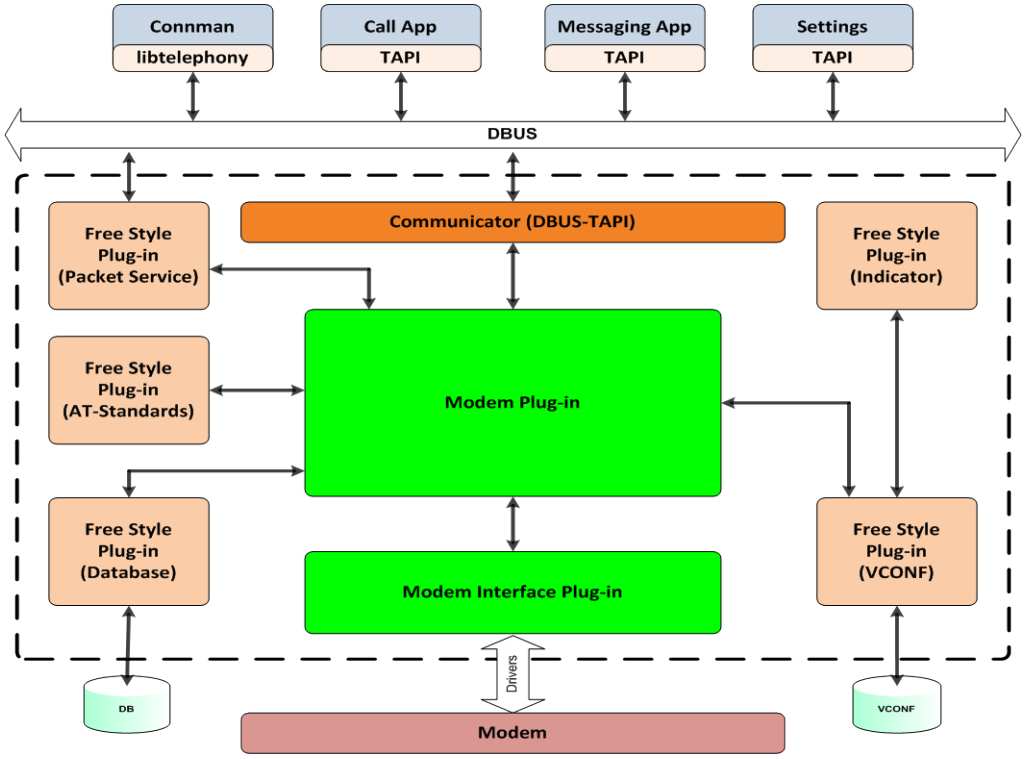

图 17. Telephony 体系结构

Telephony 库

Telephony 提供 2 个库：

- Telephony API (TAPI) 库
  TAPI 库（或简称为 TAPI）是向应用程序提供的以便与 Tizen Telephony 交互的一种标准接口。

- 核心 Telephony 库（或 libtcore）
  核心 Telephony 库（或 libtcore）为 Tizen Telephony 提供 API 框架以便进行交互工作。
  该 libtcore 提供 API 以便用于：

    - 不同服务器组件（例如服务器、Communicator、HAL、插件和核心对象）的创建、破坏和维护
    - 存储维护、队列机制和一般的实用工具
    - CMUX 支持（创建、破坏和处理）
    - AT 分析器

Telephony 插件

提供以下 Telephony 插件：

- Communicator 插件
    - 提供 DBUS Communicator 以便用于 TAPI 和电话服务器之间的接口。
- 调制解调器插件
    - 包含提供电话功能的核心功能单元。
    - 维护和管理电话状态。
    - 维护和管理与 Telephony 相关的数据库。
- Modem interface plug-in（调制解调器接口插件）
    - 电话服务器和通信处理器之间的接口。
    - 定义硬件功能和使用情况的硬件特定的插件。
    - Modem interface plug-in 也称作“硬件抽象层”(HAL)。
- 自由样式插件
    - 提供与硬件无关的完全独立的功能（通信处理器）。
    - 包括插件，例如程序包服务、存储和指示器。

移植 OAL Interface

OEM 供应商可以根据其需要在 Telephony 内移植每个可用插件。不必移植所有插件以便支持某一特定硬件。OEM 需要专门实现调制解调器和 Modem interface plug-in 以便支持其硬件。

任何 Telephony 模块都必须提供插件描述符结构（请参阅 `/libtcore/include/plugin.h`）。每个插件的描述符结构都必须命名为“plugin_define_desc”。服务器获取此符号的地址，以便提供对插件的控制以便执行其定义的功能。基于插件的“优先级”定义不同其他 Telephony 插件中插件的优先级顺序。

所有 Telephony 插件都必须按以下顺序安装：

- **调制解调器插件：**`/usr/lib/telephony/plugins/modems/`
- **其他插件：**`/usr/lib/telephony/plugins/`

3.6 Security

本节介绍 Security 模块。

3.6.1 访问控制 (Smack)

下图阐释了 Tizen 安全模型。

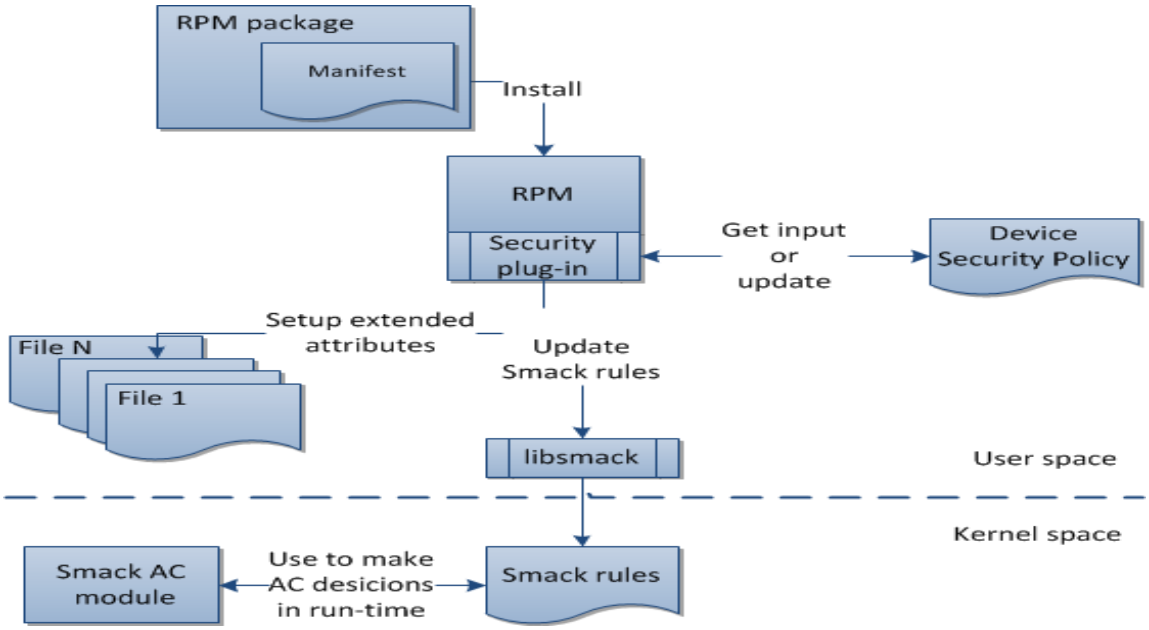

图 18. 安全模型

为了实现安全模型，Tizen 2.3 使用：

- 任意访问控制 (DAC)：文件系统权限和访问控制列表 (ACL)
  http://en.wikipedia.org/wiki/Discretionary_Access_Control
- 具有内核 LSM Smack 的强制访问控制 (MAC)
  http://en.wikipedia.org/wiki/Mandatory_access_control

Tizen 2.3 中的 SMACK

Smack 支持 2 个主要安全要求：

- 以应用程序颗粒度控制有权限的资源
- 具有受控的共享的应用程序隔离

基于 Smack 的安全模型

- 权限
  应用程序使用的服务必须控制调用方是否具有足够的权限来调用各 API。在 Tizen 2.3 中，此级别的访问控制是使用针对 IPC 机制的非常详细的 Smack 策略实现的。换言之，权限由 smack 规则构成。您可以在 `/usr/share/privilege-control/*` 中确认这一点。

    - 有关 Tizen 权限的概述，请参阅：https://developer.tizen.org/dev-guide/2.2.1/org.tizen.gettingstarted/html/tizen_overview/privilege.htm
    - 有关 Tizen 权限的列表，请参阅：https://www.tizen.org/ko/privilege?langredirect=1

- Manifest（清单）
  Web 或本机应用程序开发人员必须在清单文件中描述他们要使用的权限。此外，每个 RPM package（RPM 程序包）都必须具有一个清单文件，在该清单文件中，开发人员可以指定其应用程序必须在其中运行的访问控制域以及针对该应用程序的潜在的附加安全策略。

- 安装应用程序
  安装程序使用该清单文件的内容设置已安装应用程序或 RPM package 的安全上下文。安装程序使用 libprivilege-control 组件处理 Smack 安全数据库。安装程序在 `/opt/dbspace/.rules-db.db3` 中更新应用程序的 smack 规则的数据库。RPM 程序包的 smack 规则存储于 `/etc/smack/accesses.d/*` 中。

- Smack 审核
  默认情况下，审核所有被拒绝的事件。Kernel-space 拒绝事件保留在 dmesg 上。此外，IPC 机制上的任何 user-space 拒绝事件都记录在 dlogutil 上。

参考

对于 smack，请参阅：

http://en.wikipedia.org/wiki/Smack_(Linux_security_module)

有关清单和安装应用程序，请参阅：

https://wiki.tizen.org/wiki/Security/WebApps_and_Smack#Manifest_for_WebApps

有关清单和安装 RPM package（平台模块），请参阅：

https://wiki.tizen.org/wiki/Security/Application_installation_and_Manifest#Manifest_file

3.6.2 证书管理

在公钥基础结构 (PKI) 架构中，使用证书来证明公钥的所有权。因为 Tizen 使用公钥架构，所以，实现 Cert-svc（证书服务）和 Cert-svc-vcore 来有效地管理证书，并且只允许使用有效的签名来安装应用程序。

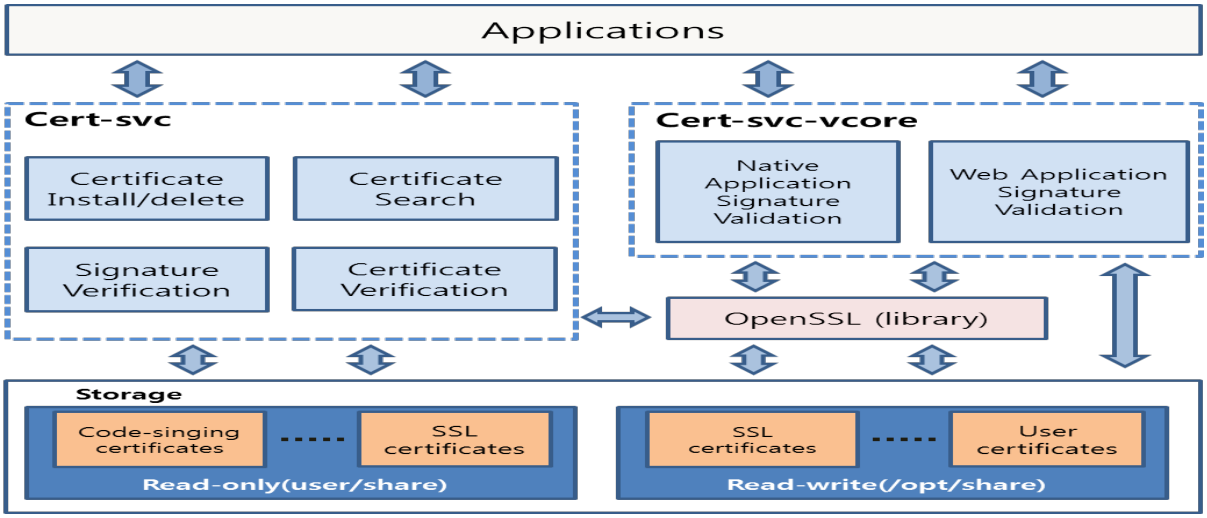

图 19. 证书管理

下表列出了证书组件的主要功能。

表 2. 主要功能

	功能	说明
Cert-svc	Certificate Installation（证书安装）	将证书安装到指定位置。
	Certificate Deletion（证书删除）	删除不需要的证书。
	Certificate Information Extraction（证书信息提取）	从 X.509 结构证书提取信息。
	Certificate Search（证书搜索）	使用用户请求信息搜索证书。
	Certificate Verification（证书验证）	检查是否正常发布证书验证。
	Signature Verification（签名验证）	使用消息、签名和证书验证签名。
Cert-svc-vcore	Native App Signature Verification（本机应用程序签名验证）	检查对本机应用程序签名的验证。
	Web App Signature Verification（Web 应用程序签名验证）	检查对 Web 应用程序签名的验证。

参考

- 证书服务编程指南版本 0.1

3.6.3 防病毒（CSR 框架）

CSR（内容筛选和声誉）框架由内容安全性框架和 Security Plug-in（安全性插件）构成。在 Tizen 中，它们是共享库。

- 内容筛选：
  使调用方模块和应用程序能够扫描内容（数据、文件）。

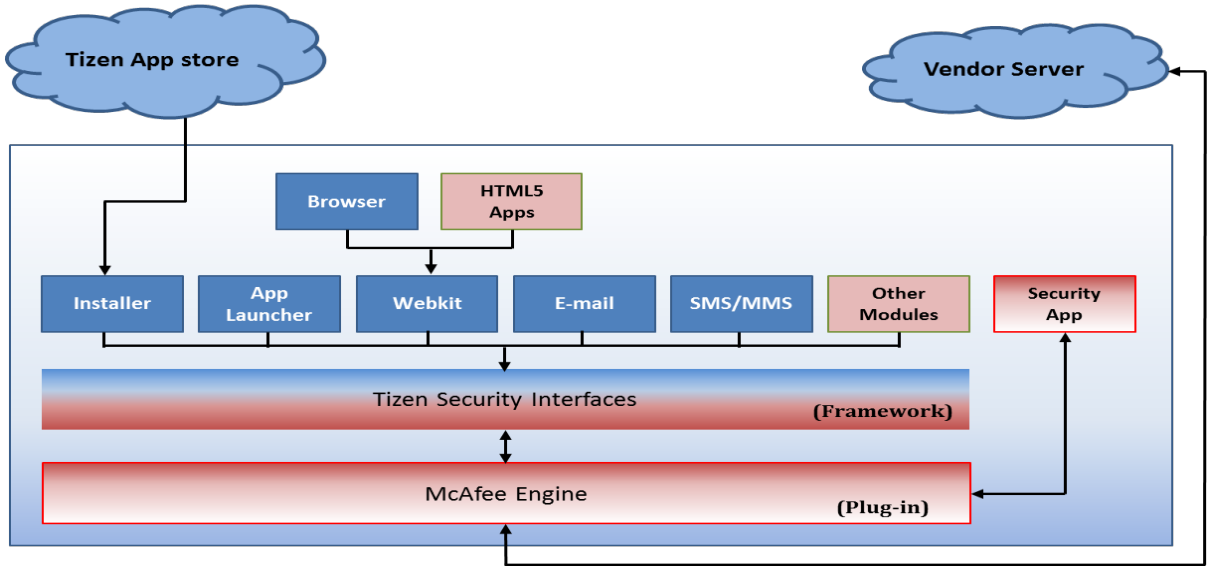

图 20. 内容筛选

- 声誉（Web 保护）：
  可以保护世界各地的用户免受基于 Web 的恶意软件威胁、浏览器安全漏洞和身份盗窃。

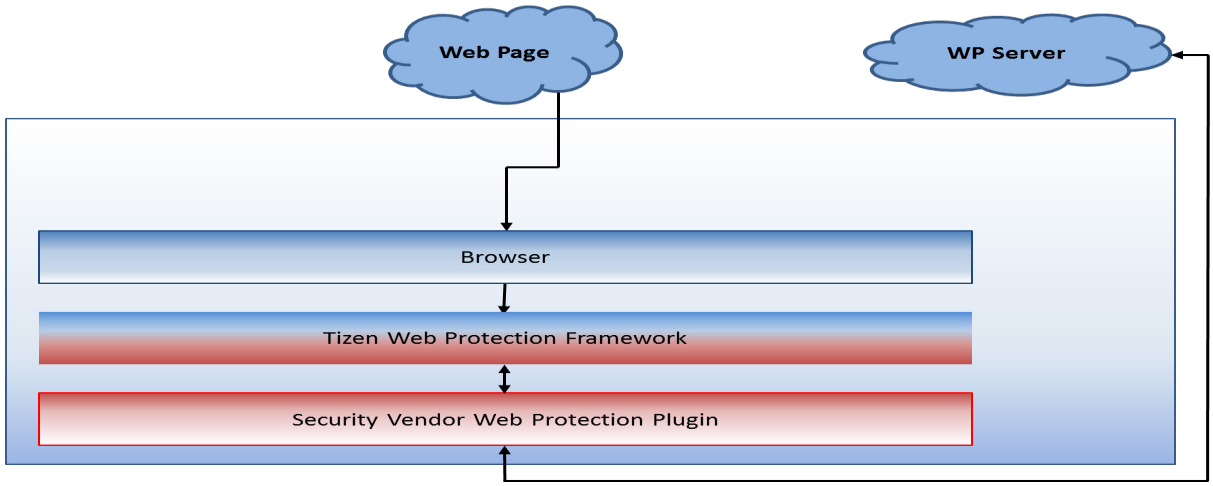

图 21. 声誉

内容安全性框架

- 共享库：libsecfw.so
  该文件直接链接到调用安全 API 的系统组件。它负责插件（引擎）加载和重新加载。
  请始终尝试从 `/opt/usr/share/sec_plugin/` 路径加载新插件。

Security Plug-in

- 共享库：libengine.so（内容筛选）、libwpengine.so（声誉）
  Security Plug-in 在运行时加载并且随安全应用程序包一起安装。

防病毒应用程序包格式

- 程序包格式必须符合 Tizen 应用程序格式，这是一个简单的压缩文件。
- 程序包包含：
  - `/bin/…`（应用程序可执行文件）
  - `/res/…`（图标、图片和其他资源文件）
  - `/data/…`（应用程序数据）
  - `/lib/plugin/libengine.so` 或 `libwpengine.so`
    （安装程序将此文件复制到 `/opt/usr/share/sec_plugin/libengine.so` 或 `libwpengine.so`。）
  - `/database/…`
    （引擎数据。例如，签名数据库。每个人都可以读取该目录。）
  - `/info/manifest.xml`（包含正确的应用程序类型。例如 <category=“sec-app” />。）

Tizen 安装程序增强功能

- 检查签名/证书并且确保它是由受信任方进行验证的。
- 将 `/lib/plugin/libengine.so` 复制到 `/opt/usr/share/sec_plugin/libengine.so` 或 libwpengine.so.
- 将所有文件复制到 `/opt/usr/apps/[package id]/…`。
- 将 `/opt/usr/apps/[package id]/database` 的 smack 标签设置为“sec_database”。
- 使用内容安全性框架的所有应用程序都已具有用于读取“sec_database”的规则。
- 为安全性应用程序提供权限。
  （用于安全性检查的权限，例如对文件系统或其他应用程序的访问。）

参考

- Tizen 内容安全性框架建议（文档版本 1.0.2 2013，McAfee，Inc.）

3.7 Location

Location 模块提供基于位置的服务 (LBS)，包括位置信息、卫星信息和 GPS 状态。

主要功能是：

- GPS（全球定位系统）
- 获取当前位置、上次已知位置、精确性、距离和速率
- 获取 GPS 和 GLONASS 的卫星信息
- 在用户进入或离开预定义的通称为地理分隔的边界集（例如校园出勤区域或相邻边界）时通知用户。

Location 框架

下图阐释 Location 框架。

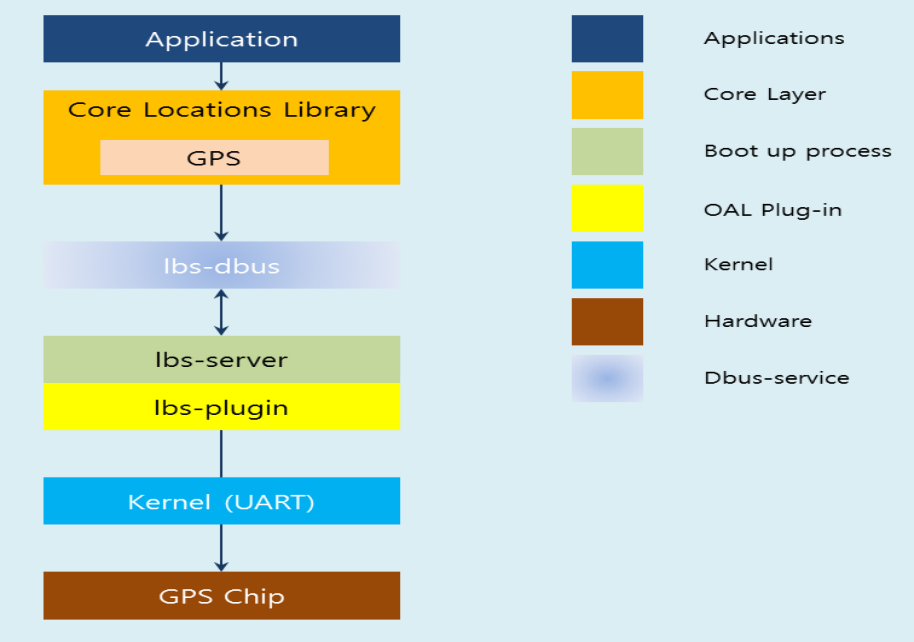

图 22. Location 框架

- 位置库：
    - 包含本机位置可用来获取服务的位置提供程序。
    - GPS 提供位置信息、速度和卫星信息。它用于获取某一设备的当前位置。

- Lbs dbus：
    - 这是用于在 Location 模块和 GPS 管理器守护程序之间进行通信的 IPC。
- LBS 服务器：
    - 通过与 GPS 芯片进行通信提供位置、速度、NMEA 和卫星信息。
    - LBS 服务器的功能：
      - GPS 初始化/取消初始化和打开/关闭控制。
      - 为位置库提供位置结果。
      - 针对基于会话状态的会话终止的位置会话管理确定。
      - 用于 GPS 接收器的串行接口。
      - 使 GPS 芯片集能够支持独立的 GPS 定位方法。
      - 支持独立的操作模式，在该模式中，接收器提供所有自己的数据的 GPS 接收器设备需要并且执行所有位置计算，并且无需连接到外部网络或服务器。

移植 OAL Interface

LBS 插件是基于供应商特定的 GPS 设备的 Tizen LBS 服务器实现的。

LBS 插件是作为共享库实现的，而 lbs-server 在运行时加载特定的 LBS 插件。LBS 插件必须使用预定义的接口写入。通过在程序包规格文件中添加以下命令，将 lbs-server-plugin-dev 源程序包安装在 OBS 上：

- `BuildRequires:pkgconfig(lbs-server-plugin)`

在 lbs-server-plugin-dev 程序包内，源文件位于：

- `/usr/include/lbs-server-plugin/*.h`
- `/usr/lib/pkgconfig/lbs-server-plugin.pc`

`gps_plugin_intf.h` 文件包括用于 lbs-server 和其 GPS 插件之间的通信的 API 接口：

```
typedef struct {
   /** 初始化插件模块并且注册针对事件交付的回调 */
   int (*init) (gps_event_cb gps_event_cb, gps_server_param_t * gps_params);
   /** 取消初始化插件模块 */
   int (*deinit) (gps_failure_reason_t *reason_code);
   /** 请求针对插件模块的特定操作 */
   int (*request) (gps_action_t gps_action, void *data, gps_failure_reason_t *reason_code);
} gps_plugin_interface;

const gps_plugin_interface *get_gps_plugin_interface();
```

`get_gps_plugin_interface()` 必须在 LBS 插件中导出。它向 lbs-server 提供 gps_plugin_interface 结构，而 lbs-server 通过这些接口进行通信。在启动 gps-manager 时，将加载 GPS 插件并且调用 `init()` 函数。此时，必须初始化 GPS 设备（例如，用于电源控制和固件下载）。

4. 优化

通常，使用 2 个主要方法来执行优化任务。第一个方法是静态（源代码）分析，它通常仅在相当简单的应用程序中是有帮助的。在复杂软件中常常很难应用静态分析，例如，提供具有不同执行线程、系统 IPC 使用率和第三方模块（插件）支持的许多服务的复杂软件。在该情况下，您必须针对不同的使用情形在运行时分析应用程序行为，而第二种方法（即动态分析）更有效。Tizen 平台提供用于动态应用程序分析的必需的工具。

该动态分析过程划分为几个基本阶段：

- 分析会话准备
- 数据收集
- 结果的可视化

在准备阶段中，开发人员选择为进行分析而当前需要的数据：兴趣点（例如库或函数）、要分析的功能（例如内存、文件 IO 和网络）以及其他参数（例如抽样期间）。所有这些数据都是在分析会话期间在目标设备上收集的，并且发送回主机系统。主机对接收的数据进行分析和存储，执行所需的分析，并且以肉眼可读取的形式显示结果。该方法允许开发人员观察程序在复杂环境中的行为方式以及不同的系统方面是如何影响它的。下图阐释了 Tizen 应用程序的整个分析过程。

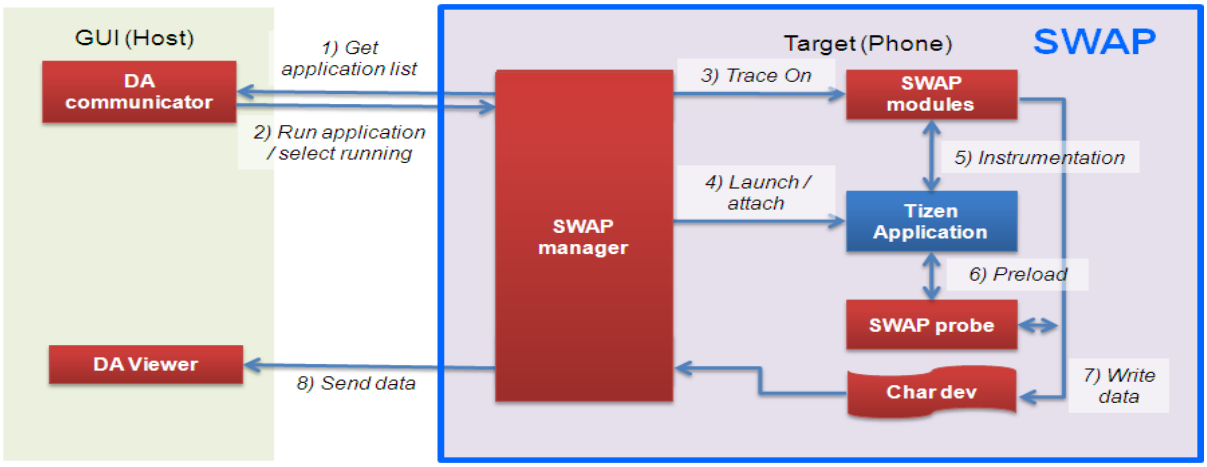

图 23. 分析过程

性能的系统范围的分析器 (SWAP)

性能的系统范围的分析器 (SWAP) 是一个动态的二进制工具引擎，允许您执行系统范围的性能、内存、耗电量分析和优化。它基于 kprobe 机制，并且提供工具来收集在运行时对应用程序和整个系统行为进行分析所需的所有数据。SWAP 用作在目标设备上分析数据集合的后端。

SWAP 关键功能：

- Kernelspace 分析：
    - 可以在命中给定地址处的说明时执行自定义处理程序。
    - 函数分析：在进入和退出指定的函数时执行自定义处理程序（例如，perform、参数或返回值分析或时间戳记录）。
    - 例如，截取处理程序、计划程序、驱动程序、系统调用、IPC、IO、图形和输入事件的分析。
- Userspace 分析：
    - 具有自定义进入/退出处理程序的函数工具。
    - 多线程和多进程应用程序支持。
    - 库工具，包括从所有正在运行的进程收集数据的系统范围的模式。
    - “已在运行”分析：无需应用程序/系统重新启动。
- 轻松的数据提取：
    - 例如，函数参数（kernel 或 userspace）、任务注册和当前系统参数。
- 多架构支持：
    - ARM、ARM64、x86 和 MIPS。
- 无需源代码或调试信息。
- 无需程序/内核重新编译。
- 模块化体系结构：
    - 轻松实现针对特定任务（例如，内存分析、性能度量或耗电量评估）的基于 SWAP 的新工具。
- 可作为单独模块编译或内置于内核映像中。

动态分析程序 (DA)

动态分析器是设计为基于从目标设备收集的数据执行应用程序行为分析的 GUI 前端。它提供：

- 时间线图表：CPU 负荷、内存、缓存、网络使用情况、UI 事件、应用程序生存期和自定义图表
- 汇总：失败的 API、资源泄露、功能分析和警告
- 分析组件：文件 IO、线程、UI、网络、OpenGL® ES、计划和系统信息
- UX 和其他信息：调用跟踪、记录和重放、源代码链接、基于范围的分析、运行时屏幕快照、服务/混合应用程序支持、平台库支持和多进程支持

DA 关键功能：

- 查找可能的优化点，例如瓶颈和不需要的等待。
- 分析问题的原因，例如泄露和功能使用错误。
- 效率分析：网络、OpenGL® ES

附件 A. 词汇表

表 3. 词汇表

术语或缩写	说明
ALSA	高级 Linux 声音体系结构
EAP	可扩展身份验证协议
Tizen	用于设备（例如移动设备、电视和 IVI）的基于 Linux 的开放源操作系统
V4L2	Video4Linux2
WLAN	无线局域网
WPA	Wi-Fi 保护的访问
WPS	Wi-Fi 保护的设置