Introduction to SLP
Samsung Linux Platform

삼성전자 DMC 연구소
전해돈
2010.10.05
공개 SW 개발자 대회

Contents

LiMo Foundation 현황

- 2007년 1월에 공식 출범, 현재(10.7) 46개 회원사(11 Board 포함)
- 목표: Open, H/W-Independent, Linux-based Platform for Mobile Devices

Board Members (11)

제조사	사업자	기 타
SAMSUNG, LG, Panasonic, NEC	vodafone, verizon wireless, NTT DoCoMo, Telefonica, SK telecom	ACCESS, WIND RIVER

Core Members – Without Board Seat (2)

Aplix, orange

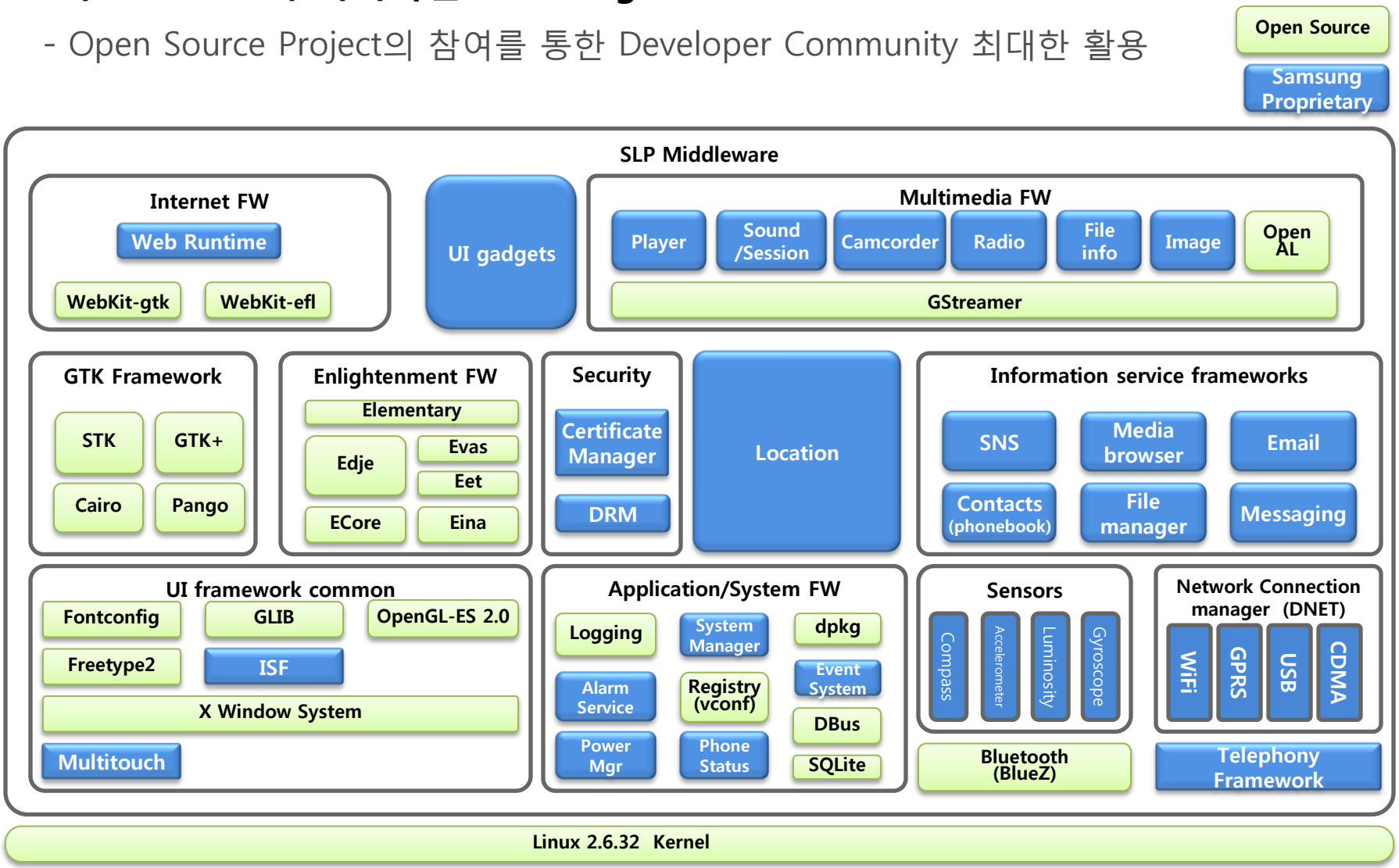
Associate Members (33)

Acrodea, ARM, Broadcom, ETRI, KDDI, EB, Huawei, alleh kt, Broadsoft, Ericsson, immersion, Kvaleberg, Marvell, Movial, Opera software, Renesas, McAfee, mozilla corporation, pv, Samsung SDS, Motorola, Openplug, redbend software, SFR, SK innoace, swisscom, ZTE中兴, SoftBank, TELECOM, myriad, Texas Instruments, MOBILE

SLP Overview

최신 Device에 최적화된 Samsung Linux Platform

- Open Source Project의 참여를 통한 Developer Community 최대한 활용

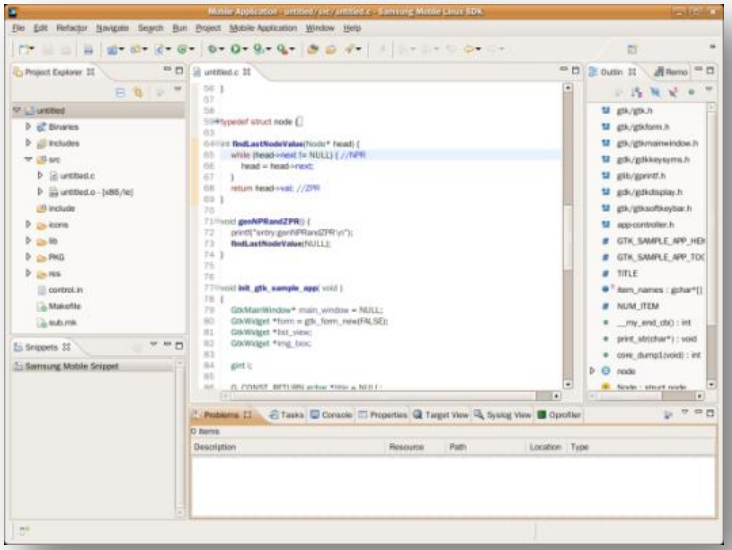


Application Basics

- **Applications are written in C/C++.**
  - Uses platform components (open source + middleware).
  - EFL/GTK can be used to design the UI.
- **Each application runs in its own Linux process.**
- **Follows lifecycle defined by application framework.**
- **Applications are controlled by the Linux kernel.**
- **Multitasking support.**
  - Applications can execute in background.
- **Application Relaunch support Multiple Instance Support**
  - Default behavior is to resume previous state
  - Applications can choose to be re-launched instead of resume.
- **Debian packaging system is used for applications.**

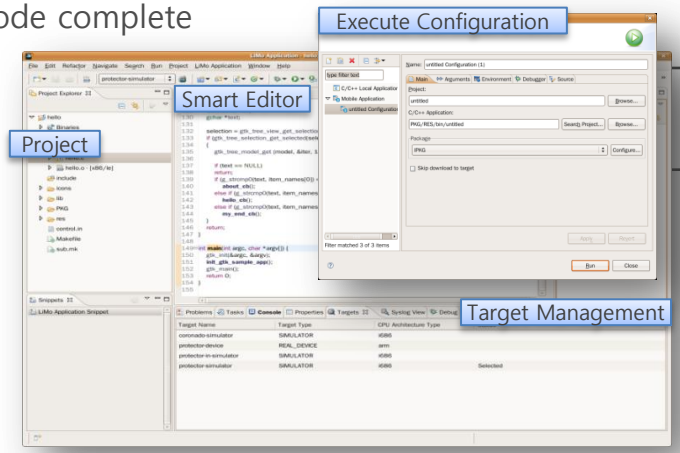
SDK (Software Development Kit)

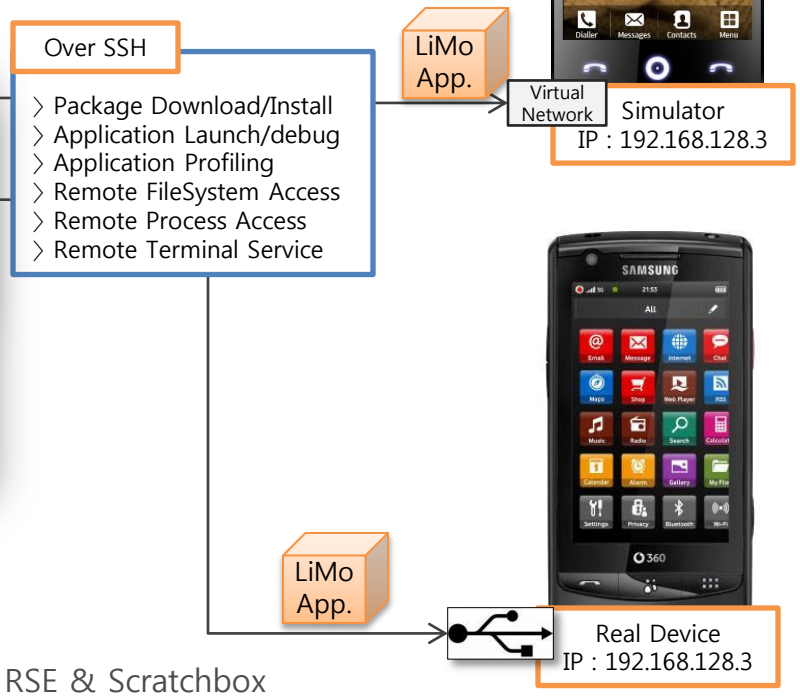
- Definition
  - A set of development tools that allows a software engineer to create applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar platform
- Components
  - Application programming interface (API) in the form of files
  - A certain embedded system or a simulator (emulator)
  - Tools and utilities such as debugging aids, IDE and profiler
  - User/development/API guides, tutorials, technical notes, sample code



SDK Tool Feature - IDE

- Eclipse-based
  - Open Source Project, Host Support : Ubuntu 8.04, 9.10
- Project Management
  - Project Template / Sample Applications
  - Convenient Build Configuration (using pkg-config)
- Smart Editor
  - API Tooltip (Hover) with HTML Link
  - Snippet view support for often used codes
  - Auto code complete
- Execution
  - Consistent GUI for launch / debug / profile
- Target Management
  - Convenient controlling for all the remote targets using RSE & Scratchbox





SDK Tool Feature – Simulator

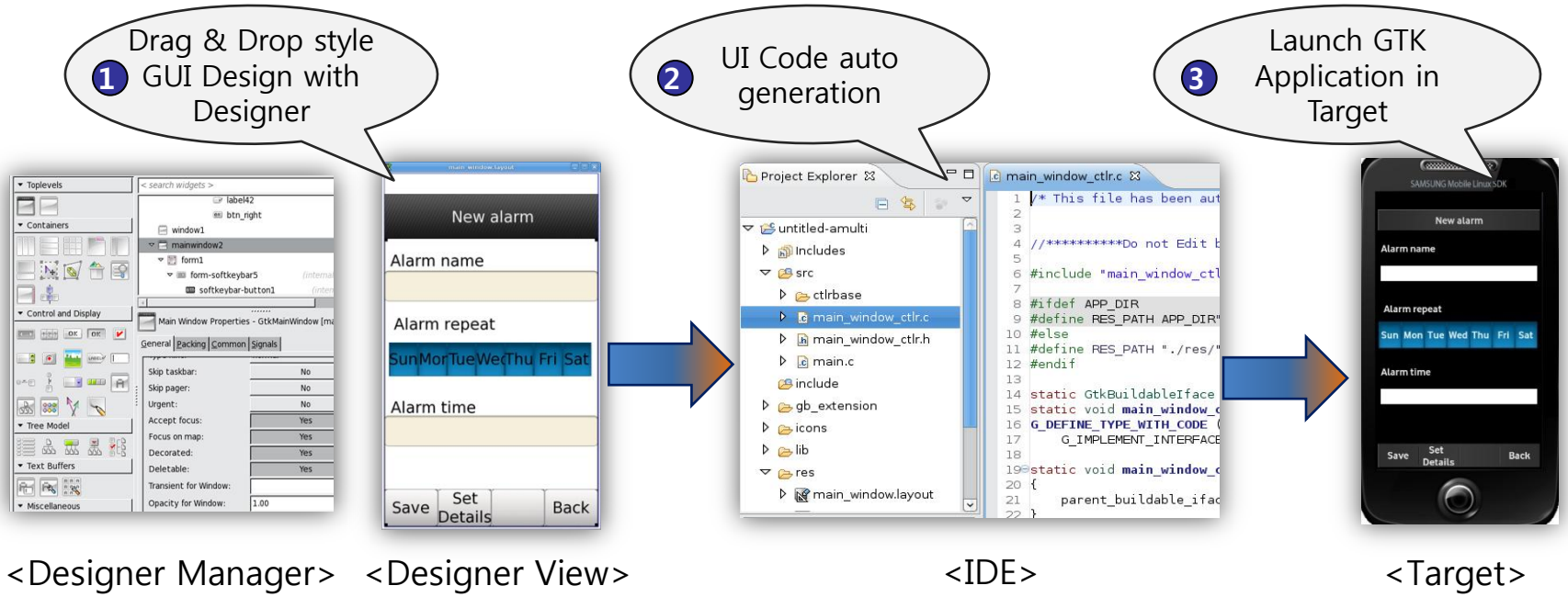
- QEMU-based H/W emulation
  - LiMo based OS over Host OS (Ubuntu)
- Support KVM (Kernel-based Virtual Machine)
  - KVM allows a user space program to utilize the hardware virtualization features
- Support various virtual device
  - Touch, Network, Sound, OpenGL, Rotate device, GPS, SD card etc
- Telephony Emulator
  - Incoming & Outgoing Call, Sending & Receiving SMS, Supplementary service

SDK Tool Feature –GUI Builder

- Support easy and rapid UI design
  - XML based WYSIWYG GUI Editor
  - Support Portrait/Landscape screen mode
  - Support various mobile gtk/stk widget
  - UI Code auto generation

- Support various mobile UI Elements
  - Support various type widget over 40

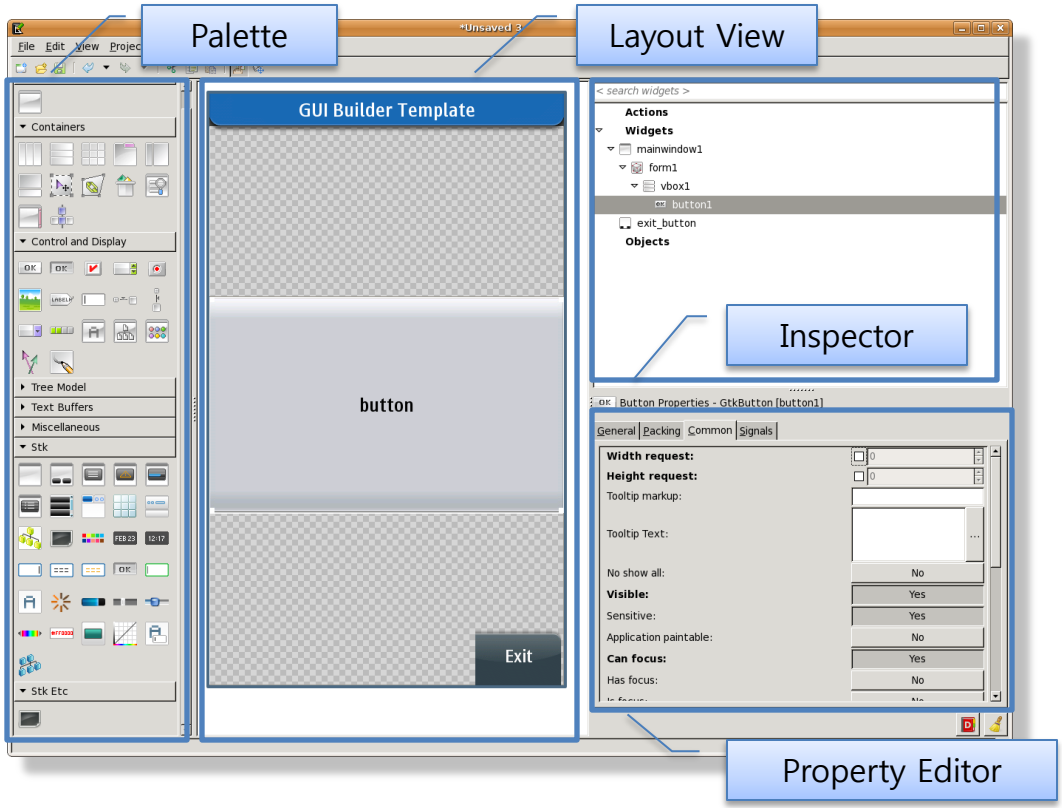
Category	Count
GTK Widget	40
STK Widget	29
* EFL Widget	30



<Designer Manager> <Designer View> <IDE> <Target>

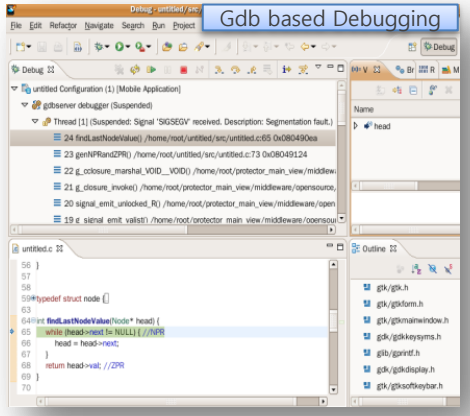
SDK Tool Feature- GTK Designer

- **Palette : List available widgets which are used by designer**
- **Layout View : Display the layout**
- **Inspector : Show the hierarchy of widgets including current xml file**
- **Property Editor : Edit the value of properties of each widget**
  - General / Packing / Common / Signal

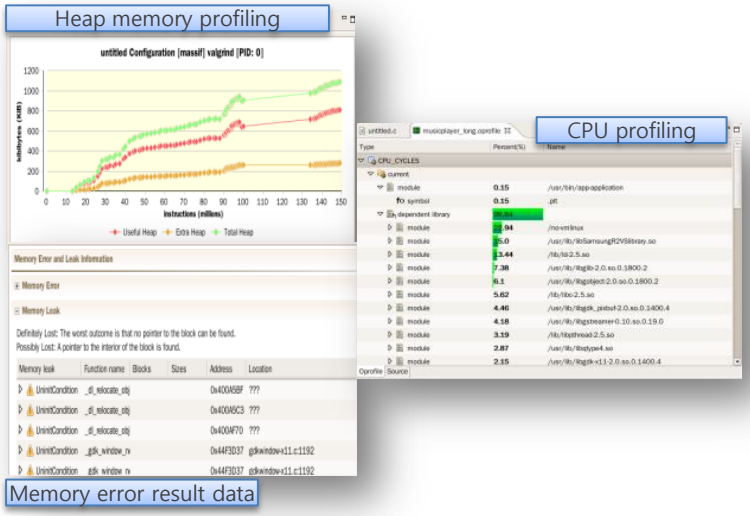


SDK Tool Feature – Debugging & Profiling

- **Various debugging method support with gdb and logUtil**
  - Normal/Attach Debugging
  - Postmortem Core-dump Debugging
  - Debug Message Viewer using LogUtil View



- **Powerful Memory, CPU usage Analysis**
  - Valgrind-based Memory Profiler
  - Memory Leak/Error, Dynamic Heap Analysis
  - OProfile-based Performance (CPU) Profiler



Dev Site Screen Shot

SAMSUNG Linux Platform SLP-SDK

All Sites

SLP-SDK | Intro | Support | Downloads | DEV Guide | Notice | Store | Admin_site_usage_report

View All Site Content
Intro
- SDK Overview
- Platform Overview

Support
Downloads
Notice
Recycle Bin

SLP-SDK

Development Guide for great SLP Application

Developer Library Everything you need to know about developing and distributing with SLP.

API Reference Library This document provides descriptions for all SLP APIs

Download SLP SDK

The SLP SDK provides what you needs to creat great apps-you gen get the tools,sample code, and document. **Learn more**

Getting Started Explore the basic concepts behind building your own app with a quick overview of the developer tools.
Learn more

Overview of SLP

Samsung Linux Platform Fundamental overview infomation

Learn more

SLP Application FW Guide

Learn more

Application Dev Process with SDK Explore the process of developing a SLP application including the tools you need to use and the main steps.
Learn more

Application Store Guide Publish your applications.
Learn more

Developement Tool video Guide Learn more

Notice

New SDK release version SDK-SLP-20100929-pre2 10/1/2010 6:09 PM
by 서강훈/Mobile S/W Platform Lab(DMC연)/E4(선임)/삼성전자
ftp://165.213.149.206/pub/Tools/SDK/SLP2.0_SDK/Linux_SDK/20101001/SDK-SLP-20100929-pre2.tar.gz
All versions of SDK Platform release packages are downloadable for free. The TAR files we provide through [Download(Binary)] are basically composed of Samsung…

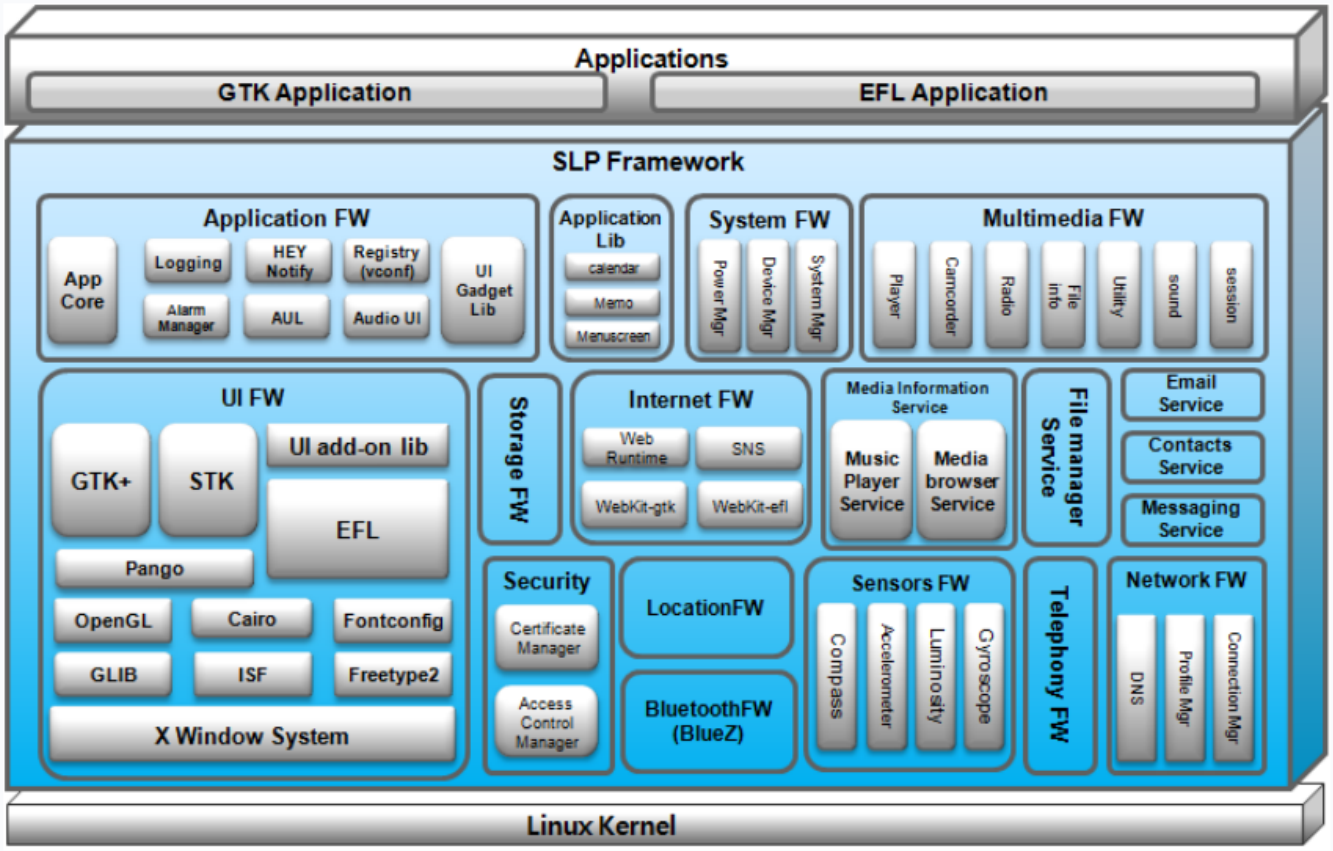
Dev Site Screen Shot

SLP API 1001__12:16 version

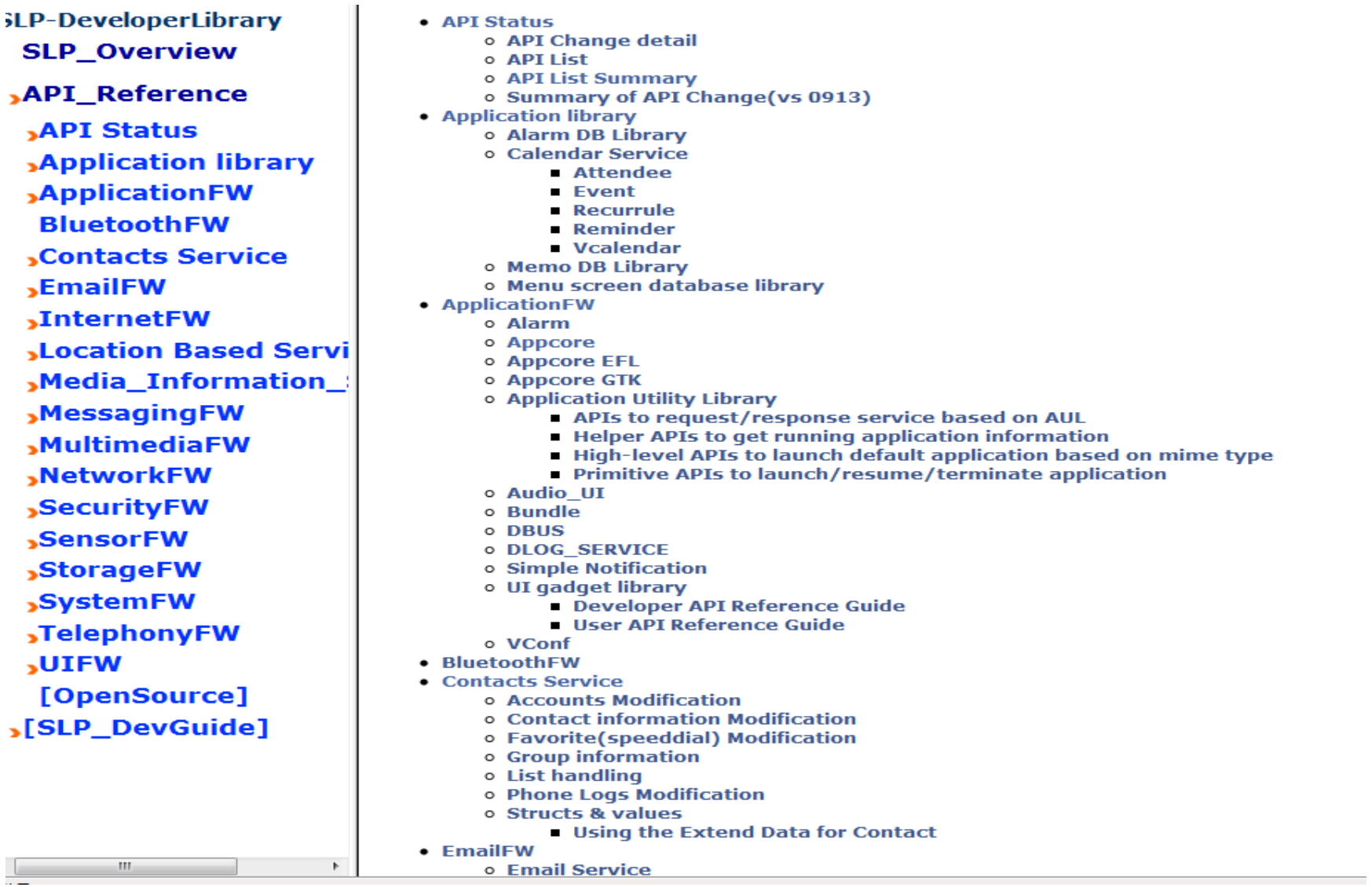
Platform Architecture

SLP software stack has a layered architecture consisting of

- a. Applications
- b. Middleware & System libraries
- c. Linux Kernel



Dev Site Screen Shot



App Developer's Perspective(1/2)

- **Commercial Pragmatism**
  - In the last two years, mobile SW and apps have become the essential marketing playbook for mobile industry vendors.
  - At the same time, software developers have grown to be much more knowledgeable, pragmatic and savvy about the economic implications of mobile development.
- **Market penetration**
  - The most important reason for selecting a platform
  - Developers care more about addressable market and monetisation potential than any single technical aspect of a platform.
- **Platform concurrency**
  - Most developers work on multiple platforms – on average, 2.8 platforms per developer
  - Among iPhone and Android developers, 20% releases apps in both Apple and Android App Store
- **Mindshare vs addressable market disconnect**
  - Symbian is deployed in around 390 million handsets (Q2 2010), and claims over 6,000 apps
  - Apple's iPhone has seen 30x more applications while being deployed at just 60 million units
- **Developer bias**
  - Most developers have a head-strong affinity towards the platform they have invested time in

App Developer's Perspective(2/2)

- **Learning curve and efficiency**
  - On average, Symbian takes 15 months or more to learn; Android is less than 6 months
  - A Symbian developer needs to write almost three times more code than an Android developer, and twice as much code as an iPhone developer.
- **Development environment**
  - Top pain points for mobile emulators and debuggers are slow speed and poor target device mirroring.
- **UI tools**
  - Ability to build compelling UIs is still far from the reach of most mobile developers.
  - Around 50 out of 100 Symbian, BlackBerry and Windows Phone perplatform respondents are annoyed with the difficulty in creating great UIs.
- **Support**
  - The majority of developers (> 80%) rely on community or unofficial forums for support, while websites are used for support by only 40%.
- **Open source**
  - Android and iPhone developers are three times more likely to lead open source communities, compared to Symbian
  - The single key drawback to open source was the confusion created by open source licenses

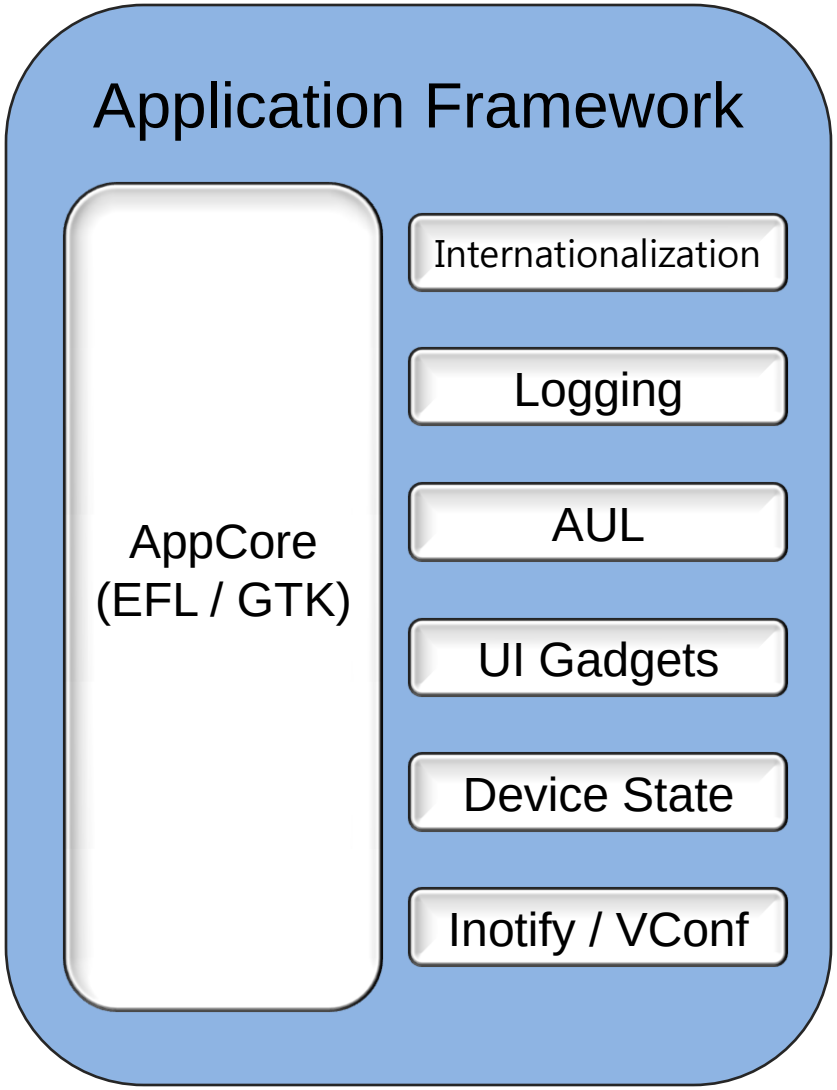
Conclusion

- **The success of the platform and application developers are aligned**
- **Samsung is aware of that and plan to support app developers**
  - Platform 경쟁력 제고를 위한 Ecosystem 구축
    - 개발자 Community 및 Business 파트너 사이의 Virtuous Cycle 체계 구축
  - 11월말: Platform & SDK Contribution to LiMo
- **Samsung will ship SLP phones from 2011**
  - 2011. 2월 MWC: SLP Phone 발표
- **Now is the time to learn and develop SLP applications**

Appendix

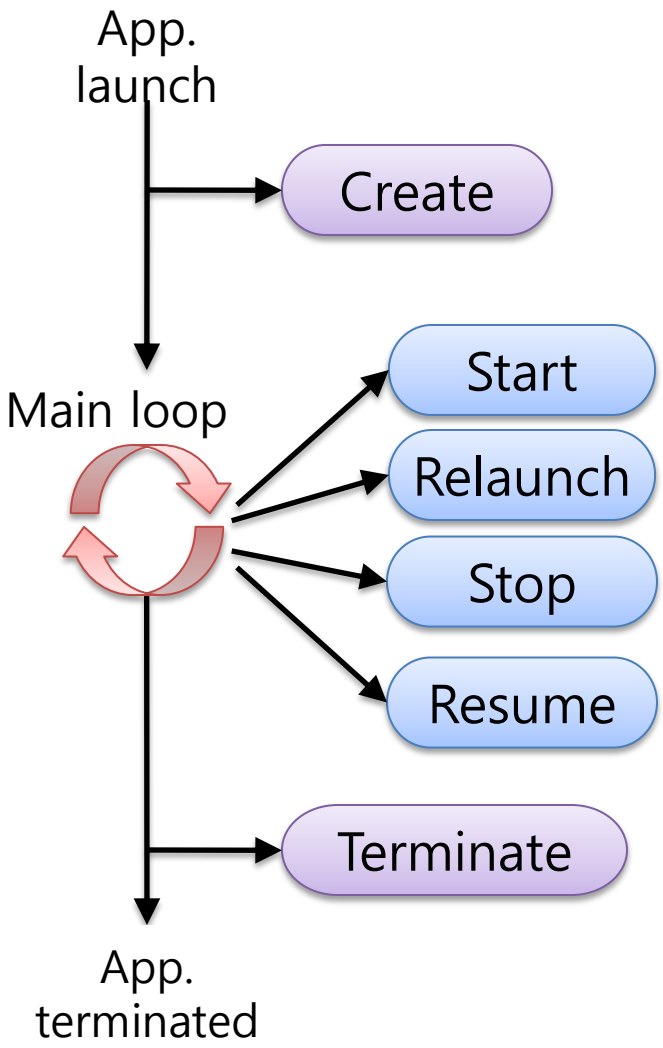
- Application Framework
- Application Lifecycle

Application Framework



- **Application functions**
  - Main window for application
  - Lifecycle management
  - Internationalization
  - Logging
- **Sharing functions**
  - Application Utility Library (AUL)
  - UI Gadgets
- **Device state functions**
  - Power, Memory, Rotation
- **Other functions**
  - Inotify, Alarm
  - Vconf

Application Lifecycle (1/4)



Operation	Description
CREATE	Called once before the main loop. Initialize the application such as window creation, data structure allocation, and etc.
START	Called when Application is first running and ready.
RELAUNCH	Called at the first idler and every "relaunch" message. Reset the application states and data structures.
STOP	Called when the window of the application becomes invisible. Recommend to block the actions related to the visibility during this state.
RESUME	Called when the window of application becomes visible. Resume the paused actions.
TERMINATE	Called once after the main loop. Release the resources.

*AUL=Application Utility Library

Application Lifecycle (2/4)

Basic Application

```
static int app_procedure(int event, void *event_param, void *user_param)
{
    switch (event) {
        case APPCORE_EVENT_CREATE:
            app_create(user_param);
            return 1;
        case APPCORE_EVENT_RESET:
            app_start(user_param);
            return 1;
        case APPCORE_EVENT_LOW_MEMORY:
            /* cannot handle */
            return 0;
        case APPCORE_EVENT_TERMINATE:
            app_terminate(user_param);
            return -1;
...
}
int main(int argc, char *argv[])
{
    return appcore_gtk_main("AppName", argc, argv, app_procedure, 0, user_param);
}
```

→ Application handled the event

→ App. Framework handle the event

→ Quit the application

Application Lifecycle (3/4)

Application + Background Scheduling

```
static int app_procedure(int event, void *event_param, void *user_param)
{
    switch (event) {
...
        case APPCORE_EVENT_PAUSE:
            app_stop(user_param);
            return 1;
        case APPCORE_EVENT_RESUME:
            app_resume(user_param);
            return 1;
...
}
int main(int argc, char *argv[])
{
    return appcore_gtk_main("AppName", argc, argv, app_procedure, 0, user_param);
}
```

→ "Home" button pressed
App. goes to background

→ Application Icon pressed
App. comes to foreground.

Application Lifecycle (4/4)

Application + Background Scheduling + Re-Launch

```
static int app_procedure(int event, void *event_param, void *user_param)
{
    switch (event) {
...
        case APPCORE_EVENT_RESET:
            app_relaunch(event_param, user_param);
            return 1;
...
}
int main(int argc, char *argv[])
{
    return appcore_gtk_main("AppName", argc, argv, app_procedure, 1, user_param);
}
```

→ **Handle re-launch request**