Faysal Aziz

Android Platform Internals and AOSP Architecture Masterclass

From Android Framework to Kernel: Understanding the Complete Android System Stack

FA-0098 · two-day

Software Development · DevOps, Cloud & Infrastructure

Introduction

Modern Android platform development requires engineers to understand far beyond application development. Organizations building customized Android platforms, embedded Android products, BSPs, middleware, system services, and hardware integrations need engineers who can navigate the complete Android software stack, from Java framework services through Binder IPC, native C++ layers, HAL implementations, and Linux kernel interactions.

This intensive two-day program provides a practical understanding of Android Open Source Project (AOSP) internals, focusing on real-world architecture, communication flows, system-level debugging, and hardware integration techniques used in commercial Android products. The course incorporates current Android platform evolution, including the industry's transition from HIDL to AIDL-based HAL implementations and modern Binderized architectures.

The program is delivered by an instructor with over 30 years of industry experience in embedded systems, Linux, Android platform engineering, device drivers, middleware development, and large-scale commercial software systems. The training emphasizes real-world implementation practices, architecture analysis, source code navigation, debugging methodologies, and industry-demanded skills rather than purely academic concepts.

Learning outcomes

Upon completion of this course, participants will be able to:

- Understand the complete Android platform architecture and AOSP source tree organization
- Navigate Android framework internals and system services architecture
- Analyze interactions between Java framework components and native C++ services
- Understand Binder IPC architecture and Android inter-process communication mechanisms
- Design and troubleshoot AIDL and Binder-based communication flows
- Understand JNI integration between Java and native code
- Analyze Android HAL architecture and hardware abstraction mechanisms
- Understand modern AIDL-based HAL implementations and Binderized HAL design
- Trace end-to-end execution flow from application framework to hardware layer
- Understand Linux kernel interaction with Android system components
- Analyze Android security architecture including SELinux enforcement
- Perform architecture-level debugging across framework, native, HAL, and kernel layers
- Understand Android platform customization and hardware integration strategies

Prerequisites

- Strong Java programming knowledge
- Basic C/C++ programming experience
- Familiarity with Linux operating systems and command-line environments
- Understanding of operating system fundamentals
- Basic knowledge of Android application architecture
- Experience using development tools and debugging environments
- Exposure to software architecture concepts is beneficial

Training outline

1. Android Platform Architecture Overview

Evolution of Android Platform Architecture

1. Android ecosystem overview
2. AOSP project structure
3. Android release architecture evolution

4. Project Treble architecture
5. Mainline modularization concepts
6. Vendor and framework separation

Android Software Stack

1. Applications layer
2. Application Framework layer
3. Android Runtime (ART)
4. Native Libraries layer
5. Hardware Abstraction Layer (HAL)
6. Linux Kernel layer
7. End-to-end component relationships

AOSP Source Tree Navigation

1. frameworks directory structure
2. system directory organization
3. hardware directory organization
4. packages directory organization
5. vendor integration areas
6. build system overview
7. Android platform build architecture

2. Android Framework Architecture and Internals

Framework Layer Architecture

1. Framework design principles
2. Core framework services
3. SystemServer architecture
4. Service lifecycle management
5. Service registration mechanisms
6. Framework initialization sequence

Android System Services

1. ActivityManagerService
2. PackageManagerService
3. WindowManagerService
4. PowerManagerService
5. SensorService architecture
6. Connectivity services
7. Media services architecture

Java Framework Internals

1. Framework APIs and internal APIs
2. Hidden APIs and platform APIs
3. Framework service interactions
4. Resource management architecture
5. Event handling mechanisms
6. Threading models within framework services

3. Java and Native C++ Integration

Android Native Architecture

1. Native libraries overview
2. C/C++ service architecture
3. Native daemon architecture
4. Runtime interaction models

Java Native Interface (JNI)

1. JNI architecture
2. JNI call flow
3. Native method registration
4. Data marshalling and conversion
5. Memory management considerations
6. Performance implications
7. JNI debugging techniques

Framework-to-Native Communication

1. Framework service interactions with native layers
2. Native service invocation
3. Shared memory mechanisms
4. Native library loading process
5. Runtime execution flow
6. Framework and native dependency management

4. Binder IPC Architecture

Binder Fundamentals

1. Binder design principles
2. Binder driver architecture
3. Client-server communication model
4. Binder object lifecycle
5. Binder transaction processing
6. ServiceManager architecture

Android IPC Mechanisms

1. Binder IPC overview
2. Messenger architecture
3. Shared memory mechanisms
4. Socket-based communication
5. IPC design considerations

AIDL Architecture

1. AIDL fundamentals
2. Interface definition design
3. Proxy and Stub generation
4. Parcel mechanisms
5. Transaction lifecycle
6. Service registration and discovery
7. Modern AIDL architecture

Binder Debugging and Analysis

1. dumpsys analysis
2. Binder transaction tracing
3. Binder state inspection
4. Service diagnostics
5. Performance analysis
6. Common IPC bottlenecks

5. Hardware Abstraction Layer (HAL) Architecture

HAL Fundamentals

1. Purpose of HAL
2. Framework-HAL separation
3. Hardware abstraction principles
4. Vendor implementation architecture
5. HAL lifecycle management

Evolution of Android HAL Technologies

1. Legacy HAL architecture
2. HIDL architecture overview
3. Binderized HAL concepts
4. AIDL-based HAL architecture
5. Migration from HIDL to AIDL
6. Modern Android HAL design practices

HAL Development Architecture

1. HAL interface definitions
2. Service implementation models
3. Vendor service deployment
4. Framework integration
5. HAL registration mechanisms
6. VINTF concepts
7. Vendor partition architecture

Hardware Integration Examples

1. Camera HAL architecture
2. Audio HAL architecture
3. Sensor HAL architecture
4. Connectivity HAL architecture
5. Custom hardware integration patterns

6. Linux Kernel and Android Interaction

Android on Linux

1. Linux kernel architecture
2. Android-specific kernel enhancements
3. Android boot process
4. Process and thread management
5. Memory management architecture

Device Driver Architecture

1. Driver fundamentals
2. Character device drivers
3. Platform drivers
4. Device Tree concepts
5. Hardware initialization flow
6. Driver-HAL interaction models

Kernel Communication Mechanisms

1. System calls
2. ioctl architecture
3. Netlink communication
4. Shared memory mechanisms
5. Driver interfaces
6. Event propagation architecture

Android Native Services and Kernel Integration

1. Native daemon architecture
2. Hardware service interaction
3. Resource management
4. Power management architecture
5. Performance considerations

7. Android Security Architecture

Android Security Model

1. Security architecture overview
2. Application sandboxing
3. UID and permission model
4. Process isolation mechanisms
5. Platform security boundaries

SELinux Architecture

1. SELinux fundamentals
2. Mandatory access control
3. Security domains
4. Security contexts
5. Policy architecture
6. Enforcement mechanisms
7. Vendor SELinux customization

Security Across Android Layers

1. Framework security enforcement
2. Binder security controls
3. HAL security considerations
4. Native service security
5. Kernel security integration
6. Secure IPC design

8. End-to-End Android Control Flow Analysis

Framework to Hardware Execution Path

1. Request initiation within framework
2. System service processing
3. Binder transaction flow
4. Native service interaction
5. HAL invocation sequence
6. Kernel driver communication
7. Hardware response processing

Complete Transaction Lifecycle

1. Java framework request generation
2. Binder message handling
3. Native service execution
4. HAL processing flow
5. Driver invocation
6. Hardware interaction
7. Response propagation

Architecture Case Studies

1. Camera request flow
2. Sensor data flow
3. Power management flow
4. Audio processing flow
5. Connectivity service flow

Cross-Layer Debugging Methodology

1. Framework debugging techniques
2. Native debugging approaches
3. Binder tracing workflows
4. HAL diagnostics
5. Kernel tracing methods

6. End-to-end troubleshooting strategies

9. Platform Engineering Best Practices

Android Platform Development Workflows

1. Source code navigation strategies
2. Build and deployment workflows
3. Debugging methodologies
4. Logging and tracing frameworks
5. Performance optimization approaches

Enterprise Android Platform Development

1. BSP integration considerations
2. Vendor customization strategies
3. Platform maintenance practices
4. Upstream synchronization concepts
5. Long-term support considerations

Troubleshooting and Root Cause Analysis

1. Framework issues
2. Binder communication failures
3. JNI-related problems
4. HAL integration failures
5. Driver communication issues
6. SELinux policy violations
7. System stability analysis

Recommended Duration

2 Days (Intensive Advanced Technical Workshop)

- Day 1: Android Architecture, Framework Internals, Java-C++ Integration, Binder IPC, JNI
- Day 2: HAL Architecture, Linux Kernel Interaction, Android Security, SELinux, End-to-End System Flow Analysis, Debugging Methodologies

Target Audience

- Android Platform Engineers
- AOSP Developers
- BSP Engineers
- Embedded Linux Developers
- System Software Engineers
- Middleware Developers
- Device Driver Engineers
- Android Framework Developers
- Technical Architects
- Engineering Teams Working on Customized Android Platforms

Disclaimer

This course outline is provided as a proposed training framework intended to address the stated learning objectives and technical requirements. The final content, sequencing of topics, depth of coverage, hands-on activities, demonstrations, and time allocation may be adjusted, expanded, condensed, or otherwise modified by the trainer based on participant backgrounds, organizational priorities, platform versions, available laboratory environments, and emerging industry developments. Such adjustments may be made at the trainer's professional discretion without prior notice to ensure the most effective learning experience and alignment with project-specific objectives.

© 2026 Faysal Aziz · Kuala Lumpur