THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

Generic BootLoader on Android platforms

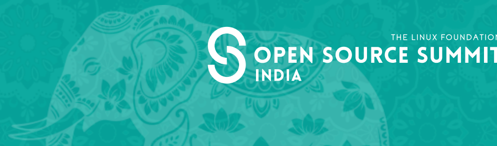
Naina Mehta, Qualcomm India Private Limited
<nainmeht@qti.qualcomm.com>

#OSSUMMMIT

Current Android Bootloader Landscape

- Fragmented Implementations
  - Each vendor maintains different bootloader codebases
  - Platform-specific implementations
- Duplication of Effort
  - Same functionality reimplemented across platforms
- Slow Security Patching
  - Updates must be replicated across multiple implementations
- Complex Upgrades
  - Barriers to adopting new Android boot framework features

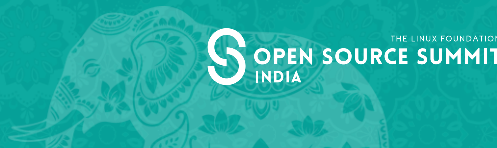
Generic BootLoader (GBL)



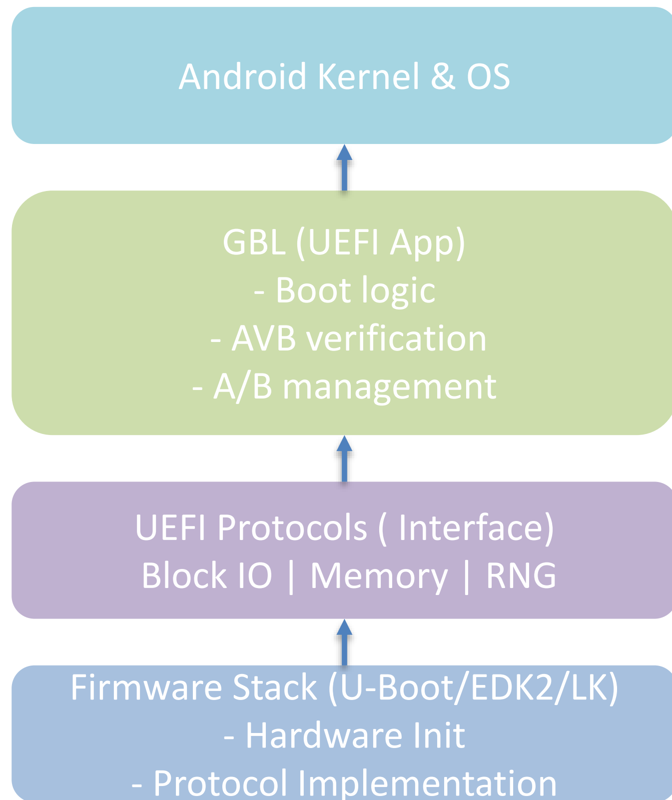
- What is GBL?
  - Rust-based bootloader provided by Google
  - Developed within AOSP
  - Designed as a UEFI application
  - Standardizes boot flow across multiple architectures
  - Memory-safe and secure by design

- Key Benefits:
  - Single codebase for the entire ecosystem
  - Faster security updates
  - Reduced maintenance overhead
  - Consistent boot behavior across platforms

Bridging Firmware and Android OS



- Supported Architectures:
  - X86
  - ARM64
  - RISC-V
- Supported firmware stacks:
  - U-Boot
  - EDK2
  - LittleKernel



Standard UEFI protocols

Protocol	Purpose	Priority
Block IO Protocol	Storage access – UFS, eMMC, etc.	Required
Random Number Generator Protocol	Enables dynamic stack canary, KASLR, bootloader entropy	Required
Memory Allocation Services	Dynamic memory for libavb	Required
Timestamp Protocol	Performance Analysis	Optional
Device Path Protocol	Logging image path at GBL start	Optional
Simple Text Input/Output Protocol	Text based interface for entering fastboot	Optional
Hash2 Protocol	Optimize hash functions during AVB signature checking	Optional

https://cs.android.com/android/kernel/superproject/+/common-android-mainline:bootable/libbootloader/gbl/docs/efi_integration.md

UEFI Memory Allocation Services

- Allocate and free memory using UEFI firmware
- Used by libavb for image verification

Name	Type	Description
AllocatePages	Boot	Allocates pages of a particular type.
FreePages	Boot	Frees allocated pages.
GetMemoryMap	Boot	Returns the current boot services memory map and memory map key.
AllocatePool	Boot	Allocates a pool of a particular type
FreePool	Boot	Frees allocated pool.

https://uefi.org/specs/UEFI/2.10/07_Services_Boot_Services.html#memory-allocation-services

DeviceTree Fixup Protocol

- Firmware to inspect the final device tree and apply necessary fixups

```
struct _DT_FIXUP_PROTOCOL {
  UINT64      Revision;
  DT_FIXUP    Fixup;
};

typedef
EFI_STATUS
(EFIAPI *DT_FIXUP)(
  IN DT_FIXUP_PROTOCOL *This,
  IN OUT VOID          *Fdt,   /* FDT buffer provided by GBL to be updated by firmware */
  IN OUT UINTN         *BufferSize, /* Size of the writable buffer */
);
```

Common DT Fixups

- Memory regions
- Command-line parameters
  - mem, Loglevel, boot_cpu , etc.
- Kaslr-seed, rng-seed (by GBL)
- intrd-start, initrd-end (by GBL)

Custom GBL protocols

- GBL_EFI_OS_CONFIGURATION_PROTOCOL
  - Devicetree selection and Bootconfig fixup
- GBL_EFI_BOOT_CONTROL_PROTOCOL
  - A/B slot management
- GBL_EFI_FASTBOOT_PROTOCOL
  - Vendor specific fastboot functionality
- GBL_EFI_FASTBOOT_TRANSPORT_PROTOCOL
  - Custom fastboot transport
- GBL_EFI_AVB_PROTOCOL
  - Support Android Verified Boot (AVB)
- GBL_EFI_AVF_PROTOCOL
  - Support Android Virtualization Framework (AVF)

GBL_EFI_OS_CONFIGURATION_PROTOCOL

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

- Firmware to provide data required for OS configuration

  Key functions:
  - FixupBootConfig: Append necessary bootconfig parameters
  - SelectDeviceTrees: Choose correct DTB/DTBO
  - SelectFitConfiguration: Select appropriate FIT configuration

GBL_EFI_OS_CONFIGURATION_PROTOCOL

- GBL_EFI_FIXUP_BOOTCONFIG
  - Append additional bootconfig parameters
  - Firmware can add platform-specific parameters
  - Libavb specific parameters must not be updated

Bootconfig parameters added by GBL

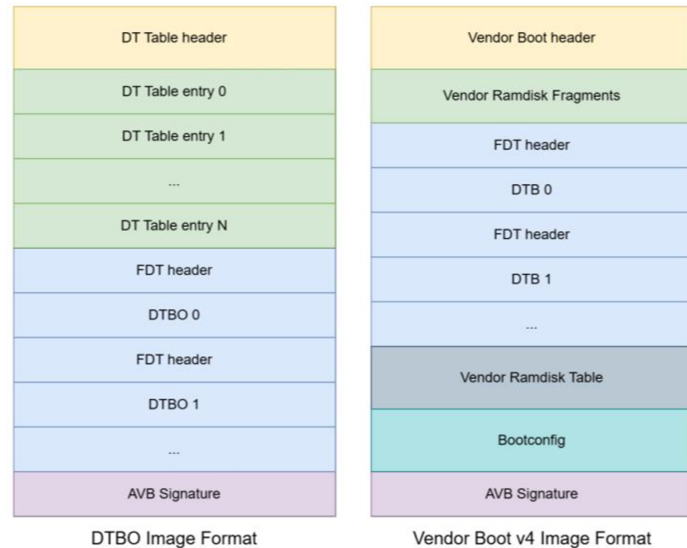
- androidboot.dtb(o)_idx
- androidboot.mode
- androidboot.force_normal_boot

GBL_EFI_OS_CONFIGURATION_PROTOCOL


```
typedef
EFI_STATUS
(EFIAPI *GBL_EFI_FIXUP_BOOTCONFIG)(
  IN GBL_EFI_OS_CONFIGURATION_PROTOCOL *Self,
  IN UINTN           BootConfigSize, /* Size of the BootConfig built by GBL */
  IN CONST CHAR8    *BootConfig, /* BootConfig built by GBL */
  IN OUT UINTN     *FixupBufferSize, /* Size of fixup BootConfig provide by GBL */
  OUT CHAR8   *Fixup /* Fixup BootConfig provide by GBL to be updated by firmware */
  );

// Sample Implementation
EFI_STATUS FixupBootConfig(
    IN  UINTN BootConfigSize,
    IN  CONST CHAR8 *BootConfig,
    OUT CHAR8 *Fixup
) {
    // Add vendor-specific parameters
    AsciiSPrint(Fixup, FixupSize,
        "androidboot.soc=%a\n"
        "androidboot.hwrev=%d\n",
        GetSocName(), GetHwRevision()
    );
    return EFI_SUCCESS;
}
```

GBL_EFI_OS_CONFIGURATION_PROTOCOL

- GBL_EFI_SELECT_DEVICE_TREES
  - GBL loads and verifies DTB(O)s
  - GBL creates a DT component registry containing DT blobs alongwith metadata
  - Firmware selects appropriate DTB and DTBOs
  - GBL applies selected DTB(O)s



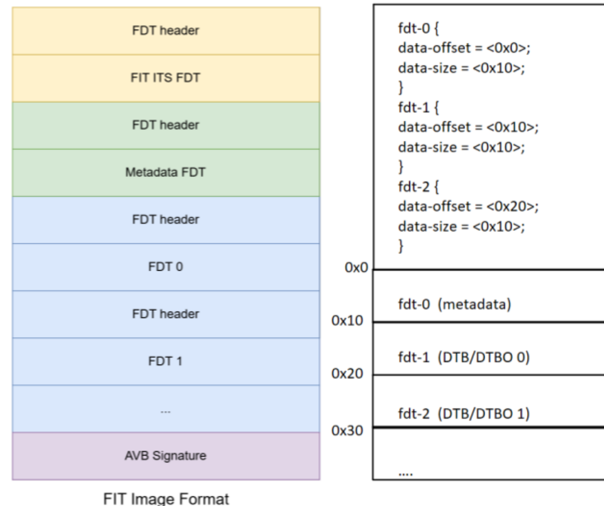
GBL_EFI_OS_CONFIGURATION_PROTOCOL

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

```
typedef
EFI_STATUS
(EFIAPI *GBL_EFI_SELECT_DEVICE_TREES)(
  IN GBL_EFI_OS_CONFIGURATION_PROTOCOL *Self,
  IN UINTN         NumDeviceTrees, /* Number of DT components in the DT array */
  IN OUT GBL_EFI_VERIFIED_DEVICE_TREE  *DeviceTrees /* Pointer to DT component registry created by GBL */
  );

//Sample Implementation
EFI_STATUS SelectDeviceTrees(
    IN UINTN NumDeviceTrees,
    IN OUT GBL_EFI_VERIFIED_DEVICE_TREE *DeviceTrees
) {
    UINT32 PlatformId = GetPlatformId();
    UINT32 SocVersion = GetBoardId();

    for (i = 0; i < NumDeviceTrees; i++) {
        if (DeviceTrees[i].PlatformId == PlatformId &&
            DeviceTrees[i].SocVersion == BoardId) {
            DeviceTrees[i].Selected = TRUE;
        }
    }
    return EFI_SUCCESS;
}
```

GBL_EFI_OS_CONFIGURATION_PROTOCOL

- GBL_EFI_SELECT_FIT_CONFIGURATION
  - If FIT is found in dtbo partition, GBL loads FIT
  - GBL passes FIT FDT and metadata to firmware
  - Firmware selects appropriate FIT configuration
  - Returns DT node offset of selected configuration



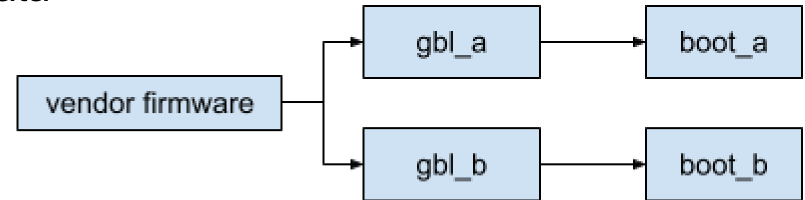
FIT Image Format

GBL_EFI_OS_CONFIGURATION_PROTOCOL

```
typedef
EFI_STATUS
(EFIAPI *GBL_EFI_SELECT_FIT_CONFIGURATION)(
  IN GBL_EFI_OS_CONFIGURATION_PROTOCOL  *Self,
  IN                        FitSize, /* Size of FIT FDT */
  IN CONST UINT8    *Fit, /* Pointer to FIT FDT */
  IN UINTN                    MetadataSize, /* Size of metadata binary */
  IN CONST UINT8    *Metadata, /* Pointer to metadata binary */
  OUT UINTN               *SelectedConfigurationOffset /* DT node offset of Selected FIT configuration*/
  );

EFI_STATUS SelectFitConfiguration(...) {
    PlatformId = GetPlatformId();
    SocId = GetSocId();

    // Find configurations node
    ConfigsNode = fdt_path_offset(Fit, "/configurations");

    // Iterate and match platform
    fdt_for_each_subnode(ConfNode, Fit, ConfigsNode) {
        if (MatchesPlatform(Fit, ConfNode,
                                PlatformId, SocId)) {
            *SelectedConfigurationOffset = ConfNode;
            return EFI_SUCCESS;
        }
    }
}
```

GBL_EFI_BOOT_CONTROL_PROTOCOL

- Multiple partition slots are required for OTA
  - Firmware to decide the bootable slot, same slot is used by GBL
  - Firmware to maintain the slot specific metadata
    - Slot count
    - Successful
    - Unbootable
    - Retry-count
  - Active slot can also be updated by fastboot



https://cs.android.com/android/kernel/superproject/+/common-android-mainline:bootable/libbootloader/gbl/docs/gbl_ab_boot_flow.md

GBL_EFI_BOOT_CONTROL_PROTOCOL

```
typedef struct GBL_EFI_BOOT_CONTROL_PROTOCOL {
  UINT64                                     Revision;
   GBL_EFI_BOOT_CONTROL_GET_SLOT_COUNT        GetSlotCount; /* Get number of boot slots */
  GBL_EFI_BOOT_CONTROL_GET_SLOT_INFO         GetSlotInfo; /* Provide slot metadata by index*/
  GBL_EFI_BOOT_CONTROL_GET_CURRENT_SLOT      GetCurrentSlot; /* Provide metadata of
                                                              current slot */
  GBL_EFI_BOOT_CONTROL_SET_ACTIVE_SLOT       SetActiveSlot; /* Set slot as active for given
                                                                       index and reset metadata */
  GBL_EFI_BOOT_CONTROL_GET_ONE_SHOT_BOOT_MODE GetOneShotBootMode; /* Get the boot mode triggered by hardware */
  GBL_EFI_BOOT_CONTROL_HANDLE_LOADED_OS      HandleLoadedOs; /* Firmware to provide the load start and size of final
                                                                kernel, device tree, ramdisk. Firmware can also
                                                                directly jump to OS and not return to GBL*/

} GBL_EFI_BOOT_CONTROL_PROTOCOL;
```

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

GBL_EFI_BOOT_MEMORY_PROTOCOL

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

- Firmware can provide reserved buffer for images preloaded by the firmware

```
typedef struct _GBL_EFI_BOOT_MEMORY_PROTOCOL {
  UINT64                         Revision;
  GBL_EFI_GET_PARTITION_BUFFER   GetPartitionBuffer; /* Reserve memory for a partition image */
  GBL_EFI_SYNC_PARTITION_BUFFER  SyncPartitionBuffer; /* Firmware can process the memory for the
                                                         partition image*/
  GBL_EFI_GET_BOOT_BUFFER        GetBootBuffer; /* Reserve buffer for GBL to construct boot images.
                                                   Can also be used for fastboot download data */
} GBL_EFI_BOOT_MEMORY_PROTOCOL;
```

GBL_EFI_FASTBOOT_PROTOCOL

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

- UEFI firmware can provide vendor/OEM specific variables and commands

```
typedef struct _GBL_EFI_FASTBOOT_PROTOCOL {
  UINT64                             Revision;
  CHAR8                              SerialNumber[GBL_EFI_FASTBOOT_SERIAL_NUMBER_MAX_LEN_UTF8]; /* Firmwar supplied
                                                                                        device serial number*/
  GBL_EFI_FASTBOOT_GET_VAR           GetVar; /* Firmware supplied fastboot variables with args */
  GBL_EFI_FASTBOOT_GET_VAR_ALL       GetVarAll; /* Vendor specific callback for GetVarAll */
  GBL_EFI_FASTBOOT_GET_STAGED        GetStaged; /* OEM supplied payload for host */
  GBL_EFI_FASTBOOT_COMMAND_EXEC      CommandExec; /* Command filtering and implementation override */
  GBL_EFI_FASTBOOT_GET_PARTITION_TYPE GetPartitionType; /* Get Partition Type */
} GBL_EFI_FASTBOOT_PROTOCOL;
```

GBL_EFI_FASTBOOT_TRANSPORT_PROTOCOL

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

- Firmware implementation for custom fastboot transport channels

```
typedef struct _GBL_EFI_FASTBOOT_TRANSPORT_PROTOCOL {
  UINT64                                     Revision;
  CONST CHAR8                                *Description; /* USB/TCP or any custom channel */
  GBL_EFI_FASTBOOT_TRANSPORT_INTERFACE_START Start; /* Setup the transport channel */
  GBL_EFI_FASTBOOT_TRANSPORT_INTERFACE_STOP  Stop; /* Stop the transport channel */
  GBL_EFI_FASTBOOT_TRANSPORT_RECEIVE         Receive; /* Receive data from the transport channel */
  GBL_EFI_FASTBOOT_TRANSPORT_SEND            Send; /* Send data to the transport channel */
  GBL_EFI_FASTBOOT_TRANSPORT_FLUSH           Flush; /* Wait for all transactions to complete */
} GBL_EFI_FASTBOOT_TRANSPORT_PROTOCOL;
```

GBL_EFI_AVB_PROTOCOL

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
INDIA

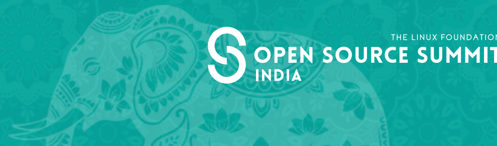
- GBL always verifies below partitions
  - boot
  - dtb
  - dtbo
  - init_boot
  - pvmfw
  - vendor_boot
  - vendor_kernel_boot

- Firmware can implement vendor-specific AVB logic for special partitions

GBL_EFI_AVB_PROTOCOL

```
UINT64                                       Revision;
  GBL_EFI_AVB_READ_PARTITION_ATTRIBUTES   ReadPartitionAttributes; /* AVB attributes for special partition*/
  GBL_EFI_AVB_READ_DEVICE_STATUS          ReadDeviceStatus; /* Firmware to provide AVB device state */
  GBL_EFI_AVB_VALIDATE_VBMETA_PUBLIC_KEY ValidateVbmetaPublicKey; /* Firmware to verify the public key */
  GBL_EFI_AVB_READ_ROLLBACK_INDEX         ReadRollbackIndex; /* Firmware to provide rollback index for index location */
  GBL_EFI_AVB_WRITE_ROLLBACK_INDEX        WriteRollbackIndex; /* Firmware to update rollback index for index location */
  GBL_EFI_AVB_READ_PERSISTENT_VALUE       ReadPersistentValue; /*  Firmware to provide persistent value for given name */
  GBL_EFI_AVB_WRITE_PERSISTENT_VALUE      WritePersistentValue; /*  Firmware to write persistent value for given name */
  GBL_EFI_AVB_HANDLE_VERIFICATION_RESULT HandleVerificationResult; /* Custom handling for verification result */
  GBL_EFI_AVB_WRITE_LOCK_STATE            WriteLockState; /* Locks or unlocks the DEVICE and CRITICAL lock */
  GBL_EFI_AVB_FACTORY_DATA_RESET          FactoryDataReset; /* Perform Factory Data Reset */
} GBL_EFI_AVB_PROTOCOL;
```

GBL image requirements



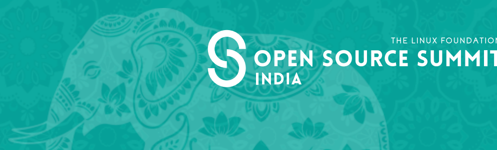
- Required Partitions
  - android_esp_a/b (FAT, 8MB minimum)
  - GBL binary

- File System
  - FAT12, FAT16, or FAT32
  - Choose based on partition size and platform requirements

- Standard UEFI Boot Path
  - /EFI/BOOT/BOOTAA64.EFI

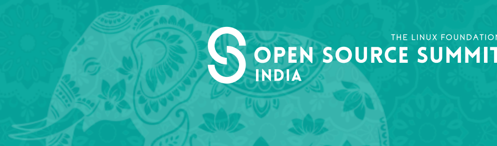
- Partition Type GUID
  - C12A7328-F81F-11D2-BA4B-00A0C93EC93B (EFI System Partition)

Challenges and key learning



- Protocol Interface Differences
  - Refactor firmware to provide standard UEFI protocols
  - Add GBL-specific protocols (Boot Control, OS Config, AVB)
- Device Tree Selection Logic
  - Move DT selection logic to firmware
  - Adapt to the structure in which DT binaries are shared by GBL
- Maintaining feature parity

Challenges and key learning



- ESP Partition Management
  - Introduce new ESP partition
  - Update firmware for loading and running GBL application
  - A/B update mechanism for GBL binary
  - GBL binary authentication
- Tooling for GBL live debugging
- Extensive test coverage

Resources

- GBL Documentation:
  - https://cs.android.com/android/kernel/superproject/+/common-android-mainline:bootable/libbootloader/gbl/docs/
  - https://source.android.com/docs/core/architecture/bootloader/generic-bootloader
  - https://source.android.com/docs/core/architecture/bootloader/generic-bootloader/gbl-dev

- GBL Code:
  - https://cs.android.com/android/kernel/superproject/+/common-android-mainline:bootable/libbootloader/gbl/

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT INDIA