

android

Vendor Native Development Kit (VNDK)

Design Principles and Practical Migration



Agenda

VNDK Overview

Dynamic Linker Support

Build System Support

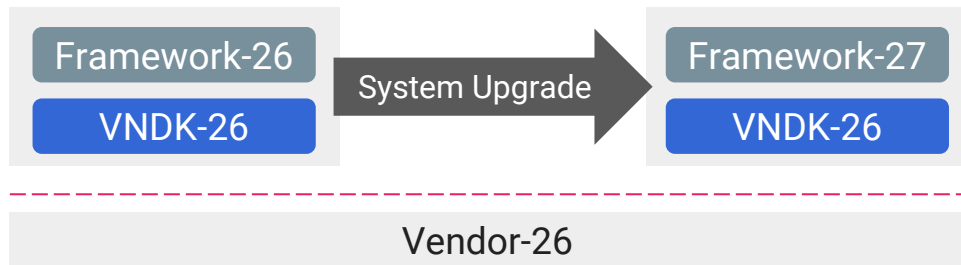
VNDK Definition Tool

JNI Libraries in Bundled APKs

VNDK Overview

What is the VNDK?

- The Vendor Native Development Kit (VNDK) is a **set of shared libraries** for vendors to implement vendor modules.
- The VNDK is part of vendor interface object (VINTF object).
- The VNDK is **versioned** and **stable**.

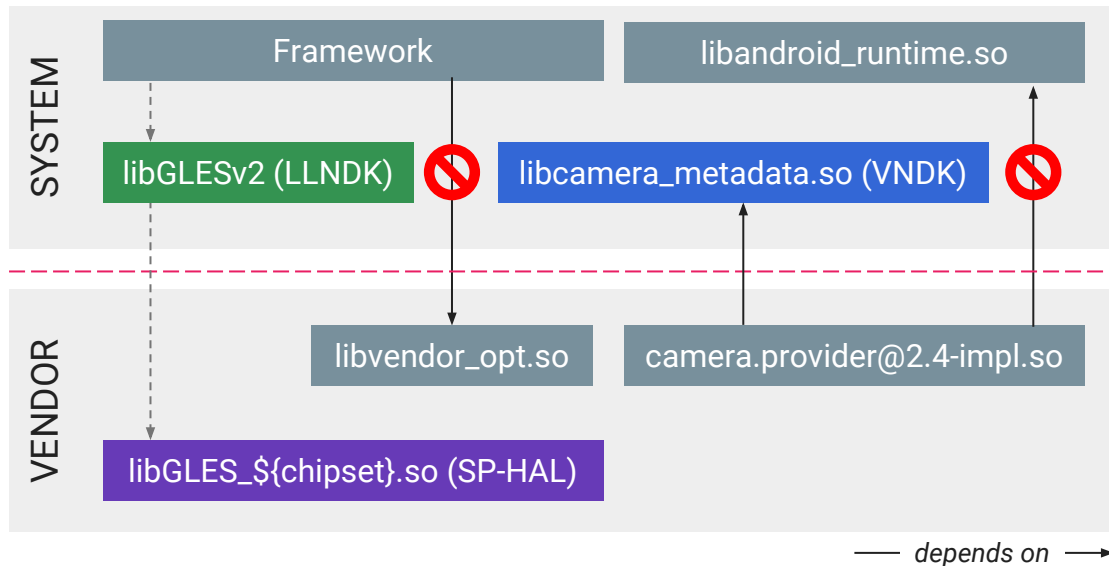


VNDK enforcement status

- VNDK is partially implemented in **Android 8 (O)**
 - VNDK-SP (for SP-HAL) is enforced.
- VNDK is fully implemented in **Android 8.1 (O MR1)**
 - VNDK-SP (for SP-HAL) is enforced.
 - Enabling VNDK is recommended.
- Future Android releases will fully enforce the VNDK. When this occurs, **ineligible libraries will not be accessible by vendor modules at build time and runtime.**

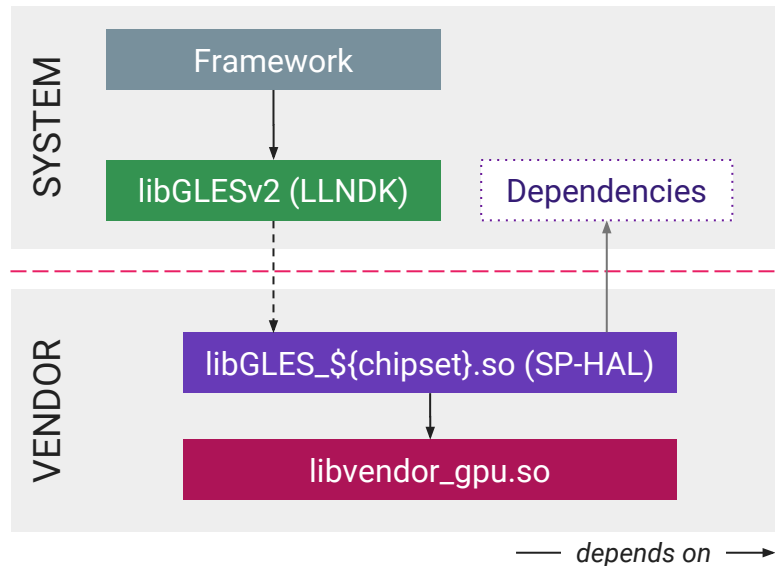
Modular Android lib dependencies

- Vendor modules should not depend on system modules **except VNDK**.
- Framework does not depend on vendor modules **except Same-Process HAL (SP-HAL)**.



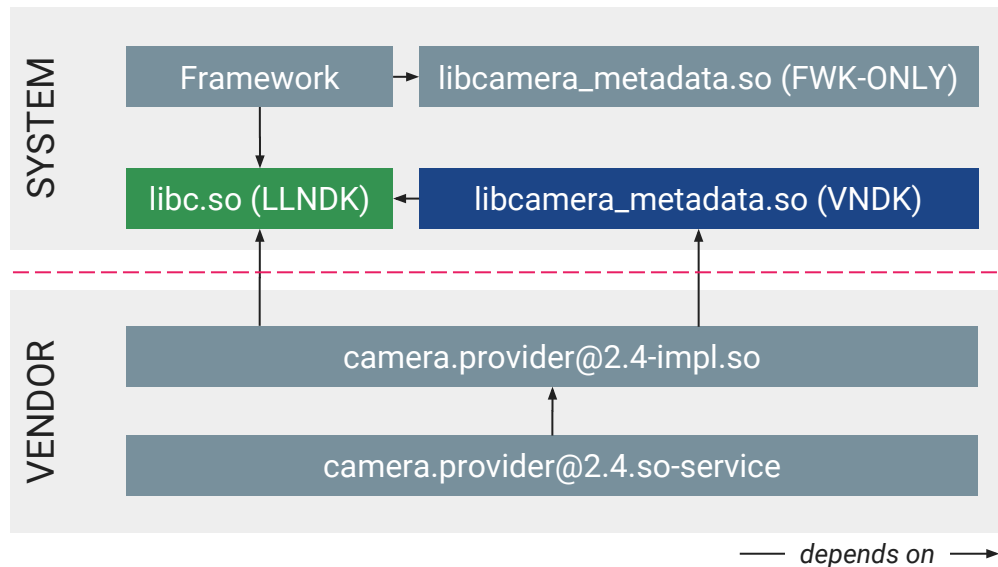
Same-Process HAL (SP-HAL)

- Several time-critical HALs are **not binderized**:
 - `android.hardware.renderscript@1.0-impl`
 - `android.hardware.graphics.mapper@1.0-impl`
 - `android.hidl.memory@1.0-impl`
 - `libEGL_${chipset}`
 - `libGLES_${chipset}`
 - `vulkan.${chipset}`
- What about dependencies? Both SP-HAL and their dependencies applies.



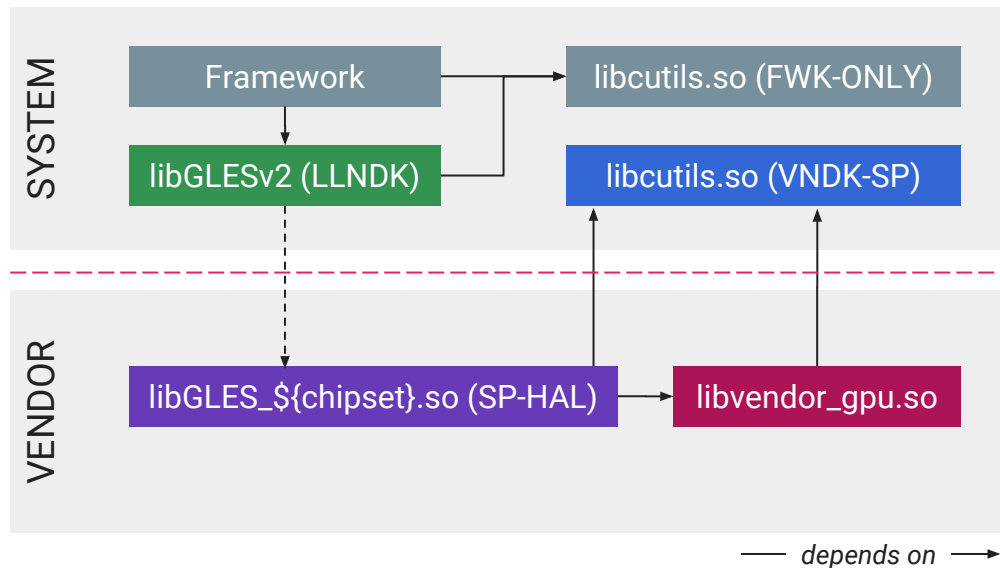
VNDK categories

- **LLNDK** (LL-NDK + SP-NDK)
 - Shared libraries with stable APIs and loosely coupled with the framework
 - System and vendor share the same file
- **VNDK**
 - Specialized variant for vendor modules.
 - May be a FWK-ONLY counterpart with the same name
- **VNDK-SP**
 - Same as VNDK
 - Can be used by SP-HALs
 - May be loaded into framework process



VNDK-SP: Dependency of Same-Process HAL

- SP-HAL must only depend on LLNDK or VNDK-SP (both SP-HAL and their dependencies apply).
- VNDK-SP and its FWK-ONLY counterpart (shared lib with same name) may be loaded into the same process.



Other categories

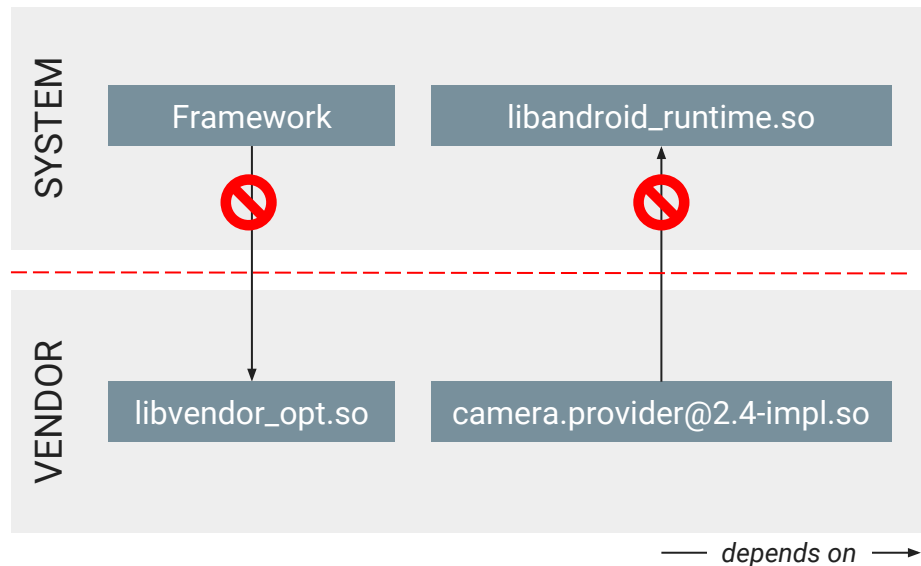
- **FWK-ONLY**

- Other shared libraries on the system partition
- Vendor modules must not depend on these libraries

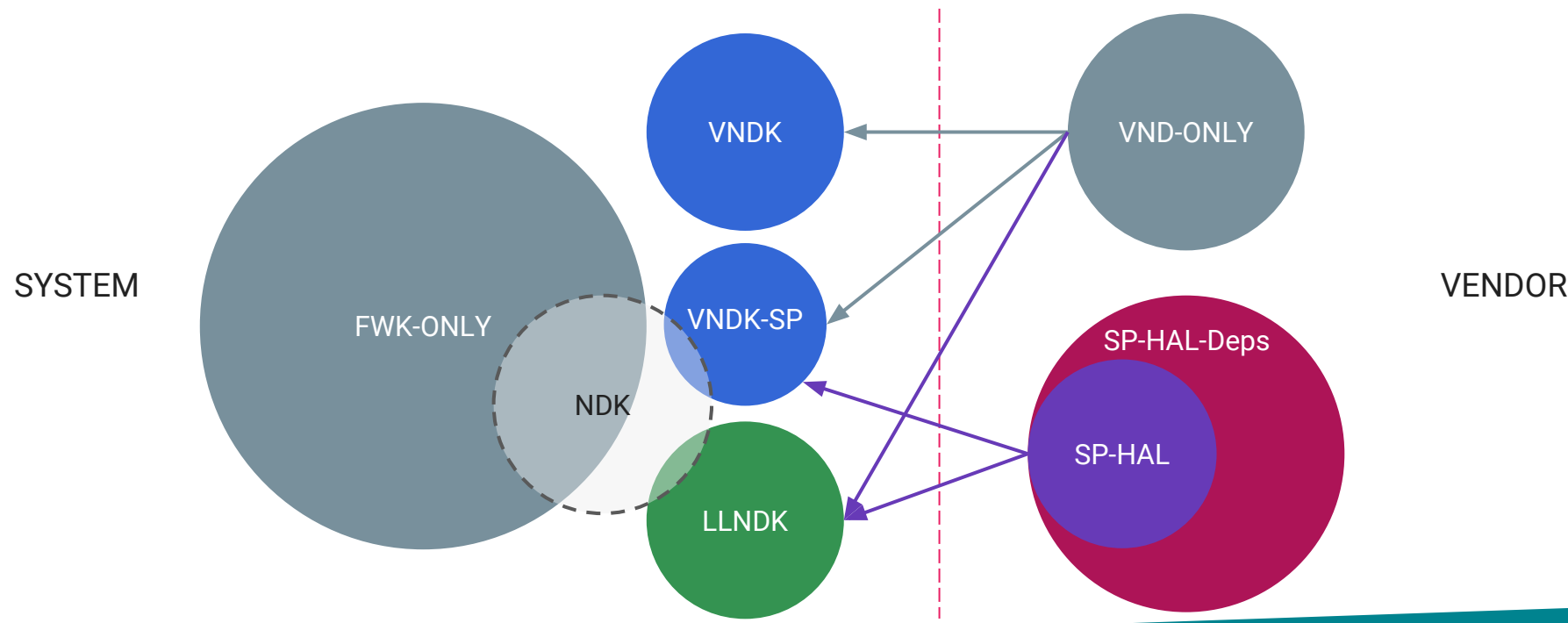
- **VND-ONLY**

- Other (i.e., non-SP-HAL) shared libraries on the vendor partition
- Framework modules must not depend on these libraries

Cross-partition dependencies must be in LLNDK, VNDK, VNDK-SP, and SP-HAL (recommended in Android 8.1 and enforced in Android 9).



Category relationships



Category libraries

LLNDK

libEGL
libGLv1_CM
libGLv2
libGLv3
libRS
libandroid_net#
libc
libdl
liblog
libm
libnativewindow
libsync
libvndksupport#

LLNDK but not NDK

VNDK-SP

android.hardware.graphics.allocator@2.0
android.hardware.graphics.common@1.0
android.hardware.graphics.mapper@2.0
android.hardware.renderscript@1.0
android.hidl.memory@1.0
android.hidl.memory@1.0-impl
libRSCpuRef
libRSDriver
libRS_internal

** In some configurations, libz belongs to LLNDK but there should be no differences.*

libbacktrace

libbase

libbcinfo

libblas

libc++

libcompiler_rt

libcutils

libhardware

libhidlbase

libhidlmemory

libhidltransport

libhwcomposer

libion

liblzma

libunwind

libunwindstack

libutils

libz*

Eligible list

- List of shared libraries that have been reviewed.
- Found in:
[`\${AOSP}/development/vndk/tools/definition-tool/datasets/eligible-list\*.csv`](#)

Some NDK libs not visible to vendor modules

libandroid.so

libaudio.so

libcamera2ndk.so

libcui18n.so

libcuc.so

libjnigraphics.so

libmediandk.so

libneuralnetworks.so

libOpenMAXAL.so

libOpenSLES.so

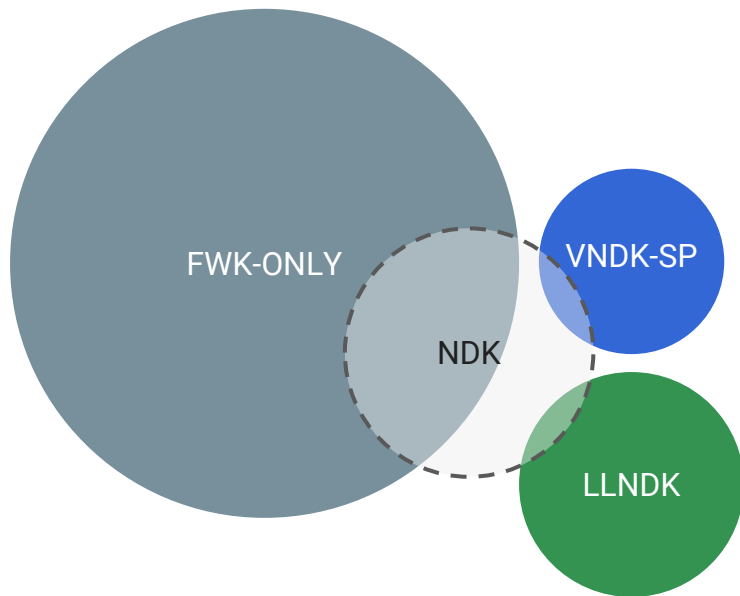
libstdc++.so*

libvulkan.so

libwebviewchromium_plat_support.so

These libraries are **highly coupled** with the framework, thus they do not belong to LLNDK.

Vendor modules must **not** depend on these shared libraries.



** Use libcpp instead of libstdc++.*

VNDK extensions

- Vendor modules may need **extra APIs** or **extra functionalities** from the VNDK libraries.
- VNDK can be extended, but they must remain **ABI compatible** to the AOSP VNDK.
 - Symbols must not be removed.
 - Exposed structures must not be altered (including struct/class layout and vtable)
- **Goal is to ensure all extensions are drop-in replacements of the AOSP VNDK shared libraries.**

```
struct Example {
    int a_;
    int bias_;
};

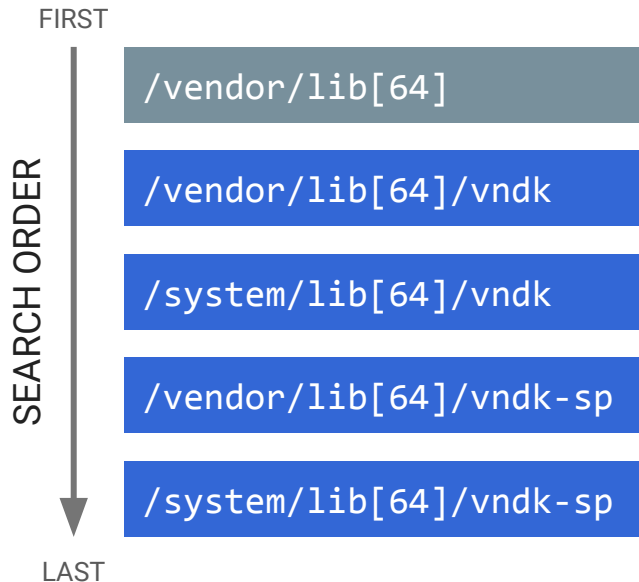
Example *example_create(int a) {
    Example *e =
        (Example *)malloc(sizeof(Example));
    e->a_ = a;
    e->bias_ = rand();
    return e;
}

int example_get_a(Example *e) {
    return e->a_ + e->bias_;
}

/* Extensions */
void example_set_bias(Example *e, int b) {
    e->bias_ = b;
}
```

Extended VNDK libraries

- Must be installed to `/vendor/lib[64]/{,vndk,vndk-sp}`
- Otherwise, vendor modules will fail **VTs** tests on **GSI**, which is required to pass compliance.
- Use as a **last resort** because extended VNDK shared libraries are not framework-only OTA updatable.
- **VNDK definition tool** can provide a preliminary set of libraries.



Degenerated VNDK (8.0) vs. Treble VNDK (8.1)

Android 8.0 (O) adopts the **degenerated VNDK directory layout**:

- VNDK-SP libraries have extra copies in `/system/lib[64]/vndk-sp`
- Both framework and vendor modules are using shared libraries in `/system/lib[64]`

Android 8.1 (O MR1) adopts the **Treble VNDK directory layout**:

- VNDK-SP libraries have extra copies in `/system/lib[64]/vndk-sp`
- VNDK libraries have extra copies `/system/lib[64]/vndk`
- Vendor modules are only using `/system/lib[64]/{vndk,vndk-sp}`
- Framework modules are only using `/system/lib[64]`

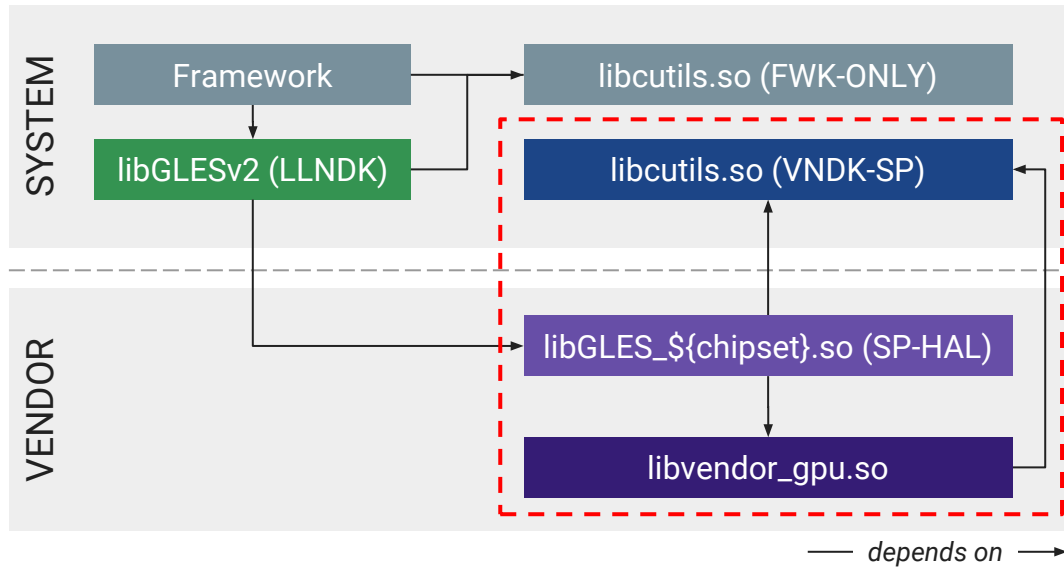
Directory layout

Category	Android 8.0 (O)	Android 8.1 (O MR1)	Independent System Updates
FWK-ONLY	/system/lib[64]	/system/lib[64]	Everything may change
LLNDK	/system/lib[64]	/system/lib[64]	New APIs or implementation
VNDK-SP	/system/lib[64]/vndk-sp	/system/lib[64]/vndk-sp	Old APIs with security fixes
VNDK-SP-EXT	/vendor/lib[64]/vndk-sp	/vendor/lib[64]/vndk-sp	N/A
VNDK	/system/lib[64] (degenerated)	/system/lib[64]/vndk	Old APIs with security fixes (only 8.1)
VNDK-EXT	/vendor/lib[64]	/vendor/lib[64]/vndk	N/A
VND-ONLY	/vendor/lib[64]	/vendor/lib[64]	N/A

Dynamic Linker Support

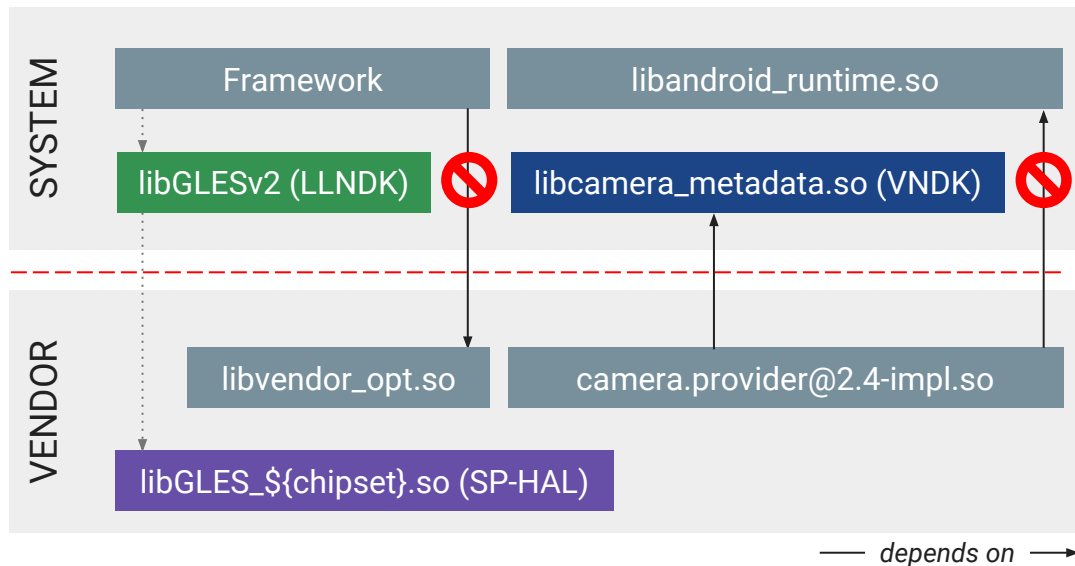
Isolate SP-HAL and VNDK-SP

- SP-HAL from the vendor partition is loaded into framework processes; may depend on **VNDK-SP**.
- Framework modules may depend on **FWK-ONLY counterpart** (shared lib with same name with VNDK-SP).
- Loading two shared libraries with the same soname causes problems (libraries may have different symbols after updates).
- Enforced in Android 8.0 (O) (PRODUCT_FULL_TREBLE:=true)



Isolate system and vendor

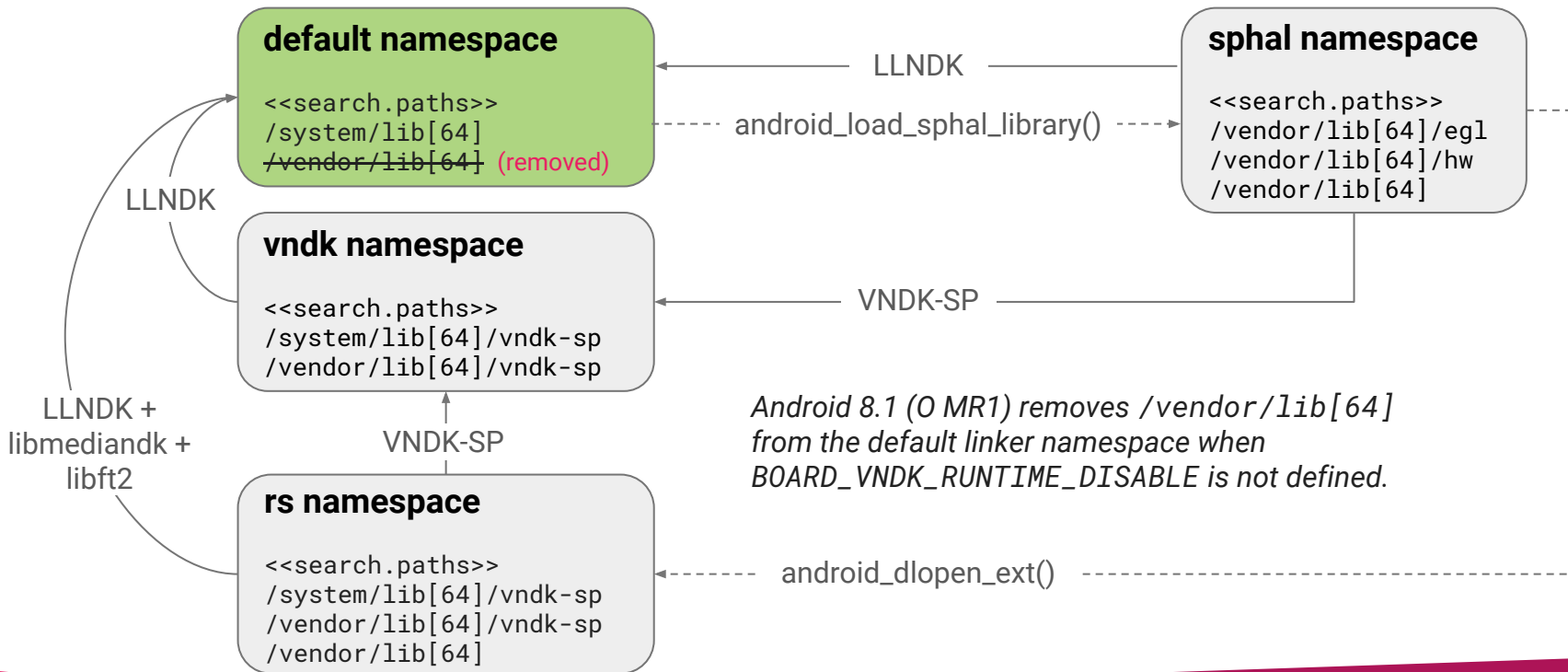
- Isolate shared library dependencies. Dynamic linker should not load shared libraries from the other partition except VNDK or SP-HAL.
- Not enforced in Android 8.0
- Recommended in Android 8.1. If `BOARD_VNDK_VERSION` is specified, enforced by default; to disable, add `BOARD_VNDK_RUNTIME_DISABLE:=true`
- Enforced in **Android 9**



Dynamic linker namespace

- **Dynamic linker** /system/bin/linker[64] is a part of Bionic that loads and links ELF shared objects at runtime. This program:
 - Is the first program being run after the kernel maps the executable into memory.
 - Is responsible to load DT_NEEDED entries and resolve undefined symbols.
 - Implements dlopen() and android_dlopen_ext().
- **Dynamic linker namespace** is the underlying mechanism that **isolates SP-HALs and VNDK-SP**. This mechanism isolates shared libraries and provides fine-grained control on:
 - Dynamic **shared libraries** resolution
 - **symbol** resolution

Framework process linker namespaces (Android 8.1)



Vendor process linker namespaces

- In Android 8.0 (O), `/system/lib[64]` is in the **default linker namespace** of vendor processes.

default namespace

```
<<search.paths>>  
/vendor/lib[64]  
/system/lib[64]
```

- In Android 8.1 (O MR1), `/system/lib[64]` is **removed** if `BOARD_VNDK_RUNTIME_DISABLE` is not defined.

default namespace

```
<<search.paths>>  
/vendor/lib[64]  
/vendor/lib[64]/vndk  
/system/lib[64]/vndk  
/vendor/lib[64]/vndk-sp  
/system/lib[64]/vndk-sp
```

LLNDK →

system namespace

```
<<search.paths>>  
/system/lib[64]
```

ld.config.txt

- Dynamic linker namespace is configured by `/system/etc/ld.config.txt`.
 - **INI file format**
 - Source code at [\\${AOSP}/system/core/rootdir/etc/ld.config*.txt](#)
- `ld.config.txt` must **not be modified**.
 - CTS verifies this file is intact.
 - Learning the file format can help in understanding how VNDK works.

ld.config.txt structure

- `dir.name` assignments specify the section that will be chosen.
- For example, the `[system]` section is chosen if the main executable of the process resides in `/system/bin`.
- Each section represents a graph with:
 - **linker namespaces** as nodes
 - links for **fallback lookups**

```
dir.system = /system/bin
dir.vendor = /vendor/bin
```

```
[system]
additional.namespaces = sphal,vndk,rs
```

```
namespace.default.isolated = true
namespace.default.search.paths = ...
namespace.default.permitted.paths = ...
```

```
namespace.sphal.isolated = true
namespace.sphal.visible = true
namespace.sphal.search.paths = ...
namespace.sphal.permitted.paths = ...
namespace.sphal.link.default.shared_libs =
```

```
[vendor]
```

ld.config.txt namespace properties

For each section, **additional.namespaces** specifies the names of other linker namespaces in addition to the **default** namespace.

For each linker namespace:

- **isolated**. Whether permitted.paths is enforced
- **permitted.paths**. Permitted path (in addition to search.paths) when isolated is true.
- **search.paths**. Directories to search when dynamic linker resolves to an *soname**.
- **visible**. Whether namespace can be found by `android_get_exported_namespace()`.

```
dir.system = /system/bin  
dir.vendor = /vendor/bin
```

```
[system]  
additional.namespaces = sphal,vndk,rs
```

```
namespace.default.isolated = true  
namespace.default.search.paths = ...  
namespace.default.permitted.paths = ...
```

```
namespace.sphal.isolated = true  
namespace.sphal.visible = true  
namespace.sphal.search.paths = ...  
namespace.sphal.permitted.paths = ...  
namespace.sphal.link.default.shared_libs =
```

```
[vendor]
```

ld.config.txt fallback links

- `namespace.${name}.link.${another}.shared_libs` specifies the *soname* that can go through the fallback link to the linker namespace `${another}`.
- If an *soname* cannot be resolved in linker namespace `${name}` and *soname* is one of the property values, the dynamic linker attempts to resolve the soname in the linker namespace `${another}`.
- Example: if `/vendor/lib/hw/vulkan.${chipset}.so` depends on `libc.so` but `libc.so` is neither in `/vendor/lib/hw` nor `/vendor/lib`, the dynamic linker attempts to find `libc.so` in the `default` namespace.

```
dir.system = /system/bin

[system]
additional.namespaces = sphal,vndk,rs

namespace.default.search.paths =
    /system/${LIB}

namespace.sphal.search.paths =
    /vendor/${LIB}/hw:/vendor/${LIB}

namespace.sphal.link.default.shared_libs =
    libc.so:libm.so
```

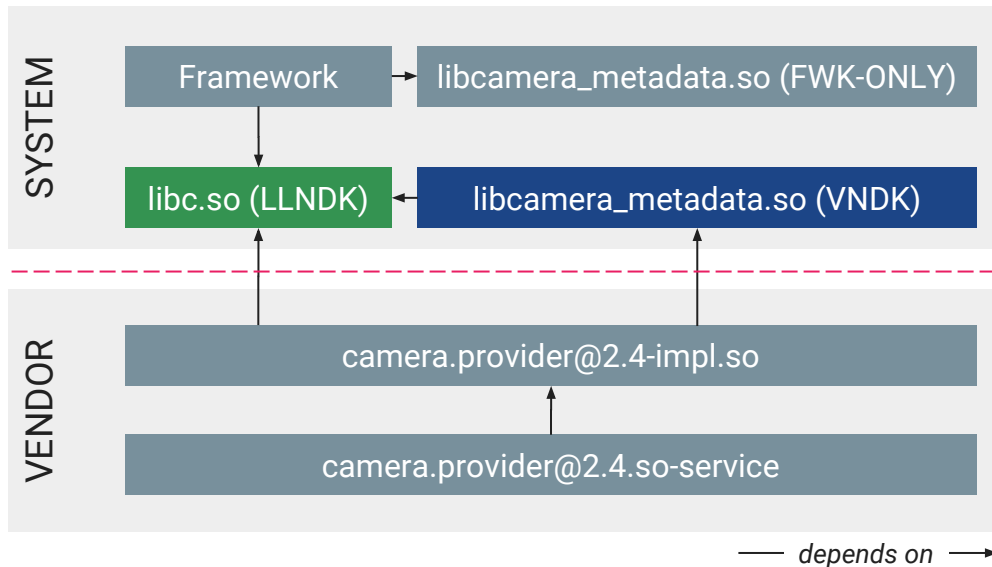
Build System Support

Motivations

- **Duplicate shared libraries when necessary**
 - Build vendor variant for users in vendor partitions
 - VNDK, VNDK-SP may be duplicated when needed
- **Make the build dependencies explicit**
 - Check whether headers, static libraries, shared libraries are available
 - Define the VNDK libraries that have to be installed into Generic System Image (GSI)
 - Generate VNDK snapshots for cross version system image development

Recap: VNDK categories

- **LLNDK** (LL-NDK + SP-NDK)
 - Shared libraries with stable APIs and loosely coupled with the framework
 - System and vendor share the same file
- **VNDK**
 - Specialized variant for vendor modules.
 - May be a FWK-ONLY counterpart with the same name
- **VNDK-SP**
 - Same as VNDK
 - Can be used by SP-HALs
 - May be loaded into framework process



Build system support (Android 8.0)

- To move a module to vendor partition:
 - Add **LOCAL_VENDOR_MODULE:=true** to Android.mk (or LOCAL_PROPRIETARY_MODULE)
 - Add **vendor:true** to Android.bp (or proprietary)
- To install a module to **both system and vendor partitions**, you need tricky build rules (see right, assigns *intermediate files* to LOCAL_PREBUILT_MODULE_FILE).

```
define define-vndk-lib
include $(CLEAR_VARS)
LOCAL_MODULE := $1.$2
LOCAL_MODULE_CLASS := SHARED_LIBRARIES
LOCAL_PREBUILT_MODULE_FILE := $(TARGET_OUT_INTERMEDIATE_LIBRARIES)/$1.so
LOCAL_STRIP_MODULE := false
LOCAL_MULTILIB := first
LOCAL_MODULE_TAGS := optional
LOCAL_INSTALLED_MODULE_STEM := $1.so
LOCAL_MODULE_SUFFIX := .so
LOCAL_MODULE_RELATIVE_PATH := $3
LOCAL_VENDOR_MODULE := $4
include $(BUILD_PREBUILT)

ifneq ($(TARGET_2ND_ARCH),)
ifneq ($(TARGET_TRANSLATE_2ND_ARCH),true)
include $(CLEAR_VARS)
LOCAL_MODULE := $1.$2
LOCAL_MODULE_CLASS := SHARED_LIBRARIES
LOCAL_PREBUILT_MODULE_FILE :=
$$($(TARGET_2ND_ARCH_VAR_PREFIX)TARGET_OUT_INTERMEDIATE_LIBRARIES)/$1.so
LOCAL_STRIP_MODULE := false
LOCAL_MULTILIB := 32
LOCAL_MODULE_TAGS := optional
LOCAL_INSTALLED_MODULE_STEM := $1.so
LOCAL_MODULE_SUFFIX := .so
LOCAL_MODULE_RELATIVE_PATH := $3
LOCAL_VENDOR_MODULE := $4
include $(BUILD_PREBUILT)
endif # TARGET_TRANSLATE_2ND_ARCH is not true
endif # TARGET_2ND_ARCH is not empty
endef
```

Build system support (Android 8.1)

- **BOARD_VNDK_VERSION := current** enables full VNDK support.
- If **BOARD_VNDK_VERSION := current** is specified in BoardConfig.mk, the build system:
 - Checks the header search path (and removes global default search paths).
 - Checks the link types of the shared libraries (i.e. vendor module can link only to LLNDK or vendor_available).
 - Builds vendor-specific VNDK libraries and install them to `/system/lib[64]/{vndk,vndk-sp}`.
 - Builds vendor-specific libraries and install them to `/vendor/lib[64]`.

VNDK-related properties in Android.bp:

- `vendor: true`
- `vendor_available: true`
- `vndk.enabled: true`
- `vndk.support_system_process: true`

vendor & vendor_available (Android 8.1)

- **vendor** specifies whether an Android.bp module is an vendor module or not.
 - If false, it cannot depend on the module with vendor equal to true.
 - If true, it can depend only on LLNDK or the module with vendor_available equal to true.
- **vendor_available** specifies whether an Android.bp module (header lib, static lib, or shared lib) is available to vendor.
 - If true and a **framework module** uses this module, the module is installed to the system partition.
 - If true and a **vendor module** uses this module, the vendor variant is built.
 - If `vndk.enabled` is false (or undefined), the module is installed to `/vendor/lib[64]`.
 - If `vndk.enabled` is true, the module is installed to `/system/lib[64]/vndk` or `/system/lib[64]/vndk-sp`.

VNDK section (Android 8.1)

- **vndk.enabled** specifies whether an Android.bp module is a VNDK library or not. It is a prerequisite to set `vendor_available` to true.
- **vndk.support_system_process** specifies whether an Android.bp module is a VNDK-SP library or not. Both `vendor_available` and `vndk.enabled` are prerequisites.

```
cc_library {  
    name: "libvendor_available",  
    vendor_available: true,  
}  
cc_library {  
    name: "libvndk",  
    vendor_available: true,  
    vndk: {  
        enabled: true,  
    },  
}  
cc_library {  
    name: "libvndksp",  
    vendor_available: true,  
    vndk: {  
        enabled: true,  
        support_system_process: true,  
    },  
}
```

target.vendor (8.1)

- **target.vendor** specifies vendor-specific build options.
 - Use the **exclude_srcs** property to exclude framework-specific source files.
 - Use the **exclude_shared_libs** property to exclude framework-specific shared libraries.

```
cc_library {  
  name: "libvnd_specific_example",  
  vendor_available: true,  
  target: {  
    vendor: {  
      exclude_srcs: ["framework_only.c"],  
      exclude_shared_libs: ["libfwk_only"],  
      cflags: ["-DEXTRA_VND_C_FLAGS"],  
      cppflags: ["-DEXTRA_VND_CPP_FLAGS"],  
    },  
  },  
}
```

Build support summary

- Define a vendor module which must be installed to vendor partition
 - `LOCAL_VENDOR_MODULE := true` (Android.mk)
 - `vendor: true` (Android.bp)
- Enable full VNDK build-time support (Android 8.1)
 - `BOARD_VNDK_VERSION := current` (BoardConfig.mk)
 - Build two variants: `vendor_available: true`
 - `VNDK:vndk.enabled: true`
 - `VNDK-SP:vndk.support_system_process: true`
- Disable runtime dynamic linker isolation between framework and vendor (Android 8.1)
 - `BOARD_VNDK_RUNTIME_DISABLE := true` (BoardConfig.mk)

VNDK Definition Tool

VNDK definition tool

- Scans the shared library dependencies
- Computes VNDK sets
- Checks for dependency violations
- Source at:

[`\${AOSP}/development/vndk/tools/definition-tool/vndk\_definition\_tool.py`](#)

Commands

- **vndk**. List VNDK libraries and other libraries that should be copied to vendor partitions.
- **check-dep**. Check for violations in shared library dependencies.
- **deps**. Print all resolved dependencies of shared libraries.
- **deps-insight**. Create HTML to show shared library dependencies.

vndk

- Lists VNDK-SP libraries and other libraries that should be copied to vendor partitions.
- Command line options:
 - **--system**: Path to your system partition directory.
 - **--vendor**: Path to your vendor partition directory.
 - **--aosp-system**: Path to GSI system partition directory (convert image with `simg2img` then mount).
 - **--tag-file**: Path to eligible list CSV file.
 - **--load-extra-deps**: Path to file specifying extra shared library dependencies.
 - **--full**: List all categories (for debugging).

```
vndk_definition_tool.py vndk \  
  --system path/system \  
  --vendor path/vendor \  
  --aosp-system path/gsi/system \  
  --tag-file eligible-list.csv \  
  --load-extra-deps deps.txt
```

```
vndk_sp:  
/system/lib/vndk-sp/libcutils.so  
  
vndk_sp_ext:  
/vendor/lib/vndk-sp/libion.so  
  
extra_vendor_libs:  
/vendor/lib/libvendor.so
```

check-dep

- Checks the dependencies and list the violating shared libraries and symbols.
- VNDK command line options plus:
 - **--module-info**: Path to `${ANDROID_PRODUCT_OUT}/module-info.json`
- Prints the following info or each violation:
 - Violating module and source path
 - Ineligible dependencies and source paths
 - Imported symbols from ineligible dependencies

```
vndk_definition_tool.py check-dep \  
  --system path/system \  
  --vendor path/vendor \  
  --aosp-system path/gsi/system \  
  --tag-file eligible-list.csv \  
  --load-extra-deps deps.txt \  
  --module-info module-info.json
```

```
/vendor/lib/libviolating.so  
  MODULE_PATH: libviolating/source  
  /system/lib/libineligible1.so  
    MODULE_PATH: ineligible1/source  
    symbol_a  
    symbol_b  
  /system/lib/libineligible2.so  
    MODULE_PATH: ineligible2/source  
    symbol_c
```

deps and deps-insight

- Debugging commands that print dependencies of shared libraries.
- **deps** prints plain text output
- **deps-insight** generates HTML for interactive investigation.
- Command line options are similar to the check-dep commands.

```
vndk_definition_tool.py deps \  
  --system path/system \  
  --vendor path/vendor \  
  --load-extra-deps deps.txt \  
  --module-info module-info.json
```

```
vndk_definition_tool.py deps-insight \  
  --system path/system \  
  --vendor path/vendor \  
  --aosp-system path/gsi/system \  
  --tag-file eligible-list.csv \  
  --load-extra-deps deps.txt \  
  --module-info module-info.json
```

JNI Libraries in Bundled APKs

JNI libraries in bundled apps (Android 8.1)

Shared libraries location	Bundled system app /system/app	Bundled vendor app /vendor/app	Downloaded app /data/app
/system/lib[64]	All	/system/etc/public.libraries.txt (NDK) + LLNDK	/system/etc/public.libraries.txt (NDK)
/vendor/lib[64]	/vendor/etc/public.libraries.txt	All	/vendor/etc/public.libraries.txt
/system/lib[64]/vndk-sp	x	Public VNDK-SP	x
/system/lib[64]/vndk	x	x	x