Open Source 西门子

Beijing · 2026

Oct. 29
Industrial Open Source

Oct. 30
Open Source Robotics

Agenda

Open Source Compliance for AOSP

Ryan Tao@Grandall Law Firm

陶冶@国浩律师事务所

Build environment

Source

android-16.0.0_r4

Product

aosp_cf_x86_64_phone

Release configuration

trunk_staging

Build system

Soong

Scope: Android 16 with Soong. Other build configurations may differ.

WHAT’S THE LICENSE OF AOSP

AOSP license data in one product build

SBOM measure	Count
File records	9,879
Package records	1,023
Installed file paths	4,660

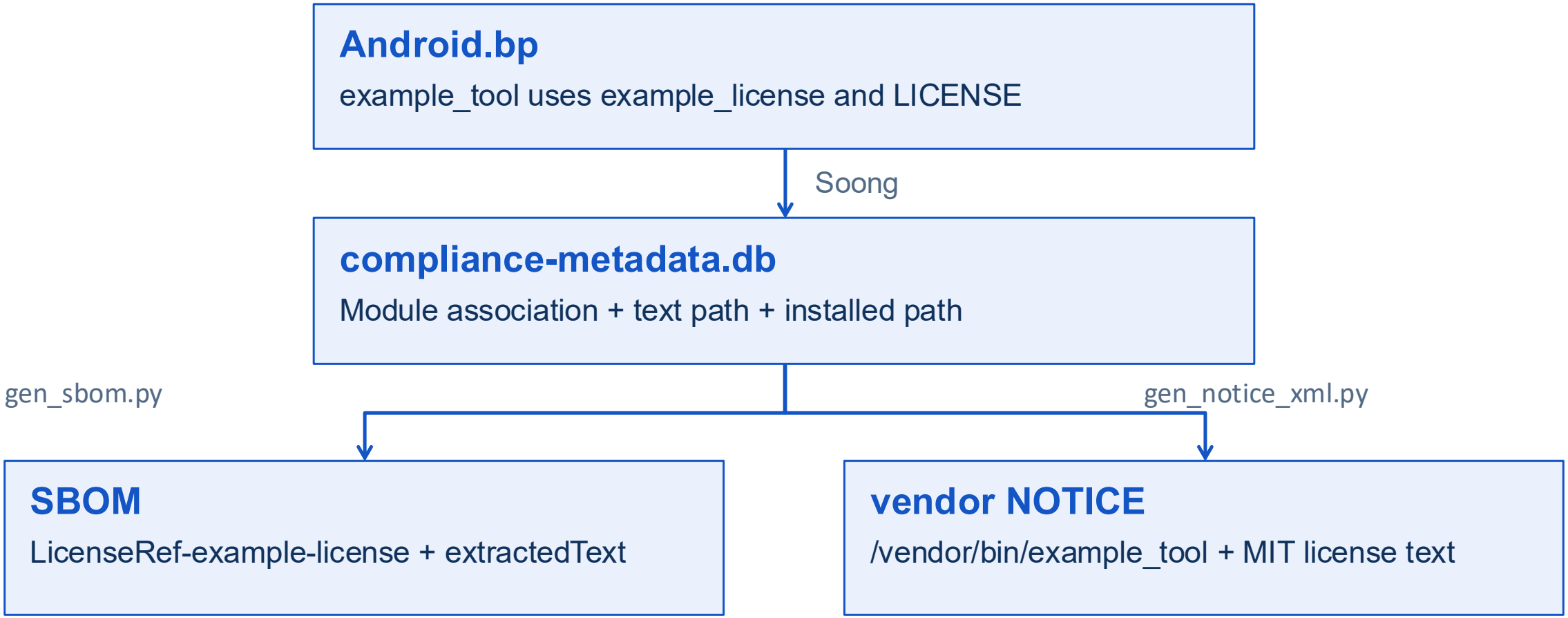
Android 16 r4, Cuttlefish x86_64.

How many different licenses?

Other measures	Count
SBOM LicenseRef records	731
Distinct license text blocks	394

Counts use the license modules behind the 731 SBOM references.

How the License Info reaches SBOM and NOTICE



Both generators read the LICENSE file referenced by license_text.

SBOM and NOTICE are generated respectively

SBOM

gen_sbom.py

Packages and files
License references
Explicit relationships

NOTICE

gen_notice_xml.py

Installed file paths
License text blocks
Selected static aggregation

An Android.bp file

```
package {
    default_applicable_licenses: [
        "example_license",
    ],
}
license {
    name: "example_license",
    license_kinds: [
        "SPDX-license-identifier-MIT",
    ],
    license_text: ["LICENSE"],
}
cc_binary {
    name: "example_tool",
    srcs: ["main.cpp"],
    vendor: true,
}
```

Package default

Applies example_license to modules without an explicit licenses property.

License text

LICENSE supplies the MIT text.

Build target

cc_binary compiles main.cpp.
The product selects example_tool for installation in /vendor/bin.

license_text

```
license {
    name: "example_license",
    license_kinds: [
        "SPDX-license-identifier-MIT",
    ],
    license_text: ["NOTICE"],
}
```

Valid text file

SBOM stores extractedText.
NOTICE includes the referenced text.

Missing license_text

License content is missing,
even with a recognized license kind.
No Build error

Missing referenced file

The build reports an error.

Default and explicit license associations

```
package {
    default_applicable_licenses: [
        "example_license",
    ],
}
license {
    name: "example_license",
    license_text: ["LICENSE"],
}
license {
    name: "other_license",
    license_text: ["OTHER_LICENSE"],
}
cc_binary {
    name: "example_tool",
    srcs: ["main.cpp"],
    licenses: ["other_license"],
    vendor: true,
}
```

Package default

example_license applies when a module omits licenses.

Explicit association

other_license is defined here. example_tool uses OTHER_LICENSE instead of the package default.

Both license text files must exist.

Missing declarations and invalid references

```
→ // Case A: no license association
  cc_binary {
     name: "example_tool",
     srcs: ["main.cpp"],
  }

  // Case B: undefined license module
  cc_binary {
     name: "other_tool",
→    licenses: ["missing_license"],
  }

  // Case C: nonexistent text file
  license {
     name: "example_license",
→    license_text: ["MISSING"],
  }
```

No associated licenses

With no package default or explicit licenses: NOTICE absent, SBOM NOASSERTION.
No Build error

Undefined module reference

Build error.

Invalid text reference

Build error.
Omitting license_text leaves a content gap in both outputs.

license_kind and conditions

```
license {
    name: "example_license",
    license_kinds: [
        "SPDX-license-identifier-AGPL-1.0",
    ],
    license_text: ["LICENSE"],
}

license_kind {
    name:
     "SPDX-license-identifier-AGPL-1.0",
    conditions: [
        "by_exception_only",
        "not_allowed",
    ],
}
```

license_kinds

References a license_kind definition.

license_kind

Defines the license classification.
It does not supply the license text.

conditions

Supplies policy labels for checks.
Build blocking requires integration.
No direct SBOM / NOTICE text change.

Generated outputs still need review

AOSP 16 r4 can produce incomplete or inconsistent SBOM and NOTICE files.

Correct component declarations are the starting point.

Six recurring gaps

01 Direct copies through PRODUCT_COPY_FILES

02 Generated .fsv_meta files

03 Missing upstream license declarations or text

04 Kernel modules from BoardConfig

05 Partitions outside default NOTICE coverage

06 Static dependencies outside three partitions

Direct file installation

PRODUCT_COPY_FILES

Product makefiles specify a source file and its destination.

PRODUCT_COPY_FILES in product makefiles

Defined in product .mk files and inherited device configuration files.

Example: device/google/cuttlefish/vsoc_x86_64/phone/aosp_cf.mk

PRODUCT_COPY_FILES += <source>:<destination>

Source frameworks/native/data/etc/aosp_excluded_hardware.xml

Destination vendor/etc/permissions/aosp_excluded_hardware.xml

PRODUCT_COPY_FILES

Direct copy, without a named Android.bp module or its license association.

prebuilt_etc

Named Android.bp module with an effective license association.

The SBOM generator’s hard-coded allowlist

build/make/tools/sbom/gen_sbom.py

```
def get_license_of_product_copy_file(file_path):
  paths = {
      'device/sample/etc/',
      'frameworks/av/media/libeffects/data/',
      'frameworks/av/media/libstagefright/data/',
      'frameworks/av/services/audiopolicy/config/',
      'frameworks/base/config/',
      'frameworks/base/data/keyboards/',
      'frameworks/base/data/sounds/',
      'frameworks/native/data/etc/',
      'hardware/google/camera/devices/EmulatedCamera/hwl/configs/',
      'system/core/rootdir/etc/',
  }
  for p in paths:
    if file_path.startswith(p):
      return 'build/soong/licenses/LICENSE'
  return ''
```

The copied file can have the wrong license

For PRODUCT_COPY_FILES sources in the hard-coded path allowlist

SBOM

Apache-2.0, regardless of the actual license.

NOTICE

No license entry for the copied file.

The SBOM rule uses the source path. It does not read the file’s Android.bp license.

Options for copied files

Module installation

use prebuilt_etc module to do direct file copies

Required direct copy

Complete SBOM and NOTICE information manually

Generated metadata files

.fsv_meta files

A separate SBOM rule for build-generated helper files.

What is a .fsv_meta file?

Build-generated metadata for fs-verity,
Android’s file-integrity verification mechanism.

The build derives it from selected files and installs it alongside them.

SBOM assigns Android Apache-2.0.
NOTICE omits the corresponding entry.

Add matching NOTICE entries when both documents must cover the same files.

Missing upstream license text

License declarations without license_text

AOSP components can omit the text required by the generators.

Upstream declarations can omit the text

prebuilts/rust/Android.bp, shortened excerpt

```
license {
    name: "prebuilts_rust_stdlib_license",
    visibility: [":__subpackages__"],
    license_kinds: [
        "SPDX-license-identifier-Apache-2.0",
        "SPDX-license-identifier-MIT",
        // Additional kinds omitted here
    ],
    // large-scale-change unable to identify
    // any license_text files
}
```

Kernel module installation

BoardConfig.mk

Device configuration can select kernel modules for installation.

BoardConfig kernel installation

BoardConfig.mk configures the device's kernel and image build.

```
# BoardConfig.mk
BOARD_SYSTEM_KERNEL_MODULES := \
    path/example.ko
```

SBOM: NOASSERTION
NOTICE: no license entry

Even with valid license_text

Soong uses a copy-based installation path, similar to PRODUCT_COPY_FILES.
The automatic build does not retrieve the configured license text.

A Soong module can fix that

```
license {
    name: "kernel_license",
    license_kinds: [
        "SPDX-license-identifier-GPL-2.0-only",
    ],
    license_text: ["COPYING"],
}
prebuilt_kernel_modules {
    name: "example_kernel_module",
    srcs: ["example.ko"],
    system_dlkm_specific: true,
    licenses: ["kernel_license"],
}
```

Module-based installation

Android.bp defines the .ko installation and explicitly associates its GPL text.

GPL text reaches both SBOM and NOTICE.

Select the module in PRODUCT_PACKAGES. Review kernel loading and dependencies separately.

NOTICE coverage by partition

Which partitions receive a NOTICE file?

Each delivered area needs a declaration and an image-build dependency.

Default NOTICE coverage

Eight partition targets are declared by default

system, system_ext, system_dlkm
product
vendor, vendor_dlkm
odm, odm_dlkm

root, ramdisk, vendor_ramdisk and other delivered areas need additional NOTICE configuration.

Declarations: build/soong/compliance/Android.bp

Additional partition configuration

Ramdisk NOTICE declaration and image dependency

```
// build/soong/compliance/Android.bp
notice_xml {
    name: "notice_xml_ramdisk",
    partition_name: "ramdisk",
}

// build/soong/fsgen/fsgen_mutators.go
"ramdisk": {
    "notice_xml_ramdisk": defaultDepCandidateProps(ctx.Config()),
},
```

The product's image build must include the NOTICE target.

Static-library license coverage

Which dependency licenses reach NOTICE?

An installed program can contain code from several static libraries.

Static libraries require dependency coverage

SBOM

Separate library entries

STATIC_LINK relationships

Library license references

NOTICE

Installed parent path with transitive static dependencies

The final artifact contains code from its static dependencies.

Static aggregation of NOTICE runs for three partitions

build/make/tools/sbom/gen_notice_xml.py, comment omitted

```
if args.partition in ('vendor', 'product', 'system_ext'):
  static_dep_modules = get_transitive_static_dep_modules(metadata, db)
  for dep in static_dep_modules:
    dep['installed_file'] = metadata['installed_file']
    installed_file_metadata_list.append(dep)
```

vendor, product, system_ext
Other partitions can retain static-library licenses in SBOM while NOTICE omits them.

Two remedies for static dependency gaps

NOTICE generator with the partition condition removed

```
# Apply the existing collection logic
# without the partition condition.
static_dep_modules = get_transitive_static_dep_modules(
    metadata, db)
for dep in static_dep_modules:
  dep['installed_file'] = metadata['installed_file']
  installed_file_metadata_list.append(dep)
```

Component option
Parent license_text includes library text.

Generator option
Remove the condition.

Result on Android 16:
Other partitions include static license text.

Thanks



Ryan Tao@Grandall Law Firm

Open Source 西门子

Beijing · 2026

Oct. 29
Industrial Open Source

Oct. 30
Open Source Robotics

Agenda