Next Steps for Software Bill of Materials (SBOM) Generation in Zephyr

Steve Winslow

Boston Tech Law

steve@swinslow.net

AGENDA

1. Intro to SBOM / SPDX
2. Zephyr and SPDX
3. Implementation Details
4. Group Discussion

WHAT IS AN SBOM?

Software Bill of Materials

"a formal record containing the details and supply chain relationships of various components used in building software" - ntia.gov/SBOM

SBOM
Software Bill of Materials
ntia.gov/sbom #SBOM

WHAT IS SPDX?

Software Package Data Exchange

"An open standard for communicating software bill of material information, including components, licenses, copyrights, and security references" - spdx.dev

ISO/IEC 5962:2021 (ISO spec, ISO PAS, latest)

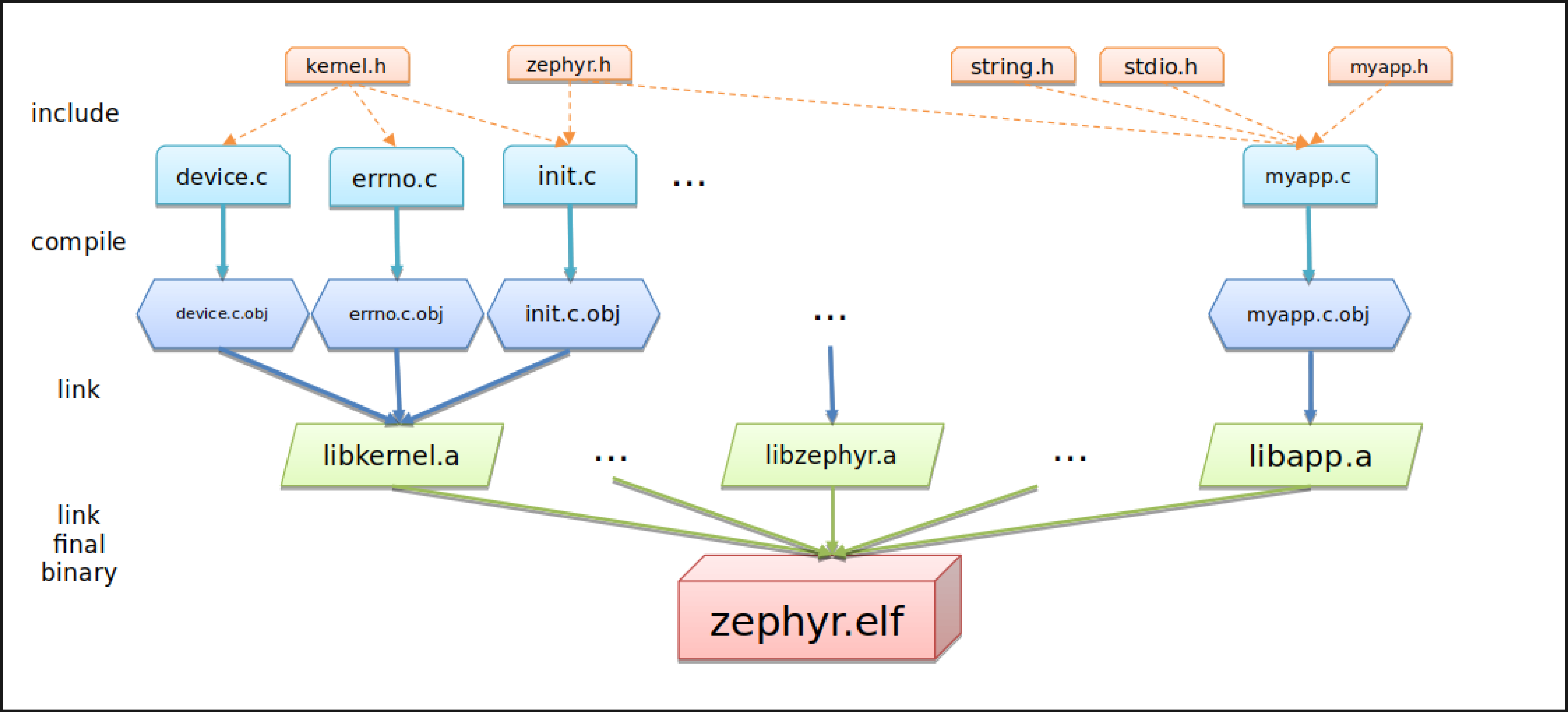
ZEPHYR AND SPDX

Added in 2.6.0: `west spdx`

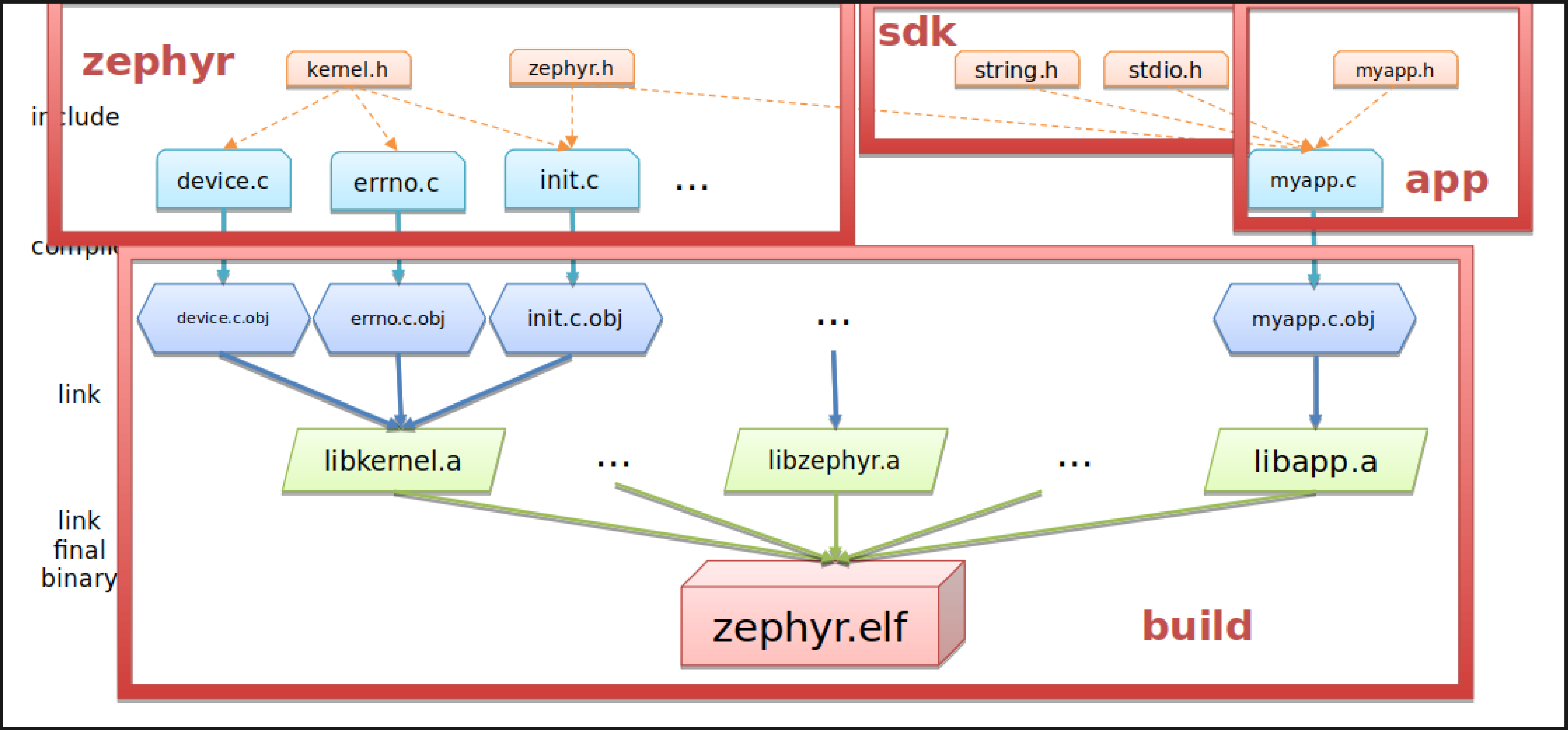
Create SPDX documents simultaneously with Zephyr `west build`; one additional command

Leverages CMake file-based API metadata

ZEPHYR AND SPDX



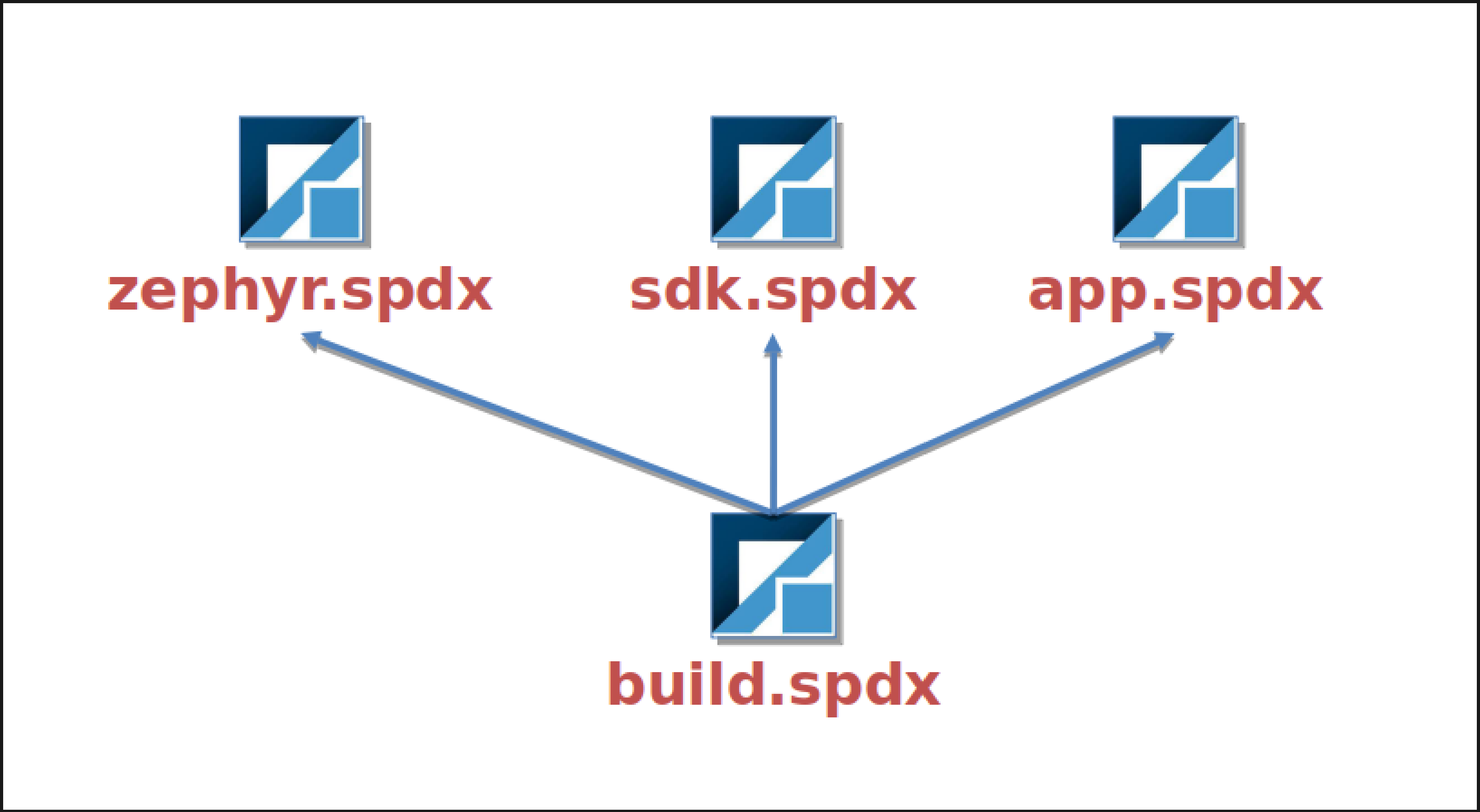
ZEPHYR AND SPDX



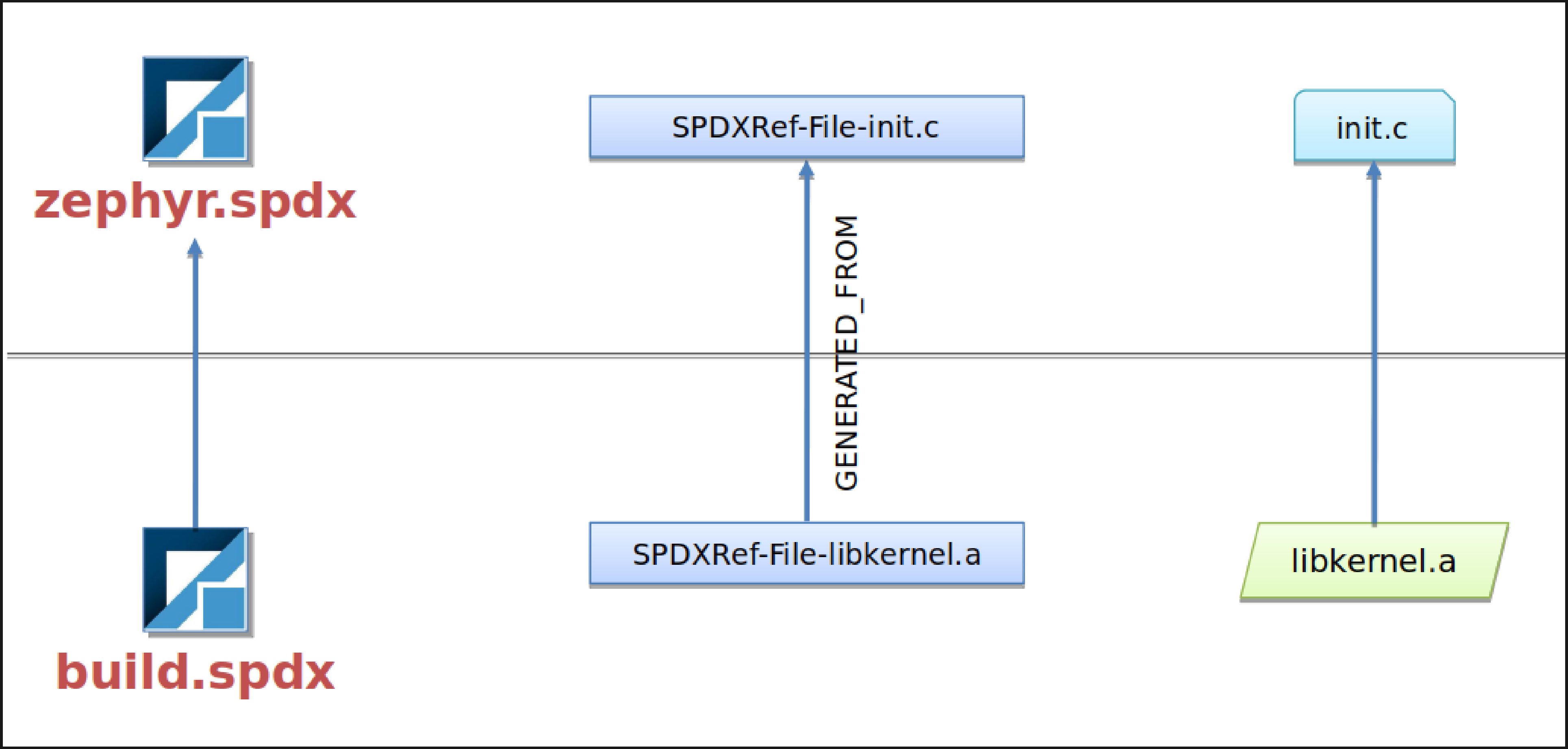
ZEPHYR AND SPDX

- SPDX Documents for:
  1. Zephyr sources
  2. (optionally) sources used from SDK
  3. application sources
  4. build outputs
- hashes for file integrity
- relationships for tracing compiles, linking
- license data from SPDX-License-Identifier tags

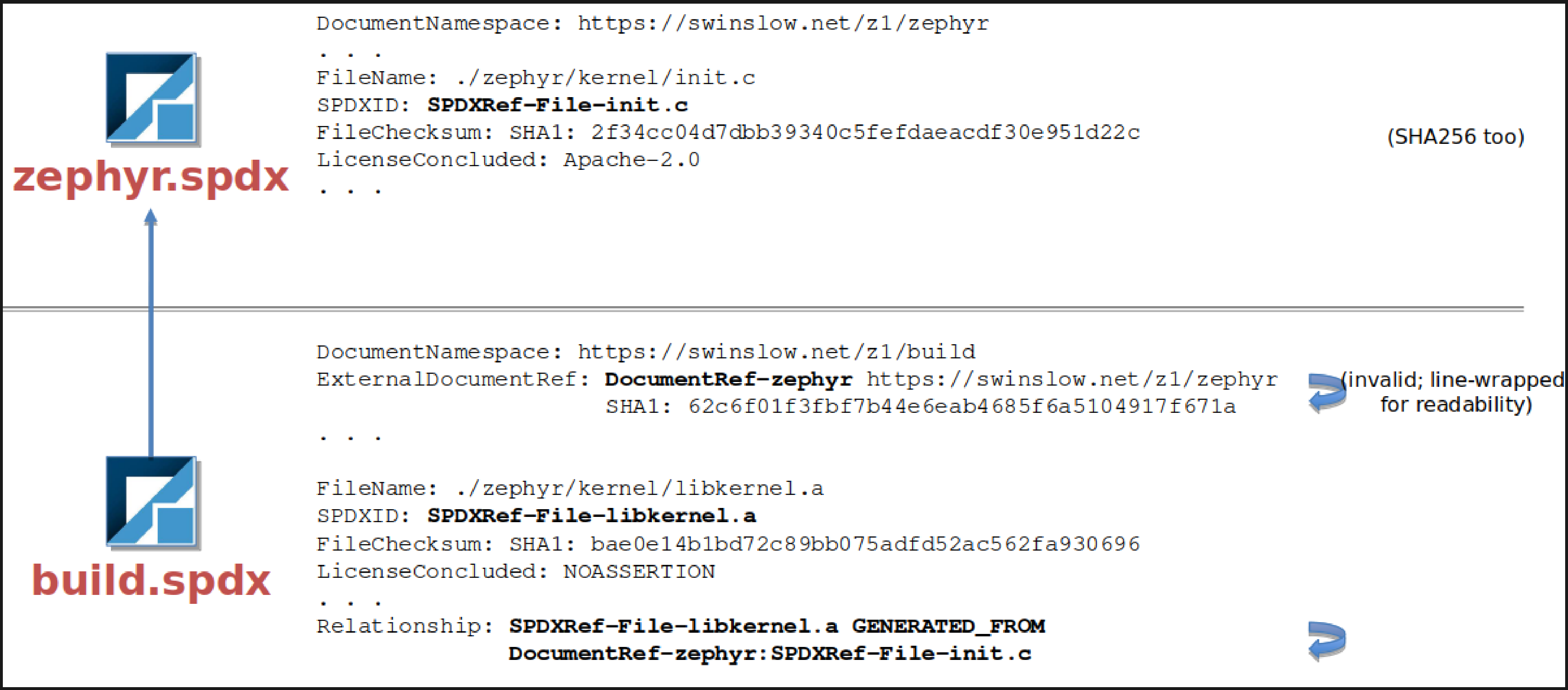
ZEPHYR AND SPDX



ZEPHYR AND SPDX



ZEPHYR AND SPDX



```
DocumentNamespace: https://swinslow.net/z1/zephyr
. . .
FileName: ./zephyr/kernel/init.c
SPDXID: SPDXRef-File-init.c
FileChecksum: SHA1: 2f34cc04d7dbb39340c5fefdaeacdf30e951d22c
LicenseConcluded: Apache-2.0
. . .
```

(SHA256 too)

```
DocumentNamespace: https://swinslow.net/z1/build
ExternalDocumentRef: DocumentRef-zephyr https://swinslow.net/z1/zephyr
                     SHA1: 62c6f01f3fbf7b44e6eab4685f6a5104917f671a
. . .

FileName: ./zephyr/kernel/libkernel.a
SPDXID: SPDXRef-File-libkernel.a
FileChecksum: SHA1: bae0e14b1bd72c89bb075adfd52ac562fa930696
LicenseConcluded: NOASSERTION
. . .
Relationship: SPDXRef-File-libkernel.a GENERATED_FROM
              DocumentRef-zephyr:SPDXRef-File-init.c
```

(invalid; line-wrapped for readability)

BENEFITS

1. traceability and security
2. evidence from build time
3. license info management
4. standardized / interchangeable

CMAKE METADATA

To activate, create empty file at:

```
$BUILD_DIR/.cmake/api/v1/query/codemodel-v2
```

When building, CMake outputs metadata at:

```
$BUILD_DIR/.cmake/api/v1/reply/
```

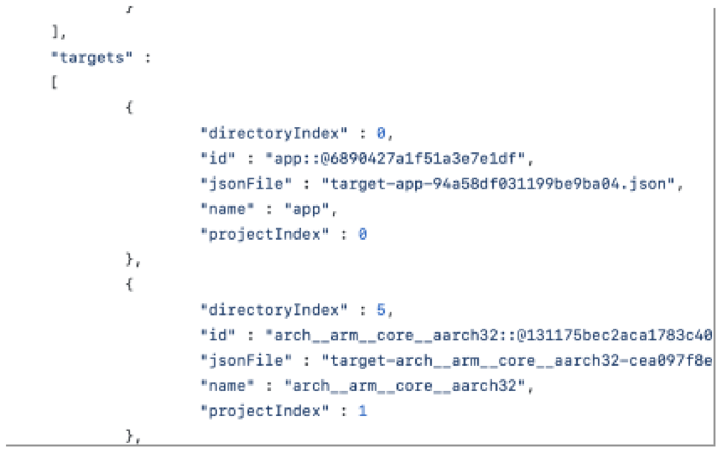
CMAKE METADATA

Outputs JSON metadata file for each build stage target:

- target file being built (e.g. `libkernel.a`)
- target dependencies (prior build stages)
- source files
- compiler command-line options used

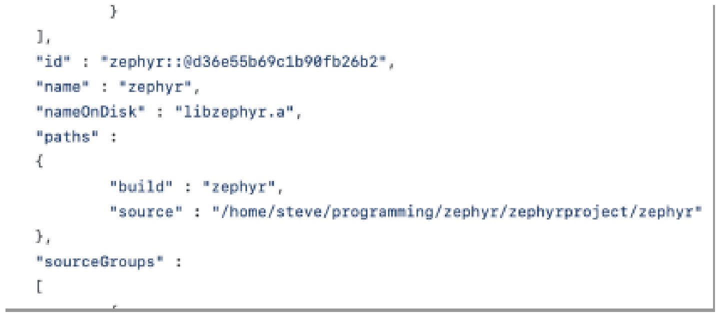
CMAKE METADATA

Codemodel JSON

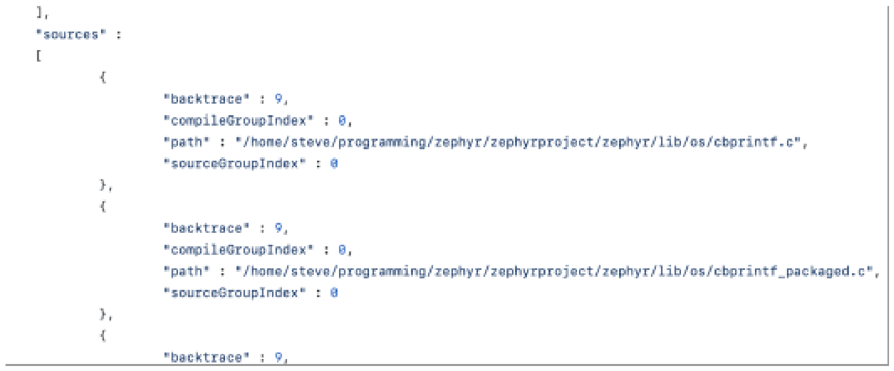


```
        ,
],
"targets" :
[
        {
                "directoryIndex" : 0,
                "id" : "app::@6890427a1f51a3e7e1df",
                "jsonFile" : "target-app-94a58df031199be9ba04.json",
                "name" : "app",
                "projectIndex" : 0
        },
        {
                "directoryIndex" : 5,
                "id" : "arch__arm__core__aarch32::@131175bec2aca1783c40
                "jsonFile" : "target-arch__arm__core__aarch32-cea097f8e
                "name" : "arch__arm__core__aarch32",
                "projectIndex" : 1
        },
```

Targets JSON

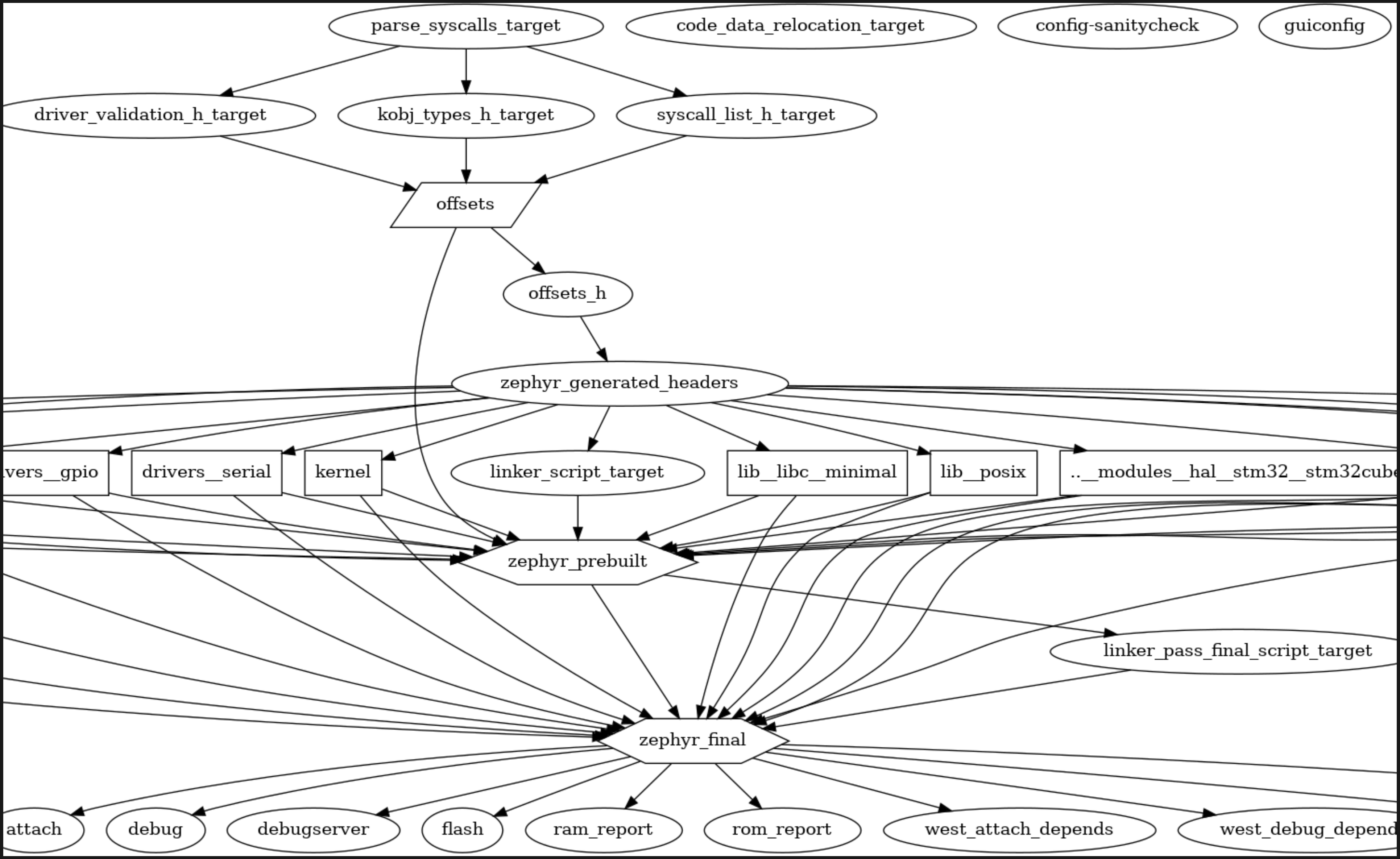


```
        }
],
"id" : "zephyr::@d36e55b69c1b90fb26b2",
"name" : "zephyr",
"nameOnDisk" : "libzephyr.a",
"paths" :
{
        "build" : "zephyr",
        "source" : "/home/steve/programming/zephyr/zephyrproject/zephyr"
},
"sourceGroups" :
[
        ,
```



```
],
"sources" :
[
        {
                "backtrace" : 9,
                "compileGroupIndex" : 0,
                "path" : "/home/steve/programming/zephyr/zephyrproject/zephyr/lib/os/cbprintf.c",
                "sourceGroupIndex" : 0
        },
        {
                "backtrace" : 9,
                "compileGroupIndex" : 0,
                "path" : "/home/steve/programming/zephyr/zephyrproject/zephyr/lib/os/cbprintf_packaged.c",
                "sourceGroupIndex" : 0
        },
        {
                "backtrace" : 9,
```

CMAKE METADATA



ZSPDX

`/scripts/west_commands/zspdx/`

Main entry point in `sbom.py`

Call from command line via `west spdx`

ZSPDX

Three stages:

- **walker**: analyze the build
  - parse CMake JSON metadata
  - analyze relationships between targets and files
  - optionally: analyze includes via dry-run recompiles
- **scanner**: scan files for hashes and license IDs
- **writer**: output SPDX documents

CURRENT DEFICIENCIES

Makes several assumptions about user's build process

Limited to files that CMake reports in metadata

Treats *all* CMake target stages as SPDX Packages

Not everyone wants source-level SBOMs

Doesn't include all NTIA minimum fields

DISCUSSION

USAGE

Who is using this now?

Who would be interested in using this?

ASSUMPTIONS

Which assumptions are wrong?

- use of CMake, SDK, ...
- structure of code trees
- relationships between files

NEXT FEATURES

What else would be useful to have?

- Simplified SBOMs (package level only)
- Include all NTIA-required fields
- Better licensing info and detection
- Pick up more details on file versions
- Analyze files differently
- Include details on build tools used
- Generate distribution tarball with sources

COMMUNITY

Are you willing to participate in improving / testing?

- collaborate on identifying use cases, build setups
- validate assumptions
- explore which files should be included vs. excluded
- add and improve features

If yes, contact me!

steve@swinslow.net

GitHub: swinslow

THANK YOU

github.com/swinslow/slides

Zephyr Developer Summit 2022