Reference Binary Artifacts
Effort

Bruce Ashfield
Overview

- History
- Problem statement
- Goals
- Landscape
- Status
- Ecosystem
- Call to action
History

- OE has a long history with packages / binaries
  - See other presentations for details
- Idea was discussed at ELCe Lyon 2019 (and earlier)
- This specific effort is part of the 5-year planning activities
  - Started in early 2021
    - ~12 observers/members across 10 companies/individuals
    - Draft Goals, phases, requirements
    - Priority and ‘vision’ discussions
    - Roughly bi-monthly meetings
- Effort renewed in April 2022
  - Tangible outputs expected by mid year
Problem statement / solution

- It is proposed to have reference “binaries” available for commonly used components to address ease of use concerns, lower barriers to adoption and test the core infrastructure for binary artifact maintenance.
- These would be available, with a defined path into the standard source based "build your own distribution" and other core pillars of the ecosystem.
Terminology

● What do we mean by "binary artifacts" and "ease of use"?

● Binary artifacts:
  ○ Outputs from a defined build that can be used / installed on a running target, or to construct a target image. The architecture and optimization are defined by the build parameters, and can impact the level of reusability

● Ease of use:
  ○ It is obvious / clear how to complete (initial) steps towards a goal
  ○ Details vary by use case
  ○ Transitioning between use cases is supported and documented
Goals / Requirements

- Don’t want to ‘adopt’ anything existing
  - Equally not looking to displace existing distros
- OEcore / Yocto Project support for binary artifact maintenance
  - Documentation, core technology, testing
- Reference artifacts available for commonly used components to address ease of use concerns, lower barriers to adoption
  - Defined path into the standard "build your own distribution" and other core pillars of the ecosystem
  - Generic architecture: x86-64, ARM64
- Defined extension to broader ecosystem / vendor artifacts
  - SDK / eSDK
  - Embrace and extend
- Rolling release / no release
Important points

- Focus on the plumbing / infrastructure / best practices
- Reuse, not reinvent
- This is a reference, not a product
- Well defined capabilities and packages, with tangible components
- Not optimized
- Extension is key: with a path to full source build
- Always consistent with the Yocto Project messaging and core competencies
- Not associated with a particular member or existing distro
More than packages (aka let’s modernize)

- **Binary:**
  - Standard/generic packages
  - Optimized packages
  - Application
  - Container build
  - License compliance
  - Board support
  - Customization
  - Universal packages

- **Source:**
  - (e)SDK
  - Application
  - License compliance
  - 'Customer' support
  - Packages
  - Container build
  - Customization

- **‘Easy(ier)’**
  - (e)SDK
  - License compliance
  - Customization
  - Board support

- **‘Hard(er)’**
  - Application
  - License compliance
  - 'Customer' support
  - Packages
  - Container build

- **Category:**
  - Developer
  - User
  - Production
Beyond packages: Things to consider

- **Reproducibility**
  - core Yocto Project capability (see reproducible-builds.org)
- **Licensing / SBOM**
  - core capability
  - multiple ways to consume it and accompany binary deliveries
- **Customization**
- **Support from the ecosystem (if modified/extended, or not)**
- **Platform Extension**
- **Application AND system developers**
- **Support, maintenance, and updates**
Status

- **Phase0**
  - Preparation, planning, and prioritization

- **Phase1**
  - Definition and documentation
    - Images, tuning, architectures, target boards (qemu*), quickstart guides, etc

- **Phase2**
  - develop, test and deploy
  - CI/CD, nightly/weekly testing, image deployment, demos (boot, extend, etc)

- **Phase3**
  - Extend and maintain
  - Containers (dockerhub, etc), ecosystem expansion, hardware reference(s) ...

Currently in phase 1.5
Ecosystem integration

- Once stable, expand outside of OEcore
- Container frameworks, IoT, etc
- Updaters (OTA or otherwise)
- Complex image formats
- Installers
- …
Call to Action

- Join at any time: we need development resources as well as input
  - Tools
  - Documentation
  - Compute resources
  - Test cycles
  - ....