X(cross) Development System
make AGL application development easier

ALS 2017
Tokyo - June 2017

Sébastien Douheret
sebastien.douheret@iot.bzh
IoT.bzh : 1st technical contributor

- Application Development
  - AGL Development Kit
  - Secure Application Framework (life cycle, cybersecurity)
  - Application Binder Framework (APIs exposure & protection)

- Integration
  - Yocto recipes
  - Releases automation & Testing (CI)
  - Renesas boards support
  - Security (MAC, Cynara, Systemd, CGroups, Namespaces,...)

- Low Level Services
  - Audio Management
  - Connectivity
  - Signaling & Events / CAN
  - SOTA
  - Secure Boot & Trusted Zone

- Community Support
  - Documentation (kickstart, developer samples, guides ...)
  - White Papers & Conferences (Genivi, AGL, Fosdem, ELC ...)
  - Renesas Community support
X(cross) Development System (XDS)

- Dedicated to Applications Developers → enable apps developer without Yocto skills
- Cross-platform build using AGL SDK toolchain
- Secure packaging (.wgt files including signatures)
- Deploy on development boards (or Qemu image)
- Remote debugging from IDE
- Easy target access (console, SSH, ...)
- Developer environment is a standard IDE
- Dashboard Web App to manage configuration and trigger actions
- Provide an XDS API → ie CI workflows or specific environments
Key Features

- **Multi-platform**: no dependencies on developer host (Linux / Windows / MacOS)

- **Easy to setup**
  Near-zero install, no admin privileges required

- **Application sources remain local**
  Compatibility with existing IT policies (e.g. corporate backup, git, ...)

- **Cross toolchain & tools embedded in a container**:
  - Local: run locally (local subsystem, virtual machine, docker container ...)
  - On-premises: run on a local build server
  - Cloud: SaaS

- **Leverage specific OS capabilities** where applicable (e.g. Linux for Windows Subsystem, docker)
Presentation video...
Targeted Use Cases

**Single Host mode**
- Host: Windows
- IDE: Eclipse
- Container: Linux Subsystem
- Sources: shared through native access

**Local Network mode**
- Host: Linux
- IDE: Visual Code
- Container: Docker
- Sources: shared through docker volume

**Cloud mode**
- Host: Linux
- IDE: Eclipse
- Container: Docker running in the Cloud
- Sources: shared through sync tool
These blocks may be bypassed when files can be shared over a local network (eg. NFS mount or docker volume)
XDS Implementation

- Written in GO (portable Linux/Windows/MacOS)
- Dashboard (webapp Angular2, TypeScript 2) to make development setup easier
- REST API + WebSocket (socket.io)
- File synchronization based on Syncthing [1] to support Cloud model
- XDS server integrated into AGL SDK docker image (see flavour xds [2])

[1]: https://syncthing.net/
[2]: https://git.automotivelinux.org/AGL/docker-worker-generator/
XDS building blocks

- **xds-server**
  - Core of the system, running in container:
    - Provide Dashboard as a webapp
    - REST API interface over HTTP to config and trigger commands
    - Websocket to get asynchronously data (commands output) or events (commands exit)
    - Control file synchronizer (Syncthing) on server/ in container
    - Manage (install, list, remove) AGL SDKs

- **xds-agent**
  - Client side part, running on developer host.
    - Control file synchronizer (Syncthing) on developer's machine
    - Target terminal *(work in progress)*

- **xds-exec**
  - Command line tool to wrap standard exec command.
Develop AGL app with XDS

2) Download xds-agent, install and start-it on desktop
3) Add a new project
4) Cross build/compile your project using either:
   • Dashboard build page
   • IDE using xds-exec
5) Deploy app
6) Source debug from IDE (work in progress)
Available as today

• Use pre-build docker image and tarballs

  docker load < agl_worker-xds-3.2.tar.xz


• Build latest version from sources

  • xds-server: https://github.com/iotbzh/xds-server#how-to-build
  • xds-agent: https://github.com/iotbzh/xds-agent
  • xds-exec: https://github.com/iotbzh/xds-make
Live Demo...
Roadmap

• To Be Done
  • Complete docker packaging and server side upgrade
  • Document REST API
  • Improve dashboard and error reporting
  • AGL SDK management (add, remove)
  • Target terminal
  • Better integration in IDE (plugin, templates)
  • Improve/document debugging support (TCF support)
  • Include security process (package signing)
  • …

• Roadmap
  • AGL AMM Fall ’17: developer preview
  • EE/CES ’18: release candidate
Contacts

- Sébastien Douheret
  <sebastien.douheret@iot.bzh>
- Stéphane Desneux
  <stephane.desneux@iot.bzh>
Gulf of Morbihan, south of Brittany, France
Why not reusing CROPS?

- Core of CROPS project(1)(2) no longer active, replaced by a single Eclipse plugin(3)
- New Eclipse plugin too much focused on Eclipse and Docker
- No RESTful API: based on sockets (doesn’t support corporate networks with firewalls as HTTP does)
- File synchronization for Cloud configuration not supported

(1): https://github.com/crops/crops
(2): https://www.youtube.com/watch?v=R54vRP0-omw
(3): https://github.com/crops/eclipse-crops