X(cross) Development System
make AGL application development easier

Sébastien Douheret
sebastien.douheret@iot.bzh
IoT.bzh

- Located in France - Brittany ( Vannes / Lorient / Rennes )
- People background:
  - 40% coming from Tizen IVI (Intel+Samsung)
  - 40% coming from Wind River
  - 20% coming from industry
- Working on AGL since 2015
- First technical contributor on AGL
Key Focuses

• Cybersecurity
  • Secure Application Framework (life cycle, MAC, Cynara, Systemd, Cgroups, Namespaces …)
  • Vehicle to Cloud, Identity management
  • Securing key platform services (audio, signaling, graphics …)

• Development Tools
  • Docker Platform development (AGL SDK)
  • Apps development from IDE: XDS (Cross Development System)
  • Platform Monitoring tools

• Integration
  • Yocto recipes, Releases automation & Testing (CI)
  • Renesas boards support

• Community Support
  • Documentation (kickstart, developer samples, guides …)
  • White Papers & Conferences (Genivi, AGL, Fosdem, ELC …)
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)
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
Architecture

Development Host
Linux / Windows / MacOS

IDE UI
(Eclipse, Visual Studio Code, ...)
XDS plugin

Command line
xds-exec

XDS-agent
App sources
Files

File Sync

XDS Server Container
Local or remote

XDS -serverDaemon

Local network or
Cloud network

REST + WS

Target
deployment

File Sync

Files
direct link
(eg. USB <=> Ethernet)

Cross Toolchain B

Cross Toolchain A

Files
App sources
(duplicated)

AGL repo

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)**
  - 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.

- **xds-gdb**
  - Command line tool to allow application debugging (based on gdb).
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

- Beta version
  - Pre-build docker image including XDS server
    https://github.com/iotbzh/xds-server#how-to-run
  - Multi-platform (Linux / Windows/ MacOS) host tools (xds-agent, xds-exec, xds-gdb)
  - Support Cloud mode (file sharing based on a synchronization tool)
  - IDE integration (manual setup)
    https://github.com/iotbzh/xds-exec#using-xds-exec-within-an-ide
  - Debugging proof of concept (based on gdb and AGL app-templates)
- Video showing development and debug
  https://vimeo.com/221493917

- Binary packages available on openSUSE Build Service:
  https://build.opensuse.org/project/show/isv:LinuxAutomotive:app-Development

- Sources available on Github
  - xds-server : https://github.com/iotbzh/xds-server
  - xds-agent : https://github.com/iotbzh/xds-agent
  - xds-exec: https://github.com/iotbzh/xds-exec
  - xds-gdb: https://github.com/iotbzh/xds-gdb
Next version

Release Candidate 1 (AGL AMM – Fall’ 17)

- Complete packaging and server side upgrade
- Improve dashboard and error reporting
- Support multiple file sharing (simple path-mapping, NFS, synchting)
- AGL SDK management (add, remove)
- Target setup and target terminal integration in dashboard
- Better integration in IDE (templates)
- Improve/document debugging support
- Document REST API
Roadmap

- AGL AMM Fall ‘17: Release Candidate 1
- EE/CES ‘18: Release Candidate
Contacts

• Sébastien Douheret
  <sebastien.douheret@iot.bzh>

• Stéphane Desneux
  <stephane.desneux@iot.bzh>
Q&A

Gulf of Morbihan, south of Brittany, France