Building & Integrating
Smart Device Link Core
What is SDL Core

• Implemented on OEM’s head unit
• Provides messaging system
• Sets protocol standard
• Supports multiple transport types
• Handles application policies
• Portable for existing HMI modules
SDL Core: Component Breakdown

- Transport Manager
- Protocol Handler
- Connection Handler
- Application Manager
- Policy Manager
- Mobile Message Handler
- HMI Message Handler
- Settings Manager
Getting Started

- Clone SDL Repository
- Clone HMI Repository
- Release branches are most stable

```
git clone https://github.com/smartdevicelink/sdl_core.git
git checkout -b release/4.0.0 origin/release/4.0.0

git clone https://github.com/smartdevicelink/sdl_hmi.git
git checkout -b release/4.0.0 origin/release/4.0.0
```
## Dependencies

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web HMI</td>
<td>Used with HTML5 HMI</td>
<td>chromium-browser</td>
</tr>
<tr>
<td>HMI2</td>
<td>Build with QT HMI</td>
<td>QT5, dbus</td>
</tr>
<tr>
<td>Extended Media Mode</td>
<td>Support Audio and Video Streaming</td>
<td>OpenGL es2, gstreamer1.0*</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Enable bluetooth transport adapter</td>
<td>libbluetooth3, libbluetooth-dev, bluez-tools</td>
</tr>
<tr>
<td>Flag</td>
<td>Description</td>
<td>Dependencies</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Testing Framework</td>
<td>Supports unit testing</td>
<td>libgtest-dev</td>
</tr>
<tr>
<td>Cmake</td>
<td>Configures SDL prior to compilation</td>
<td>cmake</td>
</tr>
<tr>
<td>Enable Logs</td>
<td>3rd Party Logging Platform</td>
<td>log4cxx</td>
</tr>
<tr>
<td>Policy DB</td>
<td>Database stores policy table</td>
<td>sqlite3</td>
</tr>
</tbody>
</table>
Build Configuration

- Setup build environment outside of Core’s directory and compile

```bash
cmake -DHMI=qt -DHMI2=ON ..../sdl_core
make
make install
```
Running SDL Core

• Copy keys and run core

   cp bin/mykey.pem src/appMain
   cp bin/mycert.pem src/appMain

   cd src/appMain
   ./smartDeviceLinkCore

• Start HMI (if using web hmi)

   cd sdl_hmi
   chromium-browser index.html
Connecting Core to HMI

• HMI Transport types
  • Web Sockets (HTML5 HMI)
  • D-Bus (Qt HMI)
  • Custom
ConnectingCore to HMI

• Communication Requirements
  • Establish Connection
  • Confirm Readiness
  • Correctly Respond to Messages
WebSocket Connection

• Create components from HMI_API
• Establish connection for each component
• SDL listens on port :8087
• WebSocket Protocol - Version 13
WebSocket Connection

• Sample Message Format

```javascript
var JSONMessage = {
    "jsonrpc": "2.0",
    "id": itemIndex,
    "method": "SDL.ActivateApp",
    "params": {
        "appID": appId
    }
};
this.client.send(JSONMessage);
```
D-Bus Connection

• Client/Server relationship
  • SDL - Client
  • HMI - Server

• Publish D-Bus Service

QDBusConnection::sessionBus().registerObject("/", this);
QDBusConnection::sessionBus().registerService("com.ford.sdl.hmi");
Modifying HMI Config

• Modify CMakeLists.txt
  • Set HMI flag
  • Set HMI Adapter (Message Adapter)
  • Set System/Build environment variables
• Bash Scripts (D-Bus & Qt)
  • FindQt.sh - Called by top level CMAKE
  • start_hmi.sh - called by InitHMI() in main.cc
Creating Custom HMI

SDL core expects that HMI supports the following behavior:
• A request to call the remote function can be synchronous or asynchronous
• All data transfer is performed through function’s arguments
• The successfully executed function returns a common (successful) response with “success” return code
• The failed call returns the successful response with “error” return code
• The notification is distributed on definite event occurred.
Creating Custom HMI: Main HMI Object

• Import Library Plugins
  • import com.ford.sdl.hmi.dbus_adapter
• Create Storage Objects
  • Data Storage
  • Settings Storage
• Create View Controller/Content Loader
Creating Custom HMI: Main HMI Object

- Initialize HMI Adapter
- Includes Components
- Initialize SDL Adapter
- Insert onReady() signal
Creating Custom HMI

- Define HMI Capabilities
  - hmi_capabilities.json
- Create HMI Component Objects
  - Basic Communication, Buttons, Voice Recognition, Text to Speech, etc’