Smart Device Link (SDL)

About

SmartDeviceLink (SDL) is a standard set of protocols and messages that connect applications on a smartphone to a vehicle head unit. This messaging enables a consumer to interact with their application using common in-vehicle interfaces such as a touch screen display, embedded voice recognition, steering wheel controls and various vehicle knobs and buttons. There are three main components that make up the SDL ecosystem.

- The Core component is the software which Vehicle Manufacturers (OEMs) implement in their vehicle head units. Integrating this component into their head unit and HMI based on a set of guidelines and templates enables access to various smartphone applications.
- The optional SDL Server can be used by Vehicle OEMs to update application policies and gather usage information for connected applications.
- The iOS and Android libraries are implemented by app developers into their applications to enable command and control via the connected head unit.

SDL Core

The Core component of SDL runs on a vehicle's computing system (head unit). Core's primary responsibility is to pass messages between connected smartphone applications and the vehicle HMI, and pass notifications from the vehicle to those applications. It can connect a smartphone to a vehicle's head unit via a variety of transport protocols such as Bluetooth, USB, Android AOA, and TCP. Once a connection is established, Core discovers compatible applications and displays them to the driver for interaction via voice or display. The core component is implemented into the vehicle HMI based on a set of integration guidelines. The core component is configured to follow a set of policies defined in a policy database and updated by a poli cy server. The messaging between a connected application and core is defined by the Mobile API and the messaging between sdl core and the vehicle is defined by the HMI API.

Mobile Proxy

The mobile library component of SDL is meant to run on the end user's smart-device from within SDL enabled apps. The library allows the apps to connect to SDL enabled head-units and hardware through bluetooth, USB, and TCP. Once the library establishes a connection between the smart device and head-unit through the preferred method of transport, the two components are able to communicate using the SDL defined protocol. The app integrating this library project is then able to expose its functionality to the head-unit through text, media, and other interactive elements.

Server

The SmartDeviceLink (SDL) server handles authentication, data collection, and basic configurations for SDL connected vehicles. In general these tasks are accomplished using JSON documents called Policy Tables that are configured by the server and then downloaded by other SDL components. The server's backend API (written in node.js) handles these types of requests and can be easily extended to handle more. Configuration of Policy Tables, or any other data, can be done using the server's front-end GUI (written in Angular.js). Customization of the front-end with your own look and feel is encourage and worry-free because these changes will not affect the underlying functionality of the server.

SHAID

The Super Helpful Application ID (SHAID) server manages Application IDs used within the SmartDeviceLink (SDL) ecosystem. This includes creation of unique application IDs, anonymous distribution of the IDs to SDL partners, and the ability to revoke an existing ID. The SHAID server can also notify subscribing SDL Servers when an ID is revoked, allowing the SDL Server to take appropriate action.

Usage Scenarios

Infrastructure

Git repositories: https://github.com/smartdevicelink

Contributing

Third party contributions are essential for making SDL great. However, we do have a few guidelines we need contributors to follow.

Issues

If writing a bug report, please make sure it has enough info. Include all relevant information.

If requesting a feature, understand that we appreciate the input! However, it may not immediately fit our roadmap, and it may take a while for us to get to your request.

Gitflow

We use Gitflow as our branch management system. Please follow gitflow's guidelines while contributing to any SDL project.

Pull Requests

- Please follow the repository's for all code and documentation.
- All feature branches should be based on `develop` and have the format `feature/branch_name`.
- Minor bug fixes, that is bug fixes that do not change, add, or remove any public API, should be based on `master`and have the format `hotfix/branch_name`.
- All pull requests should implement a single feature or fix a single bug. Pull Requests that involve multiple changes (it is our discretion what precisely this means) will be rejected with a reason.
- All commits should separated into logical units, i.e. unrelated changes should be in different commits within a pull request.
- Work in progress pull requests should have "[WIP]" in front of the Pull Request title. When you believe the pull request is ready to merge, remove this tag and @mention the appropriate SDL team to schedule a review.
- All new code *must* include unit tests. Bug fixes should have a test that fails previously and now passes. All new features should be covered. If your code does not have tests, or regresses old tests, it will be rejected.
- A great example of a pull request can be found here.

Contributor's License Agreement (CLA)

In order to accept Pull Requests from contributors, you must first sign the Contributor's License Agreement. If you need to make a change to information that you entered, please contact us.

SmartDeviceLink Mailing List

> The best place to have open community discussion about SmartDeviceLink is on the GENIVI SmartDeviceLink mailing list, and within issues and pull requests that are submitted to each of the projects on GitHub.

Related Work

Minutes and Presentations