

TOOLS FOR

SMARTDEVICELINK

SCOTT SMEREKA



- ▶ Software Engineer & Web Lead at Livio
- ▶ Background
 - ▶ Used to work on Software for Tanks
 - ▶ Join Livio January 2012
 - ▶ Livio was acquired by Ford in late 2013
 - ▶ Working on SDL and internal research projects

HOW WE HELP YOU DO WHAT YOU DO BEST — BETTER

- ▶ Slack
- ▶ Developer Portal
- ▶ Github
- ▶ Tools in Development – **New**
 - ▶ Generic HMI
 - ▶ Manticore
 - ▶ SHAID
- ▶ Timeline



+



Join **SmartDeviceLink** on Slack.

7 users online now of **233** registered.

you@yourdomain.com

GET MY INVITE

or [sign in](#).

SLACK

Open chat with the SDL community

ASK US ANYTHING

- ▶ Fastest way to find an answer to any problem, just ask!
 - ▶ But, please attempt a solution first
- ▶ Live chat with over 230 members of the SDL community on our public Slack Channel
 - ▶ Register at slack.smartdevicelink.com

SLACK CHANNELS

- ▶ Channels – Separate conversations by topics
 - ▶ Quicker answers if using correct channel
 - ▶ Frequently Asked Questions
 - ▶ Search
 - ▶ Slackbot

8/28 – 9/4 SUMMARY

- ▶ Developers are actually using slack
- ▶ 43% of traffic in public chats
- ▶ People tend to talk directly to whoever is helping them in a direct message

Your team sent a total of **997 messages** last week (that's 461 more than the week before). Of those, **43% were in public channels**, **2% were in private channels** and **55% were direct messages**. Your team also uploaded **12 files** (that's 3 more than the week before).



HACKATHON CHANNEL

► A slack channel is available for tonights #hackathon

The screenshot shows a Slack channel named #hackathon with 21 members. The channel header includes a search bar and icons for phone, settings, and a list. The conversation history shows several messages:

- A message from an unnamed user: "That is the 4.1.0 release for the sd_l_hmi portion. This is so Today open source variant and has nothing to do with STI". Below it, a message says: "For the hackathon I would propose you setup the SYNC3 emulator that you can find here: <https://developer.ford.com>".
- ophedian** (10:46 AM) replies to @timur: "Currently this is what I have setup on my PC: Android Studio, Ubuntu 14+ via VM, Sync 3 (which is running cloned all the github repositories to local (incase WiFi is flaky))."
- timur** (10:46 AM) replies: "if you have the SYNC3 emu up and running you should not need the sd_l_hmi".
- ophedian** (10:48 AM) replies: "Cool... now I am just going to worry about how a C# guy is going to learn Android Dev, Java, and Sync in 3 days 😂".
- timur** (10:53 AM) replies: "Our people will help you with SmartDeviceLink. Stackoverflow will the rest 😊".
- ophedian** (12:57 PM) replies to @timur: "that is the master plan :)".
- troy** (1:51 PM) replies to @timur: "the sync3 emulator won't switch to apps when running in virtual machine and no network connection. Have settings set to Host-only adapter. Works fine when I have a network connection and set to bridged adapter. trying to figure out how to can test with simulator in VM and no network connection on host".
- ophedian** (2:44 PM) uploads an image with the caption "I have arrived".

The image shows a blue lanyard with a badge that reads: "ctia Super Mobility 2016™", "Everything Wireless", "ABBEY", "GWAYAMBADDE", "CTO", "TRINECAST", "SHOW FLOOR".

At the bottom of the channel, there is a text input field with a plus icon and the text "Message #hackathon".

sdl

🔍 search anything

DEVELOPER PORTAL

Learn about smartdevicelink.com

OVERVIEW

- ▶ The first place you should go to find anything related to SDL is SmartDeviceLink.com
- ▶ The main goals are to help
 - ▶ Developers get their apps into vehicles
 - ▶ OEMs implement SDL in their vehicles

WHATS IN THE PORTAL?

- ▶ Guides
- ▶ Documentation
- ▶ Developer Console – **New!**
- ▶ Partner Information – **New!**
- ▶ Downloads / Links to Resources
- ▶ Latest News

GUIDES

- ▶ Guides - Explain how to do common or more complicated tasks
 - ▶ e.g. Getting Started, Configuring Core, Video Streaming
- ▶ Created based on need of the developers
- ▶ Located under **Learn** on smartdevicelink.com

The screenshot shows the 'Guides Android' page on the SDL Developer Portal. The page has a green header with the SDL logo and a search bar. A navigation menu on the left lists 'Getting Started', 'Lockscreen', and 'Remote Procedure Calls'. The main content area is titled 'Installation' and includes a 'SOURCE' section with instructions on how to download and import the source code into an Android Studio project. It also includes a 'GRADLE' section with a code snippet for adding the library as a dependency.

sd1 search anything Learn Partners Documentation Reso

Guides Android

← Guides

Filter...

Getting Started

Lockscreen

Remote Procedure Calls ▾

Installation

SOURCE

Download the latest release from [GitHub](#)

Extract the source code from the archive

Import the code in the `sd1_android_lib` folder as a module in Android Studio:

1. Click File -> New -> Import Module... -> Choose the location of `sd1_android_lib` as your Source Directory
2. The module name will automatically be set to `sd1_android_lib`, change this as desired. Click Next and then Finish

The `sd1_android_lib` is now a module in your Android Studio project, but it needs to be added as a dependency to your application. Add the following to the gradle dependencies of your project:

```
dependencies {
    compile project(path: ':sd1_android_lib')
}
```

GRADLE

The SDL Android libraries are not yet available for download from jcenter

DOCUMENTATION

- ▶ Android and iOS
 - ▶ Comprehensive documentation for the mobile libraries
 - ▶ e.g Classes, Constants, Enums, Protocols, Type Definitions, etc
- ▶ HMI
 - ▶ Describes how to integrate an OEM's user interface with SDL Core
 - ▶ e.g WebSocket Transport, Buttons, Navigation, TTS, VR, etc

The screenshot shows the SDL Developer Portal documentation page for iOS. The page has a teal header with the SDL logo, a search bar, and navigation links for 'Learn', 'Documentation', and 'Resources'. Below the header, the page title is 'Documentation iOS'. A left sidebar contains a 'Filter...' section with dropdown menus for 'Classes', 'Constants', 'Enums', 'Protocols', and 'Type Definitions'. The main content area is titled 'Constants Reference' and includes a 'Section Contents' list with the following items:

- [SDLProtocolSecurityErrorDomain](#)
- [SDLErrorDomainStreamingMediaVideo](#)
- [SDLErrorDomainStreamingMediaAudio](#)
- [SDLDefaultScreenSize](#)
- [SmartDeviceLinkVersionNumber](#)
- [SmartDeviceLinkVersionString](#)

At the bottom of the main content area, there is an 'Overview' section.

DOCUMENTATION

- ▶ SDL Server
 - ▶ In depth look at policy tables and how to update policies in a vehicle
 - ▶ e.g Policy Table Attributes and Policy Table Updates
- ▶ SHAID
 - ▶ How to integrate with the software tool used to synchronize application and user data across the OEMs
 - ▶ e.g API documentation, Authentication, and Examples

The screenshot shows the SDL Developer Portal documentation page for Application Policies. The page has a green header with the SDL logo, a search bar, and navigation links for Learn, Documentation, and Resources. The main content area is white with a green sidebar on the left. The sidebar contains a 'Filter...' section and a list of navigation items: Overview, Policy Table (expanded), Application Policies (selected), Consumer Friendly Messages, Device Data, Functional Groupings, Module Config, Module Meta, Usage and Errors, and Policy Table Update. The main content area displays the title 'Application Policies' and a paragraph explaining that an application's permissions and settings are stored in the **app_policies** property in a policy table, used to grant access to features like vehicle data and background running.

SDL

search anyth

Learn Documentation Resource

Documentation SDL Server

Filter...

Overview

Policy Table ▲

Overview

Application Policies

Consumer Friendly Messages

Device Data

Functional Groupings

Module Config

Module Meta

Usage and Errors

Policy Table Update

Application Policies

An application's permissions and settings are stored in the **app_policies** property in a policy table. The application policies are used to grant applications access to a specific set of features such as vehicle data and/or running in the background. Any other application related data, such as user-consents, can also be stored in application policies as well.

Application ID

DEVELOPER CONSOLE (NEW)

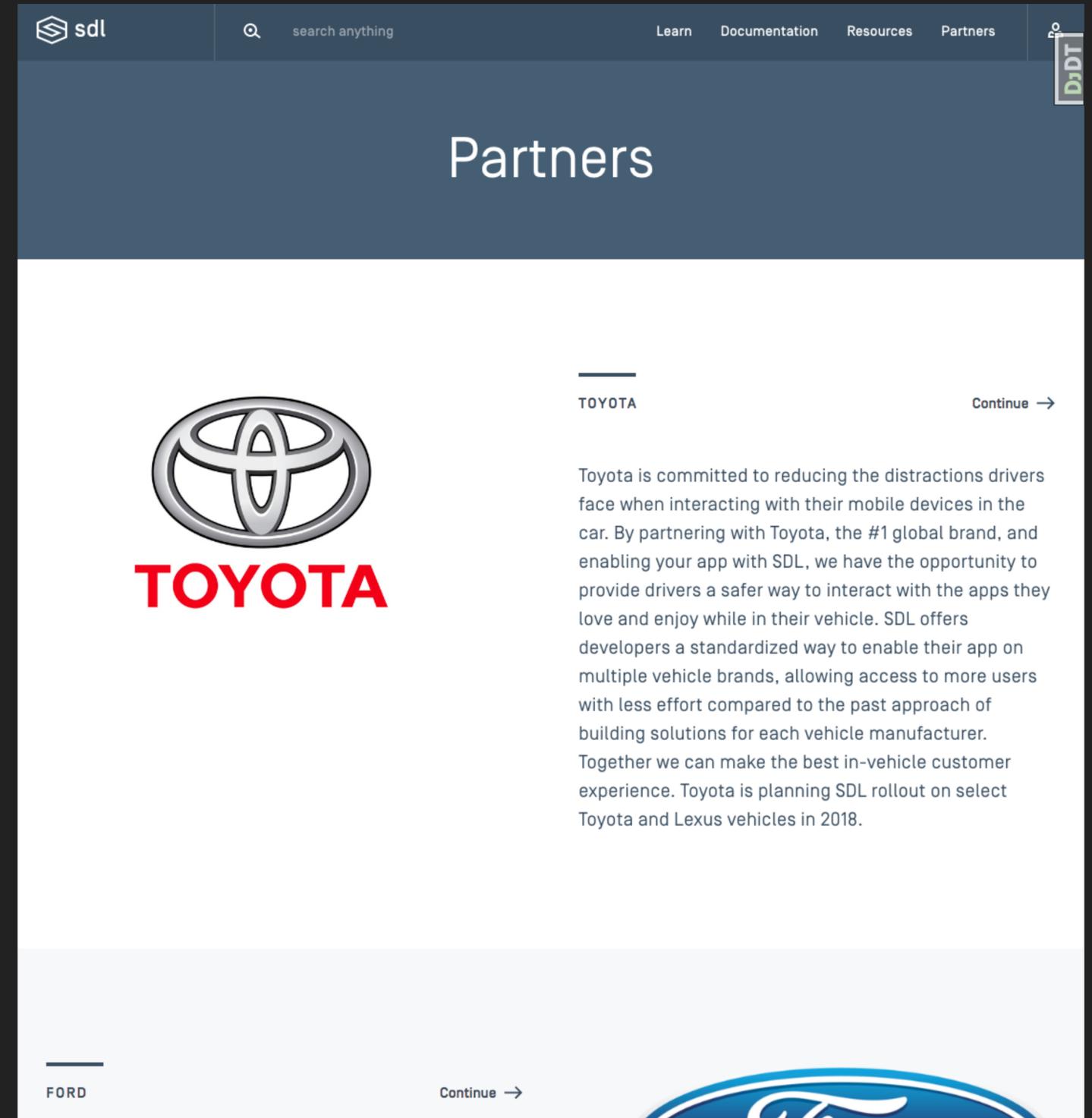
- ▶ Application IDs - Generate IDs that are globally unique across the entire SDL ecosystem
- ▶ Profile Settings - Configure your personal developer portal experience. For example
 - ▶ Default Code block themes
 - ▶ Default Language
 - ▶ Email address and user information

The screenshot shows the 'Application Details' form in the Developer Console. The form is set against a dark blue header with the 'sdl' logo and a search bar. The form fields are as follows:

- NAME:** A text input field.
- DESCRIPTION:** A text input field.
- PLATFORM:** A dropdown menu currently showing 'iOS'.
- ADD TO APP DIRECTORY:** A checkbox labeled 'Allow app to be searchable in our app directory', which is currently unchecked.
- Buttons:** A green 'Create ID' button and a red 'Cancel' button.

PARTNERS (NEW)

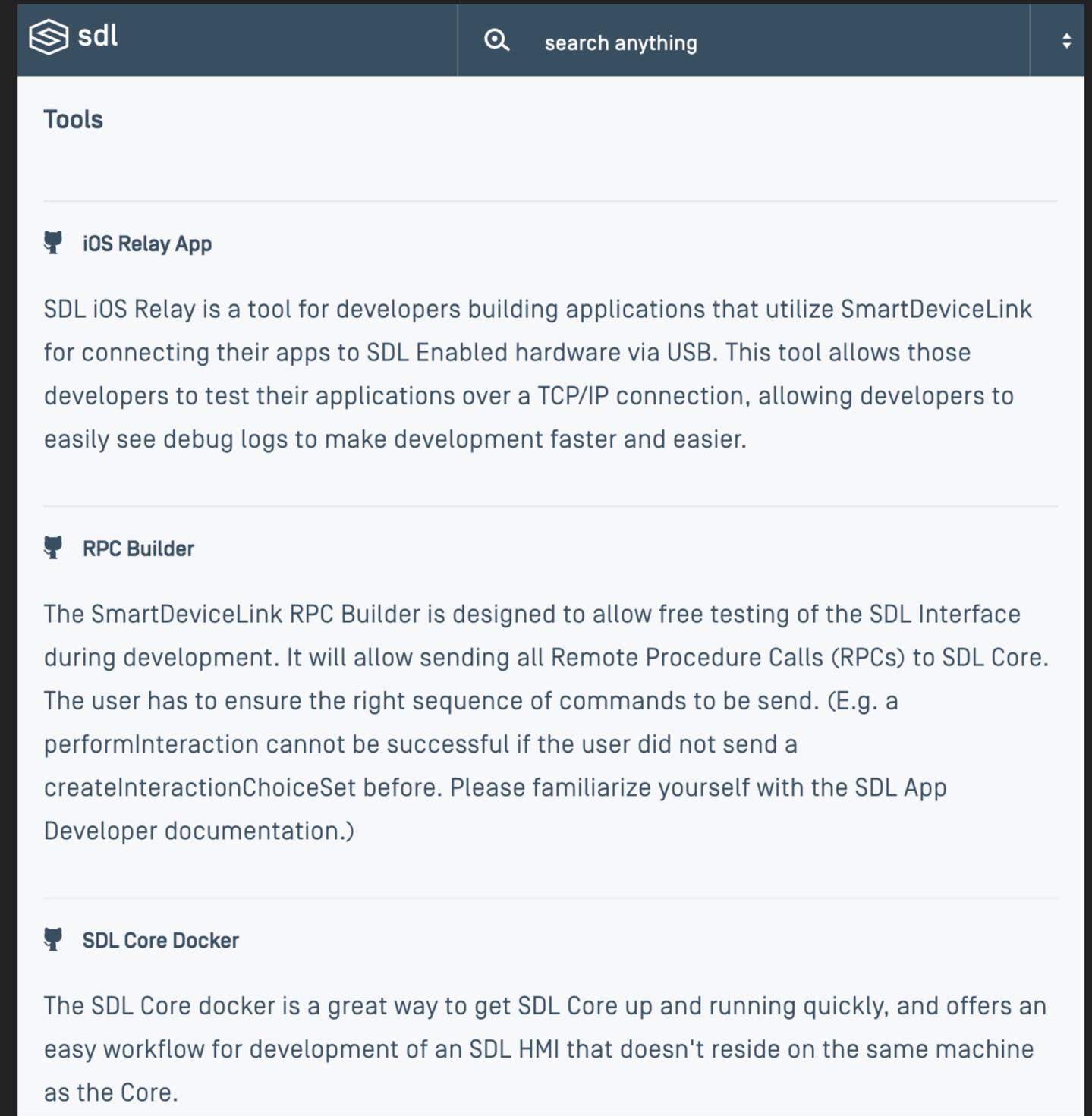
- ▶ Each OEM can create and manage their own page
 - ▶ Can see what vehicle's SDL is in
 - ▶ Helps developers contact the OEM
 - ▶ Can use our custom content management system
- ▶ Want an OEM page?
 - ▶ Email us at support@smartdevicelink.com



The screenshot shows the 'Partners' page of the SDL Developer Portal. The top navigation bar includes the SDL logo, a search bar, and links for 'Learn', 'Documentation', 'Resources', and 'Partners'. The main heading is 'Partners'. Below this, a card for Toyota is displayed. The card features the Toyota logo and the word 'TOYOTA' in red. To the right of the logo is a 'Continue →' link. The text on the card describes Toyota's commitment to reducing driver distractions and mentions a 2018 rollout for select Toyota and Lexus vehicles. At the bottom of the page, a 'FORD' card is partially visible with its own 'Continue →' link.

RESOURCES

- ▶ Downloads or Links to
 - ▶ Example applications and sample code
 - ▶ Android and iOS libraries
 - ▶ SDL Core docker container
 - ▶ Software tools
 - ▶ SDL logos and marketing materials



The screenshot shows the 'Tools' section of the SDL Developer Portal. At the top, there is a dark blue header with the 'sdl' logo on the left and a search bar containing the text 'search anything' on the right. Below the header, the word 'Tools' is displayed in a bold, dark font. The page lists three tools, each with a GitHub icon and a title:

- iOS Relay App**: A tool for developers building applications that utilize SmartDeviceLink for connecting their apps to SDL Enabled hardware via USB. This tool allows those developers to test their applications over a TCP/IP connection, allowing developers to easily see debug logs to make development faster and easier.
- RPC Builder**: The SmartDeviceLink RPC Builder is designed to allow free testing of the SDL Interface during development. It will allow sending all Remote Procedure Calls (RPCs) to SDL Core. The user has to ensure the right sequence of commands to be send. (E.g. a performInteraction cannot be successful if the user did not send a createInteractionChoiceSet before. Please familiarize yourself with the SDL App Developer documentation.)
- SDL Core Docker**: The SDL Core docker is a great way to get SDL Core up and running quickly, and offers an easy workflow for development of an SDL HMI that doesn't reside on the same machine as the Core.

SEARCH

- ▶ Easiest way to find anything is to search for it
- ▶ Searching is tracked
 - ▶ We update documentation to satisfy common search terms





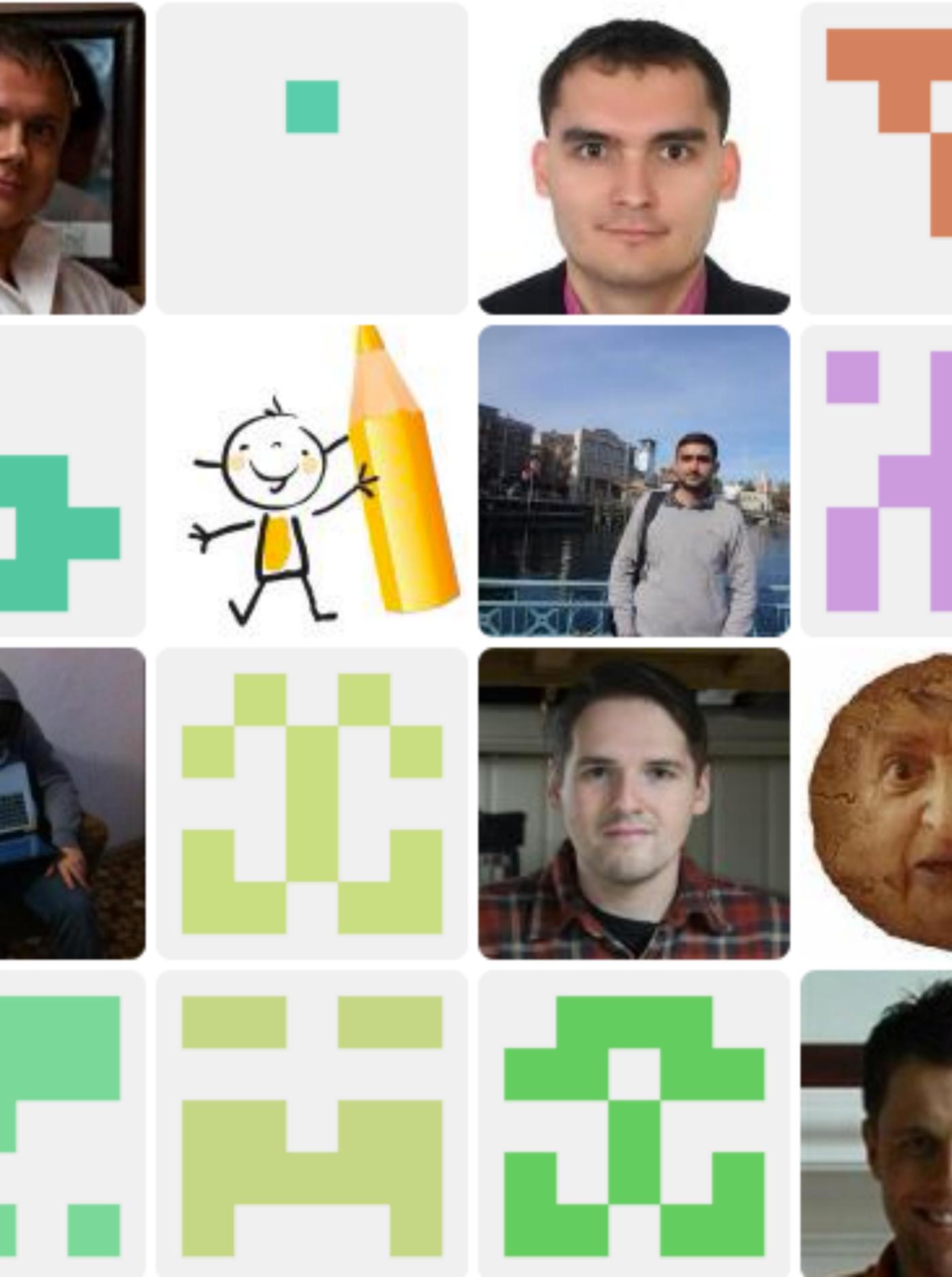
GENERIC HMI

Example SDL user interface



OVERVIEW

- ▶ The [Generic HMI](#) will become the default user interface for SDL core
 - ▶ See Joey's presentation
- ▶ The HMI is entirely web based (Javascript & React)
 - ▶ Easier for developers to test with
 - ▶ Important for other tools like Manticore



GITHUB

a.k.a. the code mines

OVERVIEW

- ▶ All **code** and **documentation** is managed on [github](#)
- ▶ All documentation is written in [DocDown](#)
- ▶ We follow [Gitflow](#) to manage each repository
- ▶ Contributions accepted using pull requests
 - ▶ New features for Android, iOS, or Core should follow the new [evolution](#) process

DOCDOWN

- ▶ DocDown is an extension of Markdown to increase the readability of documentation
 - ▶ Custom section headers
 - ▶ Configurable structure for multiple pages
 - ▶ Relative linking of content
 - ▶ Local assets
 - ▶ Pages or sections

Extended Markdown Syntax

Notes

Generic Note

Fenced by !!! . Supports inline markdown. Content can start on first line or new line. "NOTE" the default.

```
!!! NOTE
Content
!!!
```

Important Note

```
!!! IMPORTANT
Content
!!!
```

HMI Must

```
!!! MUST
Content
!!!
```

HMI May

```
!!! MAY
Content
!!!
```

CUSTOM HEADER EXAMPLE

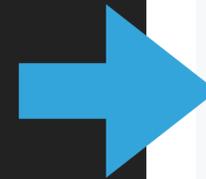
SDL sends `AlertManeuver` together with `TTS.Speak`. The purpose of this RPC is to notify the embedded navigation system about the next navigation maneuver.

!!! MAY

1. Notify the user by `TTS.Speak`, which is sent along with `AlertManeuver`.
2. Display the `AlertManeuver` dialog with an alert icon and one of the following:
 - Up to three soft buttons defined within the `softButtons` parameter.
 - HMI-defined `Close` soft button if a request without `softButtons` is received.
3. Process `SystemContext` behavior for `AlertManeuver` in the same way an `Alert` is handled.

!!!

!!! MUST



SDL sends `AlertManeuver` together with `TTS.Speak`. The purpose of this RPC is to notify the embedded navigation system about the next navigation maneuver.

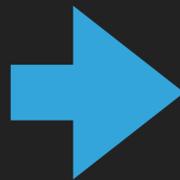
 MAY

1. Notify the user by `TTS.Speak`, which is sent along with `AlertManeuver`.
2. Display the `AlertManeuver` dialog with an alert icon and one of the following:
 - Up to three soft buttons defined within the `softButtons` parameter.
 - HMI-defined `Close` soft button if a request without `softButtons` is received.
3. Process `SystemContext` behavior for `AlertManeuver` in the same way an `Alert` is handled.

 MUST

BUILD SERVER

GITHUB COMMIT



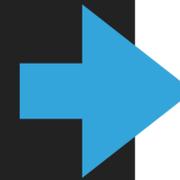
BUILD SERVER



DEVELOPER PORTAL

STATIC HTML

ELASTIC SEARCH



Documentation

HMI iOS Android SHAID SDL Ser

Where propertyName is the name of the notification, such as `Buttons.OnButtonSubscription`

MUST
The HMI must register its components, send the OnReady notification, respond to each of the IsReady RPCs, and register for the notifications it would like to receive.

INFO
If the response to any of the component `IsReady` requests

DOCDOWN PULL REQUEST

Open jacobkeeler wants to merge 1 commit into smartdevicelink:develop from jacobkeeler:feature/add_PLAY_PAUSE

Conversation 1 | Commits 1 | Files changed 1

jacobkeeler commented 9 days ago
smartdevicelink member

Adds the PLAY_PAUSE button into the Common.ButtonName enum definition.
These documentation changes are made to correspond with [smartdevicelink/sdl_core#766](#).

Addition of PLAY_PAUSE definition in ButtonName enum 0d85f3e

justinjdicow commented 9 days ago
smartdevicelink member

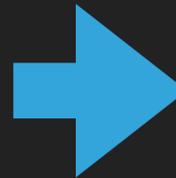
Preview this pull request at https://smartdevicelink.com/en/docs/pull_request/0d85f3eded7244c6aff1987391863a13fc10e7d4/overview/

jacobkeeler referenced this pull request in smartdevicelink/sdl_core 8 days ago
Inclusion of PLAY_PAUSE as a subscribable button #766 Open

All checks have passed Show all checks
1 successful check

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



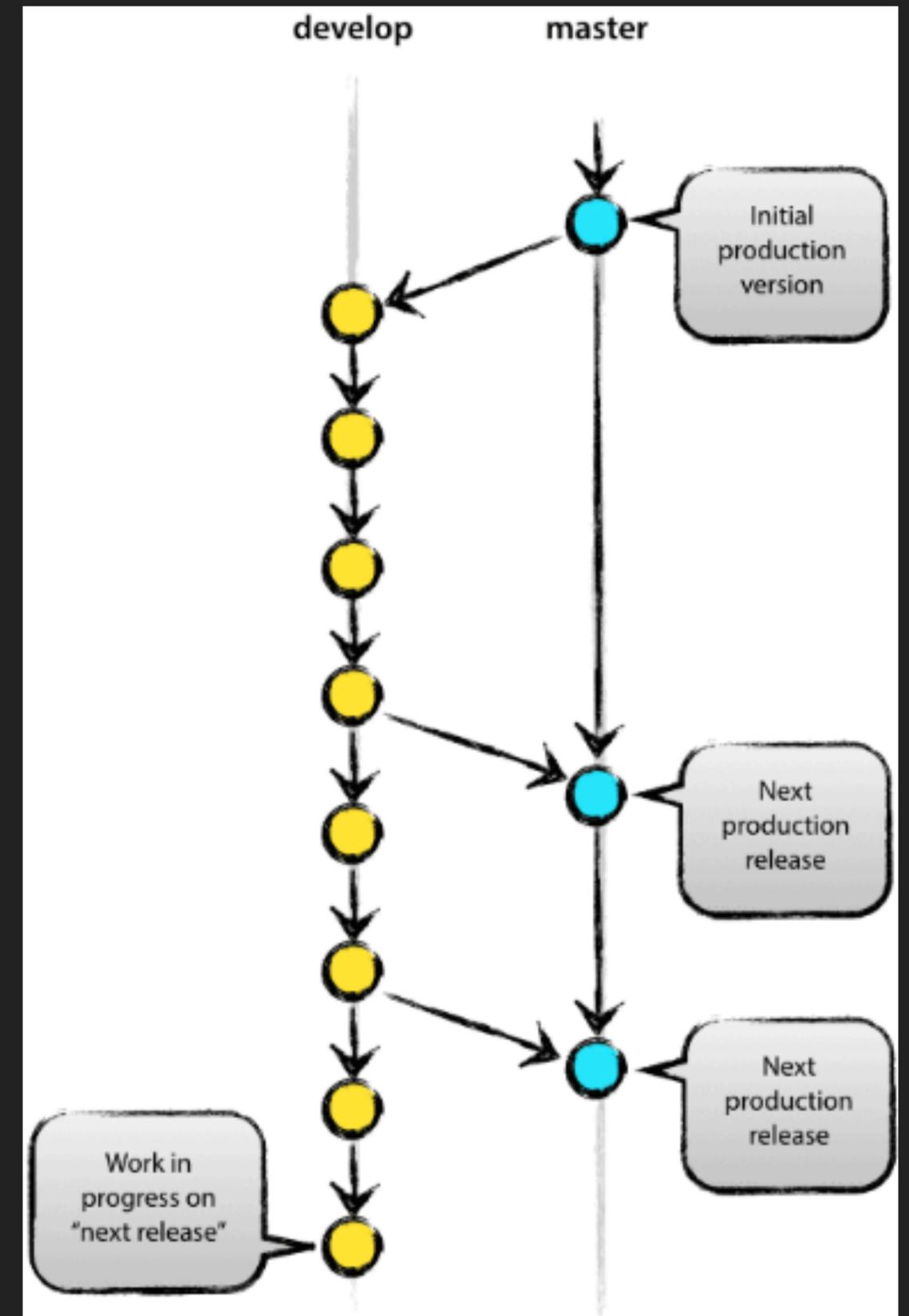
search anything | Documentation | HMI | iOS | Android | SHAID | SDL Serve

ButtonName

NAME	VALUE	DESCRIPTION
OK	0	
PLAY_PAUSE	1	
SEEKLEFT	2	
SEEKRIGHT	3	
TUNEUP	4	
TUNEDOWN	5	
PRESET_0	6	
PRESET_1	7	
PRESET_2	8	
PRESET_3	9	

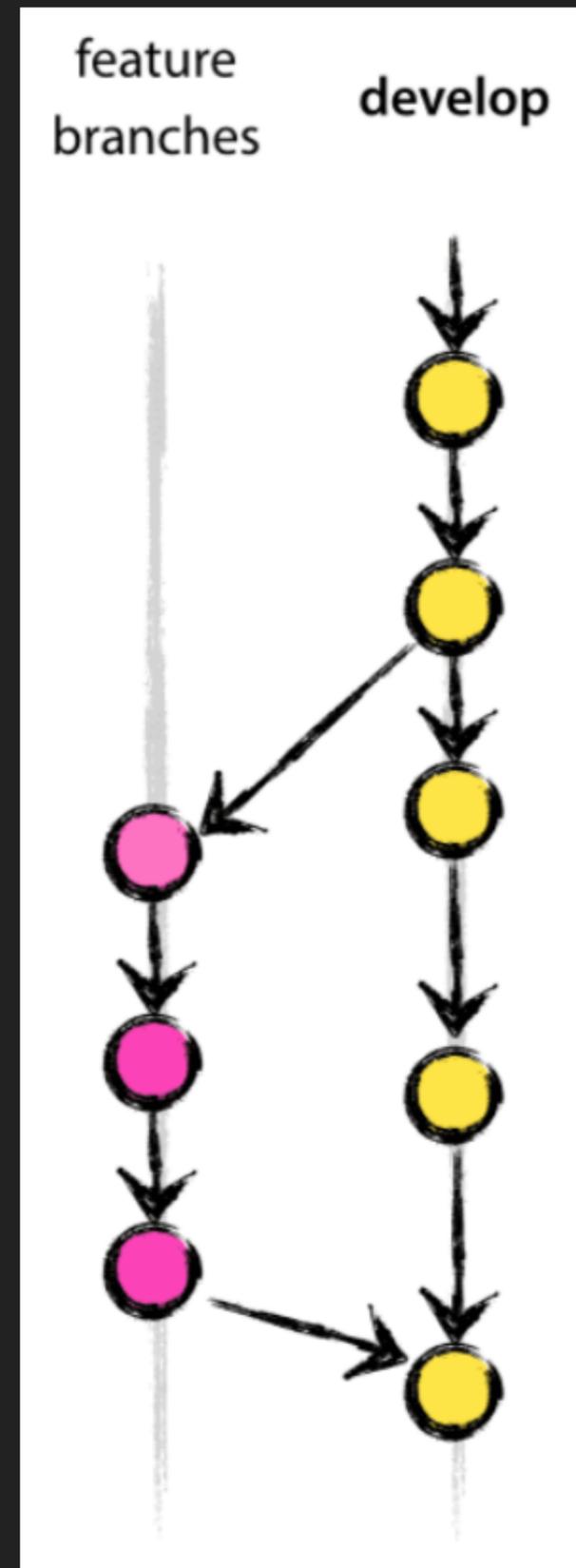
GITFLOW

- ▶ A popular development model for managing a project using git
 - ▶ Takes advantage of branching and merging
- ▶ Two main branches in a repository always exist:
 - ▶ **Master** - Production ready releases
 - ▶ **Develop** - Latest features that are complete and unreleased
- ▶ Other temporary branches may exist:
 - ▶ Feature, Release, and Hotfix



GITFLOW

- ▶ Feature Branch
 - ▶ Used to develop new features for future release
 - ▶ Branches from and to **develop**
 - ▶ Should only exist while feature is in development

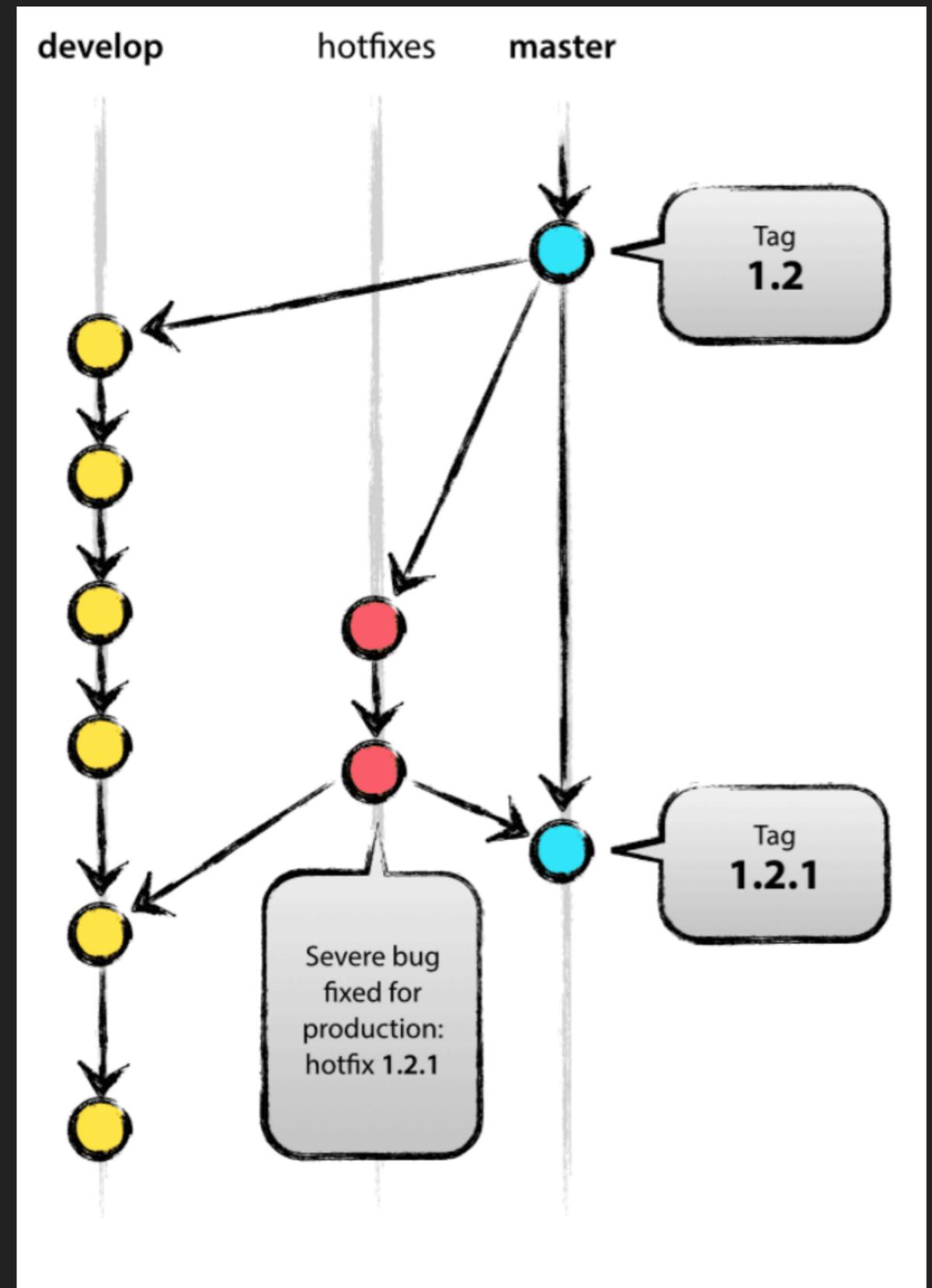


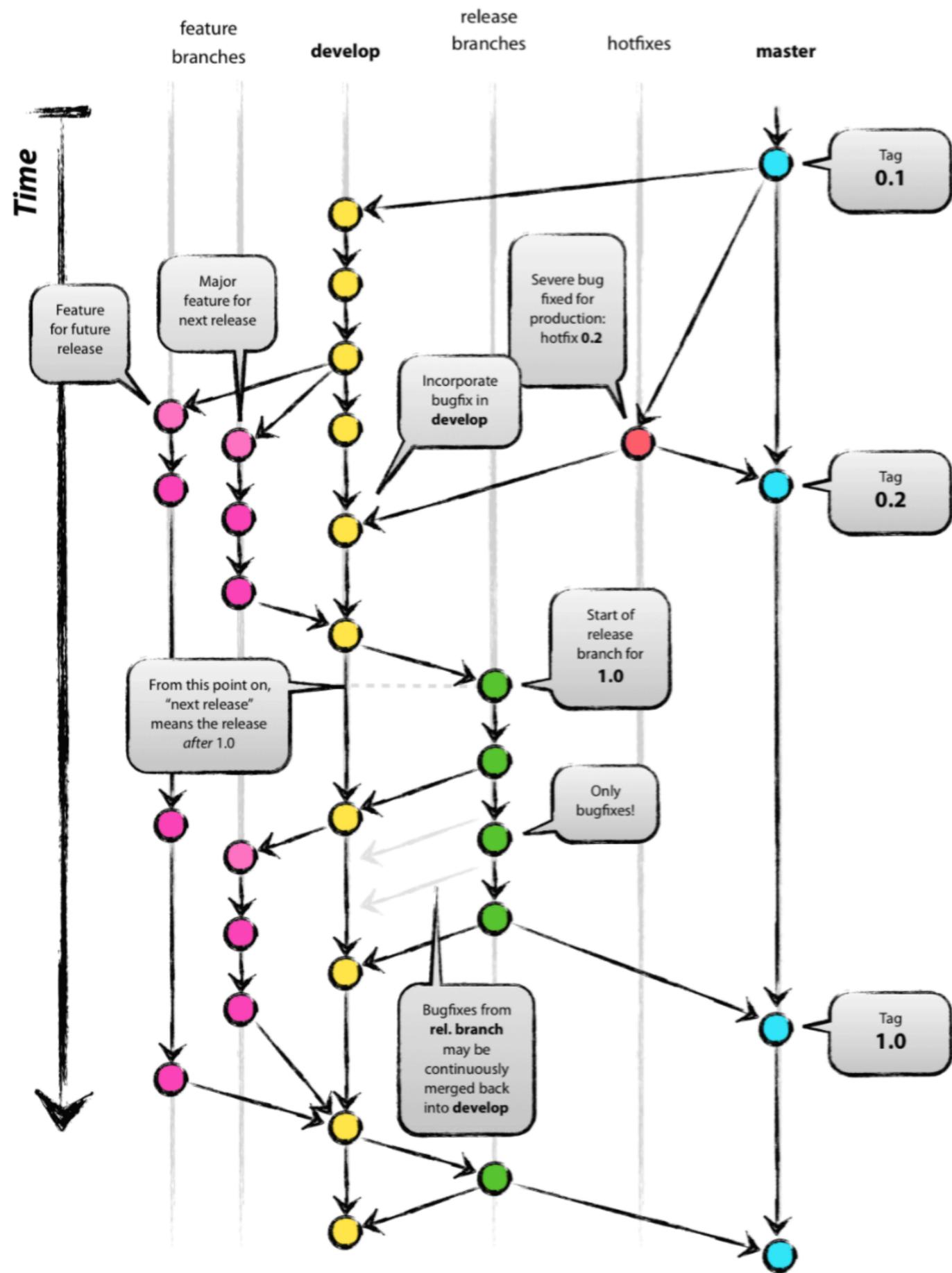
GITFLOW

- ▶ Release Branch
 - ▶ Created to prepare a new production release
 - ▶ Allows last minute dotting of i's and crossing t's
 - ▶ Branches from **develop**
 - ▶ Merges into **master** and **develop**

GITFLOW

- ▶ Hotfix Branch
 - ▶ Typically created to address an issue in production release
 - ▶ Branches from **master**
 - ▶ Merged into **master** and **develop**





HOW TO CONTRIBUTE

Using github, gitflow, and the evolution process.

STEP 0

- ▶ Sign the [Contributor's License Agreement](#) which basically says (IANAL):
 - ▶ We have a right to use your contributed code, even if you patent it
 - ▶ You can't revoke anyones right to use your contributed code
 - ▶ We can't sue you if your contributed code is bad

STEP 1

- ▶ If contributing a new feature for Android, iOS, or Core you must follow the **evolution** process. Otherwise skip this step
 - ▶ Submit your proposal to the [sdl_evolution](#) repository in a pull request using the provided [proposal template](#)
 - ▶ The proposal will be scheduled for review where the SDL community can discuss the feature
 - ▶ The SDL committee will review the feedback and either approve or deny the new feature

STEP 2

- ▶ Go to the project's github repository and search to see if the issue already exists
- ▶ If it doesn't create one in the issue tracker

The screenshot shows the GitHub interface for the repository 'smartdevicelink / sdl_android'. The 'Issues' tab is selected, showing 50 issues. The specific issue is 'Create gradle build setup #107', which is open and was created by 'mikeburke106' on Feb 10, 2015. The issue has no comments. The activity log shows that 'mikeburke106' added the 'enhancement' label, 'justinjdicow' added it to the 'Future' milestone, and 'joeygrover' referenced it in issue #305. The bottom of the page shows the 'Write' comment editor with a rich text toolbar and a 'Leave a comment' placeholder.

STEP 3

- ▶ Create a [feature](#) or [hotfix](#) branch in the desired repository
 - ▶ Features should branch from the develop branch
 - ▶ Named `issue_<Issue #>_<Feature Name>`
 - ▶ `git checkout -b feature/issue_107_androidStudio develop`
 - ▶ Hotfixes should branch from the master branch
 - ▶ Named `issues_<Issue #>`
 - ▶ `git checkout -b hotfix/issue_107 master`

STEP 4

- ▶ Create a pull request on github with the tag **[WIP]**. Include a description of the changes and link any related content
 - ▶ Issue(s)
 - ▶ Documentation
 - ▶ Links to other websites
 - ▶ Code snippets

The screenshot shows a GitHub pull request page for the repository `smartdevicelink / sdl_android`. The pull request title is "[WIP] Allow developers to use both Eclipse and Android Studio #305". It was created by `joeygrover` and is targeting the `develop` branch from the `feature/android_studio` branch. The pull request includes 4 commits and 6 files changed. A comment from `joeygrover` on June 21 describes the changes: "Includes a basic gradle build file that allows users to import the project into Android Studio and lets Eclipse users continue to use Eclipse. File structure will not change. Fixes #107". Below the comment, a list of commits is shown, including "Update gradle files for use with Android Studio", "Added bintray options to gradle file", "Added maven deploy", and "Merge branch 'develop' of https://github.com/smartdevicelink/sdl_android".

This repository Search Pull requests Issues Gist

smartdevicelink / sdl_android Watch

<> Code Issues 50 Pull requests 21 Wiki Pulse Graphs Settings

[WIP] Allow developers to use both Eclipse and Android Studio #305

Open joeygrover wants to merge 4 commits into develop from feature/android_studio

Conversation 0 Commits 4 Files changed 6

joeygrover commented on Jun 21 smartdevicelink member +

Includes a basic gradle build file that allows users to import the project into Android Studio and lets Eclipse users continue to use Eclipse.

File structure will not change.

Fixes #107

joeygrover added some commits on Jun 16

- Update gradle files for use with Android Studio ✓ c4
- Added bintray options to gradle file ✓ 93
- Added maven deploy ✓ d2
- Merge branch 'develop' of https://github.com/smartdevicelink/sdl_android ✓ 36

Add more commits by pushing to the **feature/android_studio** branch on **smartdevicelink/sdl_android**.

STEP 5

- ▶ Implement your contribution remembering to:
 - ▶ Commit in logical units
 - ▶ Add and update unit tests

The screenshot displays a GitHub pull request interface. At the top, a comment from user 'joeygrover' (smartdevicelink member) dated Jun 21 states: 'Includes a basic gradle build file that allows users to import the project into Android Studio and let Eclipse users continue to use Eclipse. File structure will not change. Fixes #107'. Below the comment is a commit history section for 'joeygrover' dated Jun 16, listing four commits: 'Update gradle files for use with Android Studio' (c44), 'Added bintray options to gradle file' (93d), 'Added maven deploy' (d24), and 'Merge branch 'develop' of https://github.com/smartdevicelink/sdl_android' (36a). A message below the commits reads: 'Add more commits by pushing to the **feature/android_studio** branch on **smartdevicelink/sdl_android**.' The status checks section shows two successful checks: 'All checks have passed' (2 successful checks) and 'This branch has no conflicts with the base branch' (Merging can be performed automatically.). A green 'Merge pull request' button is visible, along with a note: 'You can also [open this in GitHub Desktop](#) or view [command line instructions](#).' At the bottom, there is a 'Write' and 'Preview' tab, a rich text editor toolbar, and a 'Leave a comment' input field.

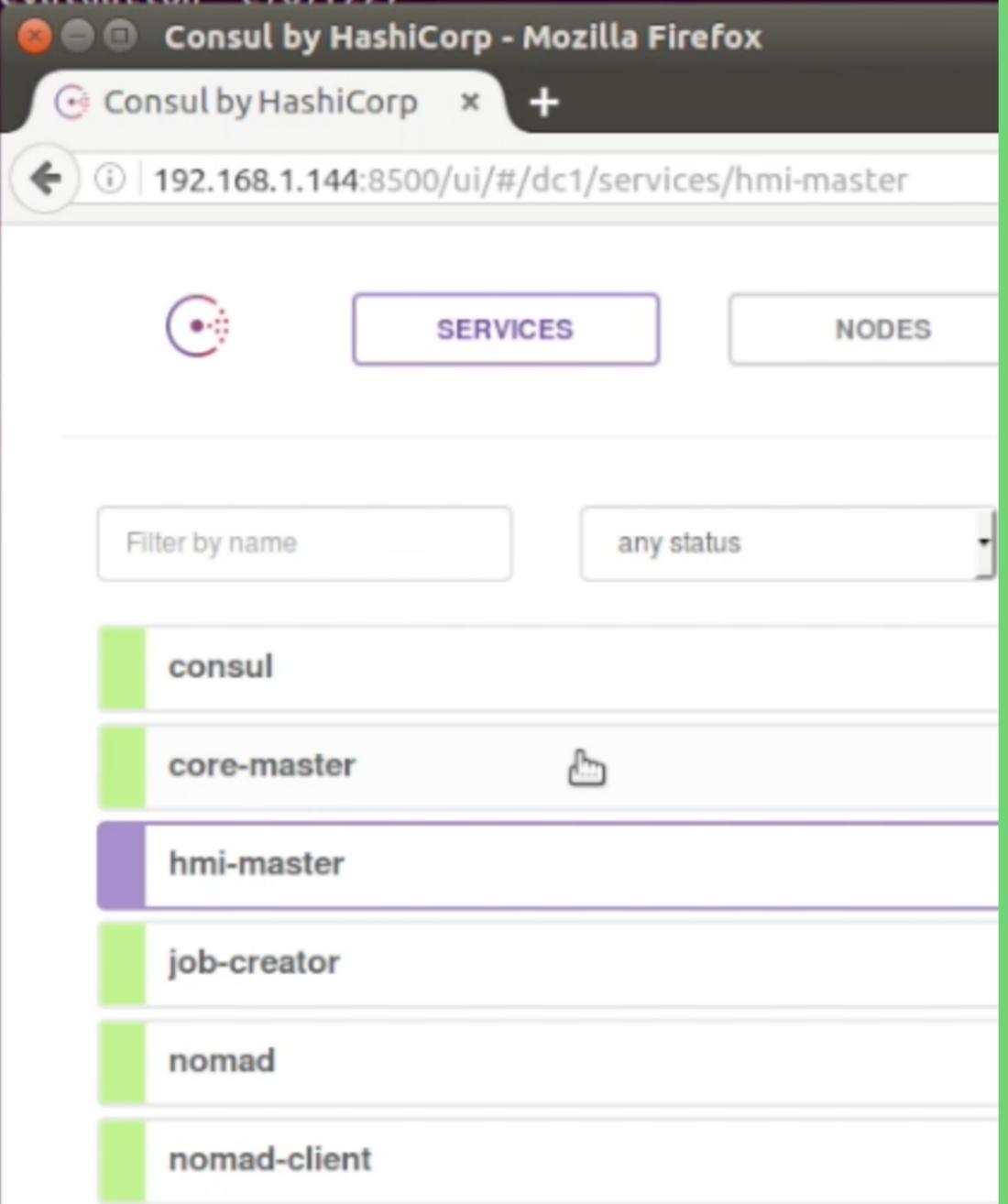
STEP 6

- ▶ When you are finished with implementation:
 - ▶ Ensure all unit tests pass the continuous integration
 - ▶ Remove the [WIP] tag
 - ▶ Add a comment mentioning one of the SDL teams
 - ▶ e.g. [@smartdevicelink/android](#) my feature is ready for review

STEP 7

- ▶ The SDL team will review and work with you to get the hotfix or feature merged
 - ▶ A feature will be merged into the **develop** branch for the next release
 - ▶ A hotfix will be merged into both **develop** and **master** branches

```
locker-user@chris-VirtualBox:~/Documents$ nomad status -address=
No running jobs
locker-user@chris-VirtualBox:~/Documents$ nomad run -address=http
=> Monitoring evaluation "e707f227"
Evaluation
Allocation
Evaluation
=> Evaluation
locker-user@chr
ID Ty
job-creator se
locker-user@chr
ID Ty
core se
amt se
job-creator se
locker-user@chr
```



MANTICORE

Testing apps just got easier...

MANTICORE

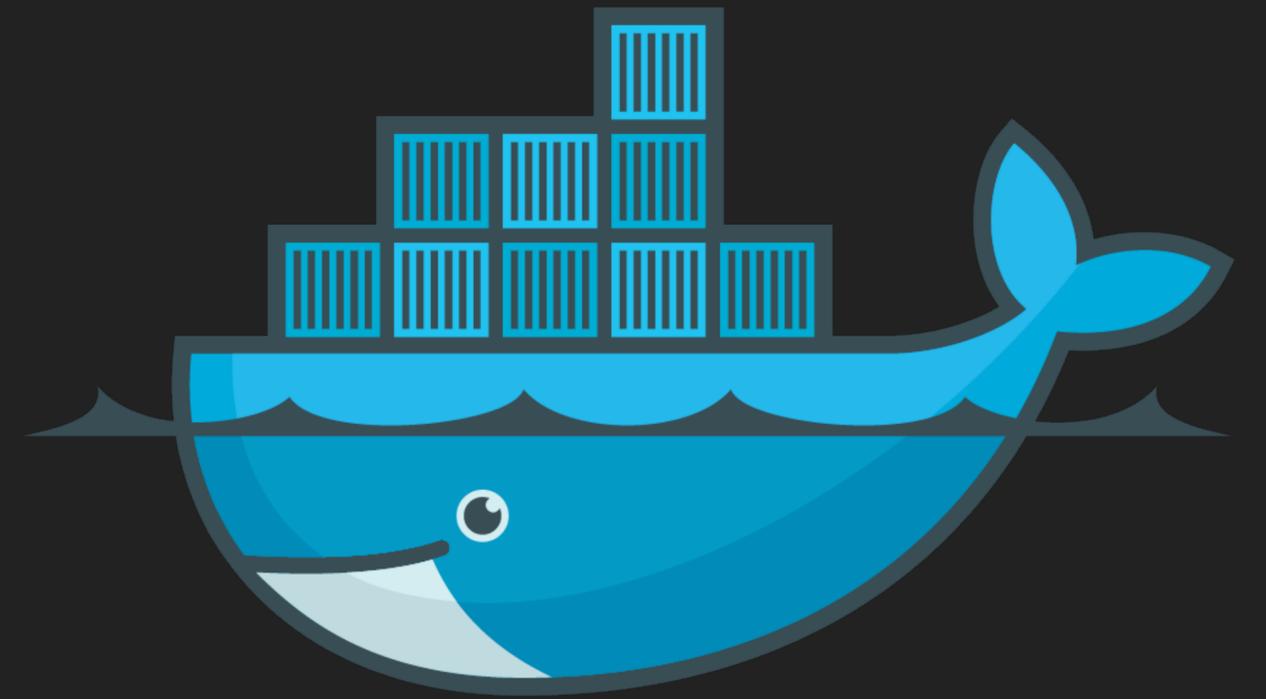
- ▶ Automagically creates an instance of SDL core and HMI in the cloud for a developer to use
 - ▶ No more virtual machines
 - ▶ No more dependency hell
 - ▶ Can work with multiple web HMIs
 - ▶ A step towards continuous integration and testing

DEMO

- ▶ Short [video demonstration](#) of Manticore and how it works
 - ▶ By Chris Rokita, a Software Engineer at Livio

DOCKER

- ▶ SDL Core & HMI docker container available on [github](#) & docker hub
- ▶ Install Docker, then download and run
 - ▶ `$ docker run -d -p 12345:12345 -p 8080:8080 -p 8087:8087 -p 3001:3001 --name core smartdevicelink/core:latest`
 - ▶ Connect app to your computer's IP address and port 12345



docker

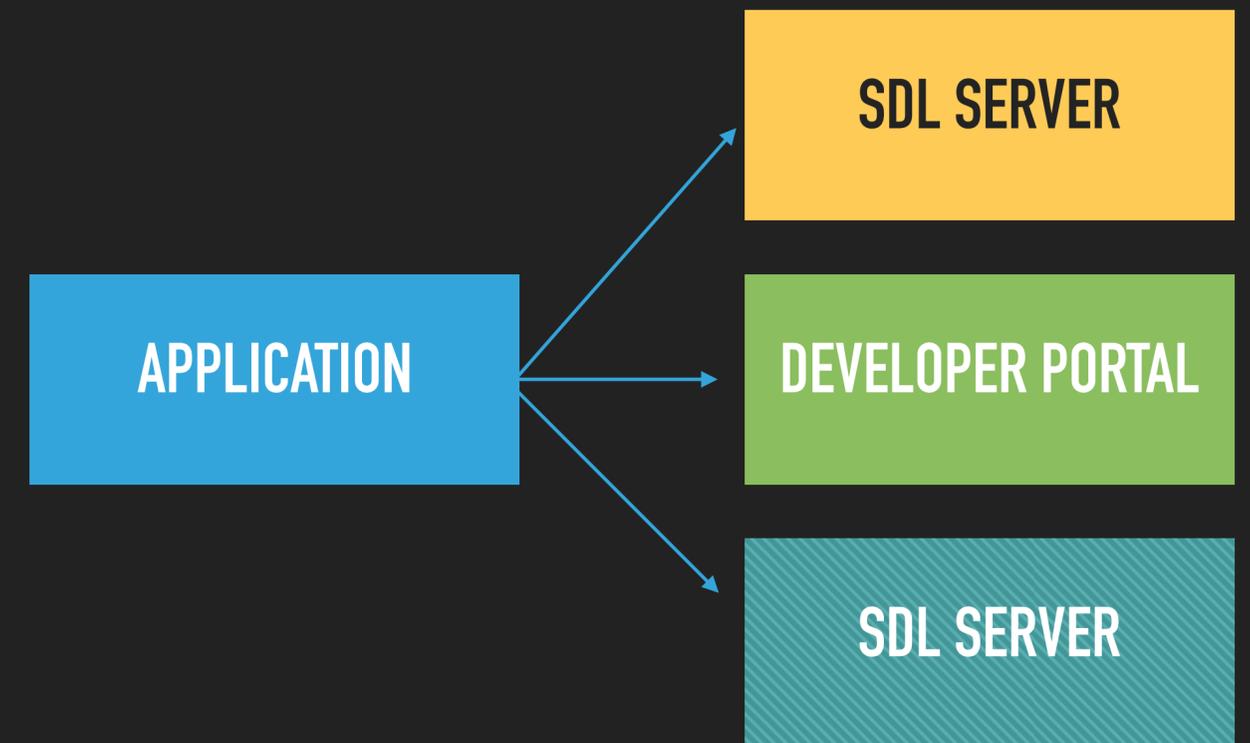


SHAID

Where did all my bits go?

PROBLEM

- ▶ Every application must be assigned a single unique application ID.
- ▶ This ID is used by SDL Core and SDL Servers to manage that application's permissions.
- ▶ However, each OEM must implement their own SDL Server.
 - ▶ How do we ensure the Application ID is unique?
 - ▶ How do we handle revoking of an ID?
 - ▶ How do we handle other application data changing, such as the package name or URL Scheme?

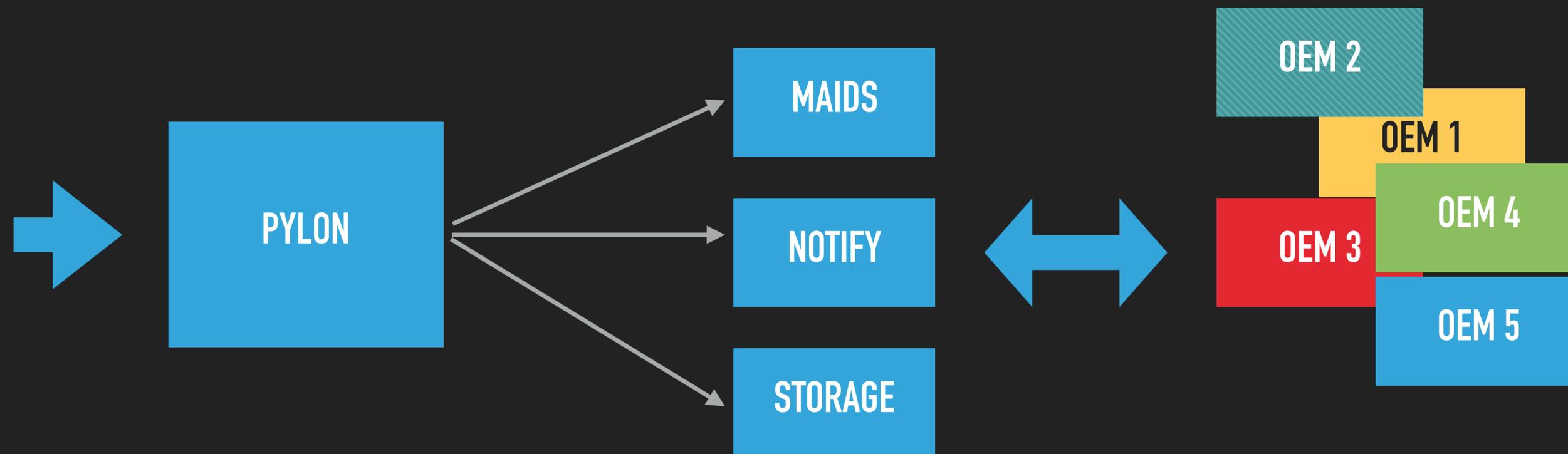


SOLUTION

- ▶ A centralized system to
 - ▶ Generate unique application IDs
 - ▶ Synchronize data
 - ▶ e.g. Application IDs, package names, permissions, url schemes
- ▶ We call this system the Super Helpful Application ID service (SHAID)

HOW DOES IT WORK

- ▶ SHAID is a series of microservices each with a very specific task
- ▶ OEM's can integrate with any of the services



APP DEVELOPERS

- ▶ What does this mean for app developers?
 - ▶ Keep your information up-to-date on smartdevicelink.com
 - ▶ Choose which companies we can share your information
 - ▶ Follow up with any notifications you receive

OEMS

- ▶ What does this mean for OEMs?
 - ▶ SHAID will generate all application IDs
 - ▶ Be able to accept these from the developer or programmatically
 - ▶ All other data synchronization is optional
 - ▶ To integrate follow the SHAID API documentation on smartdevicelink.com
 - ▶ Only application ID generation is currently available until SHAID v2

**MANTICORE
V 1**

**GENERIC HMI
V 1**

**GENERIC HMI
V 2**

**SHAID
V 1.5**

**SHAID
V 2**

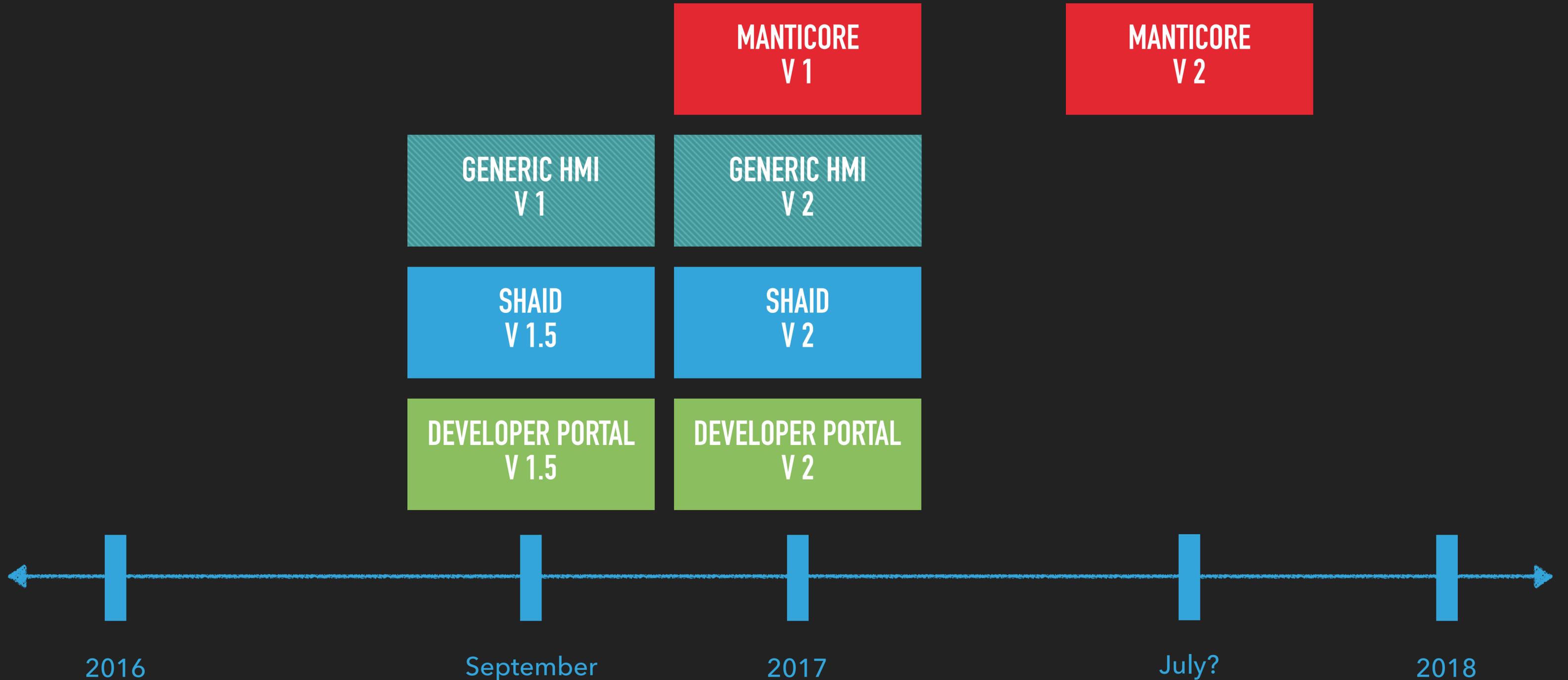
**DEVELOPER PORTAL
V 1.5**

**DEVELOPER PORTAL
V 2**

TIMELINE

So when can I use this stuff?

ESTIMATED TIMELINES





QUESTIONS