Automotive Push Service

Workshop 2

17.04.2024

COVESA

Accelerating the future of connected vehicles

Agenda

- Push notifications: a (very) brief recap
- What happened since the last workshop
  - UnifiedPush Quote
  - Forvia Reference Implementation
- Discussion on open topics
- Outlook

Push Notifications

A (Very) Brief Recap

COVESA
Accelerating the future of connected vehicles

Problem Statement

"A push notification (also known as a *server push notification*) is the delivery of information to a computing device from an application server where the request for the transaction is initiated by the server rather than by an explicit request from the client" [1]

- Common use-cases:
  - Delivery of messages
  - Incoming VoIP calls
  - Live updates in a running app (e.g., update list of upcoming meetings)

Problem

- Apps get killed by the system and cannot rely on being able to communicate with their backend
- There exists no standardized push notification service for AOSP/AAOS

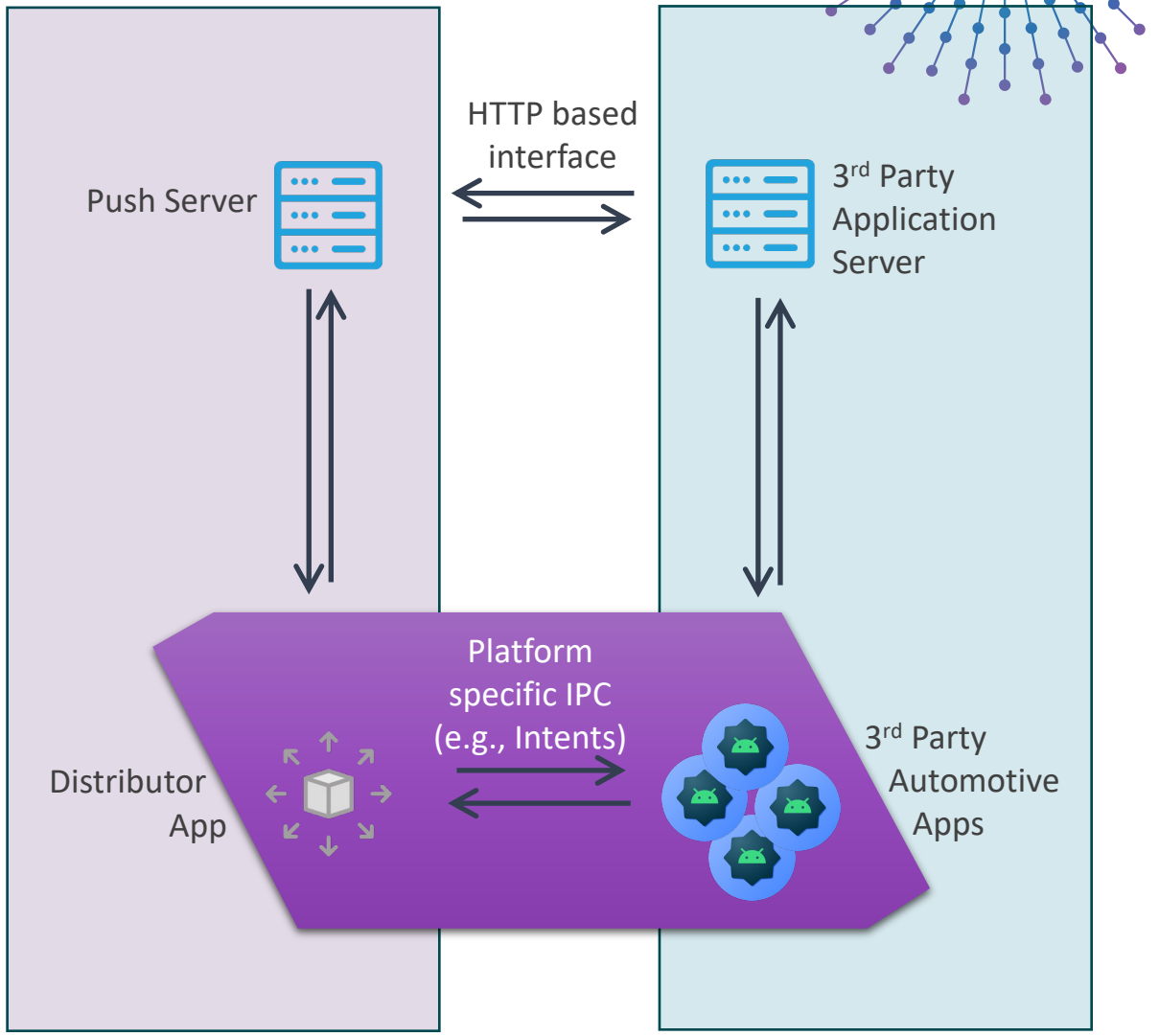
What is out there

- Apple Push Notification service – only usable on Apple devices
- Firebase Cloud Messaging – only usable with Google's Play services / Google Automotive Services
- Huawei Push Kit – only usable on Huawei devices
- Amazon Device Messaging – only usable on Amazon devices

- Pushy – a push service that supports AOSP
- Web Push (RFC8030) – THE standard for browsers
- UnifiedPush – a Web Push inspired protocol for native applications on Android and Linux

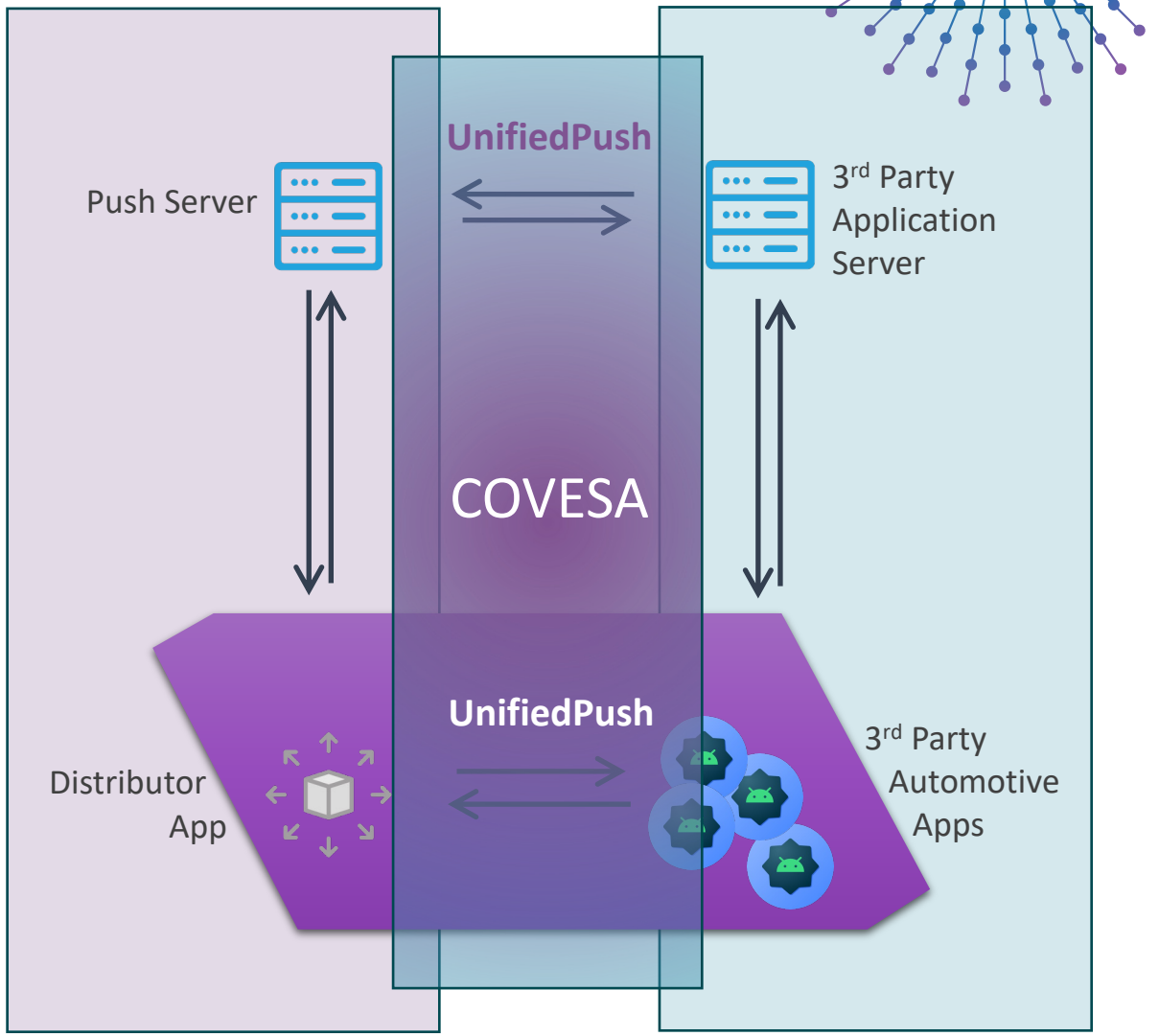
The Architecture of a Push Notification Service

- A push notification service (PNS) generally consists of four parts
- Notifications are triggered by an HTTP POST request to the respective push-server
- The exact type of the connection between the distributor and the push server varies
- The mechanism used by the distributor to talk to the application depends on the platform

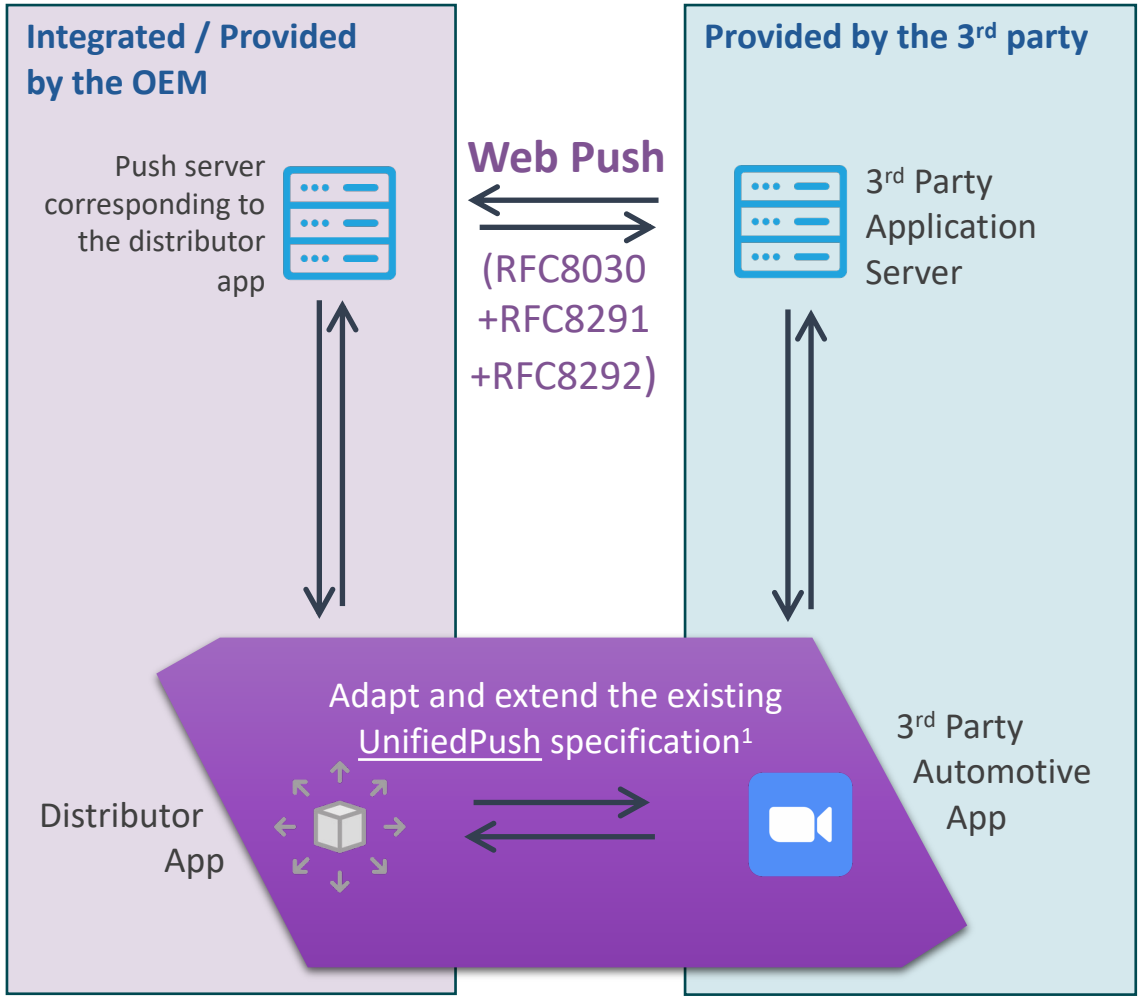


UnifiedPush

- From previous analysis performed independently by different parties – Faurecia Aptoide, BMW, FORVIA - **UnifiedPush** seemed the most promising solution:
  - Provides a specification
  - Apache 2.0 licensed, same as AOSP
- Lacks some important features:
  - **No support** for message receipts
  - **No support** for time-to-live
  - **No support** for urgency
  - **No support** for collapsible messages
  - **No support** for VAPID



Proposal



[1] Changes to the UnifiedPush specification are necessary to support all features from RFC8030 and RFC8292.

Disclaimer

- Focus will be on a one-way communication channel from 3rd party application servers to a specific app instance
- Focus is on Android Automotive Apps working independently of a customers brought in device (not phone projection)
- Its not about the notifications, its about pushing a message to the car

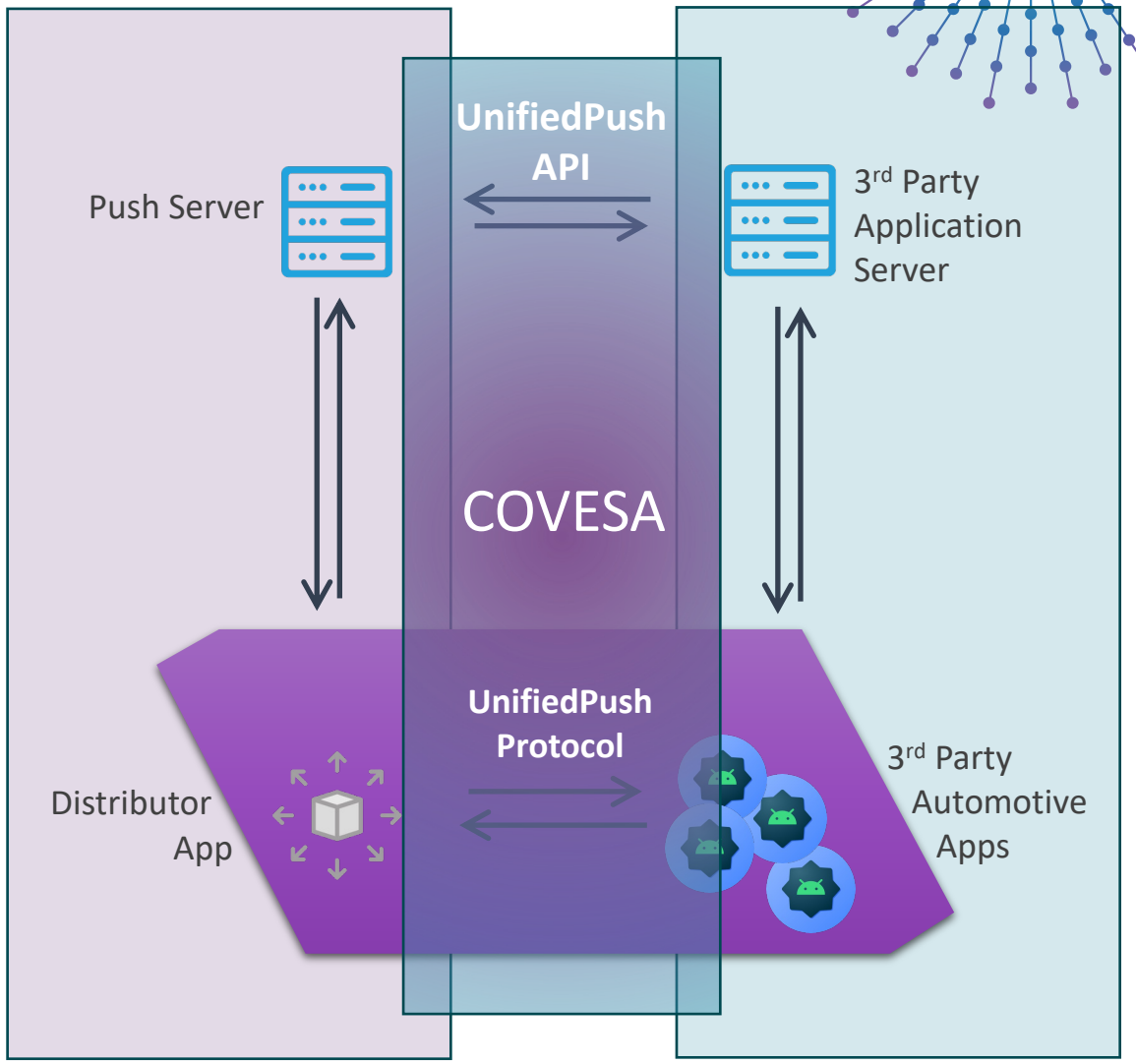
What happened since the last Workshop

Progress looks good!

COVESA
Accelerating the future of connected vehicles

What Happened Since Then

- Discussions with UnifiedPush to add the necessary features to their spec
- **Quote from UnifiedPush Maintainer to extend UnifiedPush**
- **Forvia push service implementation**



Quote to Extend UnifiedPush

Quote: Web Push support extension

Specifications

Specifications (Server and Mobile) need to be improved to extend Web Push support. This include handling:

- VAPID
- Urgency
- Time-To-Live (TTL) > 0, which requires supporting:
    - message deletion
    - message update
    - message ack

This can be done by explicitly adding all features to be compatible with RFC8030, or by defining the push server as a web push server (RFC8030). The 2nd option will introduce backward incompatible changes (requirement on headers, no discoverability request, etc), backward incompatibility friction must be estimated. Definitions may need to be updated too. Even with the 2nd option, a section must be here to help to implement a server.

Android specifications must be updated, this should not be so long.

This task is not blocking the other ones: some (small) modifications can be added before it is merged while being in development.

Time estimated: 2-3 weeks

Potential related task: open an RFC

Mandatory

COVESA
Accelerating the future of connected vehicles

Quote: Web Push support extension

Libraries update

Following the new specifications, libraries and the example app must be updated.

Time estimated: 2-3 weeks

Mandatory

Server

2 options are possible

- Writing a new service (rust agent+redis)
- Use/extend a web push server (probably require to write a gateway, or suggest pull request, I have already been in contact with some web push dev)

Potential additional tasks: packaging, documentation, CI.

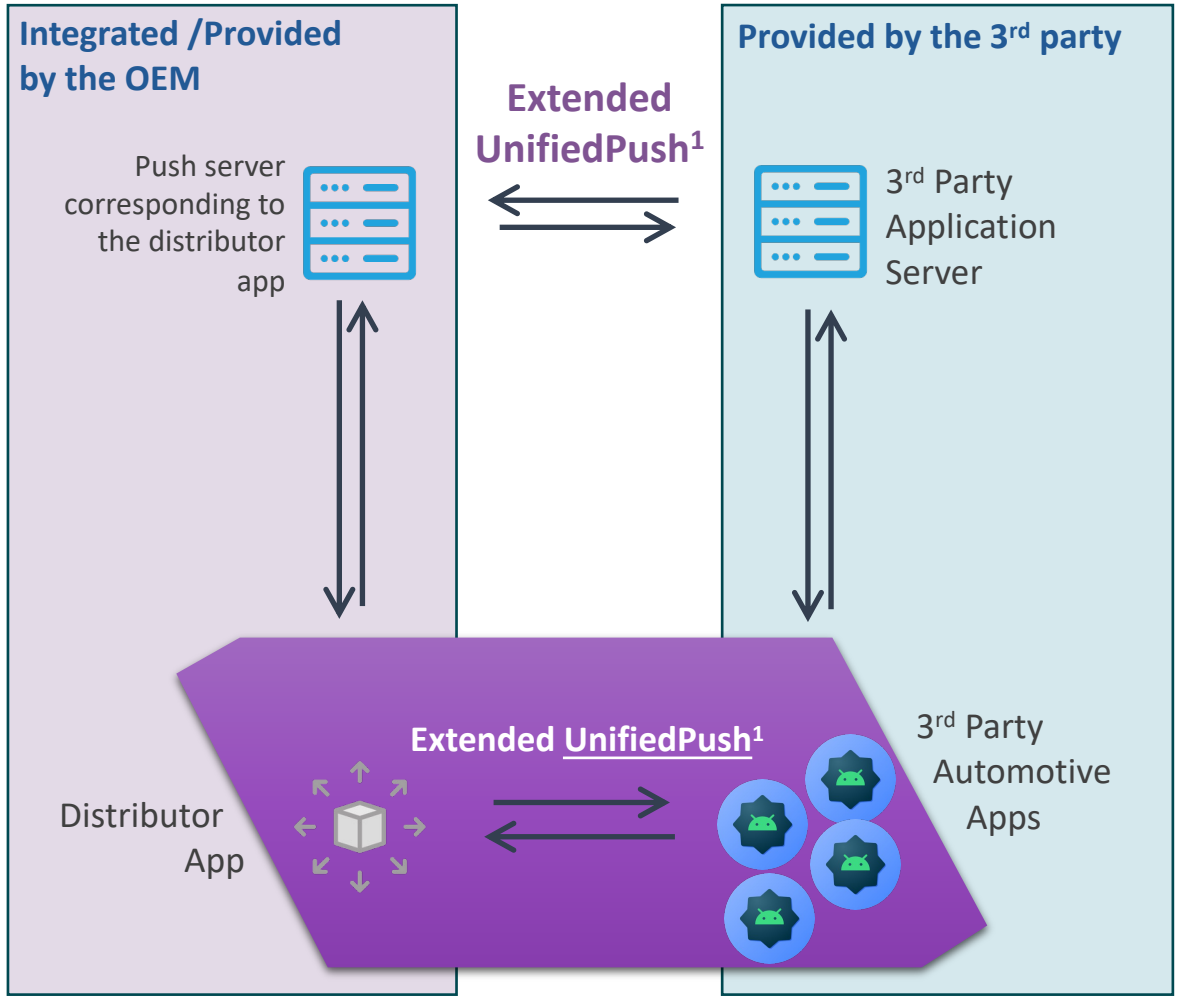
Both need to write the Android application (probably based on NextPush)

Time estimated: 10weeks

Optional

COVESA
Accelerating the future of connected vehicles

Overview: When Extending UnifiedPush



- Taken care of:
  - Libraries and the Example App
  - Specification for push server and distributor
  - Reference implementation
- **TODO OEMs**:
  - Provide a distributor app in the car / Host the reference implementation
- **TODO 3rd parties:**
  - Implement UnifiedPush support in their app and their application backend

[1] See quote from UnifiedPush maintainer S1m

Forvia Push Service

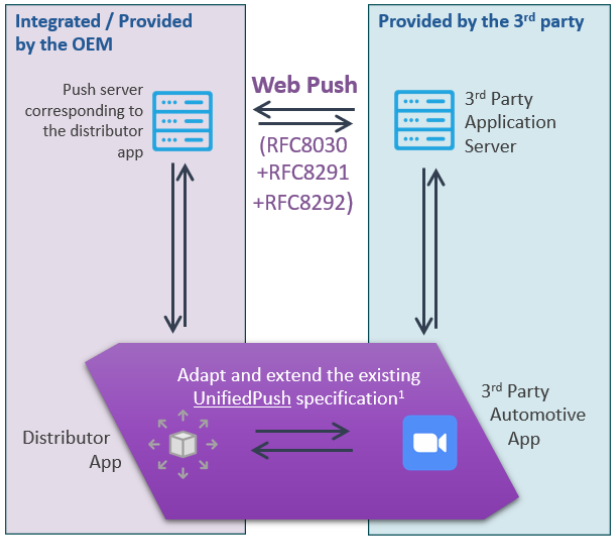
A complete implementation of the Covesa proposal

COVESA
Accelerating the future of connected vehicles

Forvia Push Service Demo

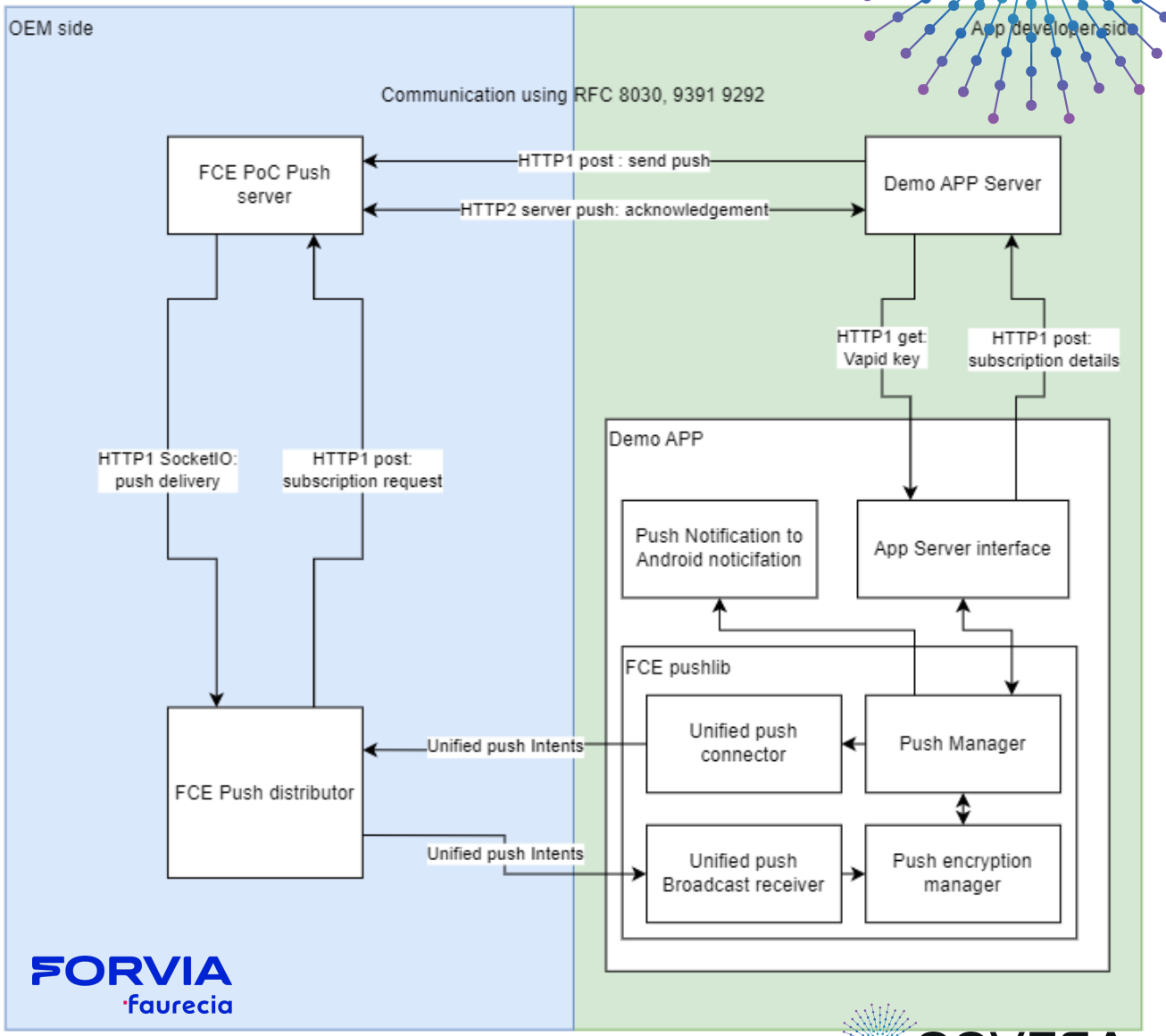
Architecture

- Covesa Proposal:



[1] Changes to the UnifiedPush specification are necessary to support all features from RFC8030 and RFC8292.

- Forvia implementation:



COVESA
Accelerating the future of connected vehicles

Discussion

Reference Implementation

UnifiedPush

- Production grade?
- Open-source
- Base on MQTT?

Forvia

- Currently a POC
- Might be made open-source

Testing Suite

- Necessary even if we have a good reference implementation?

COVESA
Accelerating the future of connected vehicles

The Details

Onboard

- Re-registration behavior / Subscription renewal
  - App data deleted → Stale subscriptions
  - Registration process fails due to connectivity issues → Possibly stale subscription
- Acknowledgement by distributor vs app
- Link push service usage to a (runtime) permission
- Battery restrictions

Backend

- Behavior for too many pending messages
- Minimum rate limits

Outlook

Outlook

- Please gather feedback until the May 6th meeting

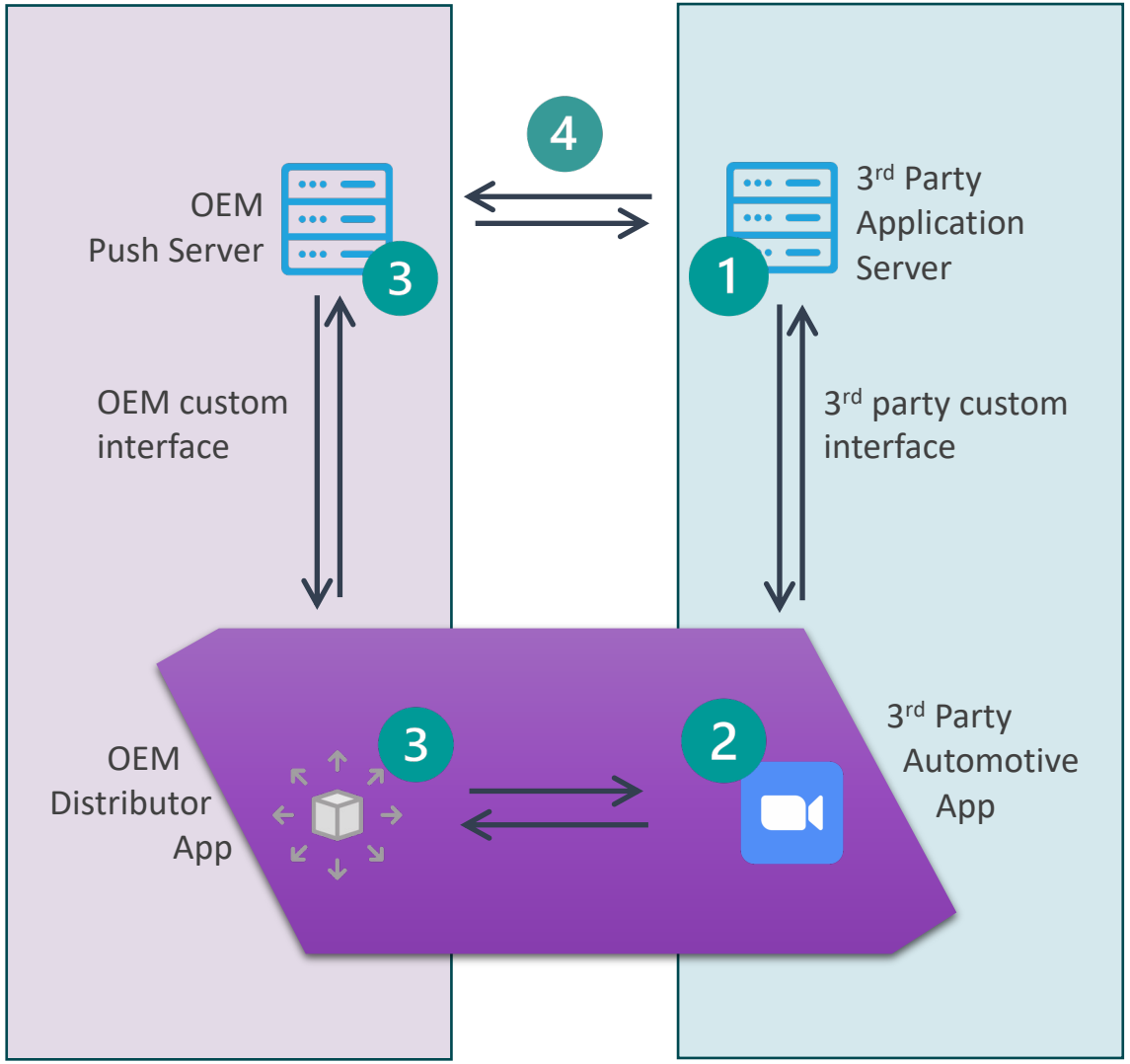
COVESA

Accelerating the future of connected vehicles

Thank you!

Backup

VAPID: Voluntary Application Server Identification



VAPID answers the following questions:

- Who sends the notification?
- Are they allowed to send notifications to the car?

How VAPID works:

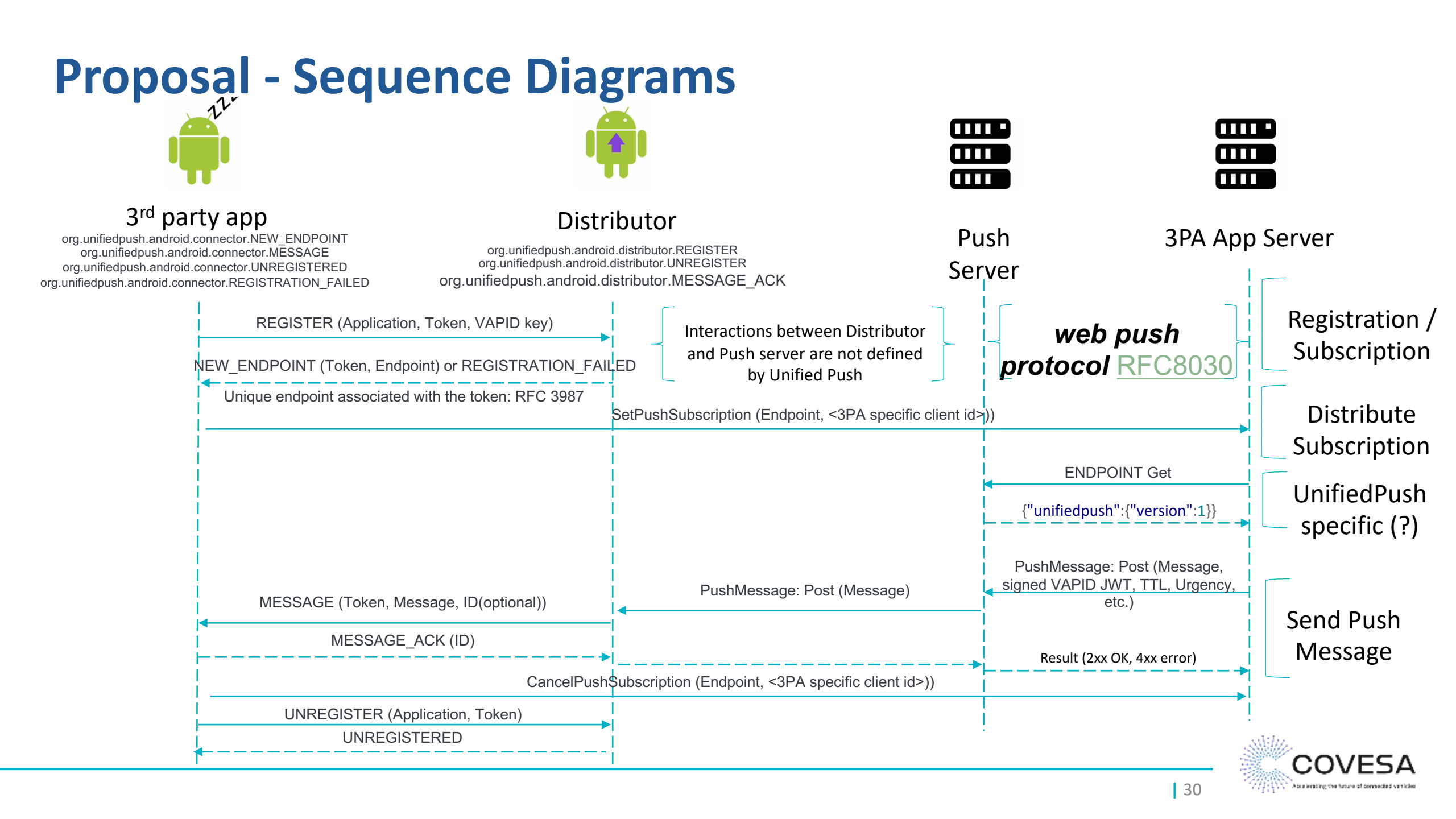
1. Create a Public/Private key pair
2. Include the public key in the app and include it when subscribing to push notifications
3. Create a new URL endpoint that only accepts requests signed with the private key associated with the provided public key
4. Include a signed JWT containing some contact information in every push notification request

COVESA
Accelerating the future of connected vehicles

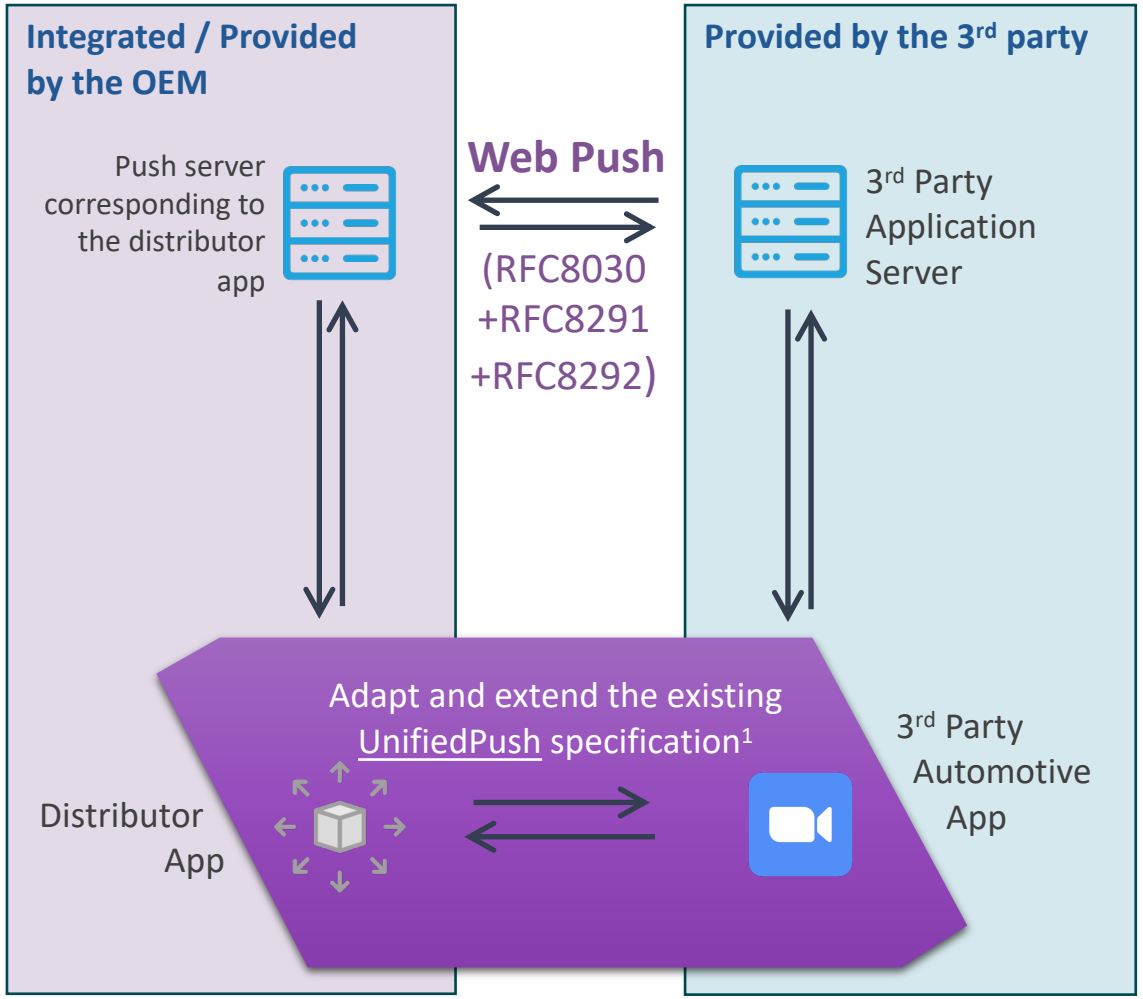
Status update: Push Notifications

- Push Notification POC on basis of Unified Push

Proposal - Sequence Diagrams



Proposal – 3rd Parties



- 3rd Parties:
  - Client side: Integrate their app with the provided connector library (slightly changed from the existing UnifiedPush one)
  - Backend side: Integrate Web Push functionality if not already existing

[1] Changes to the UnifiedPush specification are necessary to support all features from RFC8030 and RFC8292.