Automotive Push Notifications

BMW
20th November 2023

COVESA
Accelerating the future of connected vehicles

Update: Automotive Push Notifications

Current Status

- Large consensus that a specification based approach like UnifiedPush is the way to go
- Feedback so far suggests that OEMs prefer to implement their own solution
- POC1 with UnifiedPush successfully completed

Upcoming

- POC2 with mavi.io

TODO

- We need to define the new Automotive Standard
  - Inspiration:
    - RFC8030
    - RFC8291
    - RFC8292
    - UnifiedPush

My Experience

Stakeholders	Questions & Answers
Onboard The ‘Distributor’ app should be a system service and must be able to send/receive Android Intents	• Onboard should not be a big issue depending on whether a bi-directional communication channel between the car and the OEM backend already exists • Implements an Android Intent based API **Ask them:** • What infrastructure does already exist that connects the car to your backend? • Our approach is most likely compatible with that!
Backend	
Security	

My Experience

Stakeholders	Questions & Answers

Onboard	• The backend implementation is probably the most work
	• The backend team needs to develop a new backend service that supports all features COVESA agrees upon
	
	You will need to explain to them what you need:
	• They will need a thorough explanation of UnifiedPush and/or WebPush
	• Especially how URL endpoints are mapped to cars and apps
	• If they want to know how such a backend might look like:
	  • Refer them to WebPush/UnifiedPush. For both open source backends exist
	  • Also mention RFC8030

Backend There must be a publicly accessible backend server	
Security	

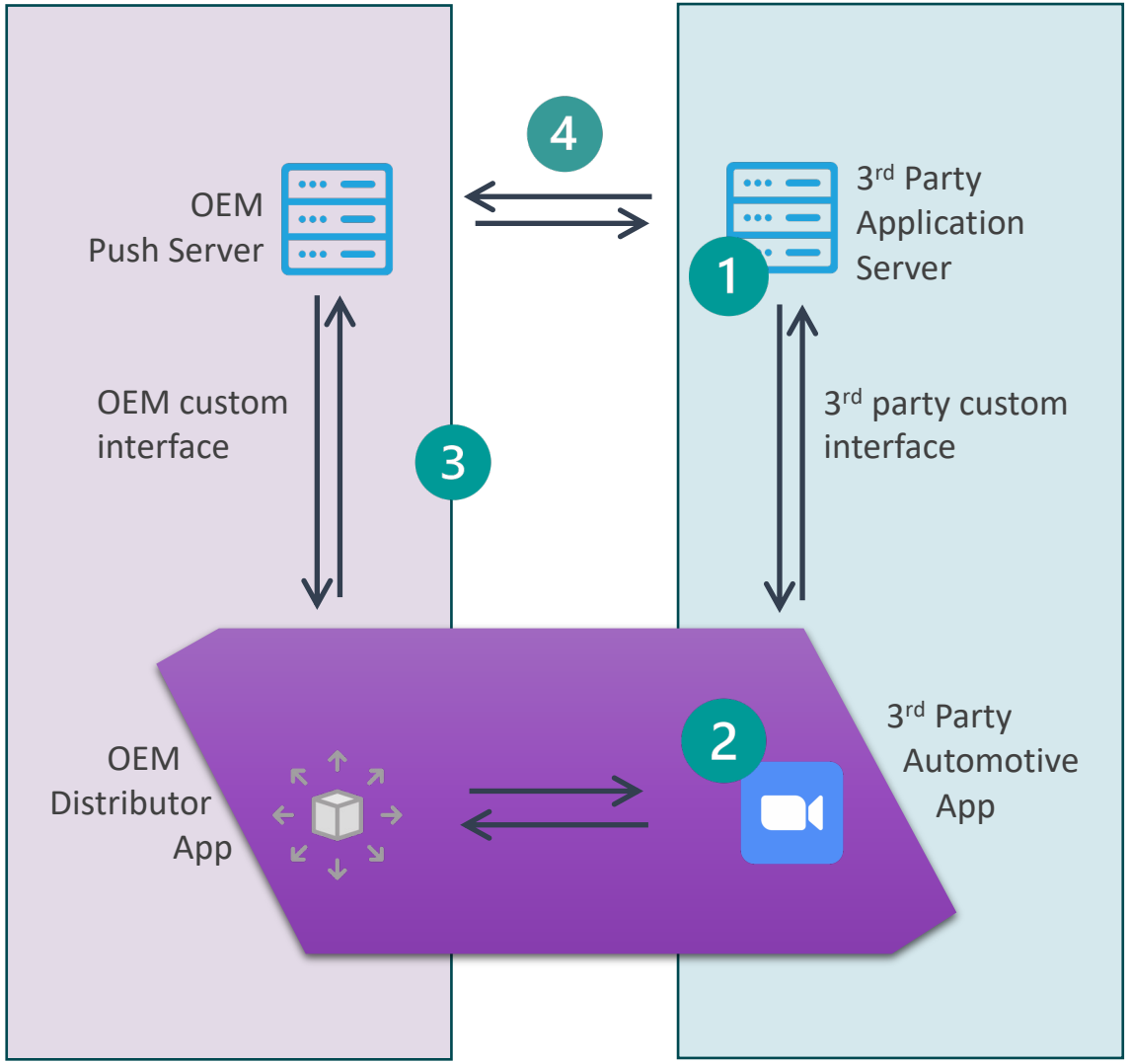
My Experience

Stakeholders	Questions & Answers

Onboard	Because the backend is (directly) connected to a car, security is very relevant!
	
	They will ask you about:
	• Application Server Authentication:
	  • This is what RFC8292 does (VAPID)
	• Message encryption
	  • This is what RFC8291 specifies

Backend	
Security The public push server will be connected to the car and send unknown payloads to it	

VAPID: Voluntary Application Server Identification



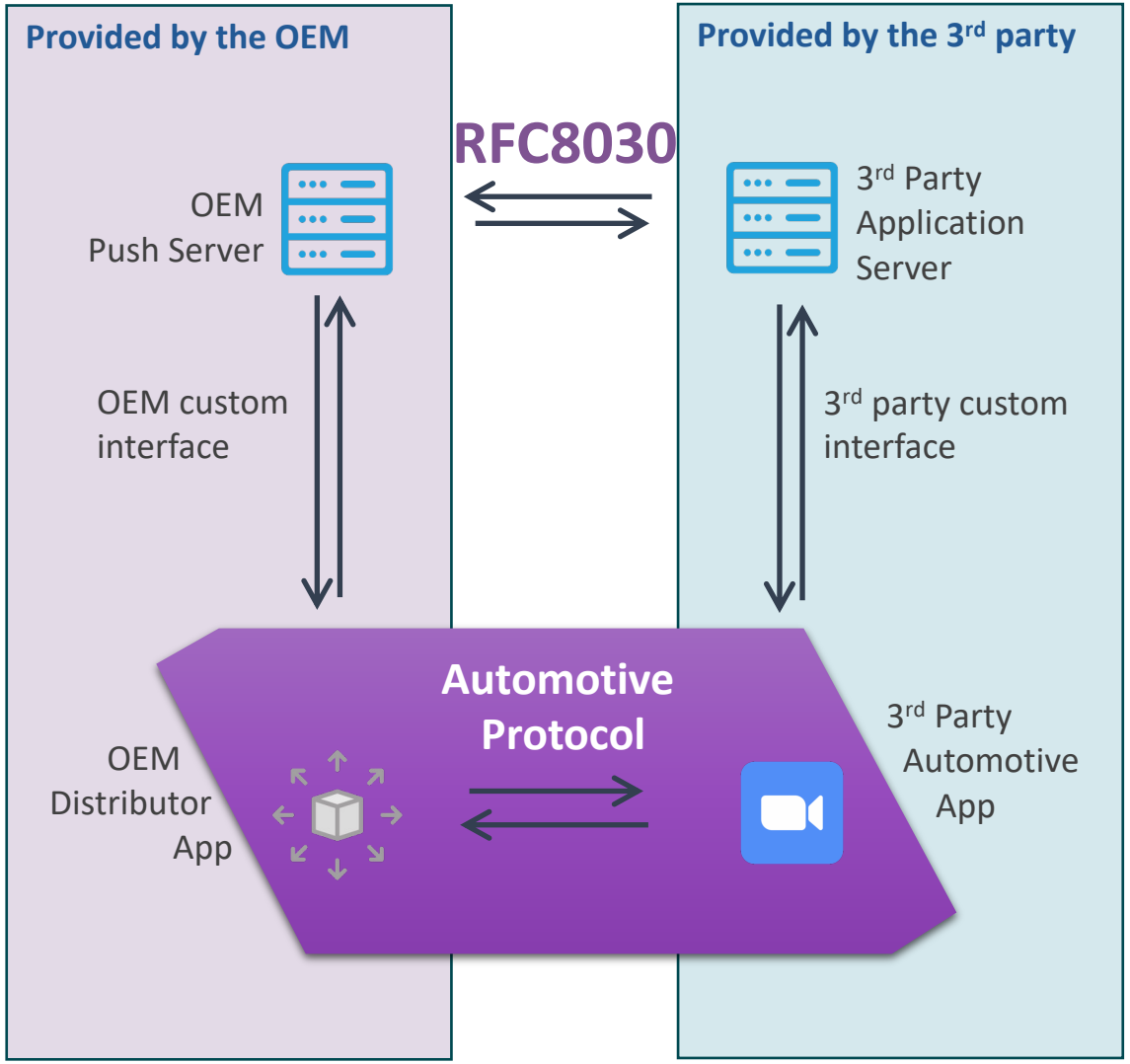
Who sends the notification?

Are they allowed to send notifications to the car?

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

Proposal for Automotive Push Notifications



- **Application server:** This is the server that hosts the application – the 3rd party backend
- **Automotive application:** The (3rd party) application receiving the push notifications. The application gets started by the distributor on incoming push notifications if it is not running
- **Push server:** This is the server that listens for incoming push messages and forwards them to the connected Push Distributor running in the car
- **Distributor:** This is the application that forwards push messages to the registered end user application. It is the application which is connected to the Push Server and must be running at all times

- **Use RFC8030 in the backend**

→ **Provides full compatibility with existing services!**

→ **Make RFC8292 (VAPID) mandatory!**

- **Adapt and extend the existing UnifiedPush protocol on the client side to support all features from RFC8030**

COVESA
Accelerating the future of connected vehicles

COVESA

Accelerating the future of connected vehicles

Thank you!

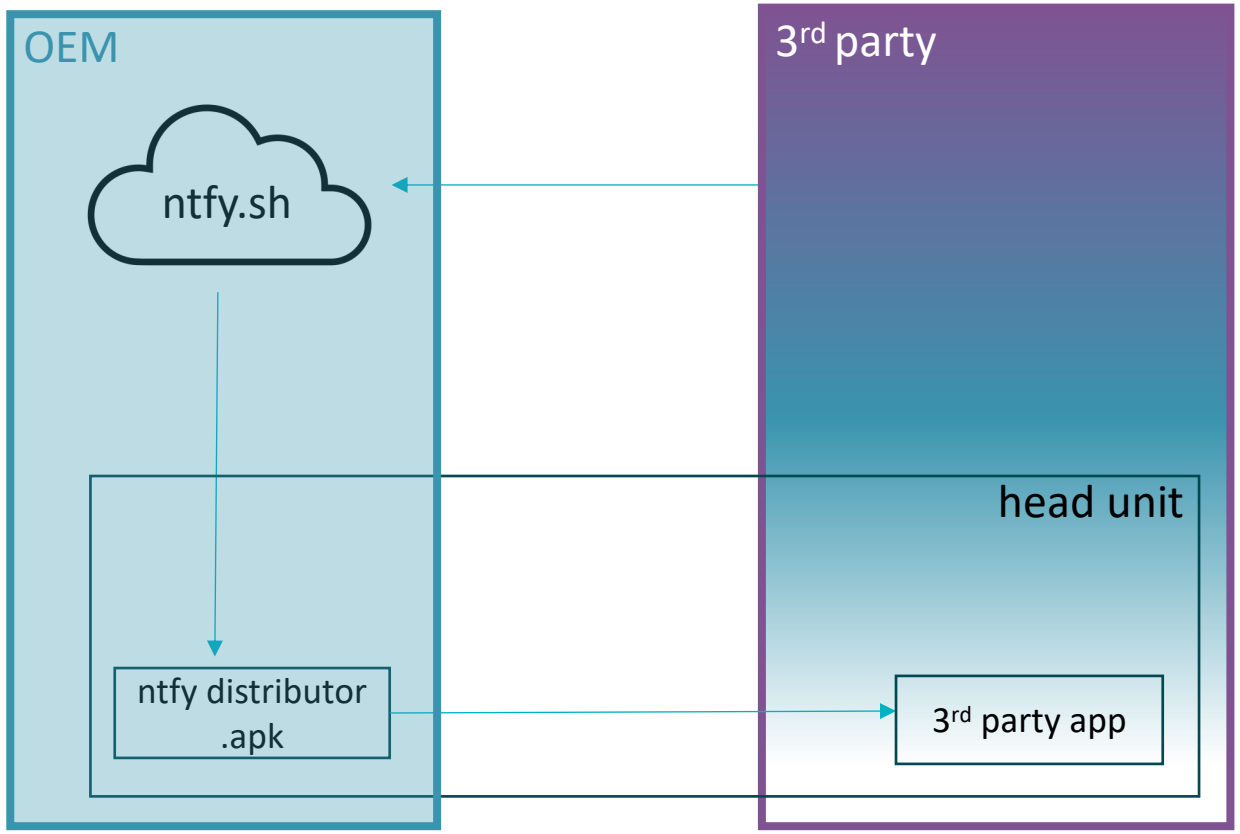
COVESA

Accelerating the future of connected vehicles

Backup

POC 2

Existing solution for OEM part / 3rd party application POC



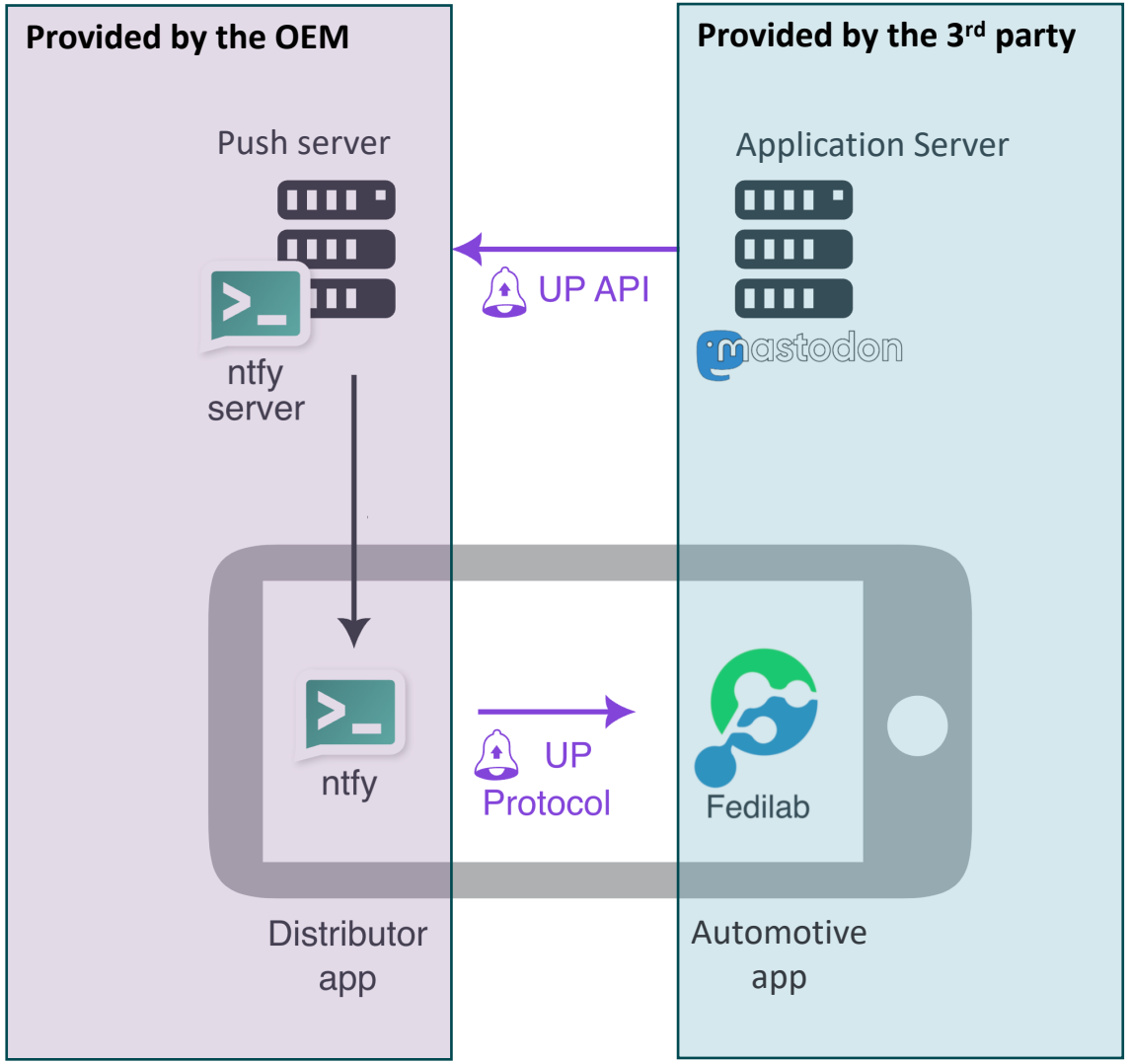
1. Use ntfy.sh as the push server and distributor
   - Easily installable on the head unit as an .apk[1]
   - Provides a free push server we can use for the POC
2. Install the 3rd party application POC that implements UnifiedPush

→ Evaluate the end-user experience

Time requirements: <1 hour once the 3rd party application POC is available

[1] https://f-droid.org/de/packages/io.heckel.ntfy/

UnifiedPush



- **Application server:** This is the server that hosts the application – the 3rd party backend
- **Automotive application:** The (3rd party) application receiving the push notifications. The application gets started by the distributor on incoming push notifications if it is not running
- **Push server:** This is the server that listens for incoming push messages and forwards them to the connected Push Distributor running in the car
- **Distributor:** This is the application that forwards push messages to the registered end user application. It is the application which is connected to the Push Server and must be running at all times
- **UP API & UP Protocol:** The only thing COVESA needs to standardize

Source: https://unifiedpush.org/