# BMW enables driver mobility via Azure Service Fabric

April 30, 2018

*This post was authored by Dr. Gregory Athas and Dr. Sean Patterson from BMW, in conjunction with Ed Price, Mark Fussell, and Stas Kondratiev from Microsoft.*

This post is part of a series (https://blogs.msdn.microsoft.com/azureservicefabric/tag/case-study/) on customers who've participated in the Azure Service Fabric (https://azure.microsoft.com/en-us/services/service-fabric/) preview program over the last year. We look at why they chose Service Fabric, and take a closer look at the design of their application, particularly from a microservices perspective. In this post, we profile BMW, their BMW Connected mobile application running on Azure, and how they designed the architecture using Service Fabric.

## BMW Technology Corporation

Under the BMW Group umbrella, the BMW Technology Corporation is a technology development company that specializes in building connected car services. The organization is concentrating on creating software services that integrate consumers' digital lives with their mobility needs to make their end-to-end driving experiences easier, safer and more pleasurable. The new BMW Connected application is a personal mobility companion. It is a personalized and intelligent app that learns a user’s mobility patterns to optimize their time.

## Enabling Driver Mobility via Service Fabric

In order to proactively act on behalf of a user, the BMW Connected application needed a solution for combining machine learned driver intents, real time telemetry from devices and up-to-date commute conditions such as traffic. In addition, this service was built to scale up to millions of drivers, and asynchronously monitor the commute needs of each driver.

Furthermore, BMW wanted to continually update the service with learned behaviors and commute condition resources. Because of these reasons as well as mobile client limitations (battery and background processing) they found that the Service Fabric Actor model was the right solution for their service.

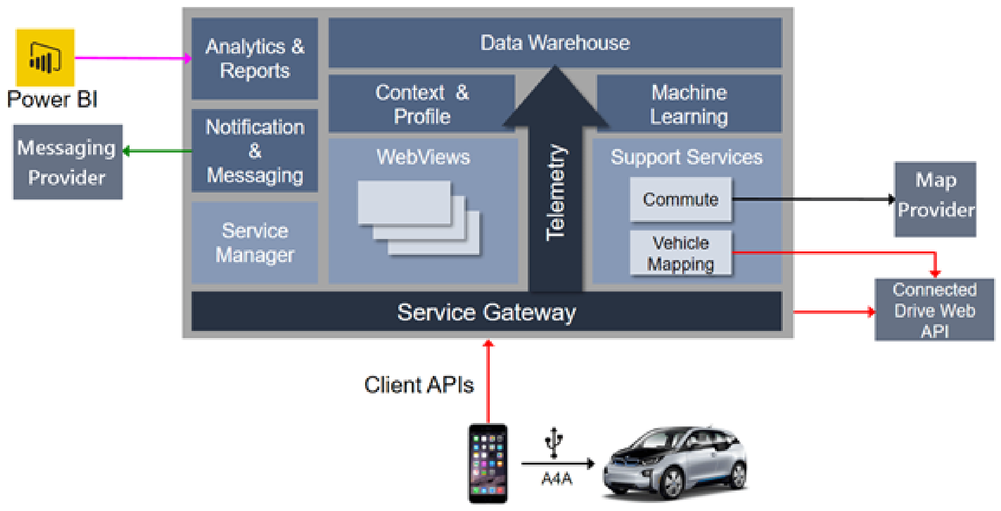
## Open Mobility Cloud

BMW’s Open Mobility Cloud (OMC) is built on Azure and communicates with the Connected mobile application. The diagram below shows the major subsystems of the OMC along with external services that it communicates with, such as Power BI, messaging services, and any map providers.

“ We’ve found actors to be a natural way to model users in our system. They allow us to focus on our core functionality while inherently supporting persistence, scalability, and resiliency.”

—Dr. Gregory Athas: Principal Software Architect
BMW Technology Corporation

## OPEN MOBILITY CLOUD (OMC) ARCHITECTURE

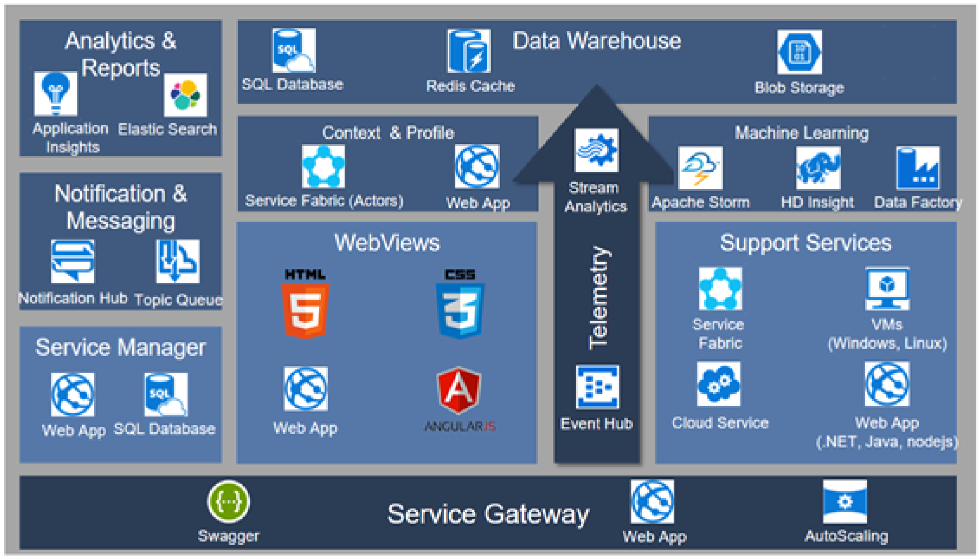


The diagram below further illustrates the Azure services used in each of the subsystems of the OMC. Next we will look deeper into the Context and Profile subsystem, which uses Service Fabric to implement the business logic for a BMW car driver.

" Azure Service Fabric is a sophisticated toolbox for creating a mix of stateless and stateful services that can be scaled and extended as our user base grows and we add new functionality."

—Dr. Sean Patterson: Senior Software Architect
BMW Technology Corporation

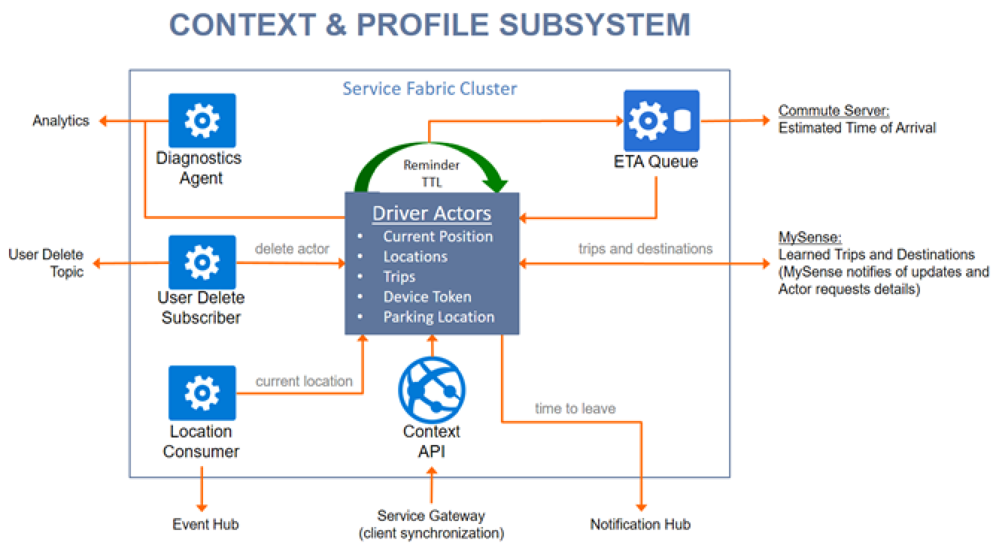
## OPEN MOBILITY CLOUD AZURE SERVICES



The Context and Profile subsystem is responsible for managing a BMW driver’s predicted and scheduled trips. Drive times for upcoming trips are asynchronously monitored relative to a driver’s current position to identify when a user should depart. Notifications are also sent to the driver’s mobile device indicating if the commute is longer than typical.

“ Actor reminders combined with Azure notifications have allowed us to offload complex client processing to the cloud.”

—Dr. Gregory Athas: Principal Software Architect
BMW Technology Corporation



The Context and Profile application consists of the following Service Fabric microservices:

- The Context API stateless service
- The Driver actor stateful service
- The Location Consumer stateless service
- The User Delete stateless service
- The ETA Queue stateful service

## Context API Stateless Service

The Context API service is responsible for implementing the APIs that allow the non-Service Fabric components of the OMC to communicate with the Context and Profile subsystem. The main API that the Context API implements is the Sync API. The mobile client calls this API to inform the Context and Profile subsystem of any new, changed, or deleted user generated locations and trips. There are plans to add additional APIs to allow additional devices and clients to communicate with the Context and Profile subsystem.

## Driver Actor Stateful Service

The Driver Actor service is responsible for keeping track of the BMW driver's profile information and determining when to generate a notification that it is time to leave for a particular trip.

One of the reasons that BMW chose to use Service Fabric for the Context and Profile subsystem is because of how well the actor model maps to their problem domain. Driver actors are easy to program because they represent just one user and Service Fabric handles most of the logic for dealing with aggregation and concurrency required to deal with a large number of users. This is inaddition to the automatic replication and failover capabilities that Service Fabric provides.

Each user's profile information is stored in their driver actor's state. BMW uses an anonymous user ID in the system as the name of their actor. One of the challenges they had was BMW's requirement that all data at rest must be encrypted. This required BMW to keep all data in the state encrypted at all times since the actor state can be written to disk at any time (i.e., after any actor method has completed). Because of this, each actor needs to encrypt its state after every method call. This has caused BMW to require higher spec VMs to handle the processing load.

Each driver actor receives data from five sources:

- Sync messages from the Context API service.
- A stream of current locations of the driver from the Location Consumer service. The location where the driver has parked their car also comes from the Location Consumer service.
- Learned destinations and predicted trips from the MySense machine learning service.
- Deleted anonymous user IDs from the User Delete service.
- Estimated trip times from the ETA Queue service.

The data received from these sources allows the driver actor to perform its functions of keeping track of the driver's profile information and calculating when to send (and sending) TTL notifications.

TTL notifications come in two flavors. For a normal trip that the user has created on the mobile app, a TTL notification is sent shortly before it is time for the user to leave for the destination location of the trip. Internally in the driver actor, reminders are used to wake up the driver actor when it is time to send a notification for a particular trip. Note that since traffic is constantly changing, it is not good enough to set a reminder for the estimated time to leave many hours in advance. Instead a reminder needs to fire at a time that is some short amount of time before departure and re-evaluate whether a notification should be sent at that time or if another reminder should be scheduled to check again at a later time whether it is time to send the notification.

When it is determined that notification needs to be sent to the driver's mobile device, an Azure notification hub (https://azure.microsoft.com/en-us/services/notification-hubs/) is contacted with the details of the notification.

## Location Consumer Stateless Service

Each mobile client sends a stream of geo-locations to an Azure Event Hub (https://azure.microsoft.com/en-us/services/event-hubs/) . The Location Consumer service is responsible for pulling current geo-locations from an Event Hub and feeding them to corresponding driver actor. The mobile client will also place an event on the Event Hub that contains a driver's parking geo-location when the user parks the car. The Location Consumer will also pull the parking geo-location from the Event Hub and send it to the correct driver actor.

## User Delete Stateless Service

The User Delete service subscribes to a Service Bus topic that contains user IDs of users who have requested that all of their personal data be deleted. This topic is subscribed by several of the non-Service Fabric components of the OMC so that all personal data in the entire OMC is deleted.

When the User Delete service receives a delete request, the driver actor is then deleted.

## ETA Queue Stateful Service

The ETA Queue service sits between the Driver Actor service and the OMC Commute server. The Commute server returns a trip drive time given a start geo-location and an end geo-location. It takes into account traffic to generate the drive time. (The Commute Server communicates with an external to BMW partner's service to acquire the drive time, as shown in the first diagram.)

The ETA Queue service is designed to decouple driver actors from the Commute server and allow the requests from the driver actors to communicate with the Commute server asynchronously. The ETA Queue service contains a request and a response queue. Requests from the driver actor are placed on the request queue and the ETA Queue service sends the requests to the Commute server in the order in which they are received. As the ETA Queue service receives responses from the Commute server they are placed on the response queue. The ETA Queue service sends the Commute server responses to the requesting actors in the order in which they are received from the Commute server. The queue elements in the ETA Queue service are stored in a Service Fabric reliable collection so that the requests and responses are not lost if the ETA Queue service goes down.

## Disaster Recovery and Monitoring

All driver actor states are backed up to a durable Azure Storage in another Azure region. If a catastrophe happens and the Service Fabric region goes down, they can spin up another Context and Profile subsystem in another Azure region and recover all the user data from the backup.

The Context and Profile subsystem contains the usual set of monitors and alarms to watch the health of the Service Fabric cluster and the services that run on the cluster. Besides the infrastructure monitoring based on the powerful out of the box Service Fabric health model, BMW has thoroughly instrumented their application code to collect and view the app diagnostic data in a single system based on Elastic Search. Alarms and prompt notifications are generated in case there's an abnormal behavior in the system.

## Summary

BMW faced the development challenge of scaling millions of drivers. With Service Fabric, they are able to complete rolling upgrades without any downtime, thanks to its automatic replication and failover capabilities. If there's an error, Service Fabric helps them effortlessly roll back to the previous version. They heavily leverage Service Fabric's actor reminders and Azure notifications to offload their complex client processing to the Azure cloud.

BMW is able to focus on the core functionality of their Connected mobile application, by leveraging Service Fabric actors to model users in their system. This helps them inherently support persistence, scalability, and resiliency. Service Fabric is their toolbox for creating a mix of stateless and stateful services that they can easily scale to millions of drivers and can extend as they grow their user base and add advanced functionality.

" With Service Fabric, we can perform a rolling upgrade of our system with no downtime. It will roll back to the previous version if there is an error."

—Sean Patterson: Senior Software Architect
BMW Technology Corporation