Distributed VSS Data Sets

Bring VSS to life
The Concept

Onboard Applications

Cloud

Mobile Applications

Realm Database

Device Sync

MongoDB Atlas

GraphQL API

Web Applications

Realm Database

Device Sync

Mobile Applications
Keep the Data in Sync

Sync Features
- Realtime
- Efficient
  - Compressed
  - Delta Only
- Conflict Resolution
- Offline First
- Dynamic Filtering
Future Outlook / Roadmap

Onboard

- Apps with Realm SDK(s)
- Dynamic Filter(s)
- Sync Service

Data Flow

Cloud Backend

- Device-Sync Backend
- MongoDB Atlas Database

Data Objects

- Domain / app specific objects
- All synced objects for a specific vehicle
- All synced objects for all vehicles in distinct collections
Documents Are Objects
Related data contained in a single, rich document

- Flexible
- Scalable
- Always On
- Freedom to run anywhere
- ...

```
{
   "_id" : ObjectId("5ad88534e3632e1a35a58d00"),
   "name" : {
      "first" : "John",
      "last" : "Doe" },
   "address" : [
      { "location" : "work",
        "address" : {  
          "street" : "16 Hatfields",
          "city" : "London",
          "postal_code" : "SE1 8DJ"},
          "geo" : { "type" : "Point", "coord" : [  
            -0.109081, 51.5065752]
            },
      },
   ],
   "dob" : ISODate("1977-04-01T05:00:00Z"),
   "retirement_fund" : NumberDecimal("1292815.75")
}
```
“Realm” - Embedded OSS Database

Offline first paradigm
• Usage: 100k+ developers; 65% of Fortune 1000; 2B+ app installs
• 47k+ Github stars
• Apache 2.0 license
• Active community involvement

Easy for developers
• Designed and built for resource constrained environments
• Just objects, with native code paradigms
• Live objects update automatically
• The class definitions are the database schema

2010
Development started by two former Nokia engineers

2016
Official announcement of Realm mobile platform

>2019
Acquisition by MongoDB

>2022
Device Sync integration into Atlas
Vehicle Signal Specification

VSS Tree

Official VSS Documentation:
https://covesa.github.io/vehicle_signal_specification/
Objects are Instances of Classes

```typescript
export class Device {
    public _id = new ObjectId;
    public name = "";
    public owner_id = "";
    public isOn = false;
    public flexibleData?: Realm.Dictionary<string>;
    public components: Array<Component> = [];

    public static schema = {
        name: 'Device',
        primaryKey: '_id',
        properties: {
            _id: 'objectId',
            name: 'string',
            owner_id: 'string',
            isOn: 'bool',
            components: 'Component[]',
            flexibleData: 'string[]'
        }
    }
}

export class Component {
    public _id = new ObjectId;
    public name = "";
    public owner_id = "";

    public static schema = {
        name: 'Component',
        primaryKey: '_id',
        properties: {
            _id: 'objectId',
            name: 'string?',
            owner_id: 'string'
        }
    }
}
```
Backend JSON Schema

Device

Component

Relationship

```json
{
  "title": "Device",
  "bsonType": "object",
  "required": [
    "_id",
    "name",
    "owner_id",
    "tsOh",
    "sensor"
  ],
  "properties": {
    "_id": {
      " bsonType": "objectId"
    },
    "name": {
      " bsonType": "string"
    },
    "owner_id": {
      " bsonType": "string"
    },
    "components": {
      " bsonType": "array",
      "items": {
        " bsonType": "objectId"
      },
      "isList": true
    },
    "tsOh": {
      " bsonType": "bool"
    },
    "flexibleData": {
      " bsonType": "object",
      "additionalProperties": {
        " bsonType": "mixed"
      }
    },
    "mixedTypes": {
      " bsonType": "mixed"
    },
    "sensor": {
      " bsonType": "long"
    }
  }
}
```

```json
{
  "title": "Component",
  "bsonType": "object",
  "required": [
    "_id",
    "owner_id"
  ],
  "properties": {
    "_id": {
      " bsonType": "objectId"
    },
    "name": {
      " bsonType": "string"
    },
    "owner_id": {
      " bsonType": "string"
    }
  }
}
```
GraphQL Schema

JSON schema is automatically converted into GraphQL schema.
Connected Vehicle Demo Architecture

**Connected Vehicle**
- IoT
- Realm SDK

**AWS Cloud**
- MongoDB Atlas
  - Device Sync
  - Database Trigger
  - GraphQL/Data API
- AWS Lambda
- Event Bridge
- AWS SNS
- SageMaker Studio
  - SageMaker
  - Model

**Connected Vehicle**
- Vehicle Owner
- Mobile

**Web Applications**
- Workshop
- Web Applications

**Resources**
Connected Vehicle End to End Architecture

Vehicle

- Realm SDK

Vehicle Owner

Mobile

Workshop

Web Applications

MongoDB Atlas

- Device Sync
- Database Trigger
- Spell icon

GraphQL/ Data API

App Services Function

Vertex AI

Google Cloud


Work in Progress
Topics for Discussion

Unsolved Challenge(s)
- Conversion of hierarchical structures into classes leads to very long class names beyond class name length limits

Collaboration / Contribution
- Currently there is no VSS tooling for conversion to JSON schema
Curious? -> Reach out

industry.solutions@mongodb.coxm