Way2i - eHail

Way2i interface for working with eHail functionality

```
@Path("/hail/")
@Consumes("application/x-protobuf")
@Produces("application/x-protobuf")
public interface Way2iHailService {
    @POST
    @Path("/vehicles")
    VehicleCongregationResponse getNearestVehicles(VehicleCongregationRequest 
request);
    @POST
    @Path("/sessions")
    HailTaxiSessionResponse hail(HailTaxiSessionRequest request);
    @GET
    @Path("/sessions/{sessionId}")
    HailTaxiSessionResponse getHailSession(@PathParam("sessionId") String sessionId);
    @PUT
    @Path("/sessions/{sessionId}/confirm")
    ResponseData confirm(@PathParam("sessionId") String sessionId, Vehicle vehicle);
    @PUT
    @Path("/sessions/{sessionId}/expire")
    ResponseData expire(@PathParam("sessionId") String sessionId);
    @PUT
    @Path("/sessions/{sessionId}/order/notes")
    ResponseData sendNotesToDriver(@PathParam("sessionId") String sessionId, String 
notes);
    @POST
    @Path("/sessions/{sessionId}/driver/callback")
    ResponseData connectRiderAndDriver(@PathParam(value = "sessionId") String 
sessionId);
    @POST
    @Path("/sessions/{sessionId}/rating")
    ResponseData rateDriver(@PathParam(value = "sessionId") String sessionId, 
RateRequest request);
}
```

Protobuff descriptions for eHail objects

```
message Position {
    required float latitude = 1;
    required float longitude = 2;
    optional float direction = 3;
    optional float speed = 4;
}
message Driver {
    optional string driverId = 1;
    optional string firstName = 2;
    optional float rating = 3;
    optional string phone = 4;
    optional bool hasPhone = 5;
}
message Vehicle {
    optional string vehicleId = 1;
    optional string tag = 2;
    optional string type = 3;
    optional Driver driver = 4;
    optional string description = 5;
    required Position position = 6;
}
message VehicleCandidate {
    optional string vehicleId = 1;
    optional string state = 2;
    optional float rating = 3;
}
message VehicleCongregationResponse {
    repeated Vehicle vehicle = 1;
    required ResponseData response = 2;
}
message VehicleCongregationRequest {
    required Position position = 1;
    required int32 radius = 2;
 }
message HailTaxiSessionRequest {
    required string callbackEndpoint = 1;
    //endpoint should be  URL like http(s)://server:port/path/to/endpoint/{sessionId}
TaxiEvent message will be sent with POST method
    required float lat = 2;
    required float lon = 3;
    required string address = 4;
    optional string notes = 5;
    required string firstName = 6;
    required string lastName = 7;
    required string email = 8;
    required string phone = 9;
    optional ServiceFee ehailFee = 10;
    optional double tipsPercentage = 11;
}
message HailTaxiSessionResponse {
   optional string taxiSessionId = 1;
   required ResponseData response = 2;
   optional string market = 3;
   repeated VehicleCandidate candidates = 4;
}
message RateRequest {
    required int32 rate = 1;
    optional string notes = 2;
}
```

Full list of system events (protobuff description)

```
enum TAXI_EVENT_TYPE {
    METER_TIME_OFF = 0;
    METER_REHIRED = 1;
    FARE_UPDATED = 2;
    METER_PAYMENT_ACKNOWLEDGED = 3;
    METER_HIRED = 4;
    WAY2RIDE_CANCELED = 5;

    HAIL_SUBMITTED = 6;
    HAIL_REVOKED = 7;
    HAIL_EXPIRED = 8;
    HAIL_CONFIRMED = 9;
    HAIL_TAXI_ARRIVED = 10;
    HAIL_RIDER_INVEHICLE = 11;
    HAIL_RIDER_NOSHOW = 12;
    HAIL_RIDE_STARTED = 13;
    HAIL_DRIVER_CANCELED = 14;
    HAIL_SERVER_CANCELED = 15;
    HAIL_PAIRING_FAILED = 16;
    HAIL_LOCATION = 17;
    HAIL_VEHICLE_ASSIGNMENT_STATE_CHANGED = 18;
}
```

Code samples

Taxi congregation

```
//build request
Way2I.VehicleCongregationRequest.Builder requestBuilder =
Way2I.VehicleCongregationRequest.newBuilder();
Way2I.Position.Builder positionBuilder = Way2I.Position.newBuilder();
requestBuilder.setPosition(positionBuilder.setLatitude(40.766974f).setLongitude(-73.87
3247f).build());
requestBuilder.setRadius(100);

//retrieve available taxis
Way2I.VehicleCongregationResponse resp =
way2iHailService.getNearestVehicles(requestBuilder.build());
//check if execution was OK
if (resp.getResponse().getStatus() == 0) {
 //display result
 resp.getVehicleList();
}
```

Submit order

```
//build request
HailTaxiSessionRequest.Builder taxiSessionRequestB =
HailTaxiSessionRequest.newBuilder();
taxiSessionRequestB.setEmail("john@doe.com");
taxiSessionRequestB.setFirstName("John");
taxiSessionRequestB.setLastName("Doe");
taxiSessionRequestB.setAddress("Main str. 123");
taxiSessionRequestB.setCallbackEndpoint("https://customer-endpoint.com/");
taxiSessionRequestB.setLat(40.766974f);
taxiSessionRequestB.setLon(-73.873247f);
taxiSessionRequestB.setPhone("+123456789");

//send request
HailTaxiSessionResponse sessionResp = hailClient.hail(taxiSessionRequestB.build());

//check if execution OK
if (sessionResp.getResponse().getStatus() == 0) {

}
```