



**Integration HLD**

**Version 0-1**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 14 Feb 2018 | 0-1 | Initial Draft for discussion | Joe Dowling.  MosaabElemam |
| 15 Feb 2018 | 0-2 | Update with further queries | Joe Dowlimg |
| 2 Mar 2018 | 0-3 | Updated with Car Track Feedback | Joe Dowling |
| 23 Mar 2018 | 0-4 | Update with technical specification | Vinay Pratap Singh |

Table of Contents

[1. Introduction 3](#_Toc512269919)

[1.1 Queries 4](#_Toc512269920)

[2. Technical Details of Interface 5](#_Toc512269921)

[2.1 Web Services 5](#_Toc512269922)

[2.2 Secure FTP 5](#_Toc512269923)

[2.3 Email Notifications 5](#_Toc512269924)

[3. Driver Message 6](#_Toc512269925)

[3.1 Interface Frequency and Schedule 6](#_Toc512269926)

[3.2 Message details 6](#_Toc512269927)

[3.3 Queries 7](#_Toc512269928)

[3.4 Implementation logic 8](#_Toc512269929)

[4. Vehicle Message 9](#_Toc512269930)

[4.1 Interface Frequency and Schedule 9](#_Toc512269931)

[4.2 Message details 9](#_Toc512269932)

[4.3 Queries 10](#_Toc512269933)

[4.4 Implementation logic 11](#_Toc512269934)

[5. License Message 12](#_Toc512269935)

[5.1 Interface Frequency and Schedule 12](#_Toc512269936)

[5.2 Message details 12](#_Toc512269937)

[5.3 Queries 12](#_Toc512269938)

[6. Scorecard Batch File 13](#_Toc512269939)

[6.1 Interface Frequency and Schedule 13](#_Toc512269940)

[6.2 Batch details 13](#_Toc512269941)

[6.3 Queries 13](#_Toc512269942)

[7. Driver Status Message 15](#_Toc512269943)

[7.1 Interface Frequency and Schedule 15](#_Toc512269944)

[7.2 Message details 15](#_Toc512269945)

[7.3 Queries 16](#_Toc512269946)

[8. Passenger Message 17](#_Toc512269947)

[8.1 Interface Frequency and Schedule 17](#_Toc512269948)

[8.2 Message details 17](#_Toc512269949)

[8.3 Queries 17](#_Toc512269950)

[9. Trip Message 18](#_Toc512269951)

[9.1 Interface Frequency and Schedule 19](#_Toc512269952)

[9.2 Message details 19](#_Toc512269953)

[9.3 Queries 19](#_Toc512269954)

[10. Incident Message 20](#_Toc512269955)

[10.1 Interface Frequency and Schedule 20](#_Toc512269956)

[10.2 Message details 20](#_Toc512269957)

[10.3 Queries 20](#_Toc512269958)

[11. General Queries 21](#_Toc512269959)

# Introduction

The Car Track system ([www.cartrackme.com](http://www.cartrackme.com)) is being implemented by APTC in conjunction with the AjmanGO project which is implementing the back-end CRM system.

There is a need to have a clear distinction between the 2 systems and to clearly define within the overall system architecture which system is the master for certain data entities.

The following table defines this ownership for the major data entities within the system:

|  |  |  |
| --- | --- | --- |
| Data Entity | Description | Master System |
| Driver | Driver details including licenses, etc but excluding driving performance scorecard. | AjmanGO |
| Vehicle | Vehicle details including license plate, maintenance details etc. | AjmanGO |
| Driver Performance | The driver performance scorecard are generated by the hardware installed in the vehicle and reported to CarTrack. | Car Track |
| Journey | Details of each journey undertaken – start and end points, distance, time, etc. | Car Track |

The interface between the systems is made up of several discreet messages as follows:

| Message | Technology | Push to … | Comments |
| --- | --- | --- | --- |
| Driver | WebService | CarTrack | AjmanGO issystem of record for Driver |
| Vehicle | WebService and/or email notification | CarTrack | AjmanGO is system of record for vehicle |
| License | WebService | CarTrack | Both drivers’ & vehicles’ |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

remainder of this document gives the details of each of these messages.

## Queries

The following table identifies queries / issues - to be / already resolved:

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
|  | Vehicle Message | Updating vehicle info electronically from AjmanGO (CRM) into Car Track (CAMS) needs to be confirmed by CT after discussion with their development team. |
|  |  |  |

# Technical Details of Interface

The interface will be achieved using 3 technological components as follows:

## Web Services

All real time interfaces will use webservices to send and receive messages as required by the interface. Full details of the format of these messages, the security credentials required for the interface and the error and recovery processes will be contained in the detailed specifications to be drafted by the CT / AjmanGO teams.

## Secure FTP

A secure FTP connection will be used for all non-real-time electronic interfaces. This includes 2 elements:

* Scorecards coming from CT,
* Driver’s photo being sent from AjmanGo corresponding to Driver details sent via Webservice.

Full details of the format of these files, the security credentials required for the transfer and the error and recovery processes will be contained in the detailed specifications to be drafted by the CT / AjmanGO teams.

## Email Notifications

In some limited cases it may be necessary to have a simple email notification being sent between the parties, where the volume of messages is likely to be so low as to not warrant an electronic interface.

Currently there are 2 cases envisaged:

* When CT engineers are installing a device on a new vehicle and the VIN (chassis) number and the registration number of the vehicle don’t match the details in the system. In this case CT will continue to install the device (assuming the registration number is correct) but will notify AjmanGO via email of the discrepancy. It will be required in AjmanGO to have a suitable process to follow up these occurrences.
* When a new vehicle is being added to the AjmanGO system if no interface exists for Vehicle Messages then an email will be sent instead.

Full details of the format of these emails and the email address to be used will be contained in the detailed specifications to be drafted by the CT / AjmanGO teams.

# Driver Message

AjmanGO is the master system for all driver related data. This includes driver details, ID and password for login to system, etc. It specifically excludes operational data like performance scorecards and trip details.

Whenever a new driver is created by the Franchisee they will go through an approval process with APTC to ensure the driver is properly registered and approved. This is all done on the AjmanGO system.

Once a driver is fully approved or whenever relevant driver details are changed on AjmanGO a message will be sent to CarTrack to create / update the driver details.

## Interface Frequency and Schedule

Immediate (real-time)

On demand

Hourly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_ Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Daily Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Weekly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Monthly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Quarterly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Annually Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Other Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

## Message details

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| <Action>  Values: ADD or MOD | Defines whether this is a new driver or whether it is an update for an existing driver. Drivers will never be deleted instead their status will be set to Invlaid – see below. | String |
| <ID> | Unique identifier for Driver. See query below | 999-9999-9999999-9 |
| <DriverValid> | Indicates whether this driver is still a valid driver or should be ignored for all processing purposes. | Boolean Flag |
| <FirstNameEN> |  | String |
| <LastNameEN> |  | String |
| <FirstNameAR> |  | String |
| <LastNameAR> |  | String |
| <MobileNumber> |  | Numeric |
| <EmailAddress> |  | Email |
| <VehicleTypes> |  | Several Codes as required |
| <PermitNumber> |  | String |
| <LicenseNumber> |  | Numeric(7) |
| <PassportNumber> |  |  |
| <Photo> | Image file will be sent via Secure FTP when required – the Unique ID will be used as the image file name. | Image file reference |
| <Username> |  | Any character string |
| <Password> |  | Any character string |

## Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
| Agreed | Unique ID | We would suggest using the Emirati ID Number as the unique identifier. This will be a 15 digit number |
| Agreed | Other data items | IN order to ensure non-duplication of data and also to have proper control of updates it is essential that AjmanGO be the master copy for the driver.  Can it be confirmed that no other personal data is maintained on CarTrack for drivers – e.g. DOB, Nationality, etc.  This has been confirmed – CT does not store DOB, Nationality or any ther personal data other than what is in these messages. |
| Agreed | Vehicle Registration | This was suggested as a driver related field – is this correct. Do vehicles not get handed on from shift to shift to different driver – therefore a driver may drive different vehicles over time.  It has been agreed that AjmanGO will send the vehicle types that are covered by a drivers permit. This can be for several types of vehicles – e.g. car, light bus etc. |
| Agreed | Mobile Number | We hold this in international format – it will be passed to CT as 00971…. . |
| Agreed | ID/Passport Number | Passport Number is different to Emirati ID. We already suggest the Emirati ID as the Unique reference – Therefore suggest that this field is Passport Number Only – which will be optional |
| Agreed | Permit Number | Although unlikely a Driver may have several Permits. Therefore, this field will only be the most recent Taxi Driving Permit Number available within AjmanGO. |
| Agreed | CarTrack Username | Currently in CT the username for a driver is the Cell Number. This will be used for drivers in AjmanGO as well so both systems will be aligned. |
|  | Password | It looks likely that the password for drivers within the CT system will be a 4 digit number, AjmanGO can accommodate this. However, if CT will allow the driver to change their password then there must be a way for CT to pass this new password back to AjmanGO. It is critical that both systems support a consistent password value across both systems. |
| Agreed | Repeat Password | We don’t need a repeated password as an element of an interface. |

## Implementation logic

Following is the algorithm steps to push information in to theCarTrack.

An application will be developed  as .net Console application which will run all the time in background and will have below flow.

1.  Read “AjmanGo”all the  records not processed so far.

2.  Bring all the record in memory cache and start loop through until all the records are processed.

  2.1 Read record with **“<Action>,<ID>,<DriverValid>,<FirstNameEN>,<LastNameEN>,<FirstNameAR>,<LastNameAR>,<MobileNumber>,<EmailAddress>,**

**<VehicleTypes>,<PermitNumber>,<LicenseNumber>,<PassportNumber>,<Photo>,<Username>,<Password>”** column and prepare a message and create a log file to store the audit.

  2.2 Call the CarTrackservice“**Service Name”**to push the above data and wait for the response.

  2.3 update log file and audit record with the CT response ID (“CTDR0999-999-999”) and status (success or failure).

3. End loop.

4. Process new batch

**Note:**

1.Driver web server schedule in our APTC server.

2.CarTrack service required for Drivers

***Codescript for Driver Input Parameter***

AjmanGO a message will be sent to CarTrack to create / update the Dirver details with web service.

Following is the code to push information in to the CarTrack.

**Class**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Text;

using Newtonsoft.Json;

using System.Net.Http;

namespace Ajman\_Go\_DriverService

{

class Program

{

static void Main(string[] args)

{

ConsumeEventSyncconsumeEventSync = new ConsumeEventSync();

consumeEventSync.GetDriverInformation();

}

}

public class ConsumeEventSync

{

//Get all driver information

public void GetDriverInformation()

{

DataTabledt = new DataTable();

try

{

DataSet ds = new DataSet();

using (SqlConnection con = new SqlConnection(@"data source =LAPTOP-V63F2LH9;initial catalog = Ajman\_Go; persist security info = True;Integrated Security = SSPI;"))

{

using (SqlCommandcmd = new SqlCommand("select \* from Driver", con))

{

cmd.CommandType = CommandType.Text;

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(ds);

}

}

dt = ds.Tables[0];

//call function to send driver inforation

SendDriverInforamtionToCarTrack(dt);

}

catch (Exception ex)

{

dt = new DataTable();

}

}

**// function is used for post information for car track**

public void SendDriverInforamtionToCarTrack(DataTabledataTable) //Get All Events Records

{

try

{

foreach (DataRow row in dataTable.Rows)

{

HttpClient cons = new HttpClient();

cons.DefaultRequestHeaders.Accept.Clear();

**//-------------------------Model Object -------------------------------------**

DriverModeldriverModel = new DriverModel();

driverModel.ID = Convert.ToInt64(row["ID"]);

driverModel.DriverValid = Convert.ToBoolean (row["DriverValid"]);

driverModel.Value = row["Action"].ToString();

driverModel.UserName = row["UserName"].ToString();

driverModel.Password = row["Password"].ToString();

driverModel.FirstNameEN = row["FirstNameEN"].ToString();

driverModel.LastNameEN = row["LastNameEN"].ToString();

driverModel.FirstNameAR = row["FirstNameAR"].ToString();

driverModel.EmailAddress = row["EmailAddress"].ToString();

driverModel.LastNameAR = row["LastNameAR"].ToString();

driverModel.MobileNumber = Convert.ToInt64(row["MobileNumber"]);

driverModel.LicenseNumber = Convert.ToInt32(row["LicenseNumber"]);

driverModel.PasportNumber = row["PasportNumber"].ToString();

driverModel.PermitNumber = row["PermitNumber"].ToString();

driverModel.VehicleType = row["VehicleType"].ToString();

driverModel.Photo = row["Photo"].ToString();

varjson = JsonConvert.SerializeObject(driverModel);

//**Note: Jsonfunction push the record to String format (for all type of string input).**

using (cons)

{

**//set authentications**

**//cons.DefaultRequestHeaders.Add("username", "sa@role.com");**

**//cons.DefaultRequestHeaders.Add("password", "lopzwsx@23");**

**// Call CarTrack web API**

HttpResponseMessage response = cons.PostAsync("http://localhost:64719/api/driver/saveDriverInformation", new StringContent(json.ToString(), Encoding.UTF8, "application/json")).Result;

response.EnsureSuccessStatusCode();

//get response from carTrackApi

UpdateResponseStatus(response);

}

}

}

catch (Exception ex)

{

throw;

}

}

**// function is used for update response in AjmanGo system**

public void UpdateResponseStatus(HttpResponseMessage response )

{

string status = string.Empty;

string description = response.Content.ToString();

if (response.IsSuccessStatusCode)

{

status = "Yes";

}

else

{

status = "No";

}

// connect database and update response

using (SqlConnection con = new SqlConnection(@"data source =LAPTOP-V63F2LH9;initial catalog = Ajman\_Go; persist security info = True;Integrated Security = SSPI;"))

{

con.Open();

using (SqlCommandcmd = new SqlCommand("update Driver set ResponseStatus='"+ status + "',ResponseDescription='"+ response.Content + "'", con))

{

cmd.CommandType = CommandType.Text;

var result = cmd.ExecuteNonQuery();

}

}

}

// function is used for get all driver information

}

**// Model Class**

public class DriverModel

{

public Int64 ID { get; set; }

public bool DriverValid { get; set; }

public string Action { get; set; }

public string UserName { get; set; }

public string Password { get; set; }

public string FirstNameEN { get; set; }

public string LastNameEN { get; set; }

public string FirstNameAR { get; set; }

public string LastNameAR { get; set; }

public Int64 MobileNumber { get; set; }

public string EmailAddress { get; set; }

public string VehicleType { get; set; }

public string PermitNumber { get; set; }

public Int32 LicenseNumber { get; set; }

public string PasportNumber { get; set; }

public string Photo { get; set; }

}

}

**Pass Parameter Input Via web API with Json format**

**JSon format Pass Parameter:**

{"ID":1,"DriverValid":true,"Action":"ADD","UserName":"","Password":"","FirstNameEN":"Arvind","LastNameEN":"Jaiswal","FirstNameAR":"Arvind","LastNameAR":"Jaiswal","MobileNumber":9312998984,"EmailAddress":null,"VehicleType":"Two Wheelar","PermitNumber":"123MNV","LicenseNumber":1234567,"PasportNumber":"ABC123","Photo":""}

**Response Codes:**

1. Success :200
2. Failure/Bad request :400

**Get Response from CarTrack web API.**

1. **Success Response Message Parameter:**

{StatusCode: 200, ReasonPhrase: 'OK', Version: 1.1, Content: System.Net.Http.StreamContent, Headers:{ Pragma: no-cache X-SourceFiles: =?UTF-8?B?QzpcVGVzdFByb2plY3RcV2ViQXBpXEFqbWFuLUdvQXBpXEFqbWFuLUdvQXBpXGFwaVxkcml2ZXJcc2F2ZURyaXZlckluZm9ybWF0aW9u?=

Cache-Control: no-cache Date: Thu, 26 Apr 2018 07:21:16 GMT Server: Microsoft-IIS/10.0

X-AspNet-Version: 4.0.30319 X-Powered-By: ASP.NET Content-Length: 9

Content-Type: application/json; charset=utf-8 Expires: -1}}

System.Net.Http.HttpResponseMessage

1. **Failed Response message Parameter :**

{Response status code does not indicate success: 400 (Bad Request).}

# Vehicle Message

AjmanGO is the master system for all vehicle related data.

Whenever a new vehicle is created by the Franchisee they will go through an approval process with APTC to ensure the vehicle is properly registered and approved.

Once a vehicle is fully approved or whenever relevant vehicle details are changed on AjmanGO a message will be sent to CarTrack to create / update the vehicle details.

## Interface Frequency and Schedule

Immediate (real-time)

On demand

Hourly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_ Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Daily Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Weekly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Monthly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Quarterly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Annually Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Other Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

## Message details

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| <Action>  Values: ADD or MOD | Defines whether this is a new vehicle or whether it is an update for an existing vehicle. | String |
| <ID> | Unique identifier for vehicle – same as Registration Number | String |
| <VehicleValid> | Indicates whether this vehicle is still a valid vehicle or should be ignored for all processing purposes. | Boolean Flag |
| <VIN> | VIN or Chassis Number | String |
| <EngineNumber> |  | String |
| >Registration< |  | String |
| >Vehicle type< | Code representing key aspects of the Vehicle – mainly number of passengers – it is also used on the Driver details to indicate which types of vehicles a driver is permitted to drive. | Code Value |
| >Make< |  | Code Value |
| >Model< |  | Code Value |
| >ModelYear< |  | 9999 |
| >Colour< |  | Code Value |

## Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
| Agreed | Unique ID | We need to check whether the registration number for a vehicle can be considered unique over the life of the vehicle – maybe use chassis number instead.  It has been agreed that the Registration number will be the UID and that the VIN number will also be held. CT processes when installing the unit on a vehicle will check that these are correct. |
| Agreed | Other data items | In order to ensure non-duplication of data and also to have proper control of updates it is essential that AjmanGO be the master copy for the vehicle.  Can it be confirmed that no other static data is maintained on CarTrack for vehicles – e.g. Chassis Number, etc. |
| Agreed | Vehicle Registration | This was suggested as a driver related field – is this correct. Do vehicles not get handed on from shift to shift to different driver – therefore a driver may drive different vehicles over time.  Vehicle Registration has been removed from the Driver Message. |
| Agreed | Vehicle Type | Can we agree a definitive list of codes (AjmanGO should be master as we are the source) |
| Agreed | Make | Can we agree a definitive list of codes (AjmanGO should be master as we are the source). |
| Agreed | Model | Can we agree a definitive list of codes (AjmanGO should be master as we are the source). |
| Agreed | Colour | Can we agree a definitive list of codes (AjmanGO should be master as we are the source). |

## Implementation logic

Following is the algorithm steps to push information in to the CarTrack.

An application will be developed  as .net Console application which will run all the time in background and will have below flow.

1.  Read “AjmanGo” all the  records not processed so far.

2.  Bring all the record in memory cache and start loop through until all the records are processed.

  2.1 Read record with **“<Action>,<ID>,<VehicleValid>,<VIN>,<EngineNumber>,<Registration>,<Vehicle type>,<Make>,<Model>,<ModelYear>,<Colour>”** column and prepare a message and create a log file to store the audit.

  2.2 Call the CarTrack service “**Service Name”** to push the above data and wait for the response.

  2.3 update log file and audit record with the CT response ID (“CTVC 999-9999-999”) and status (success or failure).

3. End loop.

4. Process new batch

**Note:**

1.Driver web server schedule in our APTC server.

2.CarTrack service required for Vehicles.

***Codescript for Vehicle Input Parameter***

AjmanGO a message will be sent to CarTrack to create / update the Vehicledetails with web service.

Following is the code to push information in to the CarTrack.

**Class**

using Newtonsoft.Json;

using System;

using System.Data;

using System.Data.SqlClient;

using System.Net.Http;

using System.Text;

namespace VehicleService

{

class Program

{

static void Main(string[] args)

{

ConsumeEventSyncconsumeEventSync = new ConsumeEventSync();

consumeEventSync.GetVehicleInformation();

}

}

public class ConsumeEventSync

{

//Get all driver information

public void GetVehicleInformation()

{

DataTabledt = new DataTable();

try

{

DataSet ds = new DataSet();

using (SqlConnection con = new SqlConnection(@"data source =LAPTOP-V63F2LH9;initial catalog = Ajman\_Go; persist security info = True;Integrated Security = SSPI;"))

{

using (SqlCommandcmd = new SqlCommand("select \* from Vehicle", con))

{

cmd.CommandType = CommandType.Text;

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(ds);

}

}

dt = ds.Tables[0];

//call function to send driver inforation

SendVehicleInforamtionToCarTrack(dt);

}

catch (Exception ex)

{

dt = new DataTable();

}

}

// function is used for post information for car track

**// function is used for post information for car track**

public void SendVehicleInforamtionToCarTrack(DataTabledataTable) //Get All Events Records

{

try

{

foreach (DataRow row in dataTable.Rows)

{

HttpClient cons = new HttpClient();

cons.DefaultRequestHeaders.Accept.Clear();

VehicleModelvehicleModel = new VehicleModel();

vehicleModel.ID = row["ID"].ToString();

vehicleModel.VehicleValid = Convert.ToBoolean(row["VehicleValid"]);

vehicleModel.Action = row["Action"].ToString();

vehicleModel.VIN = row["VIN"].ToString();

vehicleModel.EngineNumber = row["EngineNumber"].ToString();

vehicleModel.Registration = row["Registration"].ToString();

vehicleModel.VehicleType = Convert.ToInt32(row["VehicleType"]);

vehicleModel.Make = Convert.ToInt32(row["Make"]);

vehicleModel.Model = Convert.ToInt32(row["Model"]);

vehicleModel.ModelYear = Convert.ToInt32(row["ModelYear"]);

vehicleModel.Colour = Convert.ToInt32(row["Colour"]);

varjson = JsonConvert.SerializeObject(vehicleModel);

//**Note: Jsonfunction push the record to String format (for all type of string input).**

using (cons)

{

//set authentications

//cons.DefaultRequestHeaders.Add("username", "sa@role.com");

//cons.DefaultRequestHeaders.Add("password", "lopzwsx@23");

**// call web api for CarTrack**

HttpResponseMessage response = cons.PostAsync("http://localhost:64719/api/driver/saveDriverInformation", new StringContent(json.ToString(), Encoding.UTF8, "application/json")).Result;

response.EnsureSuccessStatusCode();

//get response from carTrackApi

UpdateResponseStatus(response);

}

}

}

catch (Exception ex)

{

throw;

}

}

**// function is used for update response in AjmanGo system**

public void UpdateResponseStatus(HttpResponseMessage response)

{

string status = string.Empty;

string description = response.Content.ToString();

if (response.IsSuccessStatusCode)

{

status = "Yes";

}

else

{

status = "No";

}

// connect database and update response

using (SqlConnection con = new SqlConnection(@"data source =LAPTOP-V63F2LH9;initial catalog = Ajman\_Go; persist security info = True;Integrated Security = SSPI;"))

{

con.Open();

using (SqlCommandcmd = new SqlCommand("update Driver set ResponseStatus='" + status + "',ResponseDescription='" + response.Content + "'", con))

{

cmd.CommandType = CommandType.Text;

var result = cmd.ExecuteNonQuery();

}

}

}

// function is used for get all driver information

}

**//Model**

public class VehicleModel

{

public string ID { get; set; }

public bool VehicleValid { get; set; }

public string Action { get; set; }

public string VIN { get; set; }

public string EngineNumber { get; set; }

public string Registration { get; set; }

public intVehicleType { get; set; }

public int Make { get; set; }

public int Model { get; set; }

public intModelYear { get; set; }

public intColour { get; set; }

}

}

**Pass Parameter Input Via web API with Json format**

**JSon format Pass Parameter:**

{"ID":1,"VehicleValid":true,"Action":"ADD,"VIN":"CH89677RTD","EngineNumber":"ENG99866760","Regitration":"ADU-0978","VehicleType":"02","Make":”01”,"Model":”XXXX”,"ModelYear":"2018","Color":"RED"}

**Response Codes:**

1. Success :200
2. Failure/Bad request :400

**Get Response from CarTrack web API.**

1. **Success Response Message Parameter:**

{StatusCode: 200, ReasonPhrase: 'OK', Version: 1.1, Content: System.Net.Http.StreamContent, Headers:{ Pragma: no-cache X-SourceFiles: =?UTF-8?B?QzpcVGVzdFByb2plY3RcV2ViQXBpXEFqbWFuLUdvQXBpXEFqbWFuLUdvQXBpXGFwaVxkcml2ZXJcc2F2ZURyaXZlckluZm9ybWF0aW9u?=

Cache-Control: no-cache Date: Thu, 26 Apr 2018 07:21:16 GMT Server: Microsoft-IIS/10.0

X-AspNet-Version: 4.0.30319 X-Powered-By: ASP.NET Content-Length: 9

Content-Type: application/json; charset=utf-8 Expires: -1 }}

System.Net.Http.HttpResponseMessage

1. **Failed Response message Parameter :**

{Response status code does not indicate success: 400 (Bad Request).}

# License Message

AjmanGO is the master system for all driver related data. This includes driver License details.

Once a driver license is fully approved and issued or whenever relevant driver license details are changed on AjmanGO a message will be sent to CarTrack to create / update the driver license details.

## Interface Frequency and Schedule

Immediate (real-time)

On demand

Hourly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_ Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Daily Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Weekly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Monthly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Quarterly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Annually Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Other Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

## Message details

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| <Action>  Values: ADD or MOD | Defines whether this is a new driver or whether it is an update for an existing driver. | String |
| <ID> | Unique identifier for Driver who holdsthe License. | 999-9999-9999999-9 |
| >IssueDate< |  | Date |
| >ExpiryDate< |  | Date |
| >LicenseNumber< |  | ??? |

## Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
|  | Message | Is this message needed by CarTrack? Yes it will be processed by CT to update the drivers details.  If easier for CT these details can be included in the Driver Message. |

# Scorecard Batch File

Car Track maintains a regular scorecard per driver. This will be passed to AjmanGO in order for the driver details to be centralized and usable in the wider APTC context.

## Interface Frequency and Schedule

Immediate (real-time)

On demand

Hourly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_ Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Daily Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Weekly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Monthly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Quarterly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Annually Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Other Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

## Batch details

The following defines the header elements for the file:

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| <StartDateTime> | Start date / Time for scorecard period | Date/Time |
| >EndDateTime< | End Date / Time for scorecard period | Date/Time |
| >DriverScore< | Contains all the rows of driver related scores | XML/JSON string |

The following defines the elements for each row:

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| <ID> | Unique identifier for Driver. | 999-9999-9999999-9 |
| >KmsTravelled< |  | Numeric |
| >Braking< |  | Integer 999 |
| >Accel< |  | Integer 999 |
| >Corner< |  | Integer 999 |
| >Idle< |  | Integer 999 |
| >Speeding< |  | Integer 999 |
| >AverageTotal< |  | Numeric |

## Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
| Agreed | Frequency | How frequently are the driver scorecards produced by CarTrack? This should be the frequency of transfer to AjmanGO. – This file will be sent daily by FTP. |
|  | Format | CarTrack have suggested that this data be passed as an Excel Spreadsheet – we would request that it be sent in an XML/JSON format similar to all other messages. |
|  | Average Total | What does this field mean? Description has been provided as follows: |

# Driver Status Message

AjmanGO is the master system for all driver related data. It is important that it contains the details about the driver events so that a full view is available of the driver history. This is required so that AjmanGO can provide their stakeholders or other entities (e.g. Police) with relevant and timely data to answer queries.

The following Driver States are supported within CarTrack:

* Available
* En-Route
* Booking Request Accepted
* Passenger loaded
* On Break
* Un-available
* Booking Requested
* Off Line

Those states marked in grey are relevant to AjmanGO and should be sent by WebService to AjmanGO.

## Interface Frequency and Schedule

Immediate (real-time)

On demand

Hourly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_ Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Daily Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Weekly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Monthly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Quarterly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Annually Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Other Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

## Message details

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| <DriverID> | Unique ID for the Driver – same as Emirati ID | 999-9999-9999999-9 |
| <DriverState> | Code as above | Code |
| <Date> |  | Date |
| <Time> |  | Time |
| <VehicleID> | Vehicle is use by driver | String |

## Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
|  |  |  |

# Passenger Message

When a passenger registers on the CarTrack app then their details should be sent to AjmanGO in order for it to have sufficient details of each registered passenger.

## Interface Frequency and Schedule

Immediate (real-time)

On demand

Hourly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_ Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Daily Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Weekly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Monthly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Quarterly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Annually Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Other Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

## Message details

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| <Action>  Values: ADD or MOD | Defines whether this is a new passenger or whether it is an update for an existing passenger. | String |
| <ID> | Unique identifier for Passenger.  Note: for flag down fares there will be a generic PassengerID of “FlagDown” | String |
| >FirstNameAR< | AR or EN or Both? | String |
| >LastNameAR< |  | String |
| >FirstNameEN< |  | String |
| >LastNameEN< |  | String |
| >MobileNumber< | Held in International format 00971… | Numeric |
| >EmailAddress< |  | Email |

## Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
| Agreed | Unique ID | Does the field shown on the “Cartaxi Portal to CRM Fields” tab in the supplied spreadsheet on row9 relate to a unique passenger ID?.  This relates to Single Sign On. |
|  |  |  |

# Trip Message

CarTrack is the master system for all operational data for the taxi network. This includes managing and booking taxi trips etc.

In order for AjmanGO to report and perform certain operational activities it is important that details of every completed trip is passed in a timely fashion to AjmanGO.

Once a passenger trip is completed then CarTrack will inform AjmanGO of the details of that trip.

The following details are held by CarTrack for each trip:

a.       booking\_id

b.      booking\_state\_id

c.       company\_code

d.      passenger\_id

e.      driver\_id

f.        vehicle\_id

g.       pickup\_address

h.      pickup\_lat

i.         pickup\_lon

j.        pickup\_time,

k.       destination\_address

l.         destination\_lat

m.    destination\_lon

n.      estimated\_trip\_fee

o.      estimated\_trip\_distance

p.      estimated\_trip\_duration

q.      additional\_instructions

r.        created\_ts

s.       updated\_ts

t.        reassigned

u.      reassigned\_reason

v.       taxi\_type\_id,

w.     number\_of\_passengers

x.       proximity\_notification\_sent

y.       driver\_eta\_distance

z.       driver\_eta\_duration

aa.   driver\_eta\_updated\_ts

Those items marked in grey are of interest to AjmanGO and are included in the proposed message structure.

## Interface Frequency and Schedule

Immediate (real-time)

On demand

Hourly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_ Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Daily Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Weekly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Monthly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Quarterly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Annually Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Other Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

## Message details

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| <ID> | Unique identifier for Trip. This is the Booking ID in CT | String |
| >PassengerID< | Either a reference to a registered passenger or else “Anonymous” denoting a non-registered passenger | String |
| >VehicleID< |  | String |
| >DriverID< |  | String |
| >PickupDateTime< |  | Date/Time |
| >PickupAddress< |  |  |
| >PickupLon< | Longitude |  |
| >PickupLat< | Latitude |  |
| >DestinationAddress< |  |  |
| >DestinationLon< | Longitude |  |
| >DestinationLat< | Latitude |  |
| >Waypoints< |  |  |
| >NumPassengers< |  | Integer |
| >TaxiType< |  | Code |

## Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
| Agreed | Trip ID | Does CarTrack have a suitable unique ref number for each trip. Yes it is the Booking ID. |
| Agreed | Locations | Are the locations (Pickup, Destination and Waypoints) structured or freeform data. These are in “Google Places API" format. |
| Agreed | TaxiType | Is this info not already available by virtue of the fact that a driver is assigned to a vehicle at the start and end of his shift? |
|  | driver\_eta\_updated\_ts | Is this the updated time of arrival. How are trips across midnight handled. |
|  | Number Passengers | Is this information available |

# Incident Message

CarTrack is the master system for all operational data for the taxi network. This includes managing and booking taxi trips etc.

In order for AjmanGO to report and perform certain operational activities it is important that details of every driver complaint / incident is passed in a timely fashion to AjmanGO.

When an incident report is submitted to CarTrack the details will be passed to AjmanGO.

At the moment the only incident recorded is when the driver presses the emergency button on his tablet and then the controller rings back to find out what the problem is. The below message therefore has the details provided as recorded in CarTrack.

## Interface Frequency and Schedule

Immediate (real-time)

On demand

Hourly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_ Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Daily Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Weekly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Monthly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Quarterly Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Annually Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

Other Day: \_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_\_Push/Pull: \_\_\_\_\_\_\_\_\_\_\_

## Message details

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Type |
| >DriverID< |  | String |
| >DateTime< | Date and time of incident | Date/Time |
| >Notes< |  | String |

## Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
| Agreed |  | Please provide details of info available. |
|  |  |  |

# General Queries

|  |  |  |
| --- | --- | --- |
| Resolved | Item | Query |
| Agreed | Enable /  Disable | This field is shown on the “Cartaxi Portal to CRM Fields” tab in the supplied spreadsheet on row9. What does this relate to – is it a unique reference for a registered passenger. |
| Agreed | Assign a Driver | This field is shown on the “Cartaxi Portal to CRM Fields” tab in the supplied spreadsheet on row34. What does this relate to – if it relates to the allocation of a booking to a specific driver then this is not required as the trip summary contains sufficient information. |
|  | Access to CarTrack Booking capability from within the AjmanGO portals | It will be necessary from the AjmanGO portal used within the call centre for the operators to access both AjmanGo and CarTrack systems. This needs to be done in a seamless manner. |
|  | Single Sign In | How can single sign on allow for the above |