MOT Testing System API

Guidelines

for connecting MOT Test Equipment

The MOT Testing System (MTS) has an Internet of Things (IoT) application programming interface (API) to allow measured testing equipment to send measured results in directly negating the need for manual keyed entry. The API currently has the capability to accept results from Roller Brake Testers, Decelerometers, Exhaust Gas Analysers and Diesel Smoke Meters and further capability is also planned.

MTS is hosted in the public cloud on Amazon Web Services (AWS) and as such makes use of the AWS IoT core functionality for its equipment API. At a high level there are currently two ways to connect MOT test equipment; the first approach uses a single client to act as a workshop manager. The workshop managers responsibility is to connect to each piece of equipment, collect its test results and send them to MTS in the prescribed JSON format. It will also receive messages from MTS in the same JSON fomat and send this data to the equipment, after first converting to a format they understand (where appropriate). This approach would be most suitable where all equipment is supplied from the same manufacturer or has the same internal API.

A screenshot of a cell phone

Description automatically generated

The other approach is for each piece of equipment to have its own MTS API client. This will perform that same conversion between the DVSA JSON format and the format that the equipment understands. The difference being that the client in this approach would only be responsible for one piece of equipment. This approach would be suitable for use in a test centre where equipment was supplied by different manufacturers and do not all have the same internal api.

A screenshot of a cell phone

Description automatically generated

It is possible for the above two approaches to be combined with some equipment connected directly and the rest through a workshop manager.

The MTS API is implemented in AWS IoT core and using the MQTT protocol with X.509 client certificates for authentication. Amazon produce software development kits (SDK) for many programming languages and it is recommended that these are used to implement the connectivity to the MTS API in a programming language of your choice.

The MTS API uses a publish and subscribe model with each site having its own MQTT topics, one for sending and another for receiving messages. The messages are all in JSON format and are described in the appendices below.

When the client starts up it should establish an MQTT connection and subscribe to its \*\*/in topic. This will be used to receive work order messages from MTS every time an MOT test begins. The data in this work order will contain the site number, the current mot test number and details of the vehicle under test.

Once the equipment has performed its measured test, the results should be added to the received work order and published back to MTS on the \*\*/out topic. This api is idempotent so messages can be sent any number of times.

A reference implementation of an API client has been written in Java and is also used by the MTS development team for their internal development and testing. This can be found at <https://github.com/bennetts101/dvsa-mot-iot-client-simulator>

Once a client has been written this will need to be tested by DVSA or GEA so that approvals can be issued for use in MOT tests.

Appendix 1

Brake Test Data

**WorkOrder (Response)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Mandatory / Optional** | **Data Type** | **Values** |
| siteNumber | M | String |  |
| testNumber | M | String |  |
| vehicle | M | Vehicle |  |
| brakeTestResult | M | BrakeTestResult |  |

**Vehicle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Mandatory / Optional** | **Data Type** | **Values** |
| make | M | String |  |
| model | M | String |  |
| vrm | M | String |  |
| vin | M | String |  |
| firstRegistrationDate | String | YYYY-MM-DD |  |
| fuelType | String | PETROL | DIESEL |  |
| vehicleClass | String | 1 - 7 |  |

**BrakeTestResult (Roller Brake Tester)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Mandatory / Optional** | **Data Type** | **Values** |
| serviceBrakeTestType | M | String | ROLLER |
| parkingBrakeTestType | M | String | ROLLER |
| vehicleWeightType | O | String | PRESENTED | MANUFACTURER |
| vehicleWeightKg | O | Number |  |
| serviceBrakeEfforts | M | Array [BrakeEffort] |  |
| parkingBrakeEfforts | M | Array [BrakeEffort] |  |

**BrakeTestResult (Decelerometer)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Mandatory / Optional** | **Data Type** | **Values** |
| serviceBrakeTestType | M | String | DECELEROMETER |
| parkingBrakeTestType | M | String | DECELEROMETER |
| serviceBrakeEfficiency | M | Number |  |
| parkingBrakeEfficiency | M | Number |  |

**BrakeEffort (Roller Brake Tester)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Mandatory / Optional** | **Data Type** | **Values** |
| axleNumber | M | Number | 1 – (number of axles) |
| nearside | M | Number |  |
| nearsideLock | M | Boolean | true | false |
| offside | M | Number |  |
| offsideLock | M | Boolean | true | false |

The data should be supplied in JSON format similar to the example below: -

{

"siteNumber": "V123456",

"testNumber": "1234567890",

"vehicle": {

"make": "Ford",

"model": "Fiesta",

"vrm": "ABC 123",

"vin": "1234567890123456",

“firstRegistrationDate”: “2019-01-31”,

“fuelType”: “PETROL”,

“vehicleClass”: “4”

},

"brakeTestResult": {

"serviceBrakeTestType": "ROLLER",

"parkingBrakeTestType": "ROLLER",

"vehicleWeightType": "PRESENTED",

"vehicleWeightKg": 1500,

"serviceBrakeEfforts": [

{

"axleNumber": 1,

"nearside": 600,

"nearsideLock": false,

"offside": 600,

"offsideLock": false

},

{

"axleNumber": 2,

"nearside": 400,

"nearsideLock": false,

"offside": 400,

"offsideLock": false

}

],

"parkingBrakeEfforts": [

{

"axleNumber": 2,

"nearside": 300,

"nearsideLock": false,

"offside": 300,

"offsideLock": false

}

]

}

}