improving public transport through technology

CMS to PID Interface Protocol

Part 2 – Core Content Messages

RTIG Library Reference: **RTIGT047-pt2-2.0.1**

June 2025

Availability: Public

List of contents

Status of this document

This document is Published.

If there are any comments or feedback arising from the review or use of this document, please contact us at secretariat@rtig.org.uk

1 Introduction

1.1 About this document

1.1.1 This document has been produced for the Real Time Information Group (RTIG) in conjunction with Transport for Wales.to support the standardisation of the interface between content management systems and displays.

1.1.2 This document is Part 2 covering Common Data Structures and Core Content Messages (applicable to all display types) and is part of a series of documents covering different aspects of the standard.

1.2 Background

1.2.1 Transport for Wales (TfW) would like to specify a standard interface between the Content Management System (CMS) and real time information Displays, that suppliers would need to comply/work with to enable TfW to procure a single CMS that can interface to multiple displays from a number of vendors.

1.2.2 The standard will specify the minimum capability that is to be expected of all displays supported through the interface (i.e. be able to represent real time vehicle arrival/departure information, text-based messages and hold the scheduled timetable for at least that day's services).

1.2.3 The interface will cater for the following:

- Basic text-based displays
- Graphical displays - in addition to the minimum capability, also be able to provide additional information such as weather, news feeds, advertising, information videos etc.
- Off grid displays - these will not have ready access to power and may not have significant data bandwidth available to show graphical content.

1.2.4 The interface should also cater for fault management data to be passed back to the CMS to enable monitoring and fault rectification.

1.2.5 Initial workshops were held (online) in September 2021 to build a high-level understanding among industry practitioners of what Transport for Wales would like to achieve.

1.2.6 This document is one of a series of documents detailing a standard CMS-to-PID interface.

1.3 Governing Principles

1.3.1 This document has been drafted with the following principles in mind:

1.3.2 Abstraction – "*no need to re-invent the wheel*".

- Lower-level concepts should be abstracted away using existing standards wherever possible; this document focuses solely on the application-level detail.

1.3.3 Clarity – "*grey areas should be minimised*".

- Committee-designed standards often evolve to support multiple different mechanisms of achieving the same result, leading to grey areas in compatibility between products that can all legitimately claim to support the standard – although in many cases, interoperability is limited.

1.3.4 Simplicity – "*less is more*".

- The more complex the interface protocol becomes, the less likelihood there is of industry-wide uptake. Conversely, by keeping the rules to a bare minimum, we hope to encourage wider adoption and compatibility by PID and CMS suppliers.

1.3.5 In summary, it is the intention that this specification should be as lightweight and as high-level as possible. As the project progresses, this document will be updated to reflect the latest status of the interface design.

1.4 Scope

1.4.1 The CMS to PID interface protocol includes:

1.4.1.1 Messages to support core public transport timetable and real time information used across all display types.

1.4.1.2 Messages to support formatting text and graphical content handling.

1.4.1.3 The communications infrastructure (i.e. network technology and protocols) that will be used to support communication between the CMS and the PIDs.

1.4.1.4 Security requirements of any parts of the network that rely on the public internet.

1.4.1.5 Multi-vendor support – i.e. the use of PIDs from multiple suppliers connected to the CMS by the same communications infrastructure.

1.4.1.6 A “discovery” process through which a PID can connect to a management service and receive configuration details. Typically, this will only happen during the initial commissioning phase, or when a PID is replaced or switched due to hardware failure.

1.4.1.7 Specification of message structure and content for managing timetabled departures, estimated departures and departure cleardowns.

1.4.1.8 “Heartbeat” messaging and general fault reporting.

1.4.2 The following elements are considered to be out of scope of the current project:

1.4.2.1 Detailed discussions about network security on private IP-based APNs or VPNs. Where an implementation requires the use of the public internet, it is recommended that the use of TLS is specified as a project requirement by the contracting body, to ensure end-to-end security and prevent “man-in-the-middle” and “denial-of-service” style attacks by malicious actors.

1.4.2.2 Integration of legacy or low-power devices – where legacy or low-power PIDs rely on a specific communications infrastructure (e.g. PMR) or a propriety messaging protocol (e.g. binary data packets) it is anticipated that a bidirectional “gateway” device could be used to propagate inbound and outbound messages between the CMS and the PID. It is recommended that where such gateway devices are to be used, the contracting body should specify the maximum acceptable latency that can be introduced by such devices as a project requirement.

1.4.2.3 Time synchronisation – it is expected that clock synchronisation between the CMS and PIDs should be managed using NTP or a similar service, and that maximum permitted “drift” between devices should be specified as a project requirement by the contracting body.

1.4.2.4 Hardware specification for PIDs – neither the physical characteristics of the customer-visible display nor the technical characteristics of the embedded computer are within scope of this specification. It is anticipated that the shape, size and dot pitch/resolution of the display will be subject to the project requirements of the contracting body, and that the specification of the embedded computer will be left to the discretion of the PID hardware supplier.

1.5 Existing Standards

1.5.1 Whilst SIRI provides a number of relevant message types for CMS-to-PID communication, it does not fully meet the requirements of this project. Where possible, in line with the governing principles, existing message types will be evaluated carefully and reused wherever possible.

1.6 Document Structure

1.6.1 To minimise versioning complexity, the interface protocol is split into multiple parts:

1.6.1.1 Part 1

- Communications Infrastructure
- Network Architecture

1.6.1.2 Part 2 (this document)

- Common Data Structures
- Core Content Messages (applicable to all display types)

1.6.1.3 Part 3

- Additional Message Types for Graphical Displays
- Additional Message Types for Low Power Displays

1.6.1.4 Part 4

- Additional Services (e.g. Fault Management, Reporting)

1.6.1.5 Part X

- Additional requirements yet to be defined

1.7 Limitations and The Future

1.7.1 This report reflects the available technology and practices which have been found to be effective at the date of publication. However, technology and its applications are evolving, and it is therefore probable that new technologies, new developments of existing technologies, and new ways to adopt and utilise them will evolve along with new business requirements.

1.7.2 Procuring authorities are encouraged to consider new requirements and approaches bearing in mind the standard promoted in this document. Where a new business requirement or technology is developed that is not supported by this standard, we encourage you to contact RTIG to discuss how we update the standard to ensure it remains fit for purpose.

1.8 Acknowledgements

1.8.1 RTIG is grateful to Transport for Wales for funding the project that initially developed this standard.

1.8.2 We are grateful to the members of the technical working group who have contributed to its development including Vix Technology, Journeo, r2p Systems UK, Elydium, Transport for London and ITxPT.

2 Common Data Structures

2.1.1 The following reusable data structures relate to low-level data objects. They do not constitute message types in their own right; they are used within multiple message types across multiple service types.

2.1.2 While there is nothing preventing the extension of these data structures through the addition of further attributes, this is not recommended.

2.2 Multi Language Text

2.2.1 The interface supports multiple languages for all text that can be displayed on a PID using the following JSON structure:

```
"multiLanguageText": [
 {
   "lang": "EN",
   "text": "Some text in English"
 },
 {
   "lang": "CY",
   "text": " Rhywfaint o destun yn Saesneg"
 },
 ...
]
```

2.2.2 Each text attribute is an array of tuple objects containing an ISO 639-1 two-character language code (“lang”) and a text string (“text”).

2.3 External Reference

2.3.1 Wherever references to external data sets are used, it is proposed that they are encoded using the following JSON structure that encapsulates the data source along with the reference code:

```
"externalRef": {
   "source": "DataSetName:ObjectName.AttributeName",
   "ref": "UniqueIdentifierWithinDataSet"
}
```

2.3.2 Whilst this interface specification should not be regarded as being “UK only”, where it is used for bus data within the UK, the following “business rules” should be applied:

2.3.2.1 Where an “operatorRef” is used in UK message content, this should be the National Operator Code of the operator running a given service, as defined by Traveline (see https://www.travelinedata.org.uk/traveline-open-data/transport-operations/about-2/). Where the data applies to a bus service being operated in England, the National Operator Code should appear as it is recorded in the DfT’s Bus Open Data Service (BODS) for any given trip, for other administrations this will be the locally agreed code. For example:

```
{
 "source": "NOC:NOC",
 "ref": "FSYO"
}
```

2.3.2.2 Where a “locationRef” is used in UK bus message content, this should be an ATCO Code defined in the NaPTAN database (see https://www.gov.uk/government/publications/national-public-transport-access-node-schema). For example:

```
{
 "source": "NaPTAN:AtcoCode",
 "ref": "490000077E"
}
```

Or for rail in the UK it is expected that this will be the CRS code for a rail station.

2.3.2.3 Where a “lineRef” is used in message content relating to bus services in England, this should correspond with the TransXChange Line id attribute used in BODS, for other administrations this will be the locally agreed code:

```
{
 "source": "BODS:Service.Line.id"
 "ref": "FSYO:PB0002307:141:X5"
}
```

2.3.2.4 Where a “directionRef” is used in message content relating to bus services in England, this should match the TransXChange direction used for a given vehicle journey in BODS for other administrations this will be the locally agreed code:

```
{
 "source": "BODS:VehicleJourney.Direction",
 "ref": "outbound"
}
```

2.3.2.5 Where a “vehicleJourneyRef” is used in message content, this should match the TransXChange vehicle journey id used for a given vehicle journey in BODS for other administrations this will be the locally agreed code:

```
{
 "source": "BODS:VehicleJourney.id",
 "ref": "VJ1"
}
```

2.3.3 Similar sets of “business rules” are likely to emerge for other modes and data standards (e.g. UK rail references to TIPLOC or DARWIN data sets, European references to NeTEx data, references to GTFS data); these should be formalised and applied in a similar manner.

2.4 Vehicle Journey Definition

2.4.1 An individual timetabled journey should be described using the following structure:

```
"vehicleJourneyDefinition": {
   "lineRef": {             // external reference
     "source": "BODS:Service.Line.id"
     "ref": "ABCB:PB0009999:111:123A"
   },
   "directionRef": {        //external reference
     "source": "BODS:VehicleJourney.Direction",
     "ref": "outbound"
   },
   "vehicleJourneyRef": {   // external reference
     "source": "BODS:VehicleJourney.id",
     "ref": "VJ1"
   },
   "operatorName": [        // multi-language text
     {
       "lang": "EN",
       "text": "The ABC Bus Company"
     }
   ],
   "operatorRef": {         // external reference
     "source": "NOC:NOC",
     "ref": "ABCB"
   },
```

```
    "publishedLineName": [   // multi-language text
      {
        "lang": "EN",
        "text": "123A"
      }
    "marketingLineName": [   // multi-language text
        {
          "lang": "EN",
          "text": "South Coast Express"
        }
    ],
    “originRef": {           // external reference
      "source": "NaPTAN:AtcoCode",
      "ref": "490000077E"
    },
    "originName": [          // multi-language text
      {
        "lang": "EN",
        "text": "Highbury"
      }
    ],
    "destinationRef": {      // external reference
      "source": "NaPTAN:AtcoCode",
      "ref": "490000054B"
    },
    "destinationName": [     // multi-language text
      {
        "lang": "EN",
        "text": "Paradise Park"
      }
    ],
    "via": [                 // multi-language text
      {
        "lang": "EN",
        "text": "City Centre, Park and Ride North"
      }
    ],
    "poi": [                 // multi-language text
      {
        "lang": "EN",
        "text": "Castle Ruins, All Saints Church"
      }
    ]
}
```

2.5 Location Configuration

2.5.1 The location configuration data structure holds details of the localities, clusters and stop to which a PID belongs, together with the public-facing location name and geographic coordinates:

```
"locationConfig": {
  "localities": [],
  "clusters": []
  "stops": [],
  "locationName": [ // multiLanguageText object
    {
      "lang": "EN",
      "text": "Cardiff Central"
    },
    {
      "lang": "CY",
      "text": "Caerdydd Canolog"
    }
  ],
  "geographicCoordinates": {
    "longitude": -2.05467,
    "latitude": 53.53774
  }
}
```

2.5.2 N.B. The arrays of localities, clusters and stops within the locationConfig are not multi-lingual as they are purely used for creating MQTT topic subscriptions; they are not used for public-facing display purposes.

2.6 Text Display Configuration

2.6.1 The text display configuration data structure holds details of available character encodings and fonts, the number of lines available for text display, the number of characters available per text display line and a flag indicating whether smooth scrolling is supported:

```
"textDisplayConfig": {
 "charEncodings": [  // array of string values
   ...
 ],
 "fonts": [          // array of string values
   ...
```

```
],
"lines": 3,          // integer value
"charsPerLine": 26,  // integer value
"canScroll": true,     // boolean value
}
```

2.6.2 Where no character encodings are specified, UTF-8 will be the default.

2.6.3 Font configuration primarily provides support for graphical displays and may be left as an empty array for text displays with no explicit font support.

2.6.4 .

2.7 Status Report

2.7.1 This provides basic information about a display's operational status.

```
"statusReport": {
 "firmwareVersion": "XYZ.789.001.A",
 "operatingSystem": "Homebrew Linux 3.11",
 "upSince": "2020-11-17T06:30:00-00:00",
 "memoryUsage": "63%",
 "loadAverage": "78%",    // one minute load average
 "diskFreeSpace": "7.6Gb"
 }
```

2.7.2 Vendors are free (and encouraged) to extend the status report structure to include further information as appropriate.

3 Common Enumerated Types

3.1 Reusable structures

3.1.1 The following reusable data structures relate to low-level data objects. They do not constitute message types in their own right; they are used within multiple message types across multiple service types.

3.2 Device Type

```
[
 CMS,
 DS,     // discovery service
 VSS,    // vendor-specific service
 PID-LED,
 PID-LCD,
 PID-TFT,
 PID-EPAPER
]
```

3.3 Display Status Type

3.3.1 Display instructions appearing within messages should always correspond to one of the following enumerated status values:

```
[
 OFF,          // display switched off (i.e. screen blank)
 SCHED_DEP,    // display should only show scheduled departures
 RT_DEP,       // display should only show real-time departures
 MIXED_DEP,    // display should show sched and r/t departures
 SCHED_ARR,    // display should only show scheduled arrivals
 RT_ARR,       // display should only show real-time arrivals
 MIXED_ARR,    // display should show scheduled and r/t arrivals
 SCHED_ARRDEP,// display should show sched arr and dep
 RT_ARRDEP,    // display should show real-time arr and dep
 MIXED_ARRDEP,// display should show sched and r/t arr and dep
 NO_DATA       // screen remains on – e.g. to display clock
]
```

3.4 Sensor Type

3.4.1 Information messages can be displayed in a variety of different ways:

```
[
 INT-TEMP, // internal temperature
 EXT-TEMP, // external temperature
 SHOCK,
 GPS,
 CELL-DATA,
 LIGHT-LEVEL
]
```

3.4.2 This list provides for some common sensor types contained within displays. Additional sensor types may be defined by vendors in other parts for example for management or accessibility.

3.5 Real-Time Status

3.5.1 Real-time departure messages can be displayed in a variety of different ways:

```
[
 ON-TIME,
 EARLY,
 LATE,
 DELAYED,
 CANCELLED
]
```

3.6 Message Display Style

3.6.1 Information messages can be displayed in a variety of different ways, this is managed having a Target Area and a Behaviour:

3.6.1.1 Message Target Area

```
[
 BOTTOM_LINE,
 FULL_SCREEN, //overrides all other content on display
 OVERLINE //full screen message leaves bottom line available for
other messages.
]
```

3.6.1.2 Message Behaviour

```
[
 SCROLL, // default
 PAGED,
 STATIC
]
```

```
Deprecated DisplayStyle to be removed in next version.
[
 SCROLL, //bottom line
 PAGED, // bottom line
 STATIC, // bottom line
 FULL-SCREEN, //overrides all other content on display
 OVERLINE, // full screen message leaves bottom line available
 for other messages.

]
```

3.7 Message Action

3.7.1 Messages can be managed in a variety of different ways:

```
[
 DELETE,
 REPLACE
]
```

3.8 Message Priority Type

3.8.1 Messages can be given a priority of either:

```
[
 STANDARD,
 EMERGENCY
]
```

3.8.2 Where a message is of priority STANDARD this shall be enhanced with messageActionPriority which shall be a positive integer, where 1 is the highest priority and the lowest priority being 99.

3.8.3 This approach reflects the complex nature of business requirements for message management whilst retaining as much simplicity as possible.

3.8.4 Where messages are received by the CMS from a third party source using SIRI SX and the ActionPriority element is set this should be mapped to messageActionPriority.

3.9 Cleardown Reason

3.9.1 Cleardown messages should specify the reason that a departure is being cleared down:

```
[
 DEPARTED,
 CANCELLED,
 SHORT_WORKING,
 DIVERTED,
 ARRIVED
]
```

3.10 Occupancy Level

3.10.1 Occupancy levels are taken from RTIGT043, and are based on the values available in SIRI v2.1 (which has itself been aligned with GTFS-RT):

```
[
 FULL,
 STANDINGAVAILABLE,
 SEATSAVAILABLE,
 UNKNOWN,
 EMPTY,
 MANYSEATSAVAILABLE,
 FEWSEATSAVAILABLE,
 STANDINGROOMONLY,
 CRUSHEDSTANDINGROOMONLY,
 NOTACCEPTINGPASSENGERS
]
```

3.11 Vehicle Mode

3.11.1 Vehicle Mode is taken from the SIRI v2.1 VehicleModesEnumeration:

```
[
 AIR,
 BUS,
 COACH,
 FERRY,
 METRO,
 RAIL,
 TRAM,
 UNDERGROUND
]
```

3.11.2 Where `vehicleMode` is not provided it should be assumed to be BUS.

3.12 Reason

3.12.1 Used in multimodal data:

```
[
 NEWDEPARTURE,
 DEPARTURECOMMENT,
 PLATFORM
]
```

3.13 Status

3.13.1 Used in delivery of content:

```
[
 DONE,
 NO_DATA
]
```

4 Core Service Messages

4.1 Unidirectional

4.1.1 The following message types are applicable to all display types. In each case these are unidirectional (i.e. single phase) message types and always have the third level of topic name set to the constant value service:

- Heartbeat
- Display Override
- Sensor Event

4.2 Heartbeat

4.2.1 Heartbeat messages are designed to be as small as possible. They should be published from the display using the topic:

```
{device_type}/heartbeat/service
```

4.2.2 They should carry the payload:

```
{
 "heartbeat": {
   "vendorId": "VENDOR0001",     // unique within project
   "deviceId": "ABCD1234567890", // unique for the vendor
   "timestamp": "2020-12-17T17:30:00-00:00"
   "vendorData": {
         "object1": 0,
         "object2": "0.2.0"
       }


 }
}
```

4.2.3 Where `vendorData` is provided this shall be minimised to control the volume of data transmitted and agreed by the implementing project to ensure that the CMS is able to support the inclusion of each object.

4.3 Display Override

4.3.1 Where the CMS wishes to override the current display status of a PID, it should publish a Display Override message to a topic that corresponds with one or more registered PIDs:

```
{displayType}/displayOverride/ /{vendor_id}/{device_id}
```

```
{
 "displayOverride": {
   "status": "OFF", // display status type enumerated value
   "timestamp": "2020-12-17T17:30:00-00:00"
 }
}
```

If a display override is active, it is expected that the displayOverride attribute in the status response from the display is set, so that the CMS is aware of the current displayOverride status.

4.4 Sensor Event

4.4.1 Modern PIDs often contain sensors that measure and monitor environmental conditions that can be reported back to the CMS – for example internal temperature, external temperature, shock detection, etc. In accordance with MQTT best practice, it is proposed that each sensor type should be given its own topic, thereby allowing subscribers to filter out messages from sensor types that are not relevant to them.

4.4.2 Sensor Event messages should use the following topic name:

```
{device_type}/sensorEvent/service/{type}
```

4.4.3 It is anticipated that the message content for the majority of sensor content will follow the following convention:

```
"sensorEvent": {
"vendorId": "VENDOR0001",      // unique within project
"deviceId": "ABCD1234567890",  // unique for the vendor
"type": "INT-TEMP",            // from sensor type enumeration
   "value": "24.7",
   "unit": "degreeCelsius",
   "data": {
     "vendor-specific-data-1": "value-1",
     "vendor-specific-data-2": "value-2",
     ...
```

```
  }
 }
```

4.4.4 At present, it is recommended that the sensor type is duplicated in the topic name and the message payload – allowing (future) devices and services to subscribe to sensor events of a particular type. In each case, the type should be taken from the sensor type enumeration.

4.4.5 N.B. The “data” attribute is designed to provide unlimited extensibility, depending on the requirements of PID vendors. It might be necessary to place some constraints on the content that is permitted here. This will require further discussion with PID vendors.

4.4.6 N.B. Specific event types for accessibility (e.g. Bluetooth beacon activations, physical push button requests, etc.) will be added in a later document part.

5 Core Content Messages.

5.1.1 The following message types are applicable to all display types. In each case these are unidirectional (i.e. single phase) message types and always have the third level of topic name set to the constant value content:

- Scheduled Departure
- Real-Time Departure
- Information Message

5.2 Scheduled Departure

5.2.1 Scheduled departure messages should use the following topic name structure:

```
CMS/scheduledDeparture/content/{locality_id}/{cluster_id}/{stop
_id}/{vendor_id}/{device_id}
```

5.2.2 Loosely based on the content of a SIRI-ST <TimetabledStopVisit> message, the following message structure has been expanded to include multi-language text and external references. In addition, support has been added for “via” text, points of interest and operator brand name:

```
{
 "scheduledDepartures": [
   {
     "departureId": "fc867e8a-4e9c-465d-847d-6bf2fb37977f",
     "vehicleMode": "BUS",
     "targetedVehicleJourney": { // vehicle journey definition

        "lineRef": {              // external reference
        "source": "BODS:Service.Line.id",
        "ref": "ABCB:PB0009999:111:123A"
      },
      "directionRef": {           // external reference
        "source": "BODS:VehicleJourney.Direction",
        "ref": "outbound"
      },
      "vehicleJourneyRef": {      // external reference
        "source": "BODS:VehicleJourney.id",
        "ref": "VJ1"
      },
      "operatorName": [           // multi-language text
        {
          "lang": "EN",
          "text": "The ABC Bus Company"
        }
      ],
```

```
"operatorRef": {            // external reference
  "source": "NOC:NOC",
    "ref": "ABCB"
},
"publishedLineName": [   // multi-language text
  {
    "lang": "EN",
    "text": "123A"
  }
],
"marketingLineName": [   // multi-language text
  {
    "lang": "EN",
    "text": "South Coast Express"
  }
],
"originRef": {          // external reference
  "source": "NaPTAN:AtcoCode",
  "ref": "490000077E"
},
"originName": [         // multi-language text
  {
    "lang": "EN",
    "text": "Highbury"
  }
],
"destinationRef": {     // external reference
  "source": "NaPTAN:AtcoCode",
  "ref": "490000054B"
},
"destinationName": [    // multi-language text
  {
    "lang": "EN",
    "text": "Paradise Park"
  }
],
"via": [                // multi-language text
  {
    "lang": "EN",
    "text": "City Centre, Park and Ride North"
  }
],
"poi": [                // multi-language text
  {
    "lang": "EN",
    "text": "Castle Ruins, All Saints Church"
  }
]
```

```
        },
        "targetedCall": {
          "visitNumber": "1",
          "quay": "7", // Platform, stand etc – Quay is the
Transmodel name
          "aimedArrivalTime": "2022-02-17T09:40:46-00:00",
          "aimedDepartureTime": "2022-02-17T09:42:47-00:00",
          "expectedArrivalTime": null,
          "expectedDepartureTime": null
          }
        },
        ...
    ]
   }
```

5.2.3 N.B. At the highest level, "scheduledDepartures" is defined as an array, allowing multiple scheduled departures to be sent in a single message payload. In practice it may be necessary to limit the number of scheduled departures in a single message to avoid excessive payload sizes (as these may be limited by MQTT brokers or bridges on the network).

5.2.4 expectedArrivalTime and expectedDepartureTime are nullable optional elements. If the element is null or not present, then this signifies that real time is not available. Do not populate with times if real time updates are not available. Real time updates should normally be provided using the realTimeDeparture topic.

5.2.5 An example of a rail departure would be:

```
{
 "scheduledDepartures": [
      {
       "departureId": "db654e8a-4e9c-465d-326b-6bf2fb379ab7",
       "vehicleMode": "RAIL",
       "targetedVehicleJourney": { // vehicle journey definition
       "lineRef": {              // external reference
         "source": "
         Darwin4Trains.Entity.ViewModels.TrainAmqViewModel",
         "ref": " DARWIN "
         },
         "directionRef": {           // external reference
           "source":
           Darwin4Trains.Entity.ViewModels.JourneyStatusLocatio
           nViewModel",
           "ref": "outbound"
           },
           "vehicleJourneyRef": {      // external reference
```

```
  "source":
  "Darwin4Trains.Entity.ViewModels.JourneyStatusLocati
  onViewModel",
  "ref": "VJ1"
},
"operatorName": [           // multi-language text
  {
    "lang": "EN",
    "text": "Transport for Wales"
  }
  {
    "lang": "CY",
    "text": "Trafnidiaeth Cymru"
  }
],
"operatorRef": {            // external reference
  "source": "NOC:NOC",
  "ref": "TFW"
},
"publishedLineName": [   // multi-language text
  {
    "lang": "EN",
    "text": "CDF - BYI"
  }
  {
    "lang": "CY",
    "text": "CDF - BYI"
  }
],
"marketingLineName": [   // multi-language text
  {
    "lang": "EN",
    "text": "Barry Island"
  }
  {
    "lang": "CY",
    "text": "Ynys y Barri"
  }

],
"originRef": {          // external reference
  "source":
  "Darwin4Trains.Entity.ViewModels.JourneyStatusLocati
  onViewModel",
  "ref": "CDF"
},
"originName": [         // multi-language text
  {
```

```
    "lang": "EN",
    "text": "Cardiff"
  }
  {
    "lang": "CY",
    "text": "Caerdydd"
  }
],
"destinationRef": {     // external reference
"source":
"Darwin4Trains.Entity.ViewModels.JourneyStatusLocation
ViewModel",
"ref": "BYI"
},
"destinationName": [    // multi-language text
  {
    "lang": "EN",
    "text": "Barry Island"
  }
  {
    "lang": "CY",
    "text": " Ynys y Barri "
  }
],
"via": [                // multi-language text
  {
    "lang": "EN",
    "text":  "Grangetown (1559),Cogan (1603),Eastbrook
    (1605),Dinas Powys (1607),Cadoxton (1611),Barry
    Docks (1614),Barry (1618)"
  }
  {
    "lang": "CY",
    "text":  "Grangetown (1559),Cogan (1603),Eastbrook
    (1605),Dinas Powys (1607),Tregatwg (1611),Dociau'r
    Barri (1614),Y Barri (1618)"
  }
],
"poi": [                // multi-language text
  {
    "lang": "EN",
    "text": "Barry Docks"
  }
  {
    "lang": "CY",
    "text": "Dociau'r Barri"
  }
]
```

```
},
"targetedCall": {
  "visitNumber": "1",
     "quay": "8", // Platform, stand etc – Quay is the
     Transmodel name
  "aimedArrivalTime": "2022-02-17T09:40:46-00:00",
  "aimedDepartureTime": "2022-02-17T09:42:47-00:00",
  "expectedArrivalTime": null,
  "expectedDepartureTime": null
  }
},
...
]
}
```

5.2.6 The amount of scheduled data published in advance will be a project-specific requirement, but it is strongly recommended that a minimum 24 hour lookahead is provided.

5.3 Real Time Departure

5.3.1 Real-time departure messages should use the following topic name structure:

```
CMS/realTimeDeparture/content/{locality_id}/{cluster_id}/{stop_
id}/{vendor_id}/{device_id}
```

5.3.2 Loosely based on the content of a SIRI-SM <MonitoredStopVisit> message, the payload of real-time departure messages has a lot in common with that of a scheduled departure message:

```
{
 "realTimeDeparture": [
   {
     "departureId": "fc867e8a-4e9c-465d-847d-6bf2fb37977f",
     "predictionTime": "2004-12-17T09:25:46-05:00",
     "vehicleMode": "BUS",
     "monitoredVehicleJourney": {  // vehicle journey definition
     "lineRef": {                  // external reference
       "source": "BODS:Service.Line.id"
       "ref": "ABCB:PB0009999:111:123A"
     },
     "directionRef": {             // external reference
       "source": "BODS:VehicleJourney.Direction",
       "ref": "outbound"
     },
```

```
  "vehicleJourneyRef": {        // external reference
    "source": "BODS:VehicleJourney.id",
    "ref": "VJ1"
  },
  "operatorName": [             // multi-language text
    {
      "lang": "EN",
      "text": "The ABC Bus Company"
    }
  ],
  "operatorRef": {            // external reference
    "source": "NOC:NOC",
    "ref": "ABCB"
  },
  "publishedLineName": [      // multi-language text
    {
      "lang": "EN",
      "text": "123A"
    }
  ],
  "marketingLineName": [   // multi-language text
    {
      "lang": "EN",
      "text": "South Coast Express"
    }
  ],
"originRef": {              // external reference
    "source": "NaPTAN:AtcoCode",
    "ref": "490000077E"
  },
  "originName": [             // multi-language text
    {
      "lang": "EN",
      "text": "Highbury"
    }
  ],
  "destinationRef": {         // external reference
    "source": "NaPTAN:AtcoCode",
    "ref": "490000054B"
  },
  "destinationName": [        // multi-language text
    {
      "lang": "EN",
      "text": "Paradise Park"
    }
  ],
  "via": [                    // multi-language text
    {
```

```
                "lang": "EN",
                "text": "City Centre, Park and Ride North"
              }
            ],
            "poi": [                    // multi-language text
              {
                "lang": "EN",
                "text": "Castle Ruins, All Saints Church"
              }
            ],
            "progressStatus": "ON-TIME", // enum value
            "occupancyLevel": "SEATS-AVAILABLE", // enum value
            "monitoredCall": {
                "monitored": "true"
                "vehicleAtStop": "true",
                "quay": "7", // Platform, stand etc – Quay is the
Transmodel name
                "aimedArrivalTime": "2022-02-17T09:39:00-00:00",
                "expectedArrivalTime":"2022-02-17T09:39:53-00:00:00",
                "aimedDepartureTime": "2022-02-17T09:40:00-00:00",
                "expectedDepartureTime":"2022-02-17T09:40:25-
                00:00:00",
                "departureStatus": "ON-TIME", // real-time status
                "arrivalStatus": "ON-TIME", // real-time status
              }
            }
          },
          ...
       ]
     }
```

5.3.3 Where monitored is false the expectedArrivalTime should be presented to the passenger.

5.3.4 Where no valid prediction is available then expectedArrivalTime and / or (as appropriate) expectedDepartureTime must not be provided or set to null, and monitored should be false.

5.3.5 Where no expected time was being provided and is subsequently provided this should trigger a prediction to be shown, and where expected time has been provided and is no longer provided it should trigger a prediction to be removed.

5.3.6 The departure id is used for three purposes:

- To match the prediction to the scheduled departure
- to replace earlier predictions that relate to the same departure

- to support rapid cleardown of the departure

5.4 Information Message

5.4.1 Information messages should use the following topic name structure:

```
CMS/informationMessage/content/{locality_id}/{cluster_id}/{stop
_id}/{vendor_id}/{device_id}/{message_id}
```

5.4.2 The inclusion of message_id allows for multiple messages to be set for any given device_id,

5.4.3 Information messages can be given different priorities and different display styles – so, for example, the supplied message text could be displayed as a low priority scrolling message about future disruption or a full-screen emergency message.

```
{
 "informationMessage": {
   "messageText": [
     {
       "lang": "EN",
       "text":  "Service 123A suspended between 10am and 5pm
       today"
     },
     {
       "lang": "CY",
       "text": " Gwasanaeth 123A wedi'i atal rhwng 10am a 5pm
       heddiw"
     }
   ],
   "earliestDisplayTime": "2022-02-17T10:00:00-00:00",
   "latestDisplayTime": "2022-02-17T17:00:00-00:00",
   "messagePriority": "STANDARD", //  message  priority  type
   enumerated value
   "messageActionPriority": 5, // positive integer 1 = high
   "messageTargetArea":  "SCROLL", //was  displayTargetArea  in
   previous versions
   "messageBehaviour": "SCROLL",      // message display style
   enumerated value, was displayStyle in previous versions
   "messageId": UUID,
   "messageAction": "DELETE" // optional action to delete
 }
}
```

5.4.4 There may be a requirement to delete an `informationMessage` from the display that has been sent previously. The optional attribute messageAction can be added to the `informationMessage` structure to replace or delete the message on the display with the messageId UUID provided when messageAction is set to DELETE only the messageId attribute is required. If a message with the provided UUID is not already present on the display, then no action should take place.

5.5 Journey Message

5.5.1 Journey messages should use the following topic name structure:

```
CMS/journeyMessage/content/{locality_id}/{cluster_id}/{stop_id}
/{vendor_id}/{device_id}/{departure_Id}
```

5.5.2 The inclusion of departure_Id allows for a message to be targeted for a specific journey (departure) from a specific (original) stop.

5.5.3 Journey messages can be given different priorities and different display styles – so, for example, the supplied message text could be displayed as a low priority scrolling message, a static or paged message.

```
{
"journeyMessage": {
 "messageText": [
   {
     "lang": "EN",
     "text": "Cancelled"
   },
   {
     "lang": "CY”,
     "text": "Canslo"
   }
   ]
   "earliestDisplayTime": "2022-02-17T10:00:00-00:00",
   "latestDisplayTime": "2022-02-17T17:00:00-00:00",
   "messagePriority": "STANDARD", //  message  priority  type
   enumerated value
   "messageActionPriority": 5, // positive integer 1 = high
   "messageBehaviour": "SCROLL",      // message display style
   enumerated value, was displayStyle in previous versions
   "messageId": UUID,
   "messageAction": "DELETE" // optional action to delete
 }
}
```

5.5.4 There may be a requirement to delete an `journeyMessage` from the display that has been sent previously. The optional attribute messageAction can be added to the `journeyMessage` structure to replace or delete the message on the display with the messageId UUID provided when messageAction is set to DELETE only the messageId attribute is required. If a message with the provided UUID is not already present on the display, then no action should take place.

6 Bidirectional Communication Messages

6.1 Bidirectional

6.1.1 The protocol defines a number of scenarios which require bidirectional request/response style communication patterns:

6.1.1.1 Display Capability

- Request from any device (but usually CMS)
- Response from PID

6.1.1.2 Device discovery

- Initiated by PID
- Response from a discovery service (could be CMS or vendor-specific service)
- Acknowledged by PID

6.1.1.3 Departure cleardown

- Requested by CMS
- Acknowledged by PID

6.1.1.4 Status reporting

- Request from any device (but usually CMS)
- Response by PID

6.1.1.5 Display Snapshots

- Request from any device (but usually CMS)
- Response by PID

6.2 Display Capability

6.2.1 A display capability request will typically be sent by a CMS to a single PID but could also be sent by another service on the MQTT network and/or sent to multiple PIDs using a topic name constructed as follows:

```
{device_type}/displayCapability/request/{vendor_id}/{device_id}
```

6.2.2 The message payload simply contains a request id:

```
{
 "displayCapabilityRequest": {
   "requestId": "4e7580d3-a6df-480b-b837-f8ad160f5a5a"
 }
}
```

6.2.3 The response will be broadcast by the PID to any display capability topic subscribers using the topic name:

```
{device_type}/displayCapability/response
```

6.2.4 The message payload includes the PID's vendor and device ids, a copy of the request id (so that the response can be matched to the request – or ignored by any other devices that subscribe to display capability responses) and a display configuration object.

```
{
 "displayCapabilityResponse": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId": "4e7580d3-a6df-480b-b837-f8ad160f5a5a",
   "rtigVersion": "2", // optional,if not present 1 assumed
   "textDisplayConfig": { // text display config
     "charEncodings": [   // array of string values
       "UTF-8"
     ],
     "lines": 3,          // integer value
     "charsPerLine": 26,  // integer value
     "canScroll": true    // boolean value
     "canPage": false, // boolean value
     "maxScrollChar": 150, // integer value

   }
 }
}
```

6.2.5 The elements `canScroll` and `canPage` describe the capability of a display. `maxScrollChar` is the maximum message length which is capable of being scrolled by the display.

6.3 Device Discovery

6.3.1 The interface supports and promotes the concept of “zero configuration” PID installations using a device discovery procedure. There are two pre-conditions:

6.3.1.1 Each PID must have a unique device identifier and a vendor identifier.

6.3.1.2 Where PSK-level security is enabled on the MQTT broker, each PID must have knowledge of the pre-shared key that will allow it to communicate over the MQTT network; where HTTPS is enabled on the MQTT broker, each PID must have the capability to process Transport Layer Security (TLS).

6.3.2 To carry out configuration automatically using the device discovery procedure, it is proposed that at first power up, the PID should send a message announcing its arrival on the MQTT network with the topic:

```
{device_type}/discovery/request
```

6.3.3 with a simple payload containing its unique vendor and device identifiers and a discovery request id:

```
{
 "discoveryRequest": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "deviceModel": "ABC DISPLAY", // optional string with model
   "requestId": "1ca491a6-3355-462e-81b5-cac9bbf4941c",
   "tenantId": "UUID" // optional identify customer tenant
 }
}
```

6.3.4 Where an encrypted device discovery process is being used the discovery request payload has an additional attribute to pass the public key of the display specific key to the CMS. This is used to pass security sensitive credential information back to the display via a RSA or EC public key encryption scheme. It can also be potentially used to share a symmetric signing key.

```
displayPublicKey: "string representing the PEM format public key"
```

6.3.5 The displayPublicKey is RSA 2048 bit and transmitted as a CSR in PEM format, without the begin and end lines, and as one string with no line breaks. When creating the CSR set the OU=vendorId/CN=deviceId in the X509 name attributes so that it can be identified easily should it be necessary.

6.3.6 New versions of this specification will update key length and type as required.

6.3.7 A discovery server on the network (this could be the CMS or the PID supplier's own discovery service) should then respond with a message with the topic:

```
{device_type}/discovery/response/{vendor_id}/{device_id}
```

6.3.8 The `vendorId` identifier should be the name of display manufacturer. This should be all capitals and be letters and / or numbers, not contain spaces but underscores are allowed.

6.3.9 The use of deviceId reduces data volumes as without it all devices will receive discovery data they are not interested in.

6.3.10 The response contains a unique response id in addition to the original request id:

```
{
 "discoveryResponse": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId": "1ca491a6-3355-462e-81b5-cac9bbf4941c",
   "responseId": "1bb1caf4-6866-441d-a946-83cf830fdc6d",
   "locationConfig": {  // locationConfiguration object
     "localities": [],
     "clusters": []
     "stops": ["4900128738"],
     "locationName": [  // multiLanguageText object
       {
         "lang": "EN",
         "text": "Cardiff Central"
       },
       {
         "lang": "CY",
         "text": "Caerdydd Canolog"
       }
     ]
   },
   "vendorConfig": {
     // vendor-specific object
     // this could be provided via the CMS or directly
     // by the vendor's own separate discovery service
   }
 }
}
```

6.3.11 The `discoveryResponse` message has an optional attribute to pass to the device the URL for a file containing a CA bundle file in PEM format.

6.3.12 When the client connects to the MQTT broker it should initially suppress CA validation and ignore any expired or invalid CAs. Once successfully connected a replacement CA bundle shall be provided. This attribute is used to communicate to the device the authorised CAs that should be downloaded and installed, prior to opening a TLS connection with the username/password or client certificate.

```
caBundle: "https://s3.cms-supplier.com/ca-bundle.pem"
```

6.3.13 The `discoveryResponse` message has an additional optional attribute to pass the username and password for the MQTT connection of the display to the CMS, because we need to know this to be able to connect with the appropriate credentials for MQTT topic security. The content of this is [user,pass] encrypted using the public key extracted from the `displayPublicKey` CSR and encoded as base64.

6.3.14 The CMS should only return one of `mqttUserPass` or `mqttClientCertificate`. Which one is used, will be a project specific requirement.

```
mqttUserPass: "nZDLA6/eN8jH3tWQg7x0X4CyxrMBohSJHRK"

or

mqttClientCertificate:
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2RwFFVfEcv1biTouMfPKIYk
awjxNEPilRn7WKuQTsOXoVw0kWyWJqnh++H7ht1b7QckgkoSNvbmZvlboNBAmRLiStf
gg6U4cOKZzFJrdPytgt6eCnZDLA6/eN8jH3tWQg7x0X4CyxrMBohSJHRK1bg86080Kz
e1M9muHITRsqNKSv9xv5l1F0ce1OsAVv+FYM1DB+zjtKmg3By3vWYpsAXkPI6wSQ43X
biZ6Z5RZvwF9XKD/nM2/s8pPZXbnNEWVTUI/b1Z6pUIf2BCovHCJLUdXV+apkvmUEsf
86Pv7ZCgHWrDDfF+93n7XfttTVeVQdbzs/PbhQ1Alee1QkojW6wIDAQAB"
```

6.3.15 Finally, the PID can acknowledge that the configuration has been applied by confirming the response id to the discovery service:

```
{device_type}/discovery/acknowledge
 {
   "discoveryAcknowledge": {
   "vendorId": "VENDOR0001VENDOR0001",
   "deviceId": "ABCD1234567890",
   "responseId": "1bb1caf4-6866-441d-a946-83cf830fdc6d"
 }
}
```

6.4 Request for Content Delivery

6.4.1 Where a display requires the delivery of a fresh set of content it should use the following requests. Scenarios where this would be used include:

- The display has completed discovery, started up for the first time and has not yet received any content to display.
- The display has received a data reset request from the CMS
- The display has restarted from a power outage
- The display has received a restart request from the CMS
- The display needs content provided by a platform other than the primary CMS and the display needs to inform the platform that content should be sent.

6.4.2 The display will publish to the following topics depending on the data required.

```
CMS/scheduledDeparture/request
CMS/realtimeDeparture/request
CMS/informationMessage/request
CMS/journeyMessage/request
```

6.4.3 The payload of this request will be as follows for the request of scheduled data:

```
{
 "scheduledDepartureRequest": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId”: "ad3ccebe-0611-4d4c-8ab3-7e1e6882b08b",
   "timestamp": "2022-01-12T10:32:14-00:00",
   "locationConfig": {
     "localities": [],
     "clusters": [],
     "stops": ["4900128738"],
     "locationName": [ // multiLanguageText object
       {
         "lang": "EN",
         "text": "Cardiff Central"
       },
       {
         "lang": "CY",
         "text": "Caerdydd Canolog"
       }
     ]
   }
 }
```

```
}
```

6.4.4 The payload of this request will be as follows for the request of real time data:

```
{
 "realtimeDepartureRequest": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId”: "ad3ccebe-0611-4d4c-8ab3-7e1e6882b08b",
   "timestamp": "2022-01-12T10:32:14-00:00",
   "locationConfig": {
     "localities": [],
     "clusters": [],
     "stops": ["4900128738"],
     "locationName": [ // multiLanguageText object
       {
         "lang": "EN",
         "text": "Cardiff Central"
       },
       {
         "lang": "CY",
         "text": "Caerdydd Canolog"
       }
     ]
   }
 }
 }
```

6.4.5 The payload of this request will be as follows for the request of information messages:

```
{
 "informationMessageRequest": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId”: "ad3ccebe-0611-4d4c-8ab3-7e1e6882b08b",
   "timestamp": "2022-01-12T10:32:14-00:00",
   "locationConfig": {
     "localities": [],
     "clusters": [],
     "stops": ["4900128738"],
     "locationName": [ // multiLanguageText object
       {
         "lang": "EN",
         "text": "Cardiff Central"
       },
       {
         "lang": "CY",
```

```
        "text": "Caerdydd Canolog"
      }
    ]
  }
 }
}
```

6.4.6 The payload of this request will be as follows for the request of information messages:

```
{
 "journeyMessageRequest": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId”: "ad3ccebe-0611-4d4c-8ab3-7e1e6882b08b",
   "timestamp": "2022-01-12T10:32:14-00:00",
   "locationConfig": {
     "localities": [],
     "clusters": [],
     "stops": ["4900128738"],
     "locationName": [ // multiLanguageText object
       {
         "lang": "EN",
         "text": "Cardiff Central"
       },
       {
         "lang": "CY",
         "text": "Caerdydd Canolog"
       }
     ]
   }
 }
}
```

6.4.7 The Display will subscribe to these topics:

```
CMS/scheduledDeparture/response/{locality_id}/{cluster_id}/{sto
p_id}/{vendor_id}/{device_id}
CMS/realtimeDeparture/response/{locality_id}/{cluster_id}/{stop
_id}/{vendor_id}/{device_id}
CMS/informationMessage/response/{locality_id}/{cluster_id}/{sto
p_id}/{vendor_id}/{device_id}
CMS/journeyMessage/response/{locality_id}/{cluster_id}/{stop_id
}/{vendor_id}/{device_id}
```

6.4.8 When the request has been processed, the CMS or the third-party content provider will send a response of the form

```
{
"realtimeDepartureResponse": {
 "requestId": "ad3ccebe-0611-4d4c-8ab3-7e1e6882b08b",
 "status": "DONE",
 "timestamp": "2022-01-12T10:32:14-00:00"
 }
}
```

6.4.9 The element `status` signifies DONE when fresh content has been delivered to the topic(s) or NO_DATA when there is no content available for the requested location.

6.4.10 This mechanism ensures the CMS has visibility of the data exchange and can to a limited extent track performance of third-party content providers.

6.5 Departure Cleardown

6.5.1 Timely cleardown of expired departures is critical to the success of any RTPI system.

6.5.2 The protocol therefore requires cleardown request and response messages to contain a timestamp, so that the CMS (or other MQTT-connected reporting services) can measure the latency between a CMS requesting a cleardown and a PID confirming that it has actioned the request.

6.5.3 Requests should be sent using the following topic name:

```
CMS/cleardown/request/{locality_id}/{cluster_id}/{stop_id}/{ven
dor_id}/{device_id}
```

6.5.4 The payload is very simple and only contains a departure id, a cleardown reason and a timestamp:

```
{
 "cleardownRequest": {
   "departureId": "fc867e8a-4e9c-465d-847d-6bf2fb37977f",
   "cleardownReason": "DEPARTED",
   "timestamp": "2022-01-12T10:32:14-00:00"
 }
}
```

6.5.5 The response is sent using the topic name:

```
{device_type}/cleardown/response
{
```

```
 "cleardownResponse": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "departureId": "fc867e8a-4e9c-465d-847d-6bf2fb37977f",
   "timestamp": "2022-01-12T10:32:19-00:00"
 }
}
```

6.6 Status Reporting

6.6.1 Status requests and responses follow a similar pattern to display capability requests and responses; the request topic is as follows:

```
{device_type}/status/request/{vendor_id}/{device_id}
```

6.6.2 The message payload simply contains a request id:

```
{
 "statusRequest": {
   "requestId": "a2047a18-75aa-4bd1-a281-5fbae13d6cf1"
 }
}
```

6.6.3 The response will be broadcast by the PID to any status topic subscribers using the topic name:

```
{device_type}/status/response
```

6.6.4 The message payload includes the PID's vendor and device ids, a copy of the request id (so that the response can be matched to the request – or ignored by any other devices that subscribe to status responses) and a status report object:

```
{
 "statusResponse": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId":   "   a2047a18-75aa-4bd1-a281-5fbae13d6cf1",   //
   taken from request
   "statusReport": {    // status report object
     "firmwareVersion": "XYZ.789.001.A",
     "operatingSystem": "Homebrew Linux 3.11",
     "upSince": "2020-11-17T06:30:00-00:00",
     "memoryUsage": "63%",
     "loadAverage": "78%",    // one minute load average
```

```
    "displayOverride": "RT_DEP" // optional, if a display
    override is active, set to enum
   }
  }
}
```

6.7 Text Snapshot

6.7.1 Status requests and responses follow a similar pattern to display capability and status reporting requests and responses; the request topic is as follows:

```
{device_type}/textSnapshot/request/{vendor_id}/{device_id}
```

6.7.2 The message payload simply contains a request id:

```
{
 "textSnapshotRequest": {
   "requestId": "ae1c5b90-c0f1-4b64-8493-eea6db12eeea"
 }
}
```

6.7.3 The response will be broadcast by the PID to any text snapshot topic subscribers using the topic name:

```
{device_type}/textSnapshot/response
```

6.7.4 The message payload includes the PID's vendor and device ids, a copy of the request id (so that the response can be matched to the request – or ignored by any other devices that subscribe to text snapshot responses) and a line-by-line representation of the text that is currently being displayed:

```
{
 "textSnapshotResponse": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId": "ae1c5b90-c0f1-4b64-8493-eea6db12eeea", // taken
   from request
   "lines": [     // array of string values
     "123A East Killingham  1025",
     "X17  Batchampton      7min",
     "No delays reported        "
   ]
 }
}
```

7 Use Cases

7.1 Message handling behaviour

7.1.1 There are separate enumerations for Message display style and Message priority type in the protocol, handling of these requires some interpretation. This section outlines the assumptions in behaviour made by the working group.

7.1.2 Full-Screen Message

7.1.2.1 Full-screen messages take precedence over any other content currently displayed on the screen. This means that any real-time information or other media scheduled for display will be temporarily overridden by the message while it is active, ensuring it is the sole content visible.

7.1.2.2 In cases where the entire message cannot be accommodated due to character limitations on the display, it will be paged through until the complete message has been shown. Once displayed in full, the message will cyclically loop from the beginning until its scheduled expiry time or until it is manually cancelled.

7.1.2.3 When multiple messages are sent to the display, the highest-priority message will take precedence. If two active messages share the same priority, the display will alternate between the first and second messages until one of the messages concludes or is cancelled. If the other message is still active, it will remain on display.

7.1.2.4 Emergency full screen messages are displayed exclusively, any lower priority messages are not shown at all, and bottom line messages are not shown. In the event of multiple active emergency messages, they will be paged in the order received by the display

7.1.3 Overline

7.1.3.1 These behave as full screen with the exception an overline message leaves bottom line available for other messages where screens do not otherwise differentiate, for example many LED displays.

7.1.4 Bottom Line Message

7.1.4.1 Each display can be customised to showcase a specific maximum number of messages simultaneously. This setting dictates the maximum number of messages, regardless of priority level, that can appear on the display at any given time.

7.1.4.2 Emergency Messages: When emergency messages are dispatched, they are exclusively displayed on the bottom line of the screen. In the

event of multiple active emergency messages, they will be paged in the order received by the display, adhering to the configured active message limit.

7.1.5 Message Priority

7.1.5.1 Message priority is set through the use of messagePriority.

7.1.5.2 Messages with a messagePriority of EMERGENCY will take predicence over all other messages.

7.1.5.3 Messages with a messagePriority of STANDARD are enhanced with messageActionPriority which shall be a positive integer, where 1 is the highest priority and the lowest priority being 99.

7.1.5.4 In the absence of emergency messages, messages are displayed based on their relative messageActionPriority and displayed in the order they are received.

7.1.5.5 If there are no active emergency messages and the number of messageActionPriority=1 messages haven't reached a configured limit (if available), messageActionPriority=2 messages will be displayed up a configured message limit (if available). These will be displayed alongside any messageActionPriority=1 messages, with messageActionPriority=1 messages appearing first within the message cycle.

7.1.5.6 This process continues for messages with higher messageActionPriority until the configurable limits for number of messages displayed is reached.

7.1.5.7 This system ensures that messages are displayed according to their priority level while adhering to any message limit set for each display unit.

7.2 Quay reallocation

7.2.1 Where a Quay is reallocated in real time the working group recommends the following behaviour.

7.2.2 For example a departure planned for quay 7 is reallocated to quay 8.

7.2.3 In this situation it is expected that a new Monitored Call is pushed to stop_id for quay 8 and the existing Monitored Call call for the original stop_id for quay 7 is updated to be quay 8 as well. This allows a message to be displayed against the departure on quay 7.

7.2.4 A journeyMessage can then be used to display a message on the display for quay 7 “This service now departs from stand 8”.