improving public transport through technology

CMS to PID Interface Protocol

Part 3 - Message Types for Graphical Displays

RTIG Library Reference: **RTIGT047-pt3-2.0**

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 3 covering message types for graphical displays 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 of interface

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

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

1.6.1.3 Part 3 (this document)

- 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 Message Types for Graphical Displays

2.1 Multimedia References

2.1.1 It is the intention that binary data (audio, video, images, etc) required by more advanced PIDs will not be encoded directly into MQTT message payloads. Instead, it is proposed that external URLs are provided as part of the message payload and that a PID subsequently retrieves multimedia data from the internet (or private network) directly using the following JSON structure:

```
{
  "multimediaRef": {
    "contentType": "video/mp4",
    "url": "https://suppliername.com/content/video/test.mp4",
    "md5": "79054025255fb1a26e4bc422aef54eb4",
    "contentExpiry": "2022-03-01T00:00:00-00:00"
}
```

2.1.2 Content types must be standard MIME types, from the list maintained by IANA (RFC6838 and RFC4855).

2.1.3 The MD5 checksum should be used to verify that the content has not been corrupted during transfer.

2.1.4 It should be noted that while MD5 is an old algorithm that is no longer secure for encryption purposes (having been compromised by multiple vulnerabilities and subsequently superseded for this purpose by algorithms such as SHA-256), it is still widely used and accepted as the de facto standard for file hashing, with implementations readily available in most environments.

2.1.5 If the device referencing the multimedia content has already downloaded the same content previously, it need only be re-download if the original content expiry timestamp has passed.

2.1.6 Known media data should be synchronised on display startup or playlists may be downloaded in the background as required.

2.1.7 Assets may be display type specific. The display needs to request assets for that display. (as an example, e-paper typically may use svg assets not png).

2.2 Graphical Snapshot

2.2.1 Based on the concepts of text snapshots (see RTIGT047 Part 2) and multimedia references (above), the graphical snapshot message pattern is defined as follows.

2.2.2 Request topic:

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

2.2.3 Request payload:

```
{
  "graphicalSnapshotRequest": {
    "requestId": " 7d79665a-ea06-4648-810b-9f689bf87a14",
   "uploadUrl":
   "https://suppliername.com/content/snapshot/202404201556/ABCD
   1234567890.png?signed=1231231234123d"
  }
}
```

2.2.4 Upon receiving a graphical snapshot request, the PID will upload a snapshot image to an internet-accessible location defined by uploadUrl and then issue a response to any graphical snapshot topic subscribers using the topic name:

```
{device_type}/graphicalSnapshot/response
```

2.2.5 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 graphical snapshot responses) and a multimedia reference to the snapshot image that has been uploaded to an internet-accessible location:

```
{
  "graphicalSnapshotResponse": {
    "vendorId": "VENDOR0001",
    "deviceId": "ABCD1234567890",
    "requestId": "7d79665a-ea06-4648-810b-9f689bf87a14",
    "multimediaRef": {
      "contentType": "image/png",
      "url":
     "https://suppliername.com/content/snapshot/202112171556/A
     BCD1234567890.png",
      "md5": "79054025255fb1a26e4bc422aef54eb4",
      "contentExpiry": "2021-12-18T00:00:00-00:00"
   }
}
```

2.3 Playlists

2.3.1 A playlist is a list of media elements (usually of the same media type, although mixed media types are permitted) that should be rendered sequentially *within a single display component*.

2.3.2 Playlists are specified as an array of elements containing a media type, a display duration and a display order:

```
[
  {
    "multimediaRef": {...}, // multimedia ref object
    "duration": 25,         // integer number of seconds
    "order": 1              // integer
  },
  {
    "multimediaRef": {...}, // multimedia ref object
    "duration": 10,         // integer number of seconds
    "order": 2              // integer
  },
  ...
]
```

2.4 Graphical Display Templates

2.4.1 A template defines a full-screen layout and must contain details of every display component (e.g. departures, advertising, message ticker, etc) that is included in the layout.

2.4.2 All PIDs must support the capability to use templates

2.4.3 Each component must display content of one of the following media types:

```
[
  IMAGE,
  VIDEO,
  PLAYLIST,
  displayStatusType, // e.g. RT_DEP, MIXED_ARRDEP, etc
  INFO_MSG,
  RSS_FEED
]
```

2.4.4 Each component is defined using the following basic data structure:

```
"graphicalDisplayComponent": {
  "top": 0,        // integer value in pixels
  "left": 0,       // integer value in pixels
  "bottom": 800,   // integer value in pixels
  "right": 1200,   // integer value in pixels
  "mediaType": "RT_DEP",    // enum value
  "contentRaw": { ... },  // playlist or null
  "contentRefs": []       // array of multimediaRef
}
```

2.4.5 Where a component is of media type `PLAYLIST`, `contentRaw` must be a playlist object (as defined above) and `contentRef` must be an empty array; for all other types, `contentRaw` must be null and `contentRef` must not be empty. Where a component is of type `IMAGE`, `VIDEO` or `RSS_FEED`, `contentRef` must contain a single multimedia reference; for other media types, multimedia references to an arbitrary combination of HTML, JavaScript and CSS files can be supplied. It should be noted that successful rendering of a particular referenced file type will always be subject to a PID's capabilities – see Graphical Display Capability below.

2.4.6 It is recommended that all required files must be specifically included in the contentRef array, and that a PID must not attempt to follow any external URLs embedded directly within any source files, in order to provide a basic level of security against the injection of malicious code.

2.4.7 A template is defined using the following data structure:

```
"graphicalDisplayTemplate": {
  "templateId": "09d57be3-253a-4670-8513-3bb4c2ba2883",
  "components": [
    { ... }, // graphicalDisplayComponent
    { ... }, // graphicalDisplayComponent
    ...
  ]
}
```

2.4.8 The CMS (or another service on the MQTT network) can request definitions of the templates currently being used by a PID using the following request topic:

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

2.4.9 The message payload simply contains a request id:

```
{
 "getAllTemplatesRequest": {
   "requestId": "5412b85a-1083-4eea-bf54-26c7bca709c1"
 }
}
```

2.4.10 In response, the PID should broadcast to any subscribers using the topic name:

```
{device_type}/getAllTemplates/response
```

2.4.11 The message payload is an array of graphical display template definitions, together with a reference to the request id:

```
{
 "getAllTemplatesResponse": {
   "requestId": "5412b85a-1083-4eea-bf54-26c7bca709c1",
   "templates": [
     "graphicalDisplayTemplate": { ... },
     "graphicalDisplayTemplate": { ... },
     ...
   ]
 }
}
```

2.4.12 The CMS can create/replace definitions of the templates currently being used by a PID using the following request topic:

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

2.4.13 The payload is the same as a `getAllTemplatesResponse`:

```
{
 "setAllTemplatesRequest": {
   "requestId": "7192044e-37ec-4a3c-a4ad-652b780d3c0b",
   "templates": [
     "graphicalDisplayTemplate": { ... },
     "graphicalDisplayTemplate": { ... },
     ...
   ]
 }
}
```

2.4.14 Note that the payload must contain the full set of all required templates for the PID as it will overwrite all previous template definitions.

2.4.15 In response, the PID should send a confirmation that the templates were received and updated successfully, using the topic:

```
{device_type}/setAllTemplates/response
```

2.4.16 The payload is a repetition of the accepted template ids. It is the responsibility of the PID to reject unsupported templates by excluding their ids from the response payload:

```
{
 "setAllTemplatesResponse": {
   "requestId": "7192044e-37ec-4a3c-a4ad-652b780d3c0b",
   "templateIds": [
     "d1613512-2baa-427e-b0d4-c5baa665ef1f",
     "defa79c9-93f9-4f40-b61e-9f473e1b9d59",
     ...
   ]
 }
}
```

2.4.17 The CMS (or another service on the MQTT network) can request the definition of the active template for a PID using the following request topic:

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

2.4.18 The message payload simply contains a request id:

```
{
 "getActiveTemplateRequest": {
   "requestId": "20cda746-b9cc-4202-860b-c96e5717c609"
 }
}
```

2.4.19 In response, the PID should broadcast to any subscribers using the topic name:

```
{device_type}/getActiveTemplate/response
```

2.4.20 The message payload is a single graphical display template definition, together with a reference to the request id:

```
{
 "getActiveTemplateResponse": {
   "requestId": "20cda746-b9cc-4202-860b-c96e5717c609",
   "graphicalDisplayTemplate": { ... }
 }
}
```

2.4.21 The CMS can set which template is currently being used by a PID using the following request topic:

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

2.4.22 The payload consists of a template id and a request id:

```
{
 "setActiveTemplateRequest": {
   "requestId": "7d3ccebe-0611-4d4c-8ab3-7e1e6882b08b",
   "templateId": "d1613512-2baa-427e-b0d4-c5baa665ef1f"
 }
}
```

2.4.23 In response, the PID should confirm whether or not the active template was updated, using the topic:

```
{device_type}/setActiveTemplate/response
```

2.4.24 The payload is a repetition of the received template ids:

```
{
 "setActiveTemplateResponse": {
   "requestId": "7d3ccebe-0611-4d4c-8ab3-7e1e6882b08b",
   "result": "OK" // enum value
 }
}
```

2.4.25 In each case, the result should be a value from the following enumeration:

```
[
  OK,
  NOT_SUPPORTED,
  NOT_FOUND,
  ERROR
]
```

2.5 Carousels

2.5.1 A carousel is effectively a playlist of templates. A carousel is defined in a similar manner to a playlist – i.e. as an array of elements – the key difference being that each element in the array is a template identifier rather than a multimedia reference:

```
[
  {
    "templateId": "09d57be3-253a-4670-8513-3bb4c2ba2883",
    "duration": 30,          // integer number of seconds
    "order": 1               // integer
  },
  {
    "templateId": "e8681125-b196-4034-86e0-8f0c50d12a87",
    "duration": 30,          // integer number of seconds
    "order": 2               // integer
  },
  ...
]
```

2.5.2 It is not necessary for all connected PIDs to support the capability to use carousels; this is likely to be a key differentiating factor between PID models and suppliers. Where carousel support is provided, a further differentiating factor will be how transitions between carousels are managed. For example, is the PID able to maintain the current state of a scrolling text ticker or media playlist that is common to sequential templates within a carousel – or will the content for such components be restarted whenever the template changes?

2.5.3 The CMS (or another service on the MQTT network) can request definitions of the carousels currently being used by a PID using the following request topic:

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

2.5.4 The message payload simply contains a request id:

```
{
 "getAllCarouselsRequest": {
   "requestId": "7ebdcb4c-cad0-4202-a59e-5a0bfd24f5c7"
 }
}
```

2.5.5 In response, the PID should broadcast to any subscribers using the topic name:

```
{device_type}/getAllCarousels/response
```

2.5.6 The message payload is an array of graphical display carousel definitions, together with a reference to the request id:

```
{
 "getAllCarouselsResponse": {
   "requestId": "7ebdcb4c-cad0-4202-a59e-5a0bfd24f5c7",
   "carousels": [
     "graphicalDisplayCarousel": { ... },
     "graphicalDisplayCarousel": { ... },
     ...
   ]
 }
}
```

2.5.7 The CMS can create/replace definitions of the carousels currently being used by a PID using the following request topic:

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

2.5.8 The payload is the same as a `getAllCarouselsResponse`:

```
{
 "setAllCarouselsRequest": {
   "requestId": "e7034aa3-f7f0-4a59-b486-493eb368672d",
   "carousels": [
     "graphicalDisplayCarousel": { ... },
     "graphicalDisplayCarousel": { ... },
     ...
   ]
 }
}
```

2.5.9 Note that the payload must contain the full set of all required carousels for the PID as it will overwrite all previous carousel definitions.

2.5.10 In response, the PID should send a confirmation that the carousels were received and updated successfully, using the topic:

```
{device_type}/setAllCarousels/response
```

2.5.11 The payload is a repetition of the received carousel ids:

```
{
 "setAllCarouselsResponse": {
   "requestId": "e7034aa3-f7f0-4a59-b486-493eb368672d",
   "carouselIds": [
     "4b4eb833-808a-435f-ac64-4a20cdf041d8",
     "a910c96b-d599-4ed6-9de1-1b4f91eede6c",
     ...
   ]
 }
}
```

2.5.12 The CMS (or another service on the MQTT network) can request the definition of the active carousel for a PID using the following request topic:

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

2.5.13 The message payload simply contains a request id:

```
{
 "getActiveCarouselRequest": {
   "requestId": "f9ec6b7b-9809-48c6-a5d2-9924d9b87ccf"
 }
}
```

2.5.14 In response, the PID should broadcast to any subscribers using the topic name:

```
{device_type}/getActiveCarousel/response
```

2.5.15 The message payload is a single graphical display carousel definition, together with a reference to the request id:

```
{
 "getActiveCarouselResponse": {
   "requestId": "f9ec6b7b-9809-48c6-a5d2-9924d9b87ccf",
   "graphicalDisplayCarousel": { ... }
 }
}
```

2.5.16 The CMS can set which carousel is currently being used by a PID using the following request topic:

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

2.5.17 The payload consists of a carousel id and a request id:

```
{
 "setActiveCarouselRequest": {
   "requestId": "10af973f-3213-436e-9db0-99b7a82f12d8",
   "carouselId": "4b4eb833-808a-435f-ac64-4a20cdf041d8"
 }
}
```

2.5.18 In response, the PID should confirm whether or not the active carousel was updated, using the topic:

```
{device_type}/setActiveCarousel/response
```

2.5.19 The payload is a repetition of the received carousel ids:

```
{
 "setActiveCarouselResponse": {
   "requestId": "10af973f-3213-436e-9db0-99b7a82f12d8",
   "result": "OK" // enum value from [ OK, NOT_FOUND, ERROR ]
 }
}
```

2.6 Graphical Display Capability

2.6.1 The graphical display capability data structure type includes details of supported media types, screen dimensions and orientation, using the following data structure:

```
"graphicalDisplayConfig": {
  "screenDimensions": {
    "x": 1920, // integer value in pixels
    "y": 1080  // integer value in pixels
  },
  "screenOrientation": LANDSCAPE,  // enum [LANDSCAPE, PORTRAIT]
  "screenAspectRatio": "16:9",     // string value using colon
  "iconsSupported": true,           // boolean value
  "imagesSupported": true,          // boolean value
  "videosSupported": true,          // boolean value
  "playlistsSupported": true,       // boolean value
  "carouselsSupported": true,       // boolean value
  "componentContinuitySupported": true, // boolean value
  "htmlSupported": true,            // boolean value
  "javaScriptSupported": true,      // boolean value
  "cssSupported": true              // boolean value
}
```

In theory, only one of `screenDimensions`, `screenOrientation` and `screenAspectRatio` should be required, but including all three is recommended for end-user readability. Where there is conflict between these values, `screenDimensions` shall always take precedence.

`componentContinuitySupported` relates to the ability to preserve the state of a component between template changes (regardless of whether a new template has been specifically requested or has been changed as part of a carousel). For example, where sequential templates contain a scrolling text or media playlist component (in the same location on the screen) will the component state be preserved during a template transition or will the ticker text or media playlist be restarted.

2.6.2 As described in RTIGT047 Part 2, a display capability request will typically be sent by a CMS using a topic name constructed as follows:

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

2.6.3 As before, the message payload simply contains a request id:

```
{
 "displayCapabilityRequest": {
   "requestId": "f3d68878-3b5a-4f6f-ba08-b4ffd5b0f1d7"
 }
}
```

2.6.4 Once again, the response will be broadcast by the PID to any display capability topic subscribers using the topic name:

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

2.6.5 However, for a multimedia display, the message payload includes a `graphicalDisplayConfig` object in addition to the contents of the equivalent message for a text display:

```
"displayCapabilityResponse": {
   "vendorId": "VENDOR0002",
   "deviceId": "ABCD1234567890",
   "requestId": "f3d68878-3b5a-4f6f-ba08-b4ffd5b0f1d7",
   "textDisplayConfig": { ... },
   "graphicalDisplayConfig": { ... }
 }
```

2.7 Graphical Assets Request

2.7.1 Icon media files can be downloaded by the display using a request response mechanism at display startup and dynamically as required.

2.7.2 It is expected that the CMS will supply the correctly sized and formatted icon files associated with the display that requested it. This may include SVG format files for e-paper displays. Bitmaps for LED displays, and PNG or JPG of appropriate resolution for TFT displays.

2.7.3 Graphical assets are stored as `iconType` and `iconRef`. The `iconType` enumeration may contain the following values

```
[
OPERATOR, // operator icon
LINE, // line icon
ROUTE, // route icon
POI, // Point of Interest icon
FEATURE // Feature icon
]
```

2.7.4 A graphicalAssetsRequest is sent using a topic name constructed as follows

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

2.7.5 The request contains the body as follows

```
{
 "graphicalAssetsRequest": {
   "vendorId": "VENDOR0001",
   "deviceId": "ABCD1234567890",
   "requestId": "1ca491a6-3355-462e-81b5-cac9bbf4941c",
   "iconType": "" //optional
   "iconRef": "" //optional
 }
}
```

2.7.6 The `iconType` and the `iconRef` are optional. If they are not supplied in the request (key not present, blank or set to null) then a full set of icons appropriate to that display is delivered.

2.7.7 The `iconType` may be supplied on its own without an `iconRef`. For example `iconType` is set to OPERATOR to receive a full set of operator logos.

2.7.8 The `iconRef` must be unique to the project, and may not be used more than once across different `iconType`

2.7.9 The `iconRef` may be requested on its own if the display finds that a specific icon is missing.

2.7.10 The response topic is constructed as follows

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

2.7.11 The response contains the body as follows

```
{
    "graphicalAssetsResponse": {
        "vendorId": "VENDOR0001",
        "deviceId": "ABCD1234567890",
        "responseId": "1bb1caf4-6866-441d-a946-83cf830fdc6d",
        "icons": [
            {
                "iconType": "OPERATOR",
                "iconRef": "SCE",
                "multimediaRefs": [
                    {
                        "contentType": "image/png",
                        "url":
"https://suppliername.com/content/images/sce-en.png",
                        "md5":
"79054025255fb1a26e4bc422aef54eb4",
                        "contentExpiry":  "2022-03-01T00:00:00-
00:00",
                        "lang": "EN"
                    },
                    {
                        "contentType": "image/png",
                        "url":
"https://suppliername.com/content/images/sce-cy.png",
                        "md5":
"79054025255fb1a26e4bc422aef54eb4",
                        "contentExpiry":  "2022-03-01T00:00:00-
00:00",
                        "lang": "CY"
                    }
                ]
            },
            {
                "iconType": "OPERATOR",
                "iconRef": "NXEM",
                "multimediaRefs": [
```

```
                    {
                        "contentType": "image/png",
                        "url":
"https://suppliername.com/content/images/sce-en.png",
                        "md5":
"79054025255fb1a26e4bc422aef54eb4",
                        "contentExpiry":  "2022-03-01T00:00:00-
00:00",
                        "lang": "EN"
                    }
                ]
            },
            {
                "iconType": "POI",
                "iconRef": "railway",
                "multimediaRefs": [
                    {
                        "contentType": "image/png",
                        "url":
"https://suppliername.com/content/images/railway-en.png",
                        "md5":
"79054025255fb1a26e4bc422aef54eb4",
                        "contentExpiry":  "2022-03-01T00:00:00-
00:00",
                        "lang": "EN"
                    }
                ]
            }
        ]
    }
}
```

2.7.12 On receiving the icon multimediaRefs, the display should download the referenced files and make them available for use.

2.7.13 When the contentExpiry date and time is reached, the display should request a new set of graphical Assets.

2.8 Icon Support

2.8.1 Since graphical displays are able to display images, the `scheduledDeparture`, `realtimeDeparture` and `informationMessage` message types (as defined in RTIGT047 Part 2) are extended to include support for the addition of one or more icons for a given departure or message, using the following array structure:

```
icons:
[
 {
    "iconRef": "NXEM", // iconRef
    "priority": 1           // integer - optional
 },
  {
    "iconRef": "railway", // iconRef
    "priority": 2          // integer - optional
  },
 ...
]
```

2.8.2 In each case, an instance of an icon is defined via an iconRef and an optional priority. As with all icon references, the intention is that the asset is downloaded and cached until the expiry time is reached – thereby avoiding repeated downloads each time an icon needs to be displayed.

2.8.3 The display is aware of the type of icon because this was provided as part of the graphicalAssetsResponse structure, so it is only necessary to provide an iconRef and optional priority in the content message.

2.8.4 Example of a scheduled departure message containing icons, showing the icon information for each departure alongside the targeted vehicle journey and targeted call definitions:

```
{
 "scheduledDepartures": [
   {
     "departureId": "fc867e8a-4e9c-465d-847d-6bf2fb37977f",
     "vehicleMode": "BUS",
     "targetedVehicleJourney": { ... }, // journey definition
   "targetedCall": { ... }, // targeted call definition
       "icons": [
  {
    "iconRef": "NXEM", // iconRef
    "priority": 1           // integer - optional
  },
  {
    "iconRef": "railway", // iconRef
    "priority": 2          // integer - optional
  },
   ...
 ],
```

```
    },
    ...
 ]
 }
```

The agreed logos are synchronised during display startup via the graphicalAsset request mechanism.

Operator and service logos are handled in the display template – if the display can handle the graphic and the logo file is available then it shall be used.

2.8.5 Example of a real-time departure message containing icons, showing the icon information for each departure alongside the monitored vehicle journey and monitored call definitions:

```
{
 "realTimeDeparture": [
   {
     "departureId": "fc867e8a-4e9c-465d-847d-6bf2fb37977f",
     "predictionTime": "2004-12-17T09:25:46-05:00",
     "vehicleMode": "BUS",
     "monitoredVehicleJourney": { ... }, // journey definition
     "progressStatus": "ON-TIME", // enum value
     "occupancyLevel": "SEATS-AVAILABLE", // enum value
       "monitoredCall": { ... }, // monitored call definition
     "icons": [
  {
    "iconRef": "NXEM", // iconRef
    "priority": 1           // integer
  },
  {
    "iconRef": "railway", // iconRef
    "priority": 2          // integer
  },
  ...
 ],
   },
   ...
 ]
}
```

2.8.6 Example of an information message containing icons, alongside the message text definition:

```
{
 "informationMessage": {
   "messageText": [ ... ],
   "earliestDisplayTime": "2022-02-17T10:00:00-00:00",
   "latestDisplayTime": "2022-02-17T17:00:00-00:00",
"messagePriority": "MEDIUM", // message priority enum
"displayStyle": "SCROLL", // message display style enum
"messageId": UUID,
     "icons": [
  {
    "iconRef": "NXEM", // iconRef
    "priority": 1           // integer
  },
  {
    "iconRef": "railway", // iconRef
    "priority": 2          // integer
  },
  ...
 ]
 }
}
```

2.8.7 An information messageText may contain placeholders within the text to reference icons which can be displayed inline. This is structured as %icon:iconRef% within the text. An example would be

"messageText": "Change here for %icon:railway% to continue your journey"

2.8.8 The messageText in a journeyMessage may also contain placeholders for icon substitution

2.8.9 No limit is imposed on icon dimensions, number of colours or file size by the protocol, but it should be recognised that icons are only intended for use within text fields; where larger images (e.g. corporate logos or advertising) are required, image (or playlist) components should be used within a template.