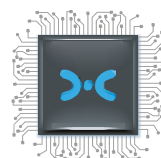


AUTO DISCOVERY REMOTE CONTROL

RML REFERENCE MANUAL



THE INTERNET OF THINGS



BETA RELEASE v0.2 // JULY 2019

DRAFT V0.2.23

xped

Contents

1	About this document	1	3.5	Controller element	56
1.1	Document purpose	1	3.5.1	Usage	56
1.2	Audience	1	3.5.2	Child elements and attributes	56
1.3	Terms and definitions	1	3.5.3	RML example	59
1.4	Reference documents	1	3.6	Configuration element.....	60
1.5	Contact information	2	3.6.1	Usage	60
2	Resource Modelling Language overview	3	3.6.2	Child elements and attributes	60
2.1	What is RML?	3	3.6.3	RML example	65
2.2	Design approach	4	3.7	Resource element.....	66
2.3	Structure.....	4	3.7.1	Usage	66
2.3.1	Root element.....	5	3.7.2	Child elements and attributes	66
2.3.2	Main elements	5	3.7.3	RML example	68
2.3.3	Language support	5	3.8	Menu element.....	69
2.3.4	Child elements and attributes.....	5	3.8.1	Usage	69
2.3.5	Icons.....	6	3.8.2	Child elements and attributes	69
2.3.6	Nicknames	6	3.8.3	RML example	71
2.3.7	Notation.....	6	4	Resource Control Protocol	72
2.4	DeB initial release	6	4.1	RML <model> element	72
3	Resource Modelling Language in detail	8	4.2	ADRC IoT Stack	72
3.1	RML root element.....	8	4.3	RCP host format	73
3.1.1	Usage.....	8	4.4	RCP wire format	73
3.1.2	Child elements and attributes.....	8	4.5	RCP protocol mapping.....	73
3.1.3	RML example	9	4.6	Transport layer.....	74
3.2	Description element.....	10	4.7	Addressing abstraction	74
3.2.1	Usage.....	10	5	Development and operations	76
3.2.2	Child elements and attributes.....	10	5.1	Integrated development environment	76
3.2.3	RML example	14	5.2	RML updates	76
3.3	Model element.....	15	Appendix A:	Complete RML example	77
3.3.1	Usage.....	15	Appendix B:	Example user interface.....	80
3.3.2	Child elements and attributes.....	15	Appendix C:	RML tags by functional groups.....	82
3.3.3	RML example	26	Appendix D:	Attributes by tag	84
3.4	View element.....	27	Appendix E:	RML reserved key words.....	89
3.4.1	Usage.....	27	Appendix F:	XML schema definition	91
3.4.2	Child elements and attributes.....	27	Appendix G:	Default icons	92
3.4.3	Extended bind syntax.....	54	Appendix H:	DeB in-built skins	99
3.4.4	RML example	54			
3.4.5	User interface examples.....	55			

Tables

Table 1: Root element - child elements	9
Table 2: Description section - child elements	13
Table 3: Model section - child elements	19
Table 4: Model section - sub-elements	22
Table 5: Model section - attributes	25
Table 6: Display widgets	31
Table 7: View section – layout elements	37
Table 8: View section - display elements	46
Table 9: View section - attributes	53
Table 10: Controller section - child elements	57
Table 11: Controller section - attributes	58
Table 12: Configuration section - child elements	61
Table 13: Configuration section – sub-elements	63
Table 14: Configuration section - attributes.....	64
Table 15: Resource section - child elements	67
Table 16: Menu section - child elements	70

Figures

Figure 1: DeB screen layout.....	7
Figure 2: User interface examples	55
Figure 3: ADRC architecture	73
Figure 4: RCP protocol mapping	74
Figure 5: Transport layer.....	75

Disclaimer: Whilst every reasonable effort has been made to ensure the accuracy of the information provided by Xped. Xped shall not be held liable for any inaccurate information of any nature, however communicated by Xped. The contents of this document are subject to change without notice

1 About this document

1.1 Document purpose

This document describes Xped's Resource Modelling Language (RML). RML is the component of Xped's Auto Discovery Remote Control (ADRC) platform that describes a Thing (electronic device) so that it can be controlled by a generic device browser app on a user's smart device or by another ADRC client.

The ADRC platform is described in the Auto Discovery Remote Control overview document, which is recommended as pre-requisite reading.

1.2 Audience

This document is a language reference for developers who intend to write files in the Resource Modelling Language (RML) to describe the API and data of an electronic device to enable that electronic device to be part of the Auto Discovery Remote Control platform.

It is assumed that readers are familiar with XML and HTML.

1.3 Terms and definitions

The following key terms are used in this document:

- ADRC
 - > Auto Discovery Remote Control – the Xped platform that allows all a user's electronic devices to be controlled from a single app on a smartphone
- Thing
 - > A Thing is any electronic device that has an on/off switch and has been enabled to connect into an ADRC personal area network.
- Item of state
 - > An item of state, also known as a property, is an individual state that can be set or read in a Thing, such as an individual control setting or data item.

For all other terms and definitions, refer to the *ADRC Glossary*.

1.4 Reference documents

Document name	Description
ADRC Glossary	Defines the terms used across all ADRC documents.
ADRC Technical Overview	Provides an end-to-end overview of Xped's Auto Discovery Remote Control platform.
ADRC Developer Guide	A how-to guide for developing RML applications to implement Things in an ADRC environment.

Resource Control Protocol Interface
Design Documents

The reference specification that defines the Resource Control Protocol.

Part 1: Host format used between the Internet of Things Gateway and the device browser or ADRC client.

Part 2: Wire format used between the Internet of Things Gateway and the Thing.

1.5 Contact information

For additional information and queries, email info@xped.com.

2 Resource Modelling Language overview

The Resource Modelling Language (RML) makes it possible to create a single app - the device browser (DeB) - that can control and monitor all Things. RML and DeB are the components of the ADRC platform that eliminate the need for developers to write a specific app for each Thing. A Thing is any electronic device that has an on/off switch and some form of connectivity.

The ADRC platform is described in the *ADRC Technical Overview* document, which is recommended as pre-requisite reading.

2.1 What is RML?

RML is based on XML, but RML is a new language that has been specifically developed to describe, control and monitor Things.

Conceptually, RML is similar to HTML. HTML describes a web page so that a web browser can draw it on a screen, whereas RML describes a Thing so that the device browser (DeB) can display the interface for the Thing to the user. The device browser can only exist because RML exists. RML standardizes the way all Things are described just as HTML standardizes the way all websites are described.

RML describes a Thing in terms of 7 key aspects:

1. Identity (manufacturer, model, version, nickname, icon)
 - the <description> element in RML
2. Functionality, data and commands used to control and monitor a Thing
 - the <model> element in RML
3. User interface
 - the <view> element in RML
4. Event handling
 - the <controller> element in RML
5. Additional resources required (such as icons, web pages, plug-ins)
 - the <resource> element in RML
6. Physical connections the Thing is capable of
 - the <configuration> element in RML
7. Administrative functions available (such as unpair a Thing, set a PIN)
 - the <menu> element in RML

The RML file is a self-describing profile for the Thing. A manufacturer creates an RML file for a Thing and loads this file into a Thing's chip. The RML file is sent to the smart device when the Thing is on-boarded. The device browser app (DeB) in the smart device uses the RML file to create the user interface to control and monitor the Thing.

Note: Normally the full RML for a Thing is stored in the Thing itself. However, if space is limited, the RML file in the Thing can simply contain a URL to where the full RML file is located.

DeB communicates with an Internet of Things Gateway (IoTG), which in turn controls all the Things in its personal area network. This personal area network is sometimes referred to as a domain. RML is also used by the IoTG. The IoTG coordinates communications between DeB (or another ADRC client) and Things. To make wireless communications with Things as efficient as possible, data is cached to reduce the power consumption and network utilisation. To minimise bandwidth utilisation, some

commands do not require a Thing to send a response or delivery confirmation to the IoTG. However, the IoTG still sends an acknowledgement to the client. RML plays a part in this by providing mechanisms that allow caching and delivery parameters to be specified.

Just as a hardware remote control for an electronic device sends commands to the manufacturer's application in a Thing, the RML file contains the required commands to interact with the manufacturer's application in a Thing.

IMPORTANT: A Thing can contain multiple RML files to cater for the following scenarios.

A Thing can have multiple hardware units within it, for example a TV with an in-built DVD player. Each hardware unit has a separate RML file.

A Thing can provide different user interfaces for different types of users, for example basic, standard, administrator. Each user interface has a different RML file.

The communications protocol that is used to transfer commands and RML files between DeB, an IoTG and a Thing is called the Resource Control Protocol (RCP). RCP is described in more detail in section 4 [Resource Control Protocol](#).

2.2 Design approach

Many of the Things that will make up the Internet of Things (IoT) will be low cost microcontrollers with limited RAM and ROM, and constrained networking abilities. For these microcontrollers to operate effectively, messaging overhead needs to be small and simple to avoid message fragmentation and minimize memory usage. The main design goal of RML and RCP has been to specify a generic protocol for the special requirements of this constrained environment.

RCP is a RESTful protocol that is familiar to those who know HTTP, but also expands these capabilities to allow for Things to send unsolicited events to clients. RCP has also been optimized for machine-to-machine (M2M) applications.

RML and DeB should be familiar to those who work with web browsers and HTML.

RML is structured around the Model View Controller (MVC) software approach. In this approach, the implementation of the user interface (View) is separated from the implementation of the data (Model) and the implementation of the control logic (Controller).

The MVC approach allows loosely coupled interfaces between the Model, View and Controller areas of an application. This supports implementations that are robust and easily modifiable, especially for complex systems.

In RML:

- The Model element defines the data and allowable states for a Thing, and also the control commands – this can be considered to be the API for the Thing
- The View element provides the graphical user interface – the layouts and widgets for a Thing
- The Controller element provides the event handlers to process the signals sent from a Thing

2.3 Structure

RML is based on XML and consists of:

- A root element
- Main elements, both mandatory and optional
- Child elements, both mandatory and optional
- Attributes

Note: As RML is based on XML, any rules or restrictions that apply to XML also apply to RML. For example, the ampersand symbol is represented by &#38;

This section introduces the overall structure of RML. The RML elements and attributes are described in detail in section 3 [Resource Modelling Language in detail](#).

2.3.1 Root element

In HTML, the root element is `<html></html>`.

In RML, the root element is `<rml></rml>`.

The `<rml>` element identifies the version of RML that is being used and the location of the RML tag definitions.

Example: `<rml version="1.0" xmlns="http://rml.xped.com">`

2.3.2 Main elements

The 7 main elements contained within the `<rml>` root element are shown below:

```
<rml version="1.0" xmlns="http://rml.xped.com">
  <description> what the Thing is </description>
  <model> how to access and control the Thing's capabilities and states </model>
  <view> how to draw the user interface for the Thing </view>
  <controller> how to handle signals sent from the Thing </controller>
  <resource> any specific resources required </resource>
  <configuration> what the physical interface connections are </configuration>
  <menu> administrative functions allowed </menu>
</rml>
```

2.3.3 Language support

Where a manufacturer supports multiple languages, instead of embedding multiple languages directly in the RML, a manufacturer creates an external language file for each of the supported languages.

The external language file is transferred to DeB at the same time the RML file is transferred, which is during the on-boarding of a Thing (triggered by the proximity tap).

Any text string that can be displayed in other languages is identified by a `tr=` attribute in the element concerned. The value of this `tr=` attribute represents a key that is used to look up an associated Unicode string in the external language file.

DeB determines the display language to use from the locale setting of the operating system of the smart device.

For those manufacturers that do not wish to provide multilingual support, the `tr=` attribute may be omitted. In this case, the text is simply taken from the text content of the element.

In the following example, only one language is supported and the text value (Off) of the `<item>` element is used:

```
<item value='0'>Off</item>
```

Whereas in this next example, the value of the `tr=` attribute 'OFF' is used to look up a Unicode string value in an external language file. If the external language file does not exist or the `tr=` key is missing from the language file, the text value of the element is used as a fall-back. If no text value is supplied, an error message is displayed.

```
<item tr='OFF' value='0'>Off</item>
```

2.3.4 Child elements and attributes

In XML, there are no firm rules about when to use attributes and when to use child elements.

In RML, there is no overlap between attributes and child elements within a main element, each is used in a specific scenario.

Some attributes and elements have the same name, but each is used for a specific purpose:

- Version
 - > The `<version>` element is a child element of the `<description>` element and represents the version of the RML file for a Thing
 - > The `version=` attribute is used in the `<rml>` root element and represents the version of the RML language that is being used
- Model
 - > The main `<model>` element is one of the 7 sections of an `<rml>` document and describes the functionality and commands for a Thing
 - > The `<model>` element that is a child element of the `<description>` element identifies the model number that a manufacturer has given to a Thing
- Enum
 - > The `<enum>` element is a child element of the `<model>` element and defines individual items of state that can be represented by a small finite set of integers, typically 2-10, e.g. on/off
 - > The `enum=` attribute is used by the `<rgbled>` element to define the colors associated with individual states specified in the `<enum>` element, e.g. for fault status indicators

2.3.5 Icons

The Internet of Things Gateway (IoTG) stores a set of default icons for a broad range of device types, including televisions, sensors, actuators, smart plugs and so on. The default icons are identified by a 4-digit hex code, for example, 8003 represents a smart plug and 0000 represents an unknown device.

The manufacturer uses the `<category>` child element in the `<description>` element to identify which default icon they wish to use.

DeB displays the category icon to the user unless the manufacturer overrides the default icon by using the `icon=` attribute in the `<resource>` element in the RML file.

2.3.6 Nicknames

The manufacturer defines a nickname for the Thing in the RML file for the Thing. This is defined in the `<nickname>` child element within the `<description>` element.

The user can overwrite the manufacturer's nickname with a nickname of their choice via DeB at on-boarding time or at a later date.

The nickname, together with the icon, is used by DeB as a label for the Thing.

2.3.7 Notation

- `<>` indicates an element
- `=` indicates an attribute
- `#` indicates an RML reserved word
- `<!-- -->` indicates a comment

2.4 DeB initial release

Xped has released an initial version of DeB to the market to enable developers to start developing and delivering their products as soon as possible.

The initial DeB release supports a standard screen layout and the most commonly used widgets, which places some restrictions on the RML it supports.

The initial release of DeB supports:

- only portrait orientation
- the following set of display widgets
 - > slide switch
 - > check box
 - > RGB LED display indicator
 - > value picker (text box with popup scrollable list)
 - > slider with up/down buttons
 - > touch track slider
 - > dial
 - > push button
 - > label for static and dynamic text
 - > editable text line
 - > color picker
 - > candy bar remote control

An example of the portrait screen layout and display widgets for a Thing's control screen is shown in [Figure 1](#).



Figure 1: DeB screen layout

3 Resource Modelling Language in detail

This section details the RML elements and attributes that enable DeB to display a user interface for a Thing so that a user can:

- on-board a Thing
- send commands to a Thing to control that Thing
- receive signals from a Thing
- perform administrative functions such as setting a PIN for a Thing or changing its nickname

Example RML is provided for each RML element to illustrate how it is used. A complete RML listing for a smart LED light bulb is provided as an example Thing and is included at [Appendix A](#).

The user interface created by the example RML for a smart LED light bulb is included at [Appendix B](#).

For additional information on writing RML files, including more detailed examples, refer to the *Resource Modelling Language Programmer's Guide*.

3.1 RML root element

The root element for RML is `<rml></rml>`.

3.1.1 Usage

There is one and only one `<rml>` element in an RML file.

3.1.2 Child elements and attributes

The format of the `<rml>` tag is `<rml version=" xmlns=" ">`

- The `version=` attribute states the version of the RML language that is being used
- The `xmlns=` attribute is the namespace showing where to find specific RML tag definitions

The child elements of the `<rml>` element are summarised in [Table 1](#) and then described in detail in the subsequent sections.

	Element name	Mandatory / optional	Purpose
1.	<description>	M	Describes the Thing and its metadata, including: <ul style="list-style-type: none"> – manufacturer – model number – common name
2.	<model>	M	Defines the data and commands of the Thing and how to access them, including: <ul style="list-style-type: none"> – commands that can be used to control the Thing e.g. to switch on/off, change a channel, select a cycle Is also known as the API for the Thing
3.	<view>	O	Describes the human interface displayed in DeB, including: <ul style="list-style-type: none"> – display widgets – screen layout – screen resolution This element is optional to cater for machine-to-machine ADRC clients that do not have a user interface
4.	<controller>	O	Defines event handlers, including: <ul style="list-style-type: none"> – signals sent from the Thing – whether server side or client side – scripts to run
5.	<resource>	O	Provides links to any additional resources required, including: <ul style="list-style-type: none"> – plug-ins (future) – icons – a link to the manufacturer's website – a link to the full RML if not contained in the Thing
6.	<configuration>	O	Describes the physical connectivity interfaces of the Thing that the Thing can use to connect to other equipment, including: <ul style="list-style-type: none"> – connectors – which interface protocol is used
7.	<menu>	O	Describes the administrative functions available, including: <ul style="list-style-type: none"> – unpair a Thing – set a PIN – factory reset – change a nickname – add a new unit – remove a unit

Table 1: Root element - child elements

3.1.3 RML example

Example RML for the <rml> element is:

```
<rml version="1.0" xmlns="http://rml.xped.com">
```

3.2 Description element

The **<description>** element is a child element of the **<rml>** element. It provides the metadata for the Thing.

3.2.1 Usage

There is one and only one **<description>** element in an RML file.

It is a mandatory element.

3.2.2 Child elements and attributes

The child elements of the **<description>** element that can be used to identify a Thing are:

<description>

<manufacturer>who made it, that is the manufacturer's name**</manufacturer>**

<model>the manufacturer's model number**</model>**

<category>one of the default ADRC icons that is used if no other icon defined**</category>**

<version>version number of this RML file**</version>**

<nickname>manufacturer-defined nickname**</nickname>**

<theme>if a Thing has multiple RML files, each file is called a theme**</theme>**

<url>a link to a url for marketing/information purposes**</url>**

<class>a reference to an ontology that defines the Thing**</class>**

</description>

These child elements are described in [Table 2](#).

Child elements of <description>	Mandatory / optional	Purpose	Sub-elements and attributes	How used
<class>	O	Refers to an external ontology (OWL class) that defines the Thing.	<ul style="list-style-type: none"> • No sub-elements • No attributes • Contains text that is a URL 	<ul style="list-style-type: none"> • Zero or many (multiple inheritance) • Allows a manufacturer to reference and use open-source device descriptions
<category>	M	<p>This references the default icon for the device.</p> <p>It maps to an index file of default icons held in the IoTG, refer to section 2.3.5.</p>	<ul style="list-style-type: none"> • No sub-elements • No attributes • Contains text that is a 4-digit hex number 	<ul style="list-style-type: none"> • One and one only • This is the icon for the Thing that DeB displays to the user, unless the manufacturer overwrites it in the <resource> element with a manufacturer-defined icon • If the default icon cannot be found, the 0000 unknown device icon is used
<manufacturer>	M	The manufacturer's name	<ul style="list-style-type: none"> • No sub-elements • No attributes • Contains text – must be alphanumeric or space 	<ul style="list-style-type: none"> • One and one only • DeB displays this to the user at on-boarding time
<model>	M	<p>The manufacturer's model number</p> <p>Note: this is the description model element, which differs from the main <model></model> element described in section 3.3.</p>	<ul style="list-style-type: none"> • No sub-elements • No attributes • Contains text – must be alphanumeric or space 	<ul style="list-style-type: none"> • One and one only • DeB displays this to the user at on-boarding time
<nickname>	M	This is the manufacturer-defined nickname. The user can overwrite this with a nickname of their choice via DeB at on-boarding time or at a later date.	<ul style="list-style-type: none"> • Contains text • The <i>tr=</i> attribute is used if support for multiple languages is required. • No attributes 	<ul style="list-style-type: none"> • One and one only • DeB displays this nickname and the default icon to the user as a label, unless the user chooses to overwrite the manufacturer-defined nickname • If only one language is supported, the

Child elements of <description>	Mandatory / optional	Purpose	Sub-elements and attributes	How used
				<p>nickname is specified directly in the <nickname> element:</p> <pre><nickname>UltraPlug</nickname></pre> <ul style="list-style-type: none"> The <i>tr=</i> attribute provides multi-lingual support: <pre><nickname tr='UPLG'>UltraPlug</nickname></pre> <p>The value associated with <i>tr=</i> represents a key that is used to look up the associated Unicode string from the external language file. DeB determines the display language to use from the locale setting of the operating system of the smartphone. Refer to section 2.3.3.</p>
<theme>	M	If a Thing consists of multiple hardware units or provides multiple user interfaces (e.g. basic, standard, advanced), there will be a separate RML file for each unit and each interface. Each file is known as a theme.	<ul style="list-style-type: none"> Contains text The <i>tr=</i> attribute is used if support for multiple languages is required No attribute 	<ul style="list-style-type: none"> One and one only The theme name is the name that will be displayed to the user in DeB to identify the individual hardware unit or interface If only one language is supported, the text is specified directly in the <theme> element: <pre><theme>Standard</theme></pre> <ul style="list-style-type: none"> The <i>tr=</i> attribute provides multi-lingual support. <pre><theme tr='STD'>Standard</theme></pre> <p>The value associated with <i>tr=</i> represents a key that is used to look up the associated Unicode string from the external language</p>

Child elements of <description>	Mandatory / optional	Purpose	Sub-elements and attributes	How used
				file. DeB determines the display language to use from the locale setting of the operating system of the smartphone. Refer to section 2.3.3 .
<version>	M	This is the version number of this specific RML file.	<ul style="list-style-type: none"> • No sub-elements • No attributes • Contains text • The format is defined by the manufacturer • The major.minor.release format is recommended 	<ul style="list-style-type: none"> • One and one only • For the manufacturer to control versioning of the RML files for a specific Thing
<url>	O	Defines a URL to a webpage that the manufacturer has provided for the Thing.	<ul style="list-style-type: none"> • No sub-elements • No attributes • Contains text that is a URL • The URL is assumed to begin with "http://www." 	<ul style="list-style-type: none"> • Zero or one • If DeB is not installed, the phone's web browser will start and go to this link when the phone is tapped on the Thing • If DeB is installed, the link will appear on the proximity page that DeB displays to the user • The user can follow the link to access the information that the manufacturer has provided. Can be used for information, marketing purposes, registration, and so on

Table 2: Description section - child elements

3.2.3 RML example

Example RML for the <description> element for an UltraPlug is:

```
<description>
  <manufacturer>Xped</manufacturer>
  <model>PSW-240AU1</model>
  < category>8003</ category>
  <version>1.0</version>
  <theme tr='STD'>Standard</theme>
  <nickname tr='UPLG'>UltraPlug</nickname>
  <class>http://ontology.xped.com/1.0/home/psw.owl#electricity</class>
  <url>xped.com/products/ultraplug.htm</url>
</description>
```

The example RML provides the following metadata for the UltraPlug:

- Manufactured by **Xped**
- The model is known as **PSW-240AU1**
- The manufacturer is using the default icon for a smart plug (**8003**) that is provided in the IoTG
 - > **Note:** this may be over-written later in the RML file by the <resource> element
- This is version **1.0** of the RML file for the Thing
- There is one RML file that describes one user interface called **Standard** for the UltraPlug
 - > The language used by the smartphone is checked against the external language file for the Thing and, where required, the text string of **Standard** will be displayed in the language used by the smartphone
 - > In this example, there is only RML file and this interface is called Standard. As there is only one interface, DeB does not display the term 'Standard' to the user
- If there are multiple RML files for a Thing to define multiple interfaces (Standard, Advanced, and so on), the device screen in DeB will contain a menu with menu items named Standard, Advanced and so on
- The manufacturer has given a nickname of **UltraPlug** for the Thing.
 - > The language used by the smartphone will be checked against the external language file for the Thing and the text string of **UltraPlug** will be displayed in the language used by the smartphone
 - > This together with the default icon forms the label that will be shown by DeB, unless the user uses the option in DeB to override the nickname
- The manufacturer has provided a link to an external ontology that can be found at: <http://ontology.xped.com/1.0/home/psw.owl#electricity>
- The manufacturer has provided a link to an external website for further information, registration, or other, that can be found at: <http://www.xped.com/products/ultraplug.htm>

3.3 Model element

The **<model>** element is a child element of the **<rml>** element. It can be considered to be the API for the Thing.

This element describes the Thing's functionality that can be controlled by a user, the possible control settings and ranges, and the commands that can be sent to a Thing.

This element defines:

- An identifier for each individual setting
- Commands that can be sent to the Thing to access or change an item of state
- The values and measurement units for each setting
- Any qualifiers, such as read-only, read-write
- A reference to an external ontology, if available

3.3.1 Usage

There is one and only one **<model>** element in an RML file.

It is a mandatory element.

3.3.2 Child elements and attributes

The child elements of the **<model>** element that can be used to describe the individual items of state for a Thing are:

<model>

<enum>

defines individual items of state that can be represented by a small finite set of integers, typically 2-10, e.g. on/off

</enum>

<range>

defines individual items of state that can be represented by a range of integers, e.g. minimum to maximum values with a step

</range>

<text>

defines individual items of state that can be represented by a text string, e.g. a read-only display or some editable text

</text>

<trigger>

defines a command with no parameters that typically triggers an action, e.g. stop, play, pause

</trigger>

<record>

defines an aggregate type that can contain **<enum>**, **<range>** and **<text>** elements, e.g. can be used to report on the complete state of a Thing

</record>

</model>

The **<model>** element must include at least one of the **<enum>**, **<range>**, **<text>**, **<trigger>** or **<record>** child elements shown above.

These elements are described in [Table 3](#).

IMPORTANT: The id= attribute in each of the <enum>, <range>, <text>, <trigger> and <record.> child elements in the <model> main element> links to the bind= attributes in the elements that define the widgets in the <view> main element.

This linkage enables the <view> element to display the user interface that reflects the functionality defined in the <model> element.

Child elements of <model>	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 4 and Table 5)	How used
<enum>	Must have at least one of : <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> 	Defines the individual states of a Thing that can be represented by a small finite set of integers, e.g. on/off switch, short list of values.	<u>Sub-elements</u> <ul style="list-style-type: none"> • <item> • <units> • <class> • <cache> <u>Attributes</u> <ul style="list-style-type: none"> • id= • path= • mode= • timeout= • state= 	<p>The <enum> element binds to different widget elements defined in the <view> element depending upon the number of values that the <enum> element represents:</p> <ul style="list-style-type: none"> • binary enum (2 values) that binds to: <ul style="list-style-type: none"> ○ an on/off <slideshow> widget or ○ a checkbox <checkbox> widget • small enum (up to 4 values) that binds to: <ul style="list-style-type: none"> ○ an <rgbled> widget ○ often used for a hardware or fault indicator (e.g. off, red, green, blue) • enum (2 or more values) that represents a text box with a popup scrollable list and binds to: <ul style="list-style-type: none"> ○ a <valuepicker> widget ○ e.g. wash cycles for a washing machine
<range>	Must have at least one of : <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> 	Defines the individual states of a Thing that can be represented by a range of integers from a minimum value to a maximum value incremented by a step, e.g. sound volume, temperature range.	<u>Sub-elements</u> <ul style="list-style-type: none"> • <units> • <class> • <cache> <u>Attributes</u> <ul style="list-style-type: none"> • id= • path= • mode= • timeout= 	<p>The <range> element binds to different widgets in the <view> element depending upon the range of values that the <range> element represents:</p> <ul style="list-style-type: none"> • small ranges are represented by a slider with up/down buttons <ul style="list-style-type: none"> ○ the <spinslider> element • large ranges are represented by <ul style="list-style-type: none"> ○ the <dial> element for read only states ○ the <slider> element for read/write states ○ the <dial> and <slider> elements together

Child elements of <model>	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 4 and Table 5)	How used
			<ul style="list-style-type: none"> • state= • min= • max= • step= 	
<record>	Must have at least one of : <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> 	Defines a collection type that can contain <enum>, <range> and <text> elements, each of which represent an individual state of the Thing.	<u>Sub-elements</u> <ul style="list-style-type: none"> • <enum> • <range> • <text> • <class> • <cache> • <source> <u>Attributes</u> <ul style="list-style-type: none"> • id= • path= • mode= • timeout= 	The <record> element can be used to query or report the complete state of a Thing or to query subsets of the complete state.
<text>	Must have at least one of : <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> 	Defines text that can be displayed to the user or entered by the user.	<u>Sub-elements</u> <ul style="list-style-type: none"> • <units> • <class> • <cache> <u>Attributes</u> <ul style="list-style-type: none"> • id= • path= 	The <text> element binds to one of 3 types of widgets in the <view> element depending upon whether the text is display only or can be edited: <ul style="list-style-type: none"> • display-only dynamic text from a Thing binds to: <ul style="list-style-type: none"> ◦ the <label> element • a single line of editable text binds to: <ul style="list-style-type: none"> ◦ the <textline> element • a larger area of editable text that supports HTML

Child elements of <model>	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 4 and Table 5)	How used
			<ul style="list-style-type: none"> • mode= • timeout= • state= • length= 	formatting and binds to: <ul style="list-style-type: none"> ○ the <textbox> element <p>Note: Static text contained in the RML file is not defined by this <text> element. Static text is defined within the <view> element by a <label> child element without a <i>bind=</i> attribute.</p>
<trigger>	Must have at least one of : <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> 	Defines a command with no parameters that typically triggers an action, e.g. stop, play, pause, display menu, and so on.	<u>Sub-elements</u> <ul style="list-style-type: none"> • <class> <u>Attributes</u> <ul style="list-style-type: none"> • id= • path= • mode= • state= • 	The <trigger> element binds to the <button> widget in the <view> element.

Table 3: Model section - child elements

The following table ([Table 4](#)) further defines the sub-elements listed in the previous table.

Sub-element	Mandatory / optional	Purpose	Attributes or value	How used
<cache>	Zero or one for each of the following elements: <ul style="list-style-type: none"> • <enum> • <range> • <text> • <record> 	<p>Defines the caching parameters for an item of state or a record.</p> <p>The <i>etag=</i> attribute specifies whether the cached entry should be tested for staleness using an etag.</p> <p>The <i>maxage=</i> attribute specifies the period of time in seconds that must elapse after which the cached value will be considered to be stale.</p> <p>Default is 60 seconds.</p>	<p>etag= valid values are:</p> <ul style="list-style-type: none"> • #true • #false <p>maxage= valid values are:</p> <ul style="list-style-type: none"> • Integer (default = 60, 0 means never stale) 	<p>In order to make network operations as efficient as possible, the Internet of Things Gateway (IoTG) may cache items of state for a Thing and satisfy client requests directly from the cache rather than communicating with a Thing over the network.</p> <p>When a client requests an item of state from a Thing and it exists in the cache and its age is less than <i>maxage=</i>, the cached response will be returned to the client.</p>
<class>	Zero or one for each of the following elements: <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> 	Refers to an external ontology (OWL class) that defines the Thing.	Contains text that is a URL	Allows a manufacturer to reference and use open-source device descriptions.

<item>	<p>One or many for each of the following elements:</p> <ul style="list-style-type: none"> • <enum> 	<p>There is an <item> element for each of the items that make up an enumeration.</p> <p>Each <item> defines a key-value pair with the <i>tr=</i> and <i>value=</i> attributes.</p>	<p><i>value=</i> integer</p> <p><i>tr=</i> alphanumeric text</p>	<p>Defines the individual settings that apply for the function defined by the <enum> element.</p> <p>If only one language is supported, the name of the key is specified in the <i>name=</i> attribute.</p> <p>Example: <item value='0'>Low</item> <item value='1'>High</item></p> <p>The <i>tr=</i> attribute provides multi-lingual support.</p> <p>Example: <item tr='Low' value='0'/> <item tr='High' value='1'/></p> <p>The value associated with the <i>tr=</i> attribute represents a key that is used to look up the associated Unicode string from the external language file. DeB determines the display language to use from the locale setting of the operating system of the smartphone. Refer to section 2.3.3.</p> <p>The <i>value=</i> attribute defines the values for a key-value pair. When the user selects one of the settings in the key-value pair, the value associated with the <i>value=</i> attribute is appended to the command in the <i>path=</i> attribute defined in the <enum> element and sent to the Thing.</p>
<source>	Optional	<p>Defines the source of the record if retrieved from a network service rather than from the device.</p>	<p><i>service=</i> <i>path=</i></p>	<p>The <i>service=</i> attribute specifies the name of the service providing the record as advertised via mDNS-SD.</p> <p>The <i>path=</i> attribute specifies the path provided by the service that responds with the record.</p>

<units>	<p>Zero or one for each of the following elements:</p> <ul style="list-style-type: none"> • <enum> • <range> • <text> 	<p>Defines the units for an item of state.</p>	<p>system= valid values are:</p> <ul style="list-style-type: none"> • #SI (International System of Units) • #derived (default) 	<p>The <i>system=</i> attribute specifies which system the unit of measure belongs to.</p>
			<p>symbol= valid values are:</p> <ul style="list-style-type: none"> • text 	<p>The <i>symbol=</i> attribute specifies the symbol used to represent the unit and this is displayed to the user.</p>
			<p>type= valid values are:</p> <ul style="list-style-type: none"> • #integer • #float (default is #integer) 	<p>The <i>type=</i> attribute specifies the numeric type used to represent the item of state.</p>
			<p>exponent= valid values are:</p> <ul style="list-style-type: none"> • integer (default is 0) 	<p>The <i>exponent=</i> attribute specifies the scale factor that has been applied to the item of state.</p> <p>The scale factor is used by DeB to derive the symbol to display to the user. For example, for <i>exponent=3</i> and <i>symbol=Wh</i>, DeB displays kWh to the user.</p> <p>The following example shows an item of state reported in Kilowatt hours:</p> <pre><range id="energy" path="/egy" min="0" max="4294967" mode="#readonly"> <units system="#derived" symbol="Wh" type="#integer" exponent="3" >Watt Hour</unit> </range></pre>

Table 4: Model section - sub-elements

The following table ([Table 5](#)) further defines the attributes listed in the previous tables.

Attribute	Mandatory / optional	Purpose	Value	How used
id=	One for each of the following elements: <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> 	Provides an identity for the item of state defined by the element.	alphanumeric text	Must be unique within the RML document. Used as the value of <i>bind=</i> attributes in the <view> element to bind the item of state to the required display widget in the user interface.
length=	One for each of the following elements: <ul style="list-style-type: none"> • <text> 	Specifies the maximum number of characters allowed in the <text> element.	Integer (default is unlimited)	Set maximum length for text.
max=	One for each of the following elements: <ul style="list-style-type: none"> • <range> 	Specifies the highest value in a numeric range.	Integer	Maximum value for a range of values.
min=	One for each of the following elements: <ul style="list-style-type: none"> • <range> 	Specifies the lowest value in a numeric range.	Integer	Minimum value for a range of values.
mode=	One for each of the following elements: <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> 	Specifies the access mode of the command that is defined by the <i>path=</i> attribute for the item of state defined by the element.	#readonly #writeonly #readwrite #disabled	Identifies whether a command sent to a Thing has read, write or read/write access, or whether it is disabled. <i>#disabled</i> is used in template RML files that have a generic model section and when a model item is not valid for a specific kind of device.

Attribute	Mandatory / optional	Purpose	Value	How used
path=	<p>One for each of the following elements:</p> <ul style="list-style-type: none"> • <enum> • <range> • <text> • <trigger> • <record> <p>Note: for the <record> collection element, there is one <i>path=</i> attribute in the <record> element and no <i>path=</i> attributes for the child elements within the <record> element</p>	<p>The command that is used to control an item of state.</p> <p>Specifies the interface and selector of the RCP path that will be sent to the Thing for the item of state defined by the element.</p>	Any valid path without the query portion	<p>Used to send a command to a Thing.</p> <p>DeB automatically constructs the complete command required to interact with a Thing from the information contained in the <i>path=</i> attribute and the setting value entered by the user. The value entered by the user will be determined by the values allowed by the <i>value=</i>, <i>min=</i>, <i>max=</i> or <i>length=</i> attributes.</p> <p>Example:</p> <pre><model> <range id ='volume' path='/vol' min='0' max='99'/> </model></pre> <p>In this example the Thing has an item of state that represents the playback volume. The path to that item of state is '/vol'. So to set the volume to 25, the command '/vol?25' would be sent to the device.</p>
service=	<p>One for each of the following elements:</p> <ul style="list-style-type: none"> • <source> 	Specifies the mDNS-SD name of the service.	Text	<p>In this example a service with the name <i>_uap._tcp</i> is being advertised using mDNS-SD (ZeroConf). In this case the service attribute will be set to:</p> <pre>service="_uap._tcp"</pre>
state=	<p>Zero or one for each of the following elements:</p> <ul style="list-style-type: none"> • <enum> • <range> • <text> 	A hint that specifies the importance placed on each model item.	#primary #secondary #configuration	<p>Used to determine which of the model items, if any, will appear in the home screen item for the Thing.</p> <p>Example 1:</p> <pre><model> <range id ='volume' path='/vol' min='0' max='99' mode='#readwrite' state='#primary'/> </model></pre>

Attribute	Mandatory / optional	Purpose	Value	How used
				<p>In example 1 a slider for the volume control will be added to the Thing's home screen item.</p> <p>Example 2:</p> <pre><model> <range id ='power path='/pwr' min='0' max='2400' mode='#readonly' state='#secondary'/> </model></pre> <p>In example 2 a text field to display the value of the state will be added to the Thing's home screen item.</p>
step=	One for each of the following elements: <ul style="list-style-type: none"> • <range> 	Specifies the increment step in a numeric range.	Integer	The increment used within a range of values.
timeout=	Zero or one for each of the following elements: <ul style="list-style-type: none"> • <enum> • <range> • <text> • <record> Default is 100 milliseconds	Specifies the maximum amount of time that may elapse until a response is received to the command that is defined in the <i>path=</i> attribute.	Integer representing milliseconds	If a response is not received within the timeframe, the command is considered to have failed.

Table 5: Model section - attributes

3.3.3 RML example

The example RML for the <model> element for an LED light bulb is:

```
<model>
  <enum id="switch" path="/on">
    <item tr="OFF" value="0">Off</item>
    <item tr="ON" value="1">On</item>
  </enum>
  <range id="dimmer" path="/dim" min="10" max="100" step="1" />
  <range id="color" path="/clr" min="0" max="16777215"/>
  <range id="energy" path="/egy" min="0" max="4294967296" step="1" mode="#readonly">
    <units system="#derived" symbol="Wh" type="#integer" exponent="0">Watt hour</units>
    <class>http://sweet.jpl.nasa.gov/2.2/quanEnergy.owl#Energy</class>
  </range>
  <trigger id="effect" path="/eff" mode="#writeonly" />
  <text id="mfgdate" path="/mdt" mode="#readonly" />
</model>
```

This example RML defines the following items of state:

- <enum>
 - > id="switch" is the path for setting the on/off state of the bulb with the value '0' representing Off.
- <range>
 - > id="dimmer" is the path for setting the dimming state of the light bulb. The minimum and maximum values and step increment are defined.
 - > id="color" is the path for setting the color state. The minimum and maximum values are defined.
 - > id="energy" is the path for reading the amount of energy the bulb has consumed.
- <trigger>
 - > id="effect" is the path for cycling through the special effects the bulb supports. Each time the trigger is activated the next effect is selected e.g. none -> slow blink -> fast blink -> SOS -> none.
- <text>
 - > id="mfgdate" is the path for reading the date the bulb was manufactured.

3.4 View element

The **<view>** element is a child element of the **<rml>** element.

The **<view>** element describes a user interface for the Thing. A Thing may have none or many user interfaces and each interface can be tailored to a particular screen size and resolution such as a smart watch, mobile phone or tablet.

This element defines:

- Orientation and layout of the screens
- Screen resolution
- Control widgets – switches, buttons, dials, sliders, checkboxes and so on
- Text – labels, display text, editable text areas
- How the user interface *binds* to the model element

3.4.1 Usage

An RML file may have zero, one or multiple **<view>** elements. This depends upon whether the ADRC client has a user interface or not and whether the interface needs to cater for different screen types and layouts. Each **<view>** element can specify a layout for a particular screen size and resolution.

3.4.2 Child elements and attributes

The child elements of the **<view>** element are summarised below in terms of elements used to:

- Determine the layout of the user interface
- Represent the individual display widgets on the screen

Layout elements

The layout elements are referred to collectively as *#layout-widgets* in some of the documentation.

How the **<view>** element and its child elements determine the layout of the user interface for a Thing can be summarised as:

<view>

Defines the screen size, resolution and orientation for the entire user interface.

Note: only portrait orientation and native size and resolution are supported in RML Lite.

<screen>

Identifies an individual screen within the user interface. There must be at least one screen element.

<devicelist>

The top-level container for the screen that is used to control a Thing. This container has a header that automatically identifies the Thing by its icon and nickname and shows the signal strength of the Thing. Optionally, the header can display one key item of state such as the amount of power the Thing is consuming.

<devicelistitem>

The container that provides the first level of layout for the display widgets that make up part of the interface for a Thing. It defines the control widget(s) and the default layout for this container. There is usually one **<devicelistitem>** for each item of state

Each **<devicelistitem>** automatically creates a metastate indicator for the control widget to show whether a control command has been successful or not.

<box>

Sub-container that is used to change the layout for a widget or group of widgets within a `<devicelistitem>` element.

There can be multiple `<box>` elements within a `<devicelistitem>` element and `<box>` elements within `<box>` elements

`</box>`

`<devicelistitem>`

`</devicelist>`

`</screen>`

`</view>`

The `<spacer>` element can be used within any of the containers to provide spacing in a layout.

Display elements

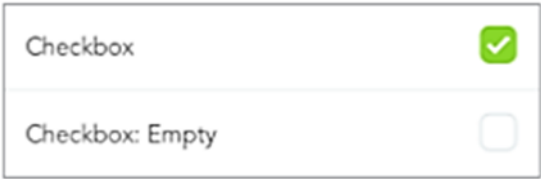

The display elements are referred to collectively as `#control-widgets` in some of the documentation.

Display elements can be grouped into:

- enumeration widgets
- range widgets
- trigger widgets
- text widgets
- special widgets

These widget types are shown in the following table ([Table 6](#)).

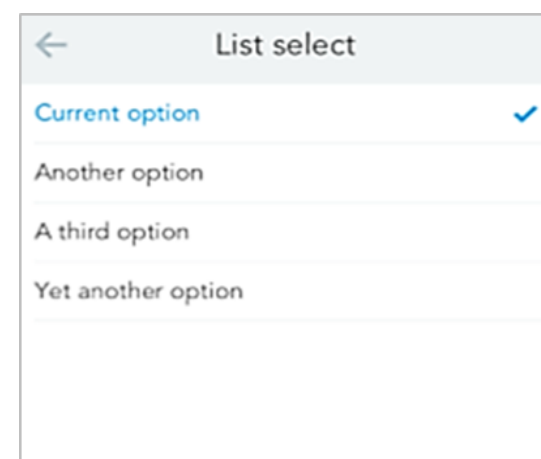
Enumeration widgets

Element	Type	Usage example	Icon example
<code><checkbox></code>	binary enum	yes/no conditions	
<code><rgbled></code>	small enum, up to 4 values	hardware fault status indicator that changes color	

<slideshow> binary enum on/off switch




<valuepicker> large enum list of options



Range widgets

Element	Type	Usage example	Icon example
<dial>	large display only ranges	display sensor measurements such as temperature	<div>Temperature</div> <div>Humidity</div>
<slider>	large ranges controlled by a touch track	LED bulb dimmer with values between 0 and 99	
<spinslider>	small ranges controlled by up/down buttons	brightness indicator with values between 0 and 15	


Trigger widgets

Element	Type	Usage example	Icon example
<button>	push button type	step through a sequence or trigger an action	

Text widgets

Element	Type	Usage example	Icon example
<label>	non-editable text	static text from an RML file (no <i>bind=</i> attribute) dynamic text from a Thing (with <i>bind=</i> attribute)	n/a
<textbox>	large area of editable text	supports HTML formatting e.g. color, highlighting	n/a
<textline>	single line of editable text	single line of text, no formatting options	n/a

Special widgets

Element	Type	Usage example	Icon example
<colorpicker>	color and brightness controls	select color and brightness for lightbulbs, LED devices	

<infrared> infrared
remote control
widget remote control
for TV, music
player, air
conditioner and
so on



Table 6: Display widgets

The layout elements and the display elements are defined in more detail in the tables below - [Table 7](#) and [Table 8](#).

Element name	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 8 and Table 9)	How used
Main element				
<view>	Zero =no user interface for the client One = one interface supported Multiple = multiple interfaces supported, such as mobile phone, smart watch, tablet	Container for a set of screens that provide a user interface for a Thing.	<u>Sub-elements</u> <ul style="list-style-type: none"> At least one <screen> element <u>Attributes</u> <ul style="list-style-type: none"> footprint= density= orientation= 	<p>Defines the user interface for a Thing.</p> <p>The attributes specify the screen orientation (portrait or landscape) and provide an indication of the screen size (<i>footprint=</i>), resolution (<i>density=</i>).</p> <p>Note: In the initial release of RML (RML Lite), only portrait mode is supported (<i>orientation='#vertical'</i>).</p>
Layout elements				
<box>	Zero to many	Is a container for other containers and widgets. Its main purpose is to control the layout of the widgets.	<u>Sub-elements</u> <ul style="list-style-type: none"> <box> <spacer> #control-widgets (display elements) <u>Attributes</u> <ul style="list-style-type: none"> layout= 	Used to group widgets together and/or change the layout from the default layout specified for the <deviceListItem>.

Element name	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 8 and Table 9)	How used
<devicelist>	One for each <screen> element	The <devicelist> element creates a title area for the screen. It is the top-level container for the other containers (<devicelistitem> and <box>) and the control widgets that determine what is displayed on the screen.	<u>Sub-elements</u> <ul style="list-style-type: none"> <devicelistitem> 	<p>Displays the heading for a Thing and a container that holds the <devicelistitem> and <box> elements that make up the user interface.</p> <p>The <devicelist> element automatically creates a title area for the screen. This title area displays:</p> <ul style="list-style-type: none"> the icon for the Thing (displayed automatically based on the icon defined in either the <resource> element (<i>icon=</i>) or the <category> element a signal strength indicator (displayed automatically) a title, which is set to the nickname of the device
<devicelistitem>	At least one for each <devicelist> element	<p>Container for control widgets.</p> <p>Usually there is one <devicelistitem> element for each item of state for a Thing. For example, a lamp might have two items of state:</p> <ul style="list-style-type: none"> an on/off switch a dimmer capability 	<u>Sub-elements</u> <ul style="list-style-type: none"> <title> <box> <spacer> <graph> #control-widgets (display elements) 	<p>Specifies the control widget(s) for an item of state and the default layout (<i>layout=</i> attribute) for this container.</p> <p>If a <title> element is present the <devicelistitem> uses the associated text as its title.</p>

Element name	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 8 and Table 9)	How used
			<u>Attributes</u> <ul style="list-style-type: none"> • bind= • layout= 	<p>Each <devicelistitem> automatically displays a meta-state indicator that makes visible the state of the communication between that widget and the device. When no meta-state indicator is visible, the on-screen state of the widget reflects the true state of the device. If the meta-state indicator is yellow, communication between the widget and the device has not been confirmed, indicating that the display state of the widget is uncertain. If the meta-state indicator is red, communication between the widget and the device has failed and the display state of the widget is unknown.</p> <p>The <devicelistitem> can have an internal layout that is either vertical or horizontal. This governs the way its child elements are displayed.</p>
<graph>	Zero or one	Graphing widget	<u>Sub-elements</u> <ul style="list-style-type: none"> • <xaxis> • <yaxis> • <title> 	<p>Binds to an id in the model element, which indicates the location of the data source.</p> <p><xaxis> specifies the field name in</p>

Element name	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 8 and Table 9)	How used
			<u>Attributes</u> <ul style="list-style-type: none"> bind= 	<p>the record that provides the x-axis data.</p> <p><yaxis> specifies the field name in the record that provides the y-axis data.</p> <p><title> Defines a text title for the graph.</p>
<screen>	One or many	<p>Represents an individual screen within the user interface.</p> <p>A screen occupies the entire physical display area.</p>	<u>Sub-elements</u> <ul style="list-style-type: none"> <devicelist> <u>Attributes</u> <ul style="list-style-type: none"> name= 	<p>A screen displays the control widgets for a Thing. These controls are specified by the <devicelistitem> elements within <devicelist> within <screen>.</p> <p>If a Thing has many controls, more than one screen may be needed to display these controls to the user. Each screen is identified by a <screen> element with a different <i>name=</i> attribute.</p> <p>The text value in <i>name=</i> is used by DeB to display a navigation tab to allow the user to select amongst the screens.</p>
<spacer>	Zero to many	Used by the <devicelistitem> and <box> container elements to define the relative	<u>Attributes</u> <ul style="list-style-type: none"> stretch= 	The <i>stretch=</i> attribute specifies an

Element name	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 8 and Table 9)	How used
		spacing between items within the same container.		<p>integer value.</p> <p>The <i>stretch=</i> attribute defines the relative sizing between this item and the other items in the container.</p> <p>A stretch can be thought of as spring with tension of 'stretch' units. For example, if there are two items in a layout, one with stretch set to 2 and the other set to 3 then the layout manager will attempt to allocate the first item 2/5 of the space and the second item 3/5 of the space.</p>
<xaxis>	One for each <graph> element	Defines the data field in the source record and axis label for the x-axis.	<u>Attributes</u> <ul style="list-style-type: none"> • bind= • tr= 	<p>The <i>bind=</i> attribute specifies the data field in the source record of the model item to use for the x-axis data.</p> <p>The text of the <xaxis> element provides the label for the axis.</p> <p>The <i>tr=</i> links to an external language file to provide multi-language support. Refer to <i>tr=</i> in section 2.3.3.</p>

<yaxis>	One for each <graph> element	Defines the data field in the source record and axis label for the y-axis.	<u>Attributes</u> <ul style="list-style-type: none"> • bind= • tr= 	<p>The <i>bind=</i> attribute specifies the data field in the source record of the model item to use for the y-axis data.</p> <p>The text of the <yaxis> element provides the label for the axis.</p> <p>The <i>tr=</i> links to an external language file to provide multi-language support. Refer to <i>tr=</i> in section 2.3.3.</p>
---------	------------------------------	--	--	---

Table 7: View section – layout elements

The display elements are defined in more detail in the following table ([Table 8](#)).

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
Display widgets				
<button>	Zero to many	<p>Is a push button type that has a range of skins that make it suitable for many applications, including audio-visual equipment and screen navigation.</p> <p>It is a trigger type element and, therefore, cannot display state.</p>	<p><u>Attributes</u></p> <ul style="list-style-type: none"> • bind= • skin= • tr= • scale= • aspect= 	<p>Binds to a <trigger> element in the <model> element, where the functionality is defined.</p> <p>The value of the <i>bind=</i> attribute is the value specified in the <i>id=</i> attribute of the <trigger> element.</p> <p>A <button> widget is used to perform an action such as stop, play, pause and screen navigation.</p> <p>There are 20+ in-built skins for the buttons.</p> <p>A text label can be specified for the button using the <i>tr=</i> attribute or the text area of the button element.</p> <p>Example</p> <pre><button tr='OK' bind='ok'>Ok</button></pre> <p>The text area is used for single language support and the <i>tr=</i> attribute is used for multi-lingual support.</p> <p>The <i>tr=</i> links to an external language file to provide multi-language support. Refer to <i>tr=</i> in</p>

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
				<p>section 2.3.3.</p> <p>The width and height of a button can be specified in the <i>aspect=</i> attribute</p> <p>The size can also be controlled through the <i>scale=</i> attribute, which scales the size calculated by the layout manager.</p>
<checkbox>	Zero to many	Toggle type that is suitable for items of state that exhibit a binary state, such as any type of true/false condition.	<u>Attributes</u> <ul style="list-style-type: none"> • <i>bind=</i> • <i>tr=</i> 	<p>Binds to an <enum> element in the <model> element, where the functionality is defined.</p> <p>The value of the <i>bind=</i> attribute is the value specified in the <i>id=</i> attribute of the <enum> element.</p> <p>A <checkbox> widget is used to display a binary enum, such as yes/no conditions.</p> <p>The text associated with the checkbox is specified by the <i>tr=</i> attribute or the text area of the element.</p> <p>Example:</p> <pre><checkbox tr='TRM' bind='terms'>I accept the terms</checkbox></pre>

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
				The <i>tr=</i> links to an external language file to provide multi-language support. Refer to <i>tr=</i> in section 2.3.3.
<colorpicker>	Zero to many	A widget that allows a color value to be selected graphically by touching it on the screen.	<u>Attributes</u> <ul style="list-style-type: none"> • <i>bind=</i> 	<p>Binds to a <range> element in the <model> element, where the functionality is defined.</p> <p>Used for items of state that represent a color.</p> <p>For example, for an LED light bulb, the user may be able to set the color of the light.</p> <p>The color is represented as an integer in the range 0 to $2^{24}-1$.</p>
<dial>	Zero to many	A <dial> is a composite analog and numeric display type that can be used to display items of state that exhibit a continuous and large range of values.	<u>Attributes</u> <ul style="list-style-type: none"> • <i>bind=</i> • <i>caption=</i> • <i>dial-color=</i> • <i>text-color=</i> • <i>tr=</i> 	<p>Binds to a <range> element in the <model> element, where the functionality is defined.</p> <p>The value of the <i>bind=</i> attribute is the value specified in the <i>id=</i> attribute of the <range> element.</p> <p>A <dial> widget is used to display a range of values to the user.</p> <p>A dial widget can be paired with a <slider> widget to provide the user with a touch track capability and with a <spinslider> widget to</p>

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
				<p>provide the user with up/down control buttons.</p> <p>If a label is required, the syntax <code><dial>dial label</dial></code> can be used.</p>
<code><infrared></code>	Zero or one	A special type for devices that have infrared remote controls.	<u>Attributes</u> <ul style="list-style-type: none"> • <code>bind=</code> • <code>skin=</code> 	<p>Binds to multiple trigger items in the model section using the extended bind syntax.</p> <p>Please refer to the extended bind syntax described in section 3.4.3.</p> <p>The skin attribute selects which appliance category is drawn. Each category has buttons applicable to that type of device.</p>
<code><label></code>	Zero to many	A text widget that can display static or dynamic text.	<u>Attributes</u> <ul style="list-style-type: none"> • <code>bind=</code> • <code>tr=</code> • <code>length=</code> • <code>wrapping=</code> • <code>text-font=</code> • <code>text-color=</code> • <code>text-size=</code> • <code>text-bold=</code> • <code>text-italic=</code> 	<p>A <code><label></code> widget can be used to display:</p> <ul style="list-style-type: none"> • static text, if the <code>bind=</code> attribute is not used • dynamic text from the Thing, if the <code>bind=</code> attribute is used <p>Where the <code>bind=</code> attribute is used, it binds to a <code><text></code> element in the <code><model></code> element.</p> <p>The value of the <code>bind=</code> attribute is the value specified in the <code>id=</code></p>

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
				<p>attribute of the <text> element.</p> <p>Example</p> <pre><label tr='EJECT'>Eject</label></pre> <p>The format of the label is defined by the <i>length=</i>, <i>wrapping=</i> and <i>text-xxx=</i> attributes, refer to Table 9.</p>
<rgbled>	Zero to many	A display-only RGB LED indicator. Can be used for hardware indicators, such as a fault status indicator.	<u>Attributes</u> <ul style="list-style-type: none"> • <i>bind=</i> • <i>enum=</i> 	<p>Binds to an <enum> element in the <model> element, where the functionality is defined.</p> <p>The value of the <i>bind=</i> attribute is the value specified in the <i>id=</i> attribute of the <enum> element.</p> <p>An <rgbled> widget is used to display a small enum of up to 4 values.</p> <p>The <i>enum=</i> attribute is used to define the colors associated with individual states specified in the <enum> element that <rgbled> binds to.</p> <p>.</p>
<slider>	Zero to many	A <slider> is a slider type that can be used to display and control an item of	<u>Attributes</u> <ul style="list-style-type: none"> • <i>bind=</i> 	Binds to a <range> element in the <model> element, where the

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
		state with a large range of integer values.	<ul style="list-style-type: none"> orientation= 	<p>functionality is defined.</p> <p>The value of the <i>bind=</i> attribute is the value specified in the <i>id=</i> attribute of the <range> element.</p> <p>A <slider> widget is used to display a large range of values in the format of a touch track.</p> <p>The touch track can be displayed either vertically or horizontally (<i>orientation=</i> attribute).</p> <p>A writeonly <slider> widget can also be used together with a <dial> widget so as to provide the user with a numeric display.</p> <p>If a label is required, a separate <label> element is used.</p>
<slideshow>	Zero to many	Toggle type that is suitable for items of state that exhibit a binary state, such as any type of two position switch.	<u>Attributes</u> <ul style="list-style-type: none"> bind= tr= 	<p>Binds to an <enum> element in the <model>element, where the functionality is defined.</p> <p>The value of the <i>bind=</i> attribute is the value specified in the <i>id=</i> attribute of the <enum> element.</p> <p>A <slideshow> widget is used to display a binary enum, such as an on/off switch.</p> <p>The labels for each of the binary</p>

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
				<p>states are specified in the <i>tr=</i> or text area of the model item.</p> <p>The text of the slideswitch element is used for single language support and the <i>tr=</i> attribute is used for multi-lingual support.</p> <p>For example</p> <pre><slideswitch bind='switch' tr='ONOFF'>On,Off</slideswitch></pre> <p>The <i>tr=</i> attribute links to an external language file to provide multi-language support. Refer to <i>tr=</i> in section 2.3.3.</p>
<spinslider>	Zero to many	A <spinslider> is a composite analog and numeric display type with up/down buttons that is suitable for a small range of values.	<u>Attributes</u> <ul style="list-style-type: none"> bind= 	<p>Binds to a <range> element in the <model> element, where the functionality is defined.</p> <p>The value of the <i>bind=</i> attribute is the value specified in the <i>id=</i> attribute of the <range> element.</p> <p>A <dial> widget is used to display a small range of values, which are controlled using up/down buttons.</p> <p>If a label is required, a separate <label> element is used.</p>

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
<textbox>	Zero to many	A text widget that allows multiple lines of text to be displayed and edited.	<u>Attributes</u> <ul style="list-style-type: none"> • bind= • length= • wrapping= • text-font= • text-color= • text-size= • text-bold= • text-italic= 	<p>Used for items of state that can be represented by text values.</p> <p>The contents can be plain text or can contain HTML formatting tags.</p> <p>The format of the text is defined by the <i>length=</i>, <i>wrapping=</i> and <i>text-xxx=</i> attributes, refer to Table 9.</p>
<textline>	Zero to many	A text widget that allows a single line of text to be displayed and edited.	<u>Attributes</u> <ul style="list-style-type: none"> • bind= • length= • text-font= • text-color= • text-size= • text-bold= • text-italic= 	<p>Used for items of state that can be represented by text values.</p> <p>The format of the text is defined by the <i>length=</i> and <i>text-xxx=</i> attributes, refer to Table 9.</p>
<valuepicker>	Zero to many	A text box with a popup scrollable list.	<u>Attributes</u> <ul style="list-style-type: none"> • bind= • text-font= • text-color= • text-size= • text-bold= • text-italic= 	<p>Binds to an <enum> element in the <model> section, where the functionality is defined.</p> <p>The value of the <i>bind=</i> attribute is the value specified in the <i>id=</i> attribute of the <enum> element.</p> <p>A <valuepicker> widget is used to display a large enum of values within a popup scrollable list.</p>

Element name	Mandatory / optional	Purpose	Attributes (refer to Table 9 for attribute details)	How used
				<p>The text items displayed in the scrollable list are the values specified in the <i>tr=</i> attribute or the text area in each of the <item> elements within the referenced <enum> element.</p> <p>For example, it could be a scrollable list of colors – red, yellow, pink, green, orange, purple, blue.</p> <p>The <i>tr=</i> attribute in the <item> element links to an external language file to provide multi-language support. Refer to <i>tr=</i> in section 2.3.3.</p> <p>The format of the text in both the text box and the popup list is defined by the <i>text-xxx=</i> attributes, refer to Table 9.</p>

Table 8: View section - display elements

The following table (Table 9) further defines the attributes listed in the previous tables.

Attribute	Mandatory / optional	Purpose	Value	How used
aspect=	O default = 1:1	Controls the aspect ratio of a display widget. In RML 1.0, only the <button> widget can accept this attribute.	integer:integer To specify width:height	Used by <button> Example: <button bind='eject' skin='eject' aspect='4:3' />
bind=	Mandatory for all widgets except a <label> element that displays static text	The <i>bind=</i> attribute binds a view item to a model item.	Text, which is the id attribute of a model element of the correct type. There is an extended bind syntax that allows enum widgets to bind to specific model values and button widgets to trigger navigation to named screens. bind=#model can be used for the extended bind syntax. See section 3.4.3 for details.	Must bind a view item to a model item of an appropriate type. For example, a slider must bind to a range type. It would be an error to bind a slider to a text type. The value of the bind attribute is the id of the model item that the view item is bound to.

Attribute	Mandatory / optional	Purpose	Value	How used
density=	O default = #mdpi	<p>The Density attribute provides an indication as to the density of the pixels on the device's screen.</p> <p>Refer to the Android on-line documentation for further information.</p>	<p>Valid values:</p> <ul style="list-style-type: none"> • #ldpi (120 dpi) • #mdpi (160 dpi) • #hdpi (240 dpi) • #xhdpi (320 dpi) • #xxhdpi (480 dpi) • #xxxdpi (640 dpi) 	<p>Used by < view></p> <p>Example:</p> <pre><view footprint='#normal' density='#xhdpi' orientation='#vertical'> </view></pre>
enum=	M	<p>The <i>enum=</i> attribute is used by the <rgbled> element to define the colors associated with individual states that are specified in the <item> tag within the <enum> tag that <rgbled> binds to.</p> <p>Usage example: a fault status indicator.</p>	<p>Valid values:</p> <ul style="list-style-type: none"> • color:value 	<p>Used by <rgbled></p> <p>Example:</p> <pre><rgbled bind="ledstatus" enum="green:0 red:1"/></pre>
footprint=	O default = #normal	<p>The Footprint attribute provides an indication as to the screen size of device.</p> <p>Refer to the Android on-line documentation for further information.</p>	<p>Valid values:</p> <ul style="list-style-type: none"> • #tiny (1") • #small (3") • #normal (4") • #large (5") • #xlarge (7") • #xxlarge (11") 	<p>Used by <view></p> <p>Example:</p> <pre><view footprint='#normal' density='#ldpi' orientation='#vertical'> </view></pre>

Attribute	Mandatory / optional	Purpose	Value	How used
layout=	O default = #horizontal	Defines the layout mode to be used by a container widget.	Valid values: <ul style="list-style-type: none"> • #vertical • #horizontal 	Used by: <box> and <devicelistitem> to change the layout from the layout specified by a higher level container. For example, when a box element is given a vertical layout, its child widgets will be stacked in a vertical manner. Example: <box layout='#vertical' stretch='2'> </box>
length=	O default = unlimited	Defines the maximum number of characters that can be held in widgets that display text.	Integer	Used to restrict the length of display text.
name=	O	Provides a text value when only one language is catered for.	Text	For <screen> elements provides the name for the screen that DeB displays to the user.
orientation=	O default = #horizontal	Specifies the orientation of a graphical widget that can be displayed in more than one orientation.	Valid values: <ul style="list-style-type: none"> • #vertical (portrait) • #horizontal (landscape) 	Used by <view> to determine the overall orientation for the user interface. Note: In the initial release of RML (RML Lite), only portrait mode is supported for the user interface (<i>orientation='#vertical'</i>). Used by the <slider> element to specify whether it is displayed vertically or horizontally.
scale=	.O default = 1.0	Controls the size of a control widget by scaling the size calculated by the layout manager.	Floating point numbers in the range 0.1 ... 1.0	Used by <button> Useful for creating controls of the same kind that have different relative sizes.

Attribute	Mandatory / optional	Purpose	Value	How used
skin=	O	Used to customise the appearance of the infrared and button widgets.	Valid values for <infrared>: <ul style="list-style-type: none"> • video • audio • tuner • aircon • projector Valid values for <button>: <ul style="list-style-type: none"> • next • prev 	Used by <button> and <infrared> A range of skins that make it suitable for many applications, including AV equipment and screen navigation.
stretch=	O	Defines the relative sizing between the item and its siblings in a layout.	Integer	Used by all display widgets, <box> and <spacer> A stretch can be thought of as a spring with a tension of 'stretch' units. For example, if there are two items in a layout, one with stretch set to 2 and the other set to 3 then the layout manger will attempt to allocated first item 2/5 of the space and the second 3/5. Example: <pre><box layout='#horizontal'> <box stretch='2'> </box> <box stretch='3'> </box> </box></pre>
text-bold=	O default = #false	The Text Bold attribute defines the bold attribute of the font used by widgets that display text.	#true #false	Used by <label>, <textbox>, <textline>, <valuepicker> Example: <pre><label text-size="9" text-bold="#true" text-</pre>

Attribute	Mandatory / optional	Purpose	Value	How used
				italic="#false" text-color="0xd1e364" wrapping="#false" >
text-color=	O default = #FFFFFF	Defines the color of the font used by widgets that display text.	Uses hexadecimal RGB in the format: #RRGGBB prefix: # red: 00...FF green: 00...FF blue: 00...FF	Used by <label>, <textbox>, <textline>, <valuepicker> Example: <label text-size="9" text-bold="#true" text-italic="#false" text-color="#d1e364" wrapping="#false" >
text-font=	O default = system font	Defines the font to be used by widgets that display text.	Text Font names on the local system	Used by <label>, <textbox>, <textline>, <valuepicker> Example: <label text-size="9" text-bold="#true" text-italic="#false" text-color="#d1e364" wrapping="#false" >
text-italic=	O default = #false	Defines the italic attribute of the font used by widgets that display text.	#true #false	Used by <label>, <textbox>, <textline>, <valuepicker> Example: <label text-size="9" text-bold="#true" text-italic="#false" text-color="#d1e364" wrapping="#false" >
text-size=	O default = system font	Defines the size in px of the font used by widgets that display text.	Integer	Used by <label>, <textbox>, <textline>, <valuepicker> Example:

Attribute	Mandatory / optional	Purpose	Value	How used
				<code><label text-size="9" text-bold="#true" text-italic="#false" text-color="#d1e364" wrapping="#false" ></code>
tr=	O	<p>Defines the key name of a Unicode string.</p> <p>The value of this attribute is a key that is used to look up a corresponding Unicode string from an external language file.</p>	Text	<p>The <i>tr=</i> attribute provides multi-lingual support.</p> <p>The text string in <i>tr=</i> is checked against the external language file to determine if there is language support for the language used by the smartphone. Refer to section 2.3.3.</p>

wrapping=	O default = #false	Defines whether text is wrapped or not.	#true #false	Used by <label>, <textbox> Example: <label text-size="9" text-bold="#true" text-italic="#false" text-color="#d1e364" wrapping="#false" >
-----------	--------------------------	---	-----------------	--

Table 9: View section - attributes

3.4.3 Extended bind syntax

Some of the display widgets that can be defined in the <view> element are aggregates of a number of sub-widgets where each sub-widget needs to bind with a separate item of state or command defined in the <model> element. For these types of widgets, the extended bind syntax is used.

An example of this type of widget is the infrared “candy bar” widget. An infrared widget consists of a collection of buttons and each button needs to bind to a different item of state in the <model> element. For example, if the infrared widget is a TV remote control, the Ch+ (channel up) button needs to bind to the item of state that defines the channel up capability in the <model> element and the Ch- (channel down) button needs to bind to the channel down item of state.

Each item of state in the <model> element has an ID, which is defined in its *id=* attribute.

To support display widgets that are aggregates of a number of sub-widgets, the *bind=* attribute accepts multiple IDs in the bind string and each ID is separated by a space. The corresponding sub-widget name is specified after the model item ID separated by a colon. For example:

```
bind="ch+:chup ch-:chdn mute:mute ..."
```

This bind string binds the sub-widget named “chup” to the model item with *id=*“ch+”, the sub-widget named “chdn” to the model item with *id=*“ch-” and so on.

Or if the model item names are exactly the same as the sub-widget names then

```
bind="#model"
```

will bind every matching model item in the model section to the sub-widget with the matching name.

3.4.4 RML example

Example RML for the <view> element for a dimmable LED light bulb is:

```
<view>
  <screen name="CONTROLS">
    <devicelist >
      <devicelistitem>
        <title tr="POWER">Power</title>
        <slideswitch bind="power"/>
      </devicelistitem>
      <devicelistitem>
        <title tr="DIMMER">Dimmer</title>
        <slider bind="dim"/>
      </devicelistitem>
      <devicelistitem>
        <title tr="COLOR">Color</title>
        <colorpicker bind="color"/>
      </devicelistitem>
    </devicelist>
  </screen>
</view>
```

3.4.5 User interface examples

The following screenshots provide some examples of user interfaces:

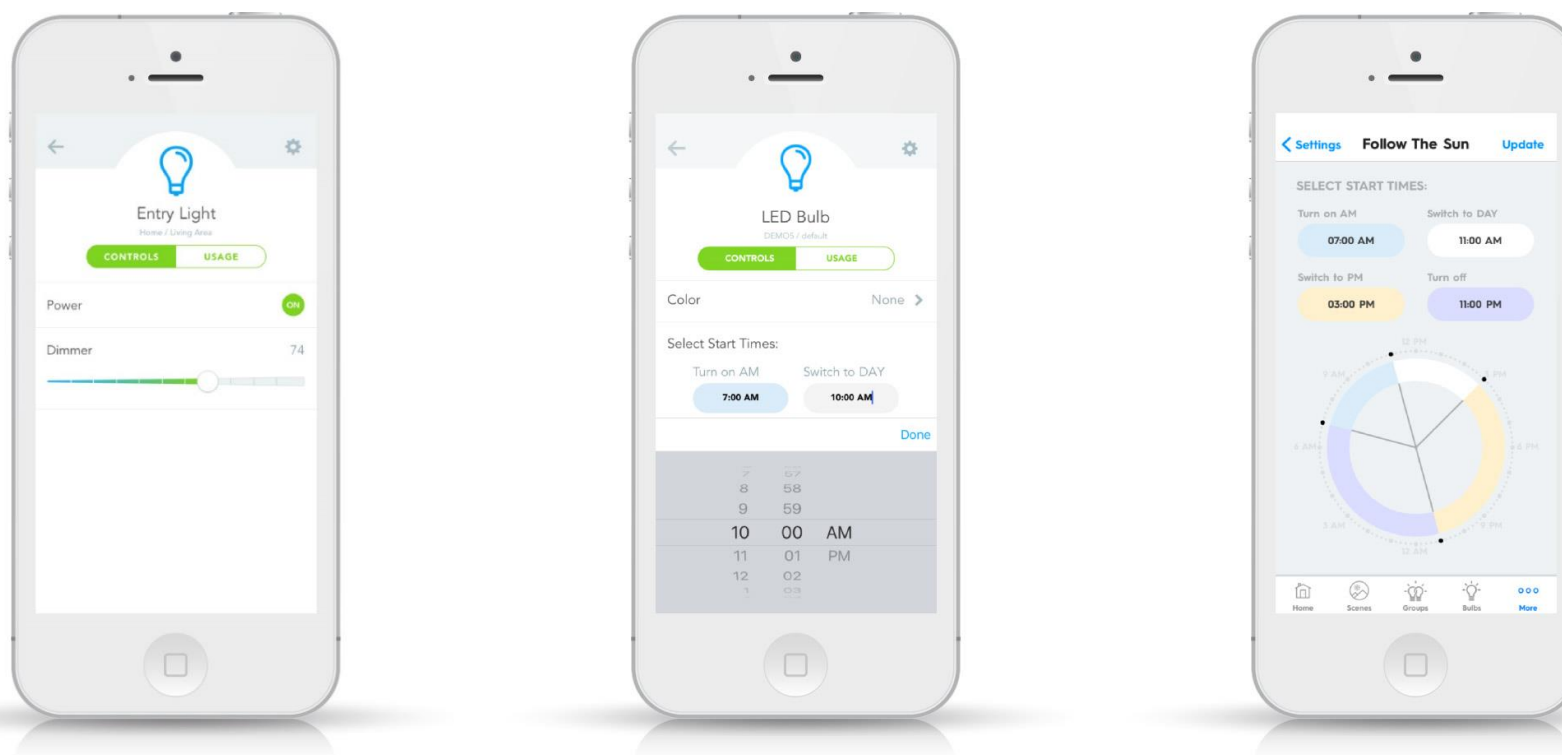


Figure 2: User interface examples

3.5 Controller element

The **<controller>** element is a child element of the root **<rml>** element.

The controller section contains event handlers that are invoked in response to signals emitted by a Thing. These events are handled using procedural logic implemented using the ECMA script language.

3.5.1 Usage

The **<controller>** element is optional and, if present, there can be only one **<controller>** element in an RML file.

3.5.2 Child elements and attributes

The child elements of the **<controller>** element that can be used to specify how to handle signals emitted from a Thing are:

<controller>

<signal>

identifies the path for the signal and the handler(s) to be invoked

<handler>

defines the code that will be executed when the signal occurs. This could be server-side code on the IoTG or client-side code in DeB

</handler>

</signal>

</controller>

These elements are described in [Table 10](#).

Element name	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 11 for attribute details)	How used
<handler>	One server-side and/or one client-side	The <handler> element defines the code that will be executed when the parent signal occurs.	<u>Attributes</u> <ul style="list-style-type: none"> script= side= 	<p>Both server-side and client-side handlers may be provided.</p> <p>A server-side handler is executed on the IoTG and may invoke a subset of the functionality provided by the ADRC server software.</p> <p>A client-side handler is executed by DeB and may invoke only those functions exposed to the scripting engine.</p>
<signal>	If <controller> element is present, at least one <signal> element is required	Registers one or more handlers for a specified device signal.	<u>Sub-elements</u> <handler> <u>Attributes</u> <ul style="list-style-type: none"> path= 	The signal is identified by its path which is specified in RCP resource path syntax. When the specified signal occurs, the associated handlers will be executed.

Table 10: Controller section - child elements

The following table (Table 11) further defines the attributes listed in the previous tables.

Attribute	Mandatory / optional	Purpose	Value	How used
path=	M	The <i>path=</i> attribute defines the interface and selector of an RCP.host message that signal handlers are to be connected to.	Any valid resource path interface and selector (refer to the RCP.host Interface Design Document)	Used by <signal> Example: <signal path='/unit'> <handler /> </signal>
script=	O For server side default = ecma For client side default = javascript	Defines the scripting language that the handler code is written in.	ecma for server side, javascript for client side	Used by <handler> Server-side scripts must be written using ECMA Script. Client-side scripts must be written in Javascript. Example: <signal selector='/unit'> <handler script="ecma" side="#server"> </handler> </signal>
side=	M	Defines on which side the script will be executed. Both server-side scripting and client-side scripting are supported.	#server #client	Used by <handler> Example: <signal path='/xert'> <handler script="javascript" side="#client"> </handler> </signal>

Table 11: Controller section - attributes

3.5.3 RML example

Example RML for the <controller> element is:

```
<signal path="/ev">
  <handler script="ecma" side="server">
    If (signal.query() == "c") {
      // do something
    } else if (signal.query() == "d") {
      // do something else
    }
  </handler>
</signal>
```

3.6 Configuration element

The **<configuration>** element is a child element of the root **<rml>** element.

It specifies details of the physical connectors in the Thing that are used to connect to other equipment. It defines the:

- Physical connectors, for example, HDMI, RS232
- Protocol supported, for example, CERC
- Type of cable required

The configuration element is not used by DeB. It is for the use of other ADRC clients that may be developed in the future, such as setup and installation wizards.

3.6.1 Usage

The **<configuration>** element is optional and, if present, there can be only one **<configuration>** element in an RML file.

3.6.2 Child elements and attributes

The child element of the **<configuration>** element that can be used to specify the physical connections in a Thing is the **<interface>** element:

<configuration>

<interface>

defines input and output interfaces/connectors for the Thing

</interface>

</configuration>

The **<interface>** element is described in [Table 12](#).

Element name	Mandatory / optional	Purpose	Sub-elements and attributes (refer to Table 13 and Table 14)	How used
<interface>	At least one	Defines a physical interface provided by a Thing for connecting to other equipment. Interfaces are usually implemented by means of a connector.	<u>Sub-elements</u> < cable > < comment > < connector > < label > < phy > < protocol > <u>Attributes</u> • id= • family=	<p>An interface may be provided for each physical connector provided by the Thing. This information may be useful during the setup and installation of the Thing into a larger system.</p> <p>For example, a setup wizard application could advise which cables should be connected to which connector on the Thing and suggest how the connection at the other end should be configured.</p>

Table 12: Configuration section - child elements

The following table (Table 13) further defines the elements listed in the previous table.

Sub-element	Mandatory / optional	Purpose	Attributes or value (refer to Table 14 for attributes)	How used
< cable >	M	Defines the cable that connects to the interface. Both ends are specified.	<u>Sub-elements</u> < comment > < connector >	<p>The cable is specified in terms of its connectors. Most cables just have two ends, however, cables with any number of ends can be specified.</p> <p>Distinguishing attributes, such as the color of the connector, can be described using the < comment > element.</p>
< comment >	O	A comment to aid the understanding of the end user.	Text	Can be used to clarify aspects of the interface or cable.
< connector >	M	Specifies the type of connector used at an end of the cable.	Optional text that represents the marking for the connector on the Thing. <u>Attributes</u> <ul style="list-style-type: none"> variant= pins= gender= 	<p>The <i>variant=</i> attribute specifies the sub-class of connector within the interface family. For example, if the interface family is 'HDMI' then possible variants of this family include A, B, C etc.</p> <p>For some families of connector it is more appropriate to specify the number of pins and the gender. For example, the DB family comes in variants with 9 and 25 pins and with male and female genders.</p>
< label >	M	Indicates the text label that identifies the interface on the equipment.	Text <u>Attributes</u> <ul style="list-style-type: none"> tr= 	<p>This label is displayed to the user to identify the interface. The value of the label should correspond to the actual text marking on the Thing.</p> <p>The <i>tr=</i> attribute provides multi-lingual support.</p> <p>The text string in <i>tr=</i> is checked against the external language file to determine if there is language support for the language used by the smartphone. Refer to section 2.3.3.</p>

Sub-element	Mandatory / optional	Purpose	Attributes or value (refer to Table 14 for attributes)	How used
<phy>	M	Defines the physical layer used by the interface technology.	Text <u>Attributes</u> • version=	Usually the physical layer is specified by the name of the technology or standard that defines it. For example, the HDMI standard for connecting AV equipment.
<protocol>	O	Defines the protocol used on the physical layer.	Text <u>Attributes</u> • version=	With some technologies several protocols may be available. For example, with HDMI the CERC command protocol may be used to send commands to the equipment.

Table 13: Configuration section – sub-elements

The following table ([Table 14](#)) further defines the attributes listed in the previous tables.

Attribute	Mandatory / optional	Purpose	Value	How used
family=	M	Identifies the connector technology family.	Text	There are many types and variants of connectors. The common families found on consumer equipment include: USB, HDMI, RCA, SCART and others.
gender=	O	The gender of the connector.	#male #female	Most connector families have parts that are classified as either male or female.
id=	M	A unique identifier for the interface.	Text	An internal identifier for the interface. This is not displayed to a user.
pins=	O	The number of pins the connector has.	Integer	Some connector families provide variants with different numbers of pins.
variant=	O	Identifies the variant of a connector within a family.	Text	Some connector families (especially digital ones) are identified by variant codes. For example, USB has type A, B etc.
version=	O	The version number of the item it is associated with.	Text	Used to identify the version number of the technology used in the <phy> and <protocol> elements.

Table 14: Configuration section - attributes

3.6.3 RML example

Example RML for the <configuration> element is:

```
<configuration>
  <interface id='HDMI1' family='HDMI'>
    <label tr='HDMI1'>HDMI1</label>
    <phy version='1.0'>hdmi</phy>
    <protocol version='2.1'>cerc</protocol>
    <cable>
      <connector variant='A' />
      <connector variant='B' />
    </cable>
    <comment tr='HDMICOM'>Connects to AV equipment via a HDMI cable</comment>
  </interface>
  <interface id='modem' family='DB'>
    <label>Modem</label>
    <phy>rs232</phy>
    <protocol>ppp</protocol>
    <cable>
      <connector pins='9' gender='#female' />
      <connector pins='9' gender='#male' />
    </cable>
    <comment>Connects to a modem via a serial cable</comment>
  </interface>
</configuration>
```

3.7 Resource element

The **<resource>** element is a child element of the root **<rml>** element.

Some Things require extra information beyond what is required to monitor and control them. For example:

- A custom graphic icon
- A user manual
- A sound bite
- A link to the complete RML for the thing
- A link to the manufacturer's website

The **<resource>** element specifies the external resources available for the Thing so that DeB can find and use them.

3.7.1 Usage

The **<resource>** element is optional and, if present, there can be only one **<resource>** element in an RML file.

3.7.2 Child elements and attributes

The child elements of the **<resource>** element that can be used to specify the external resources for a Thing are:

<resource>

<icon>

specifies a link to the location of the required icon

</icon>

<faq>

specifies a link to the location of the required FAQs

</faq>

<manual>

specifies a link to the location of the required manual

</manual>

</resource>

These elements are described in [Table 15](#).

Element name	Mandatory / optional	Purpose	Value	How used
<faq>	At least one of: <ul style="list-style-type: none"> • <icon> • <faq> • <manual> 	Specifies a URL to a webpage that provides the FAQs for the Thing	URL	<faq>http://www.virtucon.com/faq</faq>
<icon>	At least one of: <ul style="list-style-type: none"> • <icon> • <faq> • <manual> 	Specifies a URL to an icon file that the manufacturer is providing for the Thing	URL	<icon>http://www.virtucon.com/icons/thing.jpg</icon>
<manual>	At least one of: <ul style="list-style-type: none"> • <icon> • <faq> • <manual> 	Specifies a URL to a file that provides the user manual for the Thing	URL	<manual>http://www.virtucon.com/man/thing.pdf</manual>

Table 15: Resource section - child elements

3.7.3 RML example

Example RML for the <resource> element is:

```
<resource>  
  <icon> http://res.virtucon.com/PSW-240AU1D/icon.jpg</icon>  
  <manual>http://res.virtucon.com/PSW-240AU1D/manual.pdf</manual>  
  <faq>http://res.virtucon.com/PSW-240AU1D/index.htm</faq>  
</resource>
```

3.8 Menu element

The **<menu>** element is a child element of the root **<rml>** element.

The **<menu>** element describes which of DeB's built-in functions are to be made available for a Thing when a user taps a smartphone on a paired Thing. For example, whether capabilities such as 'unpair' or 'set PIN' are to be made available to the user.

These action items appear on the Proximity Action View in DeB.

3.8.1 Usage

The **<menu>** element is optional and, if present, there can be only one **<menu>** element in an RML file.

3.8.2 Child elements and attributes

The child element of the **<menu>** element that can be used to specify the in-built administrative functions for a Thing is the **<menuitem>**:

<menu>

<menuitem>

defines one of the in-built functions in DeB that will be displayed to the user when the smartphone is tapped on the Thing e.g. reset PIN

</menuitem>

</menu>

The **<menuitem>** element is described in [Table 16](#).

Element name	Mandatory / optional	Purpose	Value	How used
<menuitem>	At least one	Invokes one of the in-built functions from within DeB.	<p>Text</p> <p>Valid values:</p> <ul style="list-style-type: none"> • unpair • set-pin • factory-reset • add-unit • delete-unit 	<p>When the user taps their smartphone on a paired Thing, the proximity menu is displayed. The menu will contain the function(s) specified in the <menuitem> elements.</p> <p>‘unpair’ allows the user to unpair the Thing.</p> <p>‘set-pin’ allows the user to set a PIN for the Thing.</p> <p>‘factory-reset’ allows the user to set the Thing back to its factory defaults.</p> <p>‘add-unit’ allows the user to add sub-units to the Thing. This is only possible for Things that allow sub-units, for example, the Infrared Blaster.</p> <p>‘delete-unit’ allows the user to delete a sub-unit previously added to the Thing.</p>

Table 16: Menu section - child elements

3.8.3 RML example

Example RML for the <menu> element is:

```
<!--Add the menu items the device supports here -->
<menu>
  <menuitem>unpair</menuitem>
  <menuitem>set-pin</menuitem>
  <menuitem>factory-reset</menuitem>
</menu>
```

4 Resource Control Protocol

Resource Control Protocol (RCP) is an application layer protocol developed by Xped to transfer RML files, commands and signals between a Thing, an IoTG and an ADRC client (a device browser app or other ADRC client).

RCP will be familiar to developers who know HTTP. RCP is a RESTful protocol that has been augmented to allow for servers (Things) to send unsolicited events to clients (apps) and has been designed to keep message overhead small to cater for machine-to-machine (M2M) applications on constrained devices.

RCP has an RCP.host format (XML based) for an ADRC client to communicate with an IoTG and an RCP.wire format for an IoTG to communicate with a Thing. The IoTG translates between RCP.host and RCP.wire formats in much the same way as XML and JSON can be translated from one to another.

4.1 RML **<model>** element

DeB can automatically construct all of the operations required to interact with a Thing from information contained in the **<model>** and **<view>** elements of an RML file.

The RML elements and attributes that are directly relevant for the RCP protocol are found in the **<model>** element of an RML document.

RCP is a RESTful protocol so each item of state (control setting) for a Thing is represented by a path and this path is manipulated using a limited set of operations - Get, Put, Delete, eExecute.

The **<model>** element describes each item of state for a Thing, including the path, which can be thought of as the command that is used to operate on the item of state.

For example:

```
<model>
  <range id='volume' path='/vol' min='0' max='99' />
</model>
```

In this example, the Thing has an item of state that represents the playback volume. The path to that item of state is '/vol'. To set the volume to 25, the command is '/vol?25', which is the argument to a PUT operation.

4.2 ADRC IoT Stack

Things interact with an IoTG via a reverse proxy (firmware) in the Thing. This reverse proxy implements standardised device behaviours and is known as the ADRC IoT Stack. The IoT Stack communicates with the manufacturer's application in the Thing, which is the application that defines the functions of a Thing. The manufacturer's application may be implemented in the same hardware as the IoT Stack or it may be implemented in a separate microprocessor that interfaces with the IoT Stack via a hardware communications channel such as a UART or SPI port.

In either case, the manufacturer's application receives the RCP messages and responds to them. The payload content of these messages is entirely defined by the manufacturer of the Thing. The meaning of the payload content can also be defined in the **<model>** element of the RML file for the Thing by providing a URL to an OWL¹ class that defines it.

¹ OWL is the acronym for Web Ontology Language which has been developed by the W3C and is a part of their Semantic Web initiatives.

A high level view of the ADRC architecture, including the IoT Stack, is shown in [Figure 3](#).

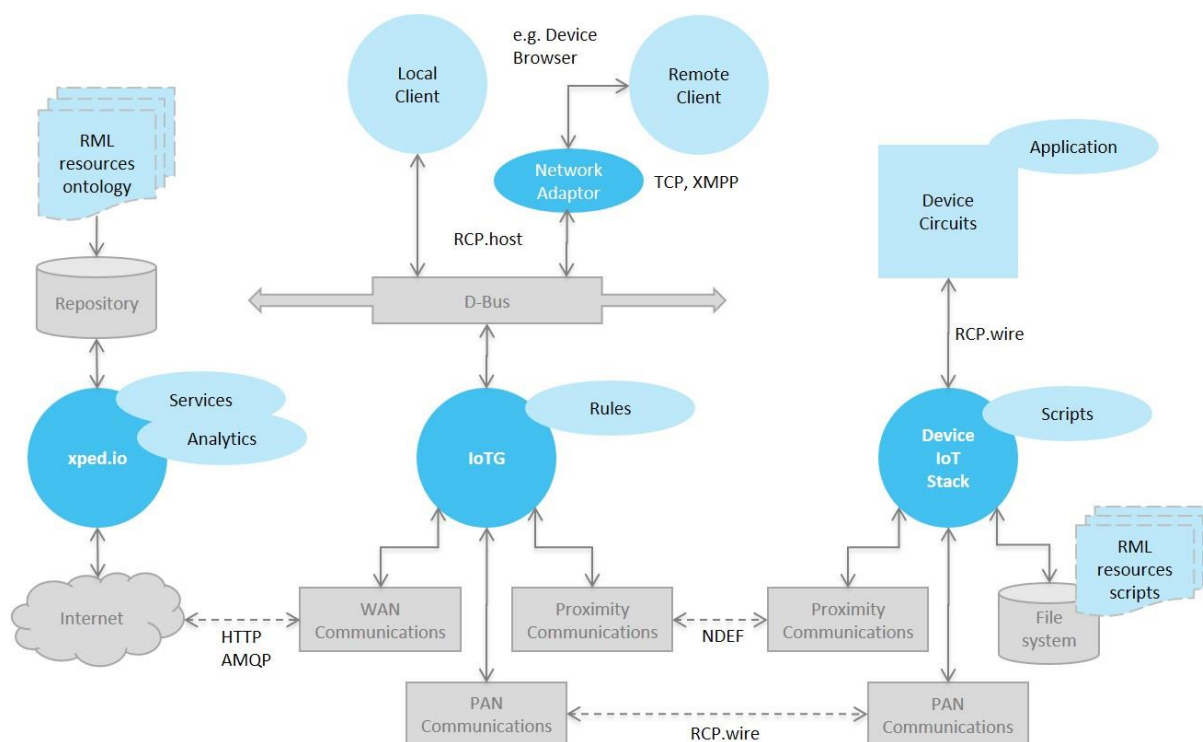


Figure 3: ADRC architecture

4.3 RCP host format

The RCP.host format that is used by an ADRC client to communicate with an IoTG is XML based. It is used to communicate between processes on a local machine using a D-Bus or between remote processes over the Internet, where it will usually tunnel through XMPP.

The RCP.host format allows for highly structured data to be interchanged and for commands to be sent to one or multiple devices and executed as individual transactions.

4.4 RCP wire format

The RCP.wire format uses a simple text based format that is framed between the ASCII STX '\2' and ETX '\3' characters. The IoT Stack understands the RCP.wire protocol, which it can receive from an IoTG over a network link or from a manufacturer's application over a hardware link, such as a UART or SPI port. Due to its self-framing structure, RCP.wire is ideally suited for transmission over these raw hardware links.

The RML files are opaque to the IoT Stack and the manufacturer's application.

The RCP.wire protocol and the RML language form the application programming interface (API) of a Thing.

4.5 RCP protocol mapping

The IoTG translates between RCP.host and RCP.wire formats. The protocol mapping is straight forward and includes the IoTG looking up the network details of the addressed Thing. The translated RCP.wire protocol is sent to the IoT Stack of the addressed device over its associated transport layer. The IoT Stack adds a routing attribute (h=<id>:<port>) to the RCP.wire message, which identifies the

incoming logical connection from the IoTG or other client. This routing attribute is opaque to the manufacturer's application, which simply includes it in any response message back to the IoT Stack.

The RCP protocol mapping is shown in [Figure 4](#).

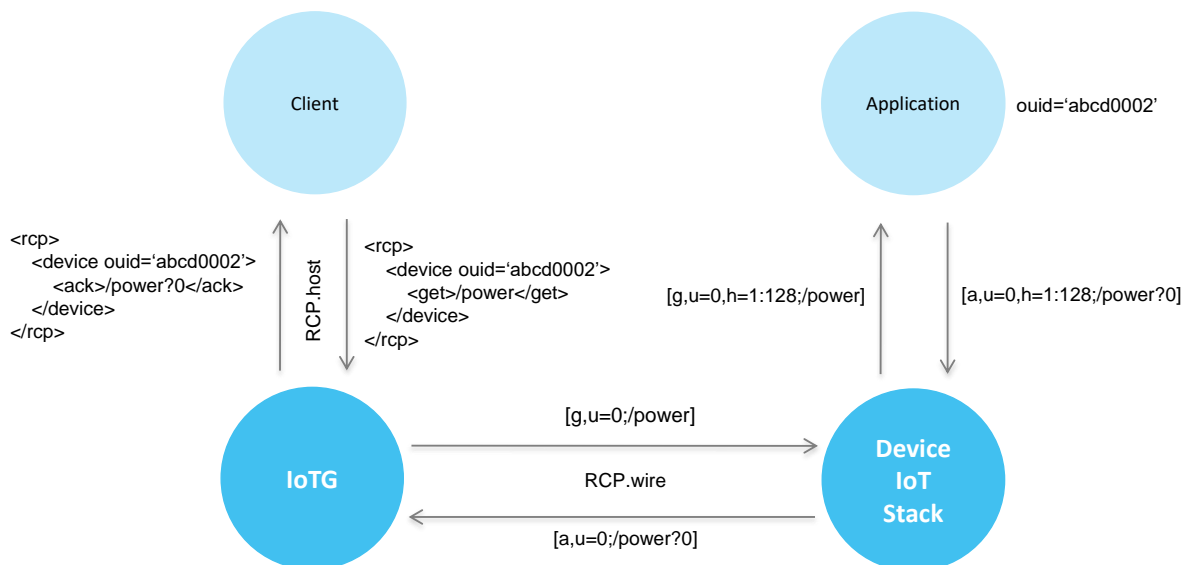


Figure 4: RCP protocol mapping

4.6 Transport layer

The IoTG provides gateway services that allow RCP.host requests from clients to be translated to RCP.wire format and transported to Things, and vice versa.

On the client side of the IoTG, clients use the most suitable transport protocol. For example, a client that needs to communicate over the Internet with an IoTG will generally need to use the XMPP protocol to transport the RCP.host messages. A client that is on a local WLAN may find it more efficient to use TCP. To support this flexibility, the IoTG provides a D-Bus interface that can be called by interface adaptors that support various networking technologies. If a client, such as DeB, is on the same local machine as the IoTG, then it can simply use the D-Bus interface directly. This approach provides a mechanism to easily support additional networking protocols between clients and the IoTG allowing implementers to develop a D-Bus interface adaptor for the required protocol.

Similarly, Things have the flexibility to use the most suitable transport protocol to communicate with an IoTG. For green-field implementations it is most cost efficient for all Things to use a common transport technology such as the 802.15.4 MAC/PHY as this simplifies the hardware requirements for the IoTG. However, it is expected that some implementers may have special requirements that require several PAN technologies to be supported. In these scenarios, the IoTG will need to include the necessary hardware to support each PAN type. Regardless of the approach used, each PAN transport protocol carries RCP.wire requests as the common protocol.

The transport layer communications are shown in [Figure 5](#).

4.7 Addressing abstraction

A Thing may have more than one hardware unit within it, for example, a TV with a built-in DVD player. ADRC uses a hierarchy of entities known as units to form an abstract model of the hardware units within a Thing. The core hardware unit, which has the network interface is always referred to as unit-0. Most common Things only have a unit-0. However, units may be arranged in tree like hierarchies. Most commonly a multi-unit Thing will have a two-tier hierarchy with unit-0 at level-0 and a number of units at level-1 below it. Each unit must have its own RML file(s) and is addressed individually. The

addressing notation used by the RCP wire protocol is: u=0 (unit-0), u=1 (unit-1), u=1.1 (unit-1 under unit-1) and so on.

In this manner, ADRC provides addressing abstraction where logical addresses for a particular unit are independent of the networking transport layer. The IoTG resolves these logical unit addresses to physical addresses before sending a command to the Thing.

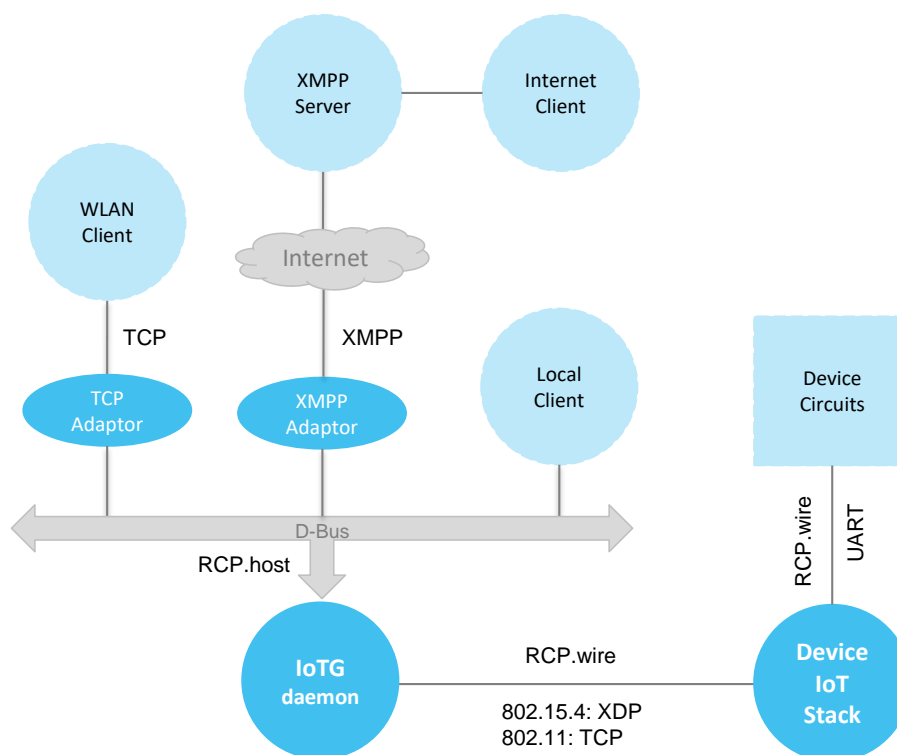


Figure 5: Transport layer

Reference information:

Detailed information on RCP is provided in the *Interface Design Documents for the Resource Control Protocol*.

5 Development and operations

5.1 Integrated development environment

Equinox is a full featured integrated development environment (IDE) provided for RML, which simplifies the implementation process for a Thing.

The RML IDE:

- is open-source and provides the developer with a working RML environment
- has been designed to specifically support RML development
- provides multi-document editing based on the Scintilla™ editor
- can be used to edit RML and Arduino code
- allows a developer to edit RML, check syntax and execute RML
- has RML templates
- a developer can use existing RML on a Thing and edit this
- automatically detects all Things in the PAN and allows the developer to fetch RML from any Thing, modify it and send it back to the Thing wirelessly
- allows RML to be run on a device emulator so the developer can interact with the user interface they are creating

A new Thing has a base RML file in it.

A developer creates the RML file for a Thing and then uses Equinox to upload that file to the Thing, generally wirelessly, to overwrite the base file.

At least one RML file is required for each Thing and is stored in the Thing's file system. The file may be compressed using gzip to save space.

5.2 RML updates

RML updates could be required to introduce new technology, correct faults, and so on.

Updates can be via:

- Xped.com (like RML app store)
- 3rd party developers
- Enthusiasts

Appendix A: Complete RML example

This is the full RML for a dimmable LED light bulb that has been used as an example of the RML language.

The user interface that DeB creates from this RML file is shown in [Appendix B](#).

```
<!-- Copyright Xped Corporation 2016 all rights reserved -->
<!-- Standard RML for 7W Dimmable LED Bulb -->
<!-- Multilingual/English version -->

<rml version="1.0" xmlns="http://rml.xped.com">
  <description>
    <manufacturer>Xped</manufacturer>
    <model>LED-E7WV</model>
    <category>8009</category>
    <version>1.0</version>
    <nickname tr="NICK">Light Bulb</nickname>
    <theme tr="THEME">Standard</theme>
    <url>xped.com/lighting/LED-E7WV</url>
  </description>

  <model>
    <enum id="switch" path="/on" state="#primary">
      <item tr="OFF" value="0">Off</item>
      <item tr="ON" value="1">On</item>
    </enum>

    <range id="dimmer" path="/dim" min="10" max="100" step="1" />

    <range id="power" path="/pwr" min="0" max="7" mode="#readonly">
      <unit system="#SI" symbol="W" type="#integer" exponent="0">Watt</unit>
      <class>http://sweet.jpl.nasa.gov/2.2/quanEnergy.owl#Power</class>
    </range>

    <range id="energy" path="/egy" min="0" max="4294967296" mode="#readonly">
      <unit system="#derived" symbol="Wh" type="#integer" exponent="0">Watt hour</unit>
      <class>http://sweet.jpl.nasa.gov/2.2/quanEnergy.owl#Energy</class>
    </range>

    <record id="today" mode="#readonly">
      <source service="_uap._tcp" path="historical/utilities/electricity/$(OUID)/today"/>
      <range id="ivl" min="0" max="95" step="1"/>
      <range id="avr" min="0" max="2500" step="1">
        <units system="#SI" symbol="W" type="#integer" exponent="0">Watt</units>
      </range>
    </record>
  </model>
</rml>
```



```

</range>
<range id="mir" min="0" max="2500" step="1">
  <units system="#SI" symbol="W" type="#integer" exponent="0">Watt</units>
</range>
<range id="mar" min="0" max="2500" step="1">
  <units system="#SI" symbol="W" type="#integer" exponent="0">Watt</units>
</range>
<range id="qty" min="0" max="4294967296" step="1">
  <units system="#derived" symbol="Wh" type="#integer" exponent="0">Watt
  hour</units>
</range>
<class>http://sweet.jpl.nasa.gov/2.2/quanEnergy.owl#Energy</class>
</record>
</model>

<view>
  <screen name="CONTROLS">
    <devicelist>
      <devicelistitem bind="switch">
        <title tr="SWITCH">Power</title>
        <slideshow bind="switch"/>
      </devicelistitem>
      <devicelistitem bind="dimmer">
        <title tr="DIMMER">Dimmer</title>
        <slider bind="dimmer"/>
      </devicelistitem>
    </devicelist>
  </screen>
  <screen name="USAGE">
    <devicelist>
      <devicelistitem>
        <title tr="USAGE">Curent usage</title>
        <box layout="#horizontal">
          <label bind="power"/>
          <label tr="WATT">watts</label>
        </box>
      </devicelistitem>
      <devicelistitem>
        <graph bind="today">
          <title tr="GTITLE">Utility monitor</title>
          <xaxis bind="today.ivl" tr="GXLABEL">Time</xaxis>
          <yaxis bind="today.avr" tr="GYLABEL">Watts</yaxis>
        </graph>
      </devicelistitem>
    </devicelist>
  </screen>
</view>

```

```
</screen>
</view>

<!-- Add the menu items the device supports here -->
<menu>
  <menuitem>unpair</menuitem>
</menu>
</rml>
```

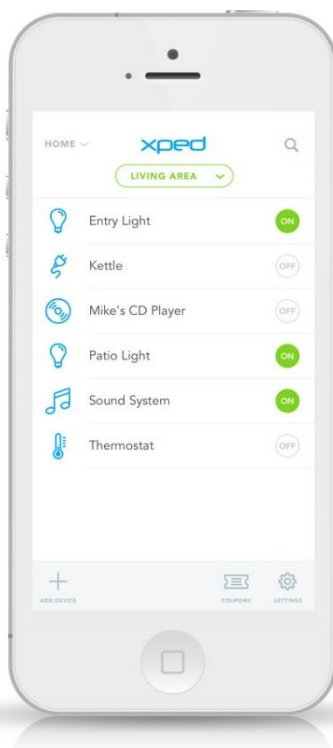
Appendix B: Example user interface

The device browser (DeB) uses the RML shown in [Appendix A](#) to create control and usage screens for an LED light bulb. The bulb hardware is shown in the following image:



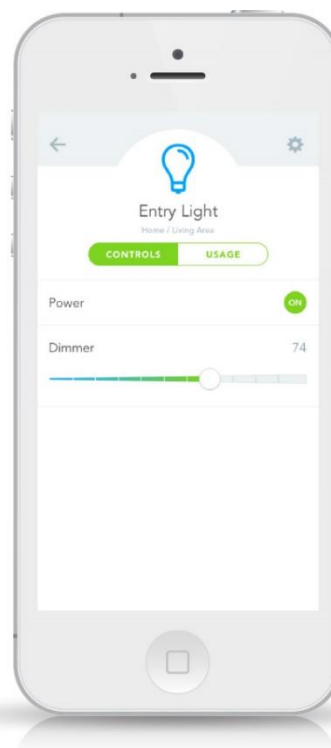
The functionality described in the RML for the LED light bulb translates into the following user interface screens within DeB, which allow a user to monitor and control the LED light bulb.

Note: In the screens shown below, the user has used DeB to change the manufacturer-defined nickname of 'Light Bulb' in the RML code to their preferred nickname of 'Entry Light'.



Home Screen

If one of the items of state for a Thing has the *state=* attribute set to “primary” in the *<model>* element, the widget to control that item of state will be displayed on the home screen. This allows convenient control of that item of state directly from the home screen. In the above example, the Things can be switched on/off directly from the home screen.



Controls Screen

All the widgets to control the light bulb are displayed on the control screen.



Usage Screen

Usage and monitoring information for the light bulb is shown on the usage screen.

Appendix C: RML tags by functional groups

Group	Tags	Children	Element table
Root element	<rml>	<description> <model> <view> <controller> <configuration> <resource> <menu>	Section 3.1 Table 1
<description>	<manufacturer> <model> <category> <version> <theme> <nickname> <class> <url>	Text Text Text Text Text Text Text URL	Section 3.2 Table 2
<model>	<enum> <range> <text> <trigger> <record> <item> <class> <units> <cache>	<item>, <class>, <units> <class>, <units> <class>, <units> None <enum>, <range>, <text>, <source> None Text Text None	Section 3.3 Table 3 Table 4
<view>	<screen> <devicelist> <devicelistitem> <box> <spacer> <button> <slideshow>	<devicelist> <devicelistitem> #control-widgets, #layout-widgets #control-widgets, #layout-widgets None <text> <text>	Section 3.4 Table 7 Table 8

	<dial>	None	
	<spinslider>	None	
	<checkbox>	<text>	
	<slider>	None	
	<label>	<text>	
	<textbox>	None	
	<textline>	None	
	<valuepicker>	None	
	<colorpicker>	None	
	<infrared>	None	
	#control-widgets	<button>, <slideshow>, <dial>, <spinslider>, <checkbox>, <slider>, <label>, <textbox>, <textline>, <valuepicker>, <colorpicker>, <infrared>, <graph>	
	#layout-widgets	<box>, <spacer>, <title>	
<controller>	<signal>	<handler>	Section 3.5
	<handler>	Text	Table 10
<configuration>	<interface>	<label>	Section 3.6
		<phy>	Table 12
		<protocol>	
		<connector>	
		<comment>	
<resource>	<icon>	URL	Section 3.7
	<faq>	URL	Table 15
	<manual>	URL	
<menu>	<menuitem>	Text	Section 3.6
			Table 16

Appendix D: Attributes by tag

Tag	Attributes	Valid values	Attribute table
<box>	layout=	#vertical, #horizontal	Table 9
<button>	bind=	Text	Table 9
	tr=	Text	
	scale=	Floating point number	
	aspect=	integer:integer	
<cache>	etag=	#true, #false	Table 4
	maxage=	integer	
<checkbox>	bind=	Text	Table 9
	tr=	Text	
<class>	none		n/a
<colorpicker>	bind=	Text	Table 9
<configuration>	none		n/a
<controller>	none		n/a
<category>	none		n/a
<description>	none		n/a
<devicelist>	none		n/a
<devicelistitem>	layout=	#vertical, #horizontal	Table 9
<dial>	tr=	Text	Table 9
	bind=	Text	
	caption=	Text	
	dial-color=	#RRGGBB	
	text-color=	#RRGGBB	
<enum>	id=	Text	Table 5
	path=	/text	
	mode=	#readonly, #writeonly, #readwrite, #disabled	
	timeout=	integer	
	state=	#primary, #secondary, #configuration	

<graph>	bind=	Text	Table 9
<handler>	path=	/text	Table 11
<infrared>	bind=	Text	Table 9
	skin=	Text representing the skin name	
<item>	tr=	Text	Table 4
	value=	integer	
<label>	bind=	Text	Table 9
	length=	integer	
	wrapping=	#true, #false	
	text-font=	Text	
	text-color=	#RRGGBB	
	text-size=	integer	
	text-bold=	#true, #false	
	text-italic=	#true, #false	
<manufacturer>	tr=	Text	
<manufacturer>	none		n/a
<menu>	none		n/a
<menuitem>	none		n/a
<model>	none		n/a
<model> child of <description>	none		n/a
<nickname>	tr=		Table 2
<range>	id=	Text	Table 5
	path=	/text	
	mode=	#readonly, #writeonly, #readwrite, #disabled	
	timeout=	integer	
	min=	integer	
	max=	integer	
	step=	integer	
	state=	#primary, #secondary, #configuration	

<record>	id=	Text	Table 5
	path=	/text	
	mode=	#readonly, #writeonly, #readwrite, #disabled	
	timeout=	integer	
<resource>	none		n/a
<rgbled>	bind=	Text	Table 9
	enum=	color:value	
<rml>	version=	Version of the RML language: e.g. "1.0"	Section 3.1.2
	xmlns=	URL of the RML name space: e.g. "http://rml.xped.com"	
<screen>	name=	Text	Table 9
<signal>	script=	javascript, ecma	Table 11
	side=	#server, #client	
<slider>	bind=	Text	Table 9
	orientation=	#vertical, #horizontal	
<slideshow>	bind=	Text	Table 9
	tr=	Text	
<source>	service =	Text	Table 5
	path=	Text	
<spacer>	stretch=	Integer	Table 9
<spinslider>	bind=	Text	Table 9
<text>	id=	Text	Table 5
	path=	/text	
	mode=	#readonly, #writeonly, #readwrite, #disabled	
	timeout=	Integer	
	length=	Integer	
	state=	#primary, #secondary, #configuration	
<textbox>	bind=	Text	Table 9
	length=	Integer	

	wrapping=	#true, #false	
	text-font=	Text	
	text-color=	#RRGGBB	
	text-size=	Integer	
	text-bold=	#true, #false	
	text-italic=	#true, #false	
<textline>	bind=	Text	Table 9
	length=	Integer	
	text-font=	Text	
	text-color=	#RRGGBB	
	text-size=	Integer	
	text-bold=	#true, #false	
	text-italic=	#true, #false	
<theme>	tr=		Table 2
<trigger>	id=	Text	Table 5
	path=	/text	
	mode=	#writeonly, #disabled	
<units>	system=	#SI, #derived	Table 4
	symbol=	Text	
	type=	#integer, #float	
	exponent=	Integer	
<url>	none		n/a
<valuepicker>	bind=	Text	Table 9
	text-font=	Integer	
	text-color=	#RRGGBB	
	text-size=	Integer	
	text-bold=	#true, #false	
	text-italic=	#true, #false	
<version>	none		n/a
<view>	footprint=	#tiny, #small, #normal, #large, #xlarge, #xxlarge	Table 9
	density=	#ldpi, #mdpi, #hdpi, #xxhdpi, #xxxhdpi	

orientation= #vertical, #horizontal

Appendix E: RML reserved key words

Keyword	Meaning
#client	Client side script
#configuration	Device configuration state
#derived	Derived units
#disabled	Attribute is disabled
#false	Attribute is false
#female	Female connector type
#float	A floating point value
#hdpi	Typically 240 dpi
#horizontal	Horizontal or landscape orientation
#integer	An integer value
#large	Typically 5" screen
#ldpi	Typically 120 dpi
#male	Male connector type
#mdpi	Typically 160 dpi
#model	Extended bind syntax See section 3.4.3
#normal	Typically 4" screen
#primary	Primary display state
#readonly	Attribute is read only
#readwrite	Attribute is read and write

#secondary	Secondary display state
#server	Server side script
#SI	System of International units
#small	Typically 3" screen
#tiny	Typically 1" screen
#true	Attribute is true
#vertical	Vertical or portrait orientation
#writeonly	Attribute is write only
#xhdpi	Typically 320 dpi
#xlarge	Typically 7" screen
#xxhdpi	Typically 480 dpi
#xxlarge	Typically 11" screen
#xxxhdpi	Typically 640 dpi

Appendix F: XML schema definition

The XMLSCHEMA definition of the RML language can be found at:

<http://www.xped.com/downloads/rml.xsd>

Appendix G: Default icons

The following table shows the default icons that are available in the Internet of Things Gateway.

Category code	Icon	Description
0000		Unknown Device
0001		TV
0002		Cable Box
0003		Satellite Box
0004		VCR
0005		DVD
0006		Audio
0007		DTV Box
0008		PVR
0009		CD
000A		Game Console

Category code	Icon	Description
000B		Smart Home
000C		Media Player
000D		Projector
000E		Blu-ray
000F		Air conditioner
8001		Charge Pad
8002		IR Blaster
8003		Power Plug
8004		Light Switch
8005		Sensor
8006		Actuator
8007		Smart Bulb
8008		Gateway

Category code	Icon	Description
8009		Contact Sensor
800A		Siren
800B		Panic Button
800C		Motion Sensor
800D		Door Lock
800E		IP Camera
800F		Multi Sensor
8010		Gas Sensor
8011		Temperature Sensor
8012		Humidity Sensor
8013		Energy Monitor
8014		CO Sensor
8015		Smoke Detector

Category code	Icon	Description
8016		Water Sensor
8017		Thermostat
8018		Barometric Sensor
8019		Light Sensor
801A		Tilt Sensor
801B		Vibration Sensor
801C		Scene Controller
801D		1-Button Fob
801E		2-Button Fob
801F		3-Button Fob
8020		4-Button Fob
8021		Battery Sensor
8022		AC Current Sensor

Category code	Icon	Description
8023		AC Voltage Sensor
8024		DC Current Sensor
8025		DC Voltage Sensor
8026		Digital I/O
8027		Digital Output
8028		Doorbell
8029		Range Extender
802A		Motor Controller
802B		Light Dimmer
802C		Garage Door Controller
802D		LED Strip
802E		C02 Sensor
802F		Hub

Category code	Icon	Description
8030		Relay
8031		Occupancy Sensor
8032		4-20 mA Sensor
F000		Lounge Room
F001		Kitchen
F002		Dining Room
F003		Bedroom
F004		Bathroom
F005		Garage
F006		Security
F007		Safety
F008		Automation
F009		Electricity

Category code	Icon	Description
F00A		Water
F00B		Gas
F00C		Lighting
F00D		Curtains
F00E		Blinds
F00F		Gate

Appendix H: DeB in-built skins

DeB has a range of in-built skins that are suitable to use as remote controls for various appliances, such as audio-visual equipment. These skins are used by the <infrared> widget.

The available skins are shown in the following table.

TBD

Contact details

Xped Registered & Corporate Office

Level 6,
412 Collins Street
Melbourne VIC 3000
AUSTRALIA

PO Box 16059
Collins St West
Melbourne VIC 8007
AUSTRALIA

Phone: + 61 3 9642 0655
Fax: + 61 3 9642 5177
Email: info@xped.com

Head Office

Suite 11, 2 Portrush Road
Payneham SA 5070
AUSTRALIA

Email: info@xped.com

