/\*! \mainpage Gateway Shadow Controller

\*

\* \section intro\_sec Introduction

\*

\* Gateway Shadow Controller is primarily the digital twin of the Cloud Shadow Controller. Both Shadow Controllers are almost identical in terms of workflow barring few features.

\* Shadows are the prime facia of a Device i.e world would not see the device but the shadow as the device and through that it will controlled. A typical shadow contains the most abstracted information of the device that is the properties and their sub-properties. Along with these each property in the shadow has two states 1) reported 2) desired. Reported is current value/state of the device whereas Desired is the state which an external body wants to reach at an interval. Shadow manages both these states.

\*

\* Further, a Shadow is nothing but a json file which is generated from RAML and represents the above mentioned reported and desired state per properties.

\* \section structure\_sec Structure Of Gateway Shadow Controller

\*

\* The structure of Shadow Controller is primarily divided into three high-level component classes.

1) Shadow Controller: This is the main manager and interface class between shadows and the Device Object. shadowcontroller is the class which is representing this interface.

2) Shadow: This is the Shadow class which manages the actual device shadow which, as I mentioned earlier, is a json file.

3) RAMLPyBinding: This is the class which does the conversion from RAML to Json object.

Although loosely coupled each of the components are linked with each other through contracts. For example, Shadow Controller class has a contract with Shadow class but not with RAMLPyBinding class, however, Shadow class and RAMLPyBinding class for the end JSON file which is our actual Shadow.

Each structure is laced with exceptions which are pertinent to the structure it is in. Exceptions are raised only when the contract between two components are not fulfilled. Now, it may be origin components' responsibility for the failure or the cloud app or mesh app. The origin is notified of this.

Let us Magnify upon each component, beginning with Shadow Controller(shadowcontroller) class.

This class does the job of knowing which shadow is the controller of the device. The major components of this class are its functions which are divided into three categories 1) setup 2) searching for & calling shadow's functions 3) forwarding the calls to the protocol layer.

In Setup, we are talking about configurations. A whole lot of configurations that can be either individual shadow dependent or wholistically. Currently, such configurations are yet to be finalized. Other important part of this section is Shadow Creation which happens when a new Device is added. At this juncture, configuration is also pushed. The Shadow is saved in a table with key as deviceId and value as the pointer of Shadow itself.

Commands such as setDeviceState and getDeviceState fall under the second category. In this shadowController class finds the shadow and calls that shadow's respective functions given the command. If it doesn't find the shadow, such a message is sent back to the originator.

Last section is the call forwarder i.e providing the gateway from right side of the equation i.e each shadow can use shadowController's functions such as SendTocloud, SendTelemetryDataCCAPP, CallTheInterface(for device interface).

shadowController class does it all and doesn't use any threads and doesn't maintain any queue for the commands. It simply forwards requests and their responses.

==========

Next section is Shadow class. This is the central component of the Shadow Controller. This class does the following tasks:-

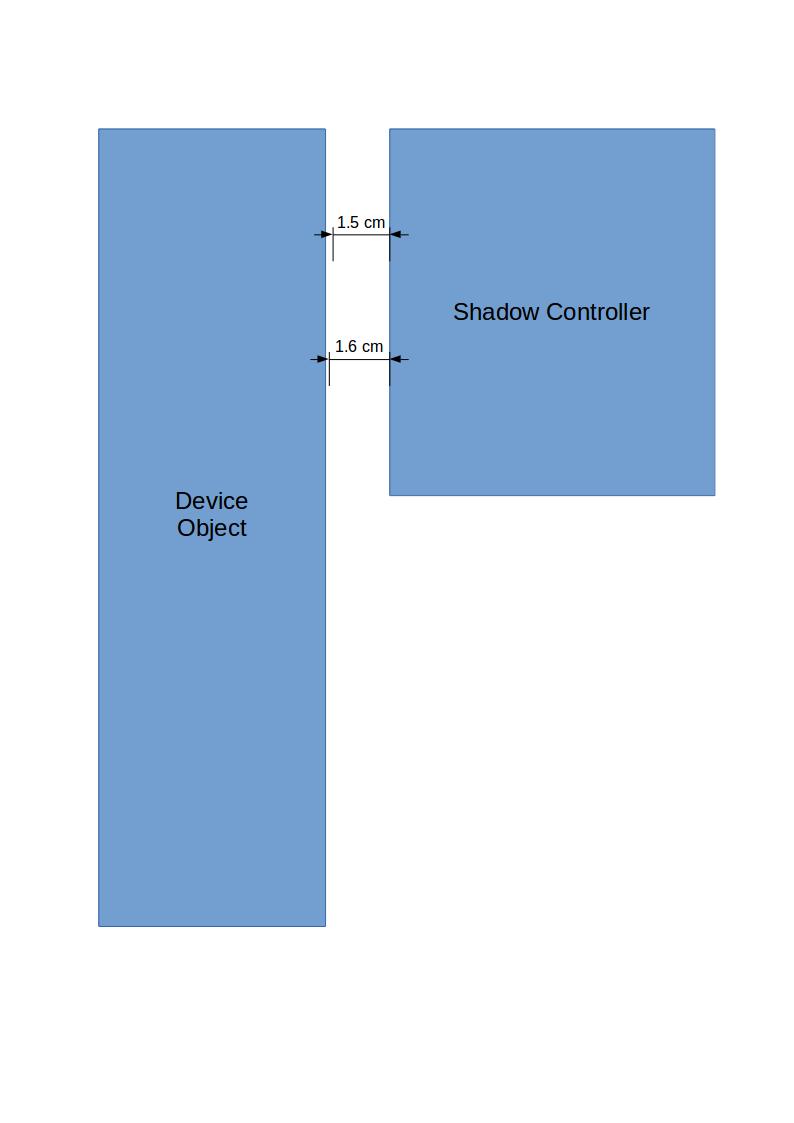
1) Maintains Json file shadows(updation of values(reported & desired), updation of files).

2) Finds Delta for the properties.

3) Gives the reported values.

Each shadow does its task independent of other shadows using threads. Each shadow has a thread associated with it. This thread is task based, if it gets the tasks then only it runs otherwise it sleeps i.e not running in a continuous loop.

**Architecture & Workflow Of Gateway Shadow Controller.**   
  
Pictorial Depiction

 Interface with Device Object :-

Shadow Controller Internal Overview :-

