1. Thing+ Embedded Overview

Thing+ Embedded is composed of the Thing+ Embedded Protocol, Thing+ Embedded SDK and Thing+ Gateway. The Thing+ Embedded Protocol defines the format of data flowing between hardware and the Thing+ Cloud server. The Thing+ Embedded SDK is a C library that implements our Thing+ Embedded Protocol. The Thing+ Gateway is a Node.js application program that implements Thing+ Embedded Protocol.

1.1 Hardware application requirements

1.1.1 Thing+ Embedded Protocol

The Thing+ Embedded Protocol can be implemented in hardware ranging from the low end such as WiFi chips to the high end. Firmware or any OS can be used. The ability to connect a MQTT server, sense sensors, actuate actuators, compose MQTT, and send HTTP messages are required in order to function correctly.

Category	**Description**
Application requirement	Sensing sensors Actuating actuators MQTT, HTTP connection Composing MQTT, HTTP message Sending status and values periodically.
Target hardware	Low end to high end
Dependency	None

1.1.2 Thing+ Embedded SDK

The Thing+ Embedded SDK is a C library that can be implemented typically in middle end to high end hardware. The SDK's depencies include libmosquitto, libcurl, libjson-c, openssl. The SDK's primary purpose is to connect hardware to the Thing+ Cloud server and compose MQTT and HTTP messages. Sensing sensors, actuating actuators, sending status messages and values are required features of any hardware used.

Category	**Descriptions**
Application requirement	Sensing sensors Actuating actuators Sending status and value periodically.
Target hardware	Middle end to high end
Dependency	libcurl libjson-c libmosquitto openssl C or C++ application

1.1.3 Thing+ Gateway

The Thing+ Gateway can be implemented on high end hardware. The Thing+ Gateway is a Node.js application program. Whatever hardware is used must have the ability to run node.js. There is another protocol between the Thing+ Gateway and hardware/device applications we call the "Device Agent". The Thing+ Gateway connects hardware to the Thing+ Cloud server and composes MQTT & HTTP messages with the purpose of sending status and data/values to the server. Sensing sensors, actuating actuators and running a JSONRPC server must be possible on any hardware you use to set up a Thing+ gateway.

Category	Descriptions
Application requirement	Sensing a sensors Actuating actuators Sending sensor event values JSONRPC Server
Target hardware	High end
Dependency	Node.js

2. Thing+ Embedded Protocol

The Thing+ defined message protocol used to communicate between hardware and the Thing+ cloud is called the "Thing+ Embedded Protocol". Thing+ Embedded Prococol is based on HTTP and MQTT. Sensor values and actuator commands are sent via MQTT between sensors and devices - data coming out of devices to a gateway is passed via HTTP.

2.1 MQTT

TBD

2.2 Thing+ MQTT Protocol

This chapter describes the MQTT Topic and data format used by Thing+ Platform. Any hardware to be integrated with the Thing+ Cloud Platform should follow the given Thing+ MQTT Specification. When the Thing+ Cloud Platform sends a command to an actuator, the command uses the same Thing+ MQTT Specification. - Thing+ MQTT Message uses QoS1. ‘RETAIN’ of all messages except that the ‘Will’ Message is not used. - Message timezone is always UTC. - A thing transmits data regarding the status of the MQTT connection, gateway and sensor. - The value for each status is defined as below,

Status	Value(strings)
turned-on	on
turned-off	off
Error occurred	err

2.2.1 MQTT connection.

The version of MQTT used by the Thing+ Platform is 3 and the 8883 port is used for MQTT communication. 8883 port should be opened before communicating with the Thing+ Platform.

** Details of the MQTT Connection Spec **

MQTT Connection SPEC	Thing+ Definition
MQTT Version	3
PORT	8883
MQTT Client ID	{gateway_id}
Clean Settion	TRUE
MQTT ID	{gateway_id}
MQTT Passworkd	{APIKEY}
Will Topic	v/a/g/{gateway_id}/mqtt/status
Will Message	err
Will Message Retain	TRUE
Keep Alive[sec]	{report_interval} x 2 (**Recommend**)

2.2.2 Transmission of the MQTT Connection status data

Hardware should transmit the status of the MQTT Connection to the Thing+ Platform after the MQTT connection has been created successfully.

> Transmit the MQTT Connection Success Status

```
TOPIC: v/a/g/__GATEWAY_ID__/mqtt/status
MESSAGE: on
```

Example

```
TOPIC: v/a/g/000011112222/mqtt/status
MESSAGE: on
```

2.2.3 Transmission of the H/W status data

Hardware should transmit its status and the valid time of the status data periodically. When Thing+ fails to get the status data of a specific piece of H/W within the valid time period, Thing+ defines its status using the error status

> Transmit the H/W Status Data

```
TOPIC: v/a/g/__GATEWAY_ID__/status
MESSAGE: __HW_STATUS__,__VALID_TIME__

__HW_STATUS__: "on" or "off"
__VALID_TIME__: Unit is msec
```

Example

```
TOPIC: v/a/g/000011112222/status
MESSAGE: on,146156169505
```

2.2.4 Transmission of the sensor status data

Hardware should transmit the status of a sensor attached to it and the valid time of the status data periodically. When Thing+ fails to get the status data within the valid time period, Thing+ defines its status using the error status

> Transmit the sensor status data

```
TOPIC: v/a/g/__GATEWAY_ID__/s/__SENSOR_ID__/status
MESSAGE: __SENSOR_STATUS__,__VALID_TIME__

__SENSOR_STATUS__ : "on" or "off"
__VALID_TIME__: Unit is msec
```

Example

```
TOPIC: v/a/g/000011112222/s/000011112222-temperature-0/status
MESSAGE: on,146156161192
```

2.2.5 Transmission of the H/W status and sensor's status data

Sensor Status Data can be transmitted along with the H/W status data. It is an efficient way to send the sensor status data combined with the H/W status data in one topic in order to save on network bandwidth cost. When hardware has multiple sensors and/or actuators, each sensor/actuator should have a unique value for identifying it from other sensors/actuators on the hardware. You can also define it by yourself. If you use this method, you can skip chapter 2.2.4

> Transmit H/W status and sensors status data

```
TOPIC: v/a/g/__GATEWAY_ID__/status
MESSAGE: __HW_STATUS__,__VALID_TIME__,__SENSOR_ID__,__SENSOR_STATUS__,__VALID_TIME__, ...(REPEAT
 FOR SENSOR), __SENSOR_ID__,__SENSOR_STATUS__,__VALID__TIME__
}

__HW_STATUS__ : "on" or "off"
__VALID_TIME__: Unit is msec
__SENSOR_STATUS__ : "on" or "off"
```

Example

```
TOPIC: v/a/g/000011112222/status
MESSAGE: on,146156169505,000011112222-onoff-0,on,146156168403,000011112222-temperature-0,off,146
156161192
```

2.2.6 Transmission of the sensor value data

A single sensor value and multiple sensor values for a specific sensor can be transmitted in one go. A sensor value should be paired with the time value when the hardware reads it. When multiple sensor values are transmitted at a time, they should be ordered by time. An array of multiple sensor values is allowed.

> Transmit of the sensor value data

```
TOPIC: v/a/g/__GATEWAY_ID__/s/__SENSOR_ID__
MESSAGE: __TIME__,__VALUE__, ...(REPEAT FOR VALUES), __TIME__,__VALUE__
```

Example

```
TOPIC: v/a/g/000011112222/s/000011112222-temperature-0
MESSAGE: 146156161000,26.5,146156162000,27.5,146156163000,30
```

> Transmit the sensor value data as array

```
TOPIC: v/a/g/__GATEWAY_ID__/s/__SENSOR_ID__
MESSAGE: [__TIME__,__VALUE__, ...(REPEAT FOR VALUES)__TIME__,__VALUE__]
```

Example

```
TOPIC: v/a/g/000011112222/s/000011112222-temperature-0
MESSAGE: [146156161000,26.5,146156162000,27.5,146156163000,30]
```

2.2.7 Transmission of the sensor value data for multiple sensors

Multiple sensor values for multiple sensors can be transmitted at a time. At this time, the sensor value should, for a single sensor, be grouped and ordered by time

> Transmit multiple sensor values for multiple sensors

```
TOPIC: v/a/g/__GATEWAY_ID__
MESSAGE: {"__SENSOR_ID__":[__TIME__,__VALUE__,...(REPEAT FOR VALUES),__TIME__,__VALUE__],"__SENS
OR_ID__":[__TIME__,__VALUE__,...,__TIME__,__VALUE__], ...(REPEAT FOR SENSORS), "__SENSOR_ID__":[
__TIME__,__VALUE__,...(REPEAT FOR VALUES),__TIME__,__VALUE__]}

Messsage includes {, }, [, ], ""
```

Example

```
TOPIC: v/a/g/000011112222
MESSAGE: {"000011112222-temperature-0":[1461563978000,27.5,1461563978000,28.5],"000011112222-hum
idity-0":[146156161000,30,146156162000,35,146156163000,40]}
```

2.2.8 Transmission of the result of a request from Thing+

Hardware should be able to report the result of a request from the Thing+ Platform. the request can be the command for an actuator or a configuration request.

> Transmit the result of a request from Thing+ - Success Case

```
TOPIC: v/a/g/__GATEWAY_ID__/res
MESSAGE: {"id":__MESSAGE_ID__,"result":__RESULT__}
```

> Transmit the result of a request from Thing+ - Failed Case

```
TOPIC: v/a/g/__GATEWAY_ID__/res
MESSAGE: {"id":__MESSAGE_ID__,"error":"code":__ERR_CODE__, "message":__ERR_MSG__}}
```

In case of *errCode*, it is the error code of the failed case. Thing+ follows the JSONRPC Error Code Rule. *errMessge* should have the details about the root cause of the error.

2.2.9 MQTT Messages from Thing+ Platform

A hardware should subscribe below topic for getting the MQTT Message published by Thing+ Platform. Each hardware can subscribe a MQTT message only for it.

> MQTT Topic and Message subscribed by a hardware

```
TOPIC : v/a/g/__GATEWAY_ID__/req
MESSAGE: {"id": __MESSAGE_ID__, "method": __METHOD__, "params": __PARAMS__}

__MESSAGE_ID__ : It is a unique id for identifying each message and reporting the result of each
 request.
__METHOD__ : List of requests from Thing+ Platform.
__PARAMS__ : : Parameters for a method. each method has its own parameters List of Methods defin
ed by the Thing+ platform as below,
```

Example

```
TOPIC: v/a/g/000011112222/req
```

2.2.10 List of Methods and Parameters from the Thing+ Platform

> Method List

Method	Description	Parameters	Param Description
timeSync	Synchronize the local time of a hardware with Thing+ server time	{"time":__TIME__}	__TIME__ : current time in UTC
controlActuator	execute a command on an actuator	{"id":__SENSOR_ID__,"cmd":__CMD__,"options",__OPTIONS__}	__SENSOR_ID__ : ID of an actuator __CMD__ : Command for an actator __OPTIONS__ : Options for a command
setProperty	Environment configuration	{"reportInterval}":__INTERVAL__	__INTERVAL__ : frequency for reporting sensor value in msec
poweroff	turn off the device	None	None
reboot	restart the H/W	None	None
restart	restar the embedded software	None	None
swUpdate	Upgrade the embedded software	None	None
swInfo	소프트웨어 Provide the version of embedded software	None	None

The mandatory methods which should be implemented for embedded software integrated with the Thing+ Platform include both timeSync and setProperty. Others are optional.

timeSync

When hardware gets an MQTT message including this method, it should reset the local time on it as the received time value.

> timeSync params

```
TOPIC: v/a/g/__GATEWAY_ID__/req

REQUEST MESSAGE: {"id":"__MESSAGE_ID__", "method":"timeSync","params":{"time":__SERVER_TIME__}}

RESPONSE IF SUCCESS : {"id":"__MESSAGE_ID__","result":""}
RESPONSE IF FAILED : {"id":"__MESSAGE_ID__","error":{"code":__ERR_CODE__, "message":"__ERR_MSG__
"}}

__SERVER_TIME__ : UTC
```

Example

```
TOPIC: v/a/g/1928dbc93871/req

REQUEST MESSAGE : {"id":"e1kcs13b9","method":"timeSync","params":{"time":1372874401865}}

RESPONSE IF SUCCESS : {"id":"e1kcs13b9","result":""}
REPONSE IF FAILED : {"id":"e1kcs13b9","error":{"code": -32000, "message": "invalid options"}}
```

setProperty

This method is for changing the report interval. The unit of reporting interval values is msec. The Thing+ Portal provides the UI for changing the report interval.

> setProperty params

```
TOPIC : v/a/g/__GATEWAY_ID__/req

REQUEST MESSAGE: {"id":"__MESSAGE_ID__","method":"setProperty","params":{"reportInterval":__INTE
RVAL__}}

RESPONSE IF SUCCESS : {"id" : "__MESSAGE_ID__", "result"}
RESPONSE IF FAILED : {"id":"__MESSAGE_ID__","error":{"code":__ERR_CODE__, "message":"__ERR_MSG__
"}}
```

Example

```
TOPIC: v/a/g/1928dbc93781/req
REQUEST MESSAGE : {"id":"e1kcs13bb","method":"setProperty","params":{"reportInterval":"60000"}}

RESPONSE IF SUCCESS : {"id":"e1kcs13bb","result":""}
RESPONSE IF ERROR : {"id":"e1kcs13bb","error":{"code":-32000,"message":"invalid interval"}}
}
```

controlActuator

This method is for sending commands to an actuator. Commands can be sent from the Thing+ Portal UI and an actuator should execute the delivered command from the Thing+ Platform. Commands and options for each actuator are defined by Thing+. The examples of commands and options for some actuators are below,

> Commonly used actuator commands and options

Actuator	Command	Option
led	on	duration
led	off	
led	blink	duration interval
powerSwitch	on	duration
powerSwitch	off	None

> controlActuator params

```
TOPIC: v/a/g/__GATEWAY_ID__/req
REQUEST MESSAGE: {"id":"__MESSAGE_ID__", "method":"controlActuator", "params":{"id":__SENSOR_ID_
_,"cmd":__CMD__, "options":{__OPTIONS__}}

RESPONSE IF SUCCESS : {"id": "__MESSAGE_ID__", "result":""}
RESPONSE IF FAILED : {"id":"__MESSAGE_ID__","error":{"code":__ERR_CODE__, "message":"__ERR_MSG__
"}}
```

Example: LED ON

```
TOPIC: v/a/g/1928dbc93781/req
REQUEST MESSAGE : {"id":"46h6f8xp3","method":"controlActuator","params":{"id":"led-1928dbc93781-
r","cmd":"on","options":{"duration":3000}}}
RESPONSE IF SUCCESS: {"id":"46h6f8xp3","result":""}
RESPONSE IF FAILED: {"id":"46h6f8xp3","error": {"code":-32000,"message":"invalid options"}}
}
```

2.3 Thing+ HTTP Protocol

TBD

3. Thing+ Embedded SDK

TBD

4, Thing+ Gateway

The Thing+ Gateway is a program implementing the Thing+ MQTT Protocol, distributed by Daliworks. The Thing+ Gateway connects hardware to the Thing+ Cloud Platform and sends sensor status and values, supports server time syncronization, discover sensors and actuators, and performs remote software updates.

The device Agent is a program made by hardware vendors for collecting sensor values and actuating actuators. The Thing+ Gateway requests sensor values and actuating actuators to the Device Agent and the Thing+ Gateway already defined protocol between Thing+ Gateway and Device Agent. The protocol is based on JSONRPC.

Using a Thing+ Gateway is easier and faster than manually implementing all of the Thing+ Embedded Protocols by hand.

4.1 Hardware Requirement

Thing+ Gateways run on embedded hardware, and is written via Node.js. Your hardware MUST have the ability to run Node.js.

> Node.js Running Enviomental

- CPU : arm, ia32, x86, x86_64
- Memory : Minimum 128MB
- OS : LINUX, WINDOWS, MAC, FREEBSD, OPENBSD, android
  - Can NOT support RTOS, MICRO OS

> Storage Requirement

Category	Size
Thing+ Gateway	11 MB
Node.js(binary)	9 MB
Node.js Modules	5 MB
StoreDb, Log	5 MB

4.2 Features

> Thing+ Gateway Features

Feature	Description
Connection	Connecting hardware and Thing+ Cloud Server
Discover	Discover connected sensors and actuators and register it.
TimeSync	Syncronize system time automatically
Store DB	Save sensors value if network is disconnected
SW Update	Upgrade Thing+ Gatweway software from remote

- Connection
  The Thing+ Gateway connects hardware to the Thing+ Cloud. One thing you need to do is launch the Thing+ Gateway with APIKEY once. The Thing+ Gateway sends sensors status and values periodically based on the report interval.

- Discover
  The Thing+ Gateway sends a connected sensors and actuators list to the Thing+ Cloud. If sensors and actuators are hot plugged, the Thing+ Gateway sends an updated list to Thing+ Cloud. If not using Discover features, the user MUST add new sensors and actuators manually via the Thing+ Portal.

- TimeSync
  Hardware time MUST be syncronized with server time. The Thing+ Gateway can, however, syncronize time automatically. If you use the Thing+ Gateway, there`s nothing required for you to do related to time syncronization. The Thing+ Gateway handles everything.

- Store DB
  If your network is disconnected, the Thing+ Gateway saves sensors values to local storage. The saved sensor values will be published after the network is reconnected.

4.3 Device Agent

Device Agent is a program that reads sensors values and actuates actuators. It is programed directly by hardware vendors. The Thing+ Gateway connects to the Device Agent with JSONRPC. The Thing+ Gateway is a JSONRPC Client and the Device Agent acts as a JSONRPC Server with a port number of 50800.

Server	**Device Agent**
PORT	**50800**
Client	Thing+ Gateway

4.3.1 JSONRPC Protocol

JSONRPC is light-weight remote procedure call protocol. It uses JSON as it's data format.(WIKI)

- Request format

property	**description**
method	Name of executino method or service
params	parameters
id	Request ID. Use this id when reponse.

- Response

property	**description**
result	The result of executed service or method
error	error object. Null if no error
id	The ID from request message

** Every message MUST end with NEW LINE(\n)**

- Error codes

Code	message
-32700	Parse error
-32600	Invald Request
-32601	Method not found
-32602	Invald params
-32603	Internal Error
-32000 to -32099	Server error

4.3.2 Device Agent method

Method	Description
discover	Request connected sensors and actuator lists.
sensor.get	Request series sensors value
sensor.set	Request actuating actuators
sensor.setNotification	Request using as event sensor
sensor.notification	Response Event sensors value and status.

4.3.2.1 "discover" method

Thing+ Gateway requests a sensor and actuator list

```
Device Agent <-- Thing+ Gateway
- Request Method : discover
- Request params : N/A

Device Agent --> Thing+ Gateway
- Response Result : [{"deviceAddress": DEV_ID, deviceModelId: DEVICE_MODEL_ID, "sensors":["id":I
D, "type":TYPE, "notification": TRUE or FALSE}, ..., {"id":ID, "type": TYPE, "notification":true
 or false}]}]
  - deviceAddress: Device ID defined by Device Agent. Each device has unique ID.

  - deviceModelId : Device Model ID defined by Thing+
  - sensors: Sensors and actuator lists
    - id: ID
    - name: NAME
    - type : Sensor types defined by Thing+
    - notification : True if event sensor.
```

- Request Example (Thing+ Gateway)

```
{"id":1,"method":"discover","params":[]}\n
```

- Response Example (Device Agent)

```
{"id":1,"result":[{"deviceAddress":"0a0b0c0d0e00", "deviceModelId": "PowerOutlet", "sensors":[{"
id":"0a0b0c0d0e00-temperature-0","type":"temperature","name":"temp0"},{"id":"0a0b0c0d0e00-temper
ature-1","type":"temperature","name":"temp1", "notification": true},{"id":"0a0b0c0d0e00-humidity
-0","type":"humidity","name":"humi0"},{"id":"0a0b0c0d0e00-onoff-0","type":"onoff","name":"di0"},
{"id":"0a0b0c0d0e00-powerSwitch-0","type":"powerSwitch","name":"do0"}]}],"error":null}\n
```


4.3.2.2 "sensor.get" method

Request a series of sensor values

```
Device Agent <-- Thing+ Gateway
- Request Method : sensor.get
- Request params : [SENSOR ID]

Device Agent --> Thing+ Gateway
- Response Result :
                    {"value": VALUE} or
                    {"status": "on"|"off"|"err"} or
                    {"status": "err", "message": ERROR REASONE}

  - value : Sensor value
  - status : Sensor status.("on"|"off"|"err"). If the sensor value is exists and status is empty
, the status is gussing it being "on"
  - message : If status is error, additional message (for error reasons(optional)).
```

- Request Example (Thing+ Gateway)


```
{"id":2,"method":"sensor.get","params":["0a0b0c0d0e00-
temperature-0"]}\n
```


- Response Example (Device Agent)


```
{"id":2,"result":{"value":5.63},"error":null}\n
{"id":2,"result":{"status":"off"},"error":null}\n
{"id":2,"result":{"status":"err","message":"initializing"},"error":null}\n
```


4.3.2.3 "sensor.set" method

Request actuating actuator

```
Device Agent <-- Thing+ Gateway
- Request Method : sensor.set
- Request Params : [Actuator ID, Command, Command Options]

Device Agent --> Thing+ Gateway
- Response Result : Result
```

- Request Example(Thing+ Gateway)


```
{"id":3,"method":"sensor.set","params":["0a0b0c0d0e00-powerSwitch-0","on",null]}\n
```


- Response Example (Device Agent)

```
{"id":3,"result":"on","error":null}\n
```

4.3.2.4 "sensor.setNotification" method

Request to set a sensor event

```
Device Agent <-- Thing+ Gateway
- Request Method : sensor.setNotification
- Request Params : [Sensor ID]

Device Agent --> Thing+ Gateway
- Response Result : "success" if success
```

- Request Example (Thing+ Gateway)

```
{"id":5,"method":"sensor.setNotification","params":["0a0b0c0d0e00-onoff-0"]}\n
```

- Response Example(Device Agent)

```
{"id":5,"result":"success","error":null}\n
```

** If using as an event sensor, the Device Agent MUST send it's sensor status on each report interval **

4.3.2.5 "sensor.notification" method

Response event with sensor values and status

```
Device Agent --> Thing+ Gateway
- Request Method : sensor.notificaion
- Request Params :
                   [Sensor ID, {"value": value}]
                   [Sensor ID, {"status": "on"|"off"|"err"}]
                   [Sensor ID, {"status": "err", "message":"ERROR REASON"}]
- Request Id : 없음

Device Agent <-- Thing+ Gateway
- Response : NONE
```

- Request Example (Device Agent)

```
{"method":"sensor.notification","params":["0a0b0c0d0e00-onoff-0",{"value": 1}]}
{"method":"sensor.notification","params":["0a0b0c0d0e00-onoff-0",{"status": "off"}]}
{"method":"sensor.notification","params":["0a0b0c0d0e00-onoff-0",{"status": "err","message":"ini
tializing"}]}
```

4.3.2.6 "ping" method

Check if the Device Agent is working or not

```
Device Agent <-- Thingp+ Gateway
- Request Method : sensor.setNotification
- Request Params : None

Device Agent --> Thing+ Gateway
- Response Result : "success"
```

- Request Example(Thing+ Gateway) `{"id":6,"method":"ping","params":[]}\n`
- Response Example(Device Agent) `{"id":6,"result":"success","error":null}\n`