Event Bus Interface

Specification

* *IoT anyware* -



Team number 1

June 2015

Revision History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description** |
| 0.1 | 07,06, 2015 | Initial draft |
| 1.0 | 18/06, 2015 | Refine and release v1.0 |
|  |  |  |
|  |  |  |
|  |  |  |

TABLE OF CONTENTS

[Chapter 1. OVERVIEW 3](#_Toc422420956)

[Chapter 2. EVENT FORMAT 3](#_Toc422420957)

[2.1 Description 3](#_Toc422420958)

[2.2 Event format 3](#_Toc422420959)

[2.3 Additional Information 3](#_Toc422420960)

[Chapter 3. TOPIC DESCRIPTION 3](#_Toc422420961)

[3.1 Description 3](#_Toc422420962)

[3.2 Topic Format 4](#_Toc422420963)

[3.3 Method Description 4](#_Toc422420964)

[Chapter 4. PAYLOAD FORMAT 5](#_Toc422420965)

[4.1 Description 5](#_Toc422420966)

[4.2 Common Key-Value for all Methods. 5](#_Toc422420967)

[4.3 Special Key-Value for each Method 5](#_Toc422420968)

[Chapter 5. PROFILE OF SENSOR/ACTUATOR 8](#_Toc422420969)

[5.1 Description of Sensor / Actuator Profile 8](#_Toc422420970)

[5.2 Thermostat Profile 9](#_Toc422420971)

[5.3 Hygrometer Profile 9](#_Toc422420972)

[5.4 Door Profile 11](#_Toc422420973)

[5.5 Light Profile 11](#_Toc422420974)

[5.6 Presence/Proximity Profile 12](#_Toc422420975)

[5.7 Alarm Profile 13](#_Toc422420976)

# Chapter 1. OVERVIEW

SA node is attached to the content-based publish-subscribe bus, thus the interface of SA node is following the event style. The events are categorized as “topic-based” and component subscribes to all events within the topic. Topic includes the unique identifier of each SA node.

Therefore, unique topic should be published or subscribed for the communication with specific SA node. This is the promise between the user application, IoT Server, and SA node.

# Chapter 2. EVENT FORMAT

## 2.1 Description

The event format consists of *Topic* and *Payload*. SA node subscribes specific *Topic* and carries out the command that is sent by publisher. In contrast, SA node publishes the specific *Topic* and sends the status to subscribers. In both cases, *Payload* includes the concrete information of command and status.

## 2.2 Event format

|  |
| --- |
| * Event = [*Topic*] + [*Payload*] * Publish ([*Topic*] , [*Payload*]) * Subscribe([*Topic*]) [*Payload*] |

## 2.3 Additional Information

1) *Payload* should be written in JSON format.

2) *Topic* is described in “Chapter 3.TOPIC DESCRIPTION”

3) *Payload* is described in “Chapter 4.PAYLOAD DESCRIPTION”

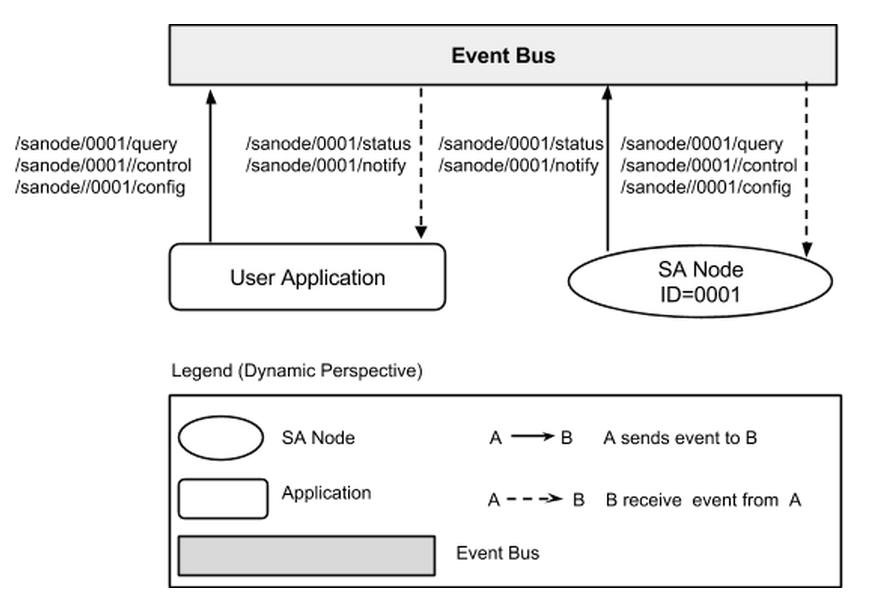
# Chapter 3. TOPIC DESCRIPTION

## 3.1 Description

The *Topic* includes *the categories of system*, *identifier of SA node*, and *Method*.

The category of SA node is “*sanode*” and the identifier is *Serial Number* of SA node.

The *Method* is classified into the *Publish Method* and *Subscribe Method* from the perspective of SA node. Please refer to ***Figure 3.1*** *: The relationship of publish and subscribe*



**Figure 3.1** The relationship of publish and subscribe

## 3.2 Topic Format

|  |
| --- |
| * /sanode/[ID of SA node]/[Method] |

## 3.3 Method Description

1) Description

It is classified into *Publish Method* and *Subscribe Method*. The purpose of Publish Method is that SA node sends the status of Sensor/Actuator or notifies the special event like the emergency message. The purpose of Subscribe Method is that *User Application* or *IoT Server* sends the command like the control of Actuator to SA node.

2) Publish Method from SA Node

|  |  |
| --- | --- |
| Method | Description |
| /sanode/[SA NODE ID]/status | Sensor/actuator status change method. |
| /sanode/[SA NODE ID]/notify | SA node message method (i.e. emergency message). |
| /sanode/[SA NODE ID]/profile | The profile of SA node method. Refer to Chapter 4. |

3) Subscribe Method To SA Node

|  |  |
| --- | --- |
| Method | Description |
| /sanode/[SA NODE ID]/control | Actuator control method. |
| /sanode/[SA NODE ID]/query | Sensor/actuator status query method. |
| /sanode/[SA NODE ID]/config | Configuration update method. |

# Chapter 4. PAYLOAD FORMAT

## 4.1 Description

*Payload* should be written in JSON format. The key-value in *Payload* are classified into command key-value for all *Methods* and specific key-value for each *Method*.

## 4.2 Common Key-Value for all Methods.

|  |  |  |
| --- | --- | --- |
| Key | Mandatory | Description |
| publisher | Optional | It includes ID of the publisher. This purpose is for the logging and sending the response at the special cases. |
| Example | | |
| { publisher : “User App” } | | |

## 4.3 Special Key-Value for each Method

1) Control Method

|  |  |  |
| --- | --- | --- |
| **Class** | Subscribe Method to SA node | |
| **Method** | **Description** | |
| control | Actuator control method. Refer to Chapter 5 for knowing each profile of Actuator | |
| **Payload** | | |
| Key | Mandatory | Description |
| name | mandatory | This indicates the Identifier of Actuator. |
| value | mandatory | This indicates the status of Actuator. |
| **Example** | | |
| [Scenario] User App send the event to SA node for controlling of Actuator.  [*Topic*] : /sanode/0001/control  [*Payload*]  {  “name” : “door”,  “value” : “open”  } | | |

2) Status Method

|  |  |  |
| --- | --- | --- |
| **Class** | Publish Method from SA node | |
| **Method** | **Description** | |
| status | Sensor/actuator status change method. Refer to Chapter 5 for knowing each profile of Sensor/actuator. | |
| **Payload** | | |
| Key | Mandatory | Description |
| name | mandatory | This indicates the Identifier of Sensor/actuator. |
| value | mandatory | This indicates the status of Sensor/actuator |
| **Example** | | |
| [Scenario] SA node send the event to User App that wants to know the status of S/A.  [*Topic*] : /sanode/0001/status  [*Payload*]  {  “name” : “door”,  “value” : “open”  } | | |

3) Query Method

|  |  |  |
| --- | --- | --- |
| **Class** | Sensor/actuator status query method. | |
| **Method** | **Description** | |
| query | Sensor/actuator status change method. | |
| **Payload** | | |
| Key | Mandatory | Description |
| **Example** | | |
| [Scenario] User App send the event to SA node to query the status of S/A  [*Topic*] : /sanode/0001/query  [*Payload*]  { } | | |

4) Config Method

|  |  |  |
| --- | --- | --- |
| **Class** | Subscribe Method to SA node | |
| **Method** | **Description** | |
| config | Configuration update method. | |
| **Payload** | | |
| Key | Mandatory | Description |
| autoTurnOffLightTime | optional | Light turns off when the house is vacant for some time. (unit : sec) |
| autoSetAlarmTime | optional | Alarm is on when the house is vacant and there is no ACK from user for some time. (unit : sec) |
| **Example** | | |
| [Scenario] User App changes the configuration of SA node.  [*Topic*] : /sanode/0001/config  [*Payload*]  {  “autoTurnOffLightTime” : “30”, // 30 sec  “autoSetAlarmTime” : “300” // 5 min  } | | |

5) Notify Method

|  |  |  |
| --- | --- | --- |
| **Class** | Publish Method from SA node | |
| **Method** | **Description** | |
| config | SA node message method | |
| **Payload** | | |
| Key | Mandatory | Description |
| type | optional | Two types. “toast” and “alert” (Default : Toast) |
| level | optional | Two levels. “info” and “warn” (Default : info) |
| message | mandatory | message string or type |
| **Example** | | |
| [Scenario] SA node sends the notification event to the User App.  [*Topic*] : /sanode/0001/notify  [*Payload*]  {  “type” : “toast”,  “level” : “info”,  “message” : “The door is manually opened while alarm is off”  } | | |

# Chapter 5. PROFILE OF SENSOR/ACTUATOR

## 5.1 Description of Sensor / Actuator Profile

Sensor/actuator profile is a data format description of the specific sensor or actuator. The profile is used for describing the SA node capability. System provides genetic sensor and actuator profiles enabling 3rd party SA node developer and/or end user application to add their product without knowing each other. In addition, 3rd party sensor or actuator profiles can be added to the system. However, profiles should be authenticated and validated before registered to the system.

Data sending to the “event bus” should be matched to the associated profile. Data validation is processed by participating entities such as SA node, end user application, and/or the system.(Extensibility)

## 5.2 Thermostat Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Profile | “id” | “com.genetic.thermostat” | |
| “value” | “access” | “r” |
| “type” | “range” |
| “valueList” | [“min”:20,”max”:45] |
| “description” | “The current temperature” |
| Example | <Example #1> SA node sends the temperature to user app  [Topic]  /sanode/0001/status  [Payload]  {  “publisher” : “0001”, //optional  “name” : “thermostat1”,  “value” : “40”  } | | |

## 5.3 Hygrometer Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Profile | “id” | “com.genetic.hygrometer” | |
| “value” | “access” | “r” |
| “type” | “range” |
| “valueList” | [“min”:20,”max”:45] |
| “description” | “The current humidity” |
| Example | <Example #1> SA node sends the humidity to user app  [Topic]  /sanode/0001/status  [Payload]  {  “publisher” : “0001”, //optional  “name” : “hygrometer1”,  “value” : “40”  } | | |

## 5.4 Door Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Profile | “id” | “com.genetic.house.door” | |
| “value” | “access” | “rw” |
| “type” | “list” |
| “valueList” | [“open”,”close”] |
| “Description” | “door” |
| Example | <Example #1> User App control the door in SA node  [Topic]  /sanode/0001/control  [Payload]  {  “publisher” : “userApp1”, //optional  “name” : “door1”,  “value” : “open”  }  <Example #2> SA node sends the status of the door to user app  [Topic]  /sanode/0001/status  [Payload]  {  “publisher” : “0001”, //optional  “name” : “door1”,  “value” : “close”  } | | |

## 5.5 Light Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Profile | “id” | “com.genetic.house.light” | |
| “value” | “access” | “rw” |
| “type” | “list” |
| “valueList” | [“on”,”off”] |
| “description | “light” |
| Example | <Example #1> User App controls the indoor light in SA node  [Topic]  /sanode/0001/control  [Payload]  {  “publisher” : “userApp1”, //optional  “name” : “indoorLight1”,  “value” : “on”  }  <Example #2> User App controls the outdoor light in SA node  [Topic]  /sanode/0001/control  [Payload]  {  “publisher” : “userApp1”, //optional  “name” : “outdoorLight1”,  “value” : “off”  } | | |

## 5.6 Presence/Proximity Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Profile | “id” | “com.genetic.proximity” | |
| “value” | “access” | “r” |
| “type” | “list” |
| “valueList” | [“vacant”,”occupied”] |
| “description” | “Presense,Proximity” |
| Example | <Example #1> SA node sends the status of proximity to user app  [Topic]  /sanode/0001/status  [Payload]  {  “publisher” : “0001”, //optional  “name” : “proximity1”,  “value” : “vacant”  } | | |

## 5.7 Alarm Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Profile | “id” | “com.genetic.alram” | |
| “value” | “access” | “rw” |
| “type” | “list” |
| “valueList” | [“on”,”off”] |
| value | on |
| off |
| Example | <Example #1> SA node sends the status of alram to user app  [Topic]  /sanode/0001/status  [Payload]  {  “publisher” : “0001”, //optional  “name” : “aram1”,  “value” : “on”  }  <Example #2> User App controls the alram in SA node  [Topic]  /sanode/0001/control  [Payload]  {  “publisher” : “userApp1”, //optional  “name” : “alram1”,  “value” : “on”  } | | |