advanech.jpg

**Sensor Network Manager API**

**and Data Format**

The document is provided to you for references and is subject to change. Please always get latest version from Advantech to sync.

|  |  |  |
| --- | --- | --- |
| Date | Version | Description / Major change |
| 2015/3/30 | 1.1 | Eric Liang, create draft document |
| 2015/10/08 | 1.2 | Eric Liang, add Get/Set API |
|  |  |  |
|  |  |  |
|  |  |  |

**Abstract**

This document defines these Advantech Sensor Network Manager APIs and Sensor Data Format for use in Advantech wireless sensor network device.

**Chapter 1**

**Overview**

**1.1 Introduction**

This document describes how to use the Advantech Sensor Network Manager (Adv SNM) module to get/set IoT sensor network information, data, or status for your sensor device.

The Advantech sensor data format is refer to IPSO Alliance Guide[1]. Use Sensor Markup Language (SENML)[3] defines media types for representing sensor measurements and device parameters in the SenML. Representations are defined in JavaScript Object Notation (JSON)[5].

It includes get IoT gateway interface capability, get/set sensor hub information / sensor data, auto register / deregister sensor network hub, and auto report sensor data.

**1.2 Terminology**

This specification makes use of the following additional terminology:

**IoT Gateway (IoTGW):** It integrates technologies and protocols for networking, embedded control, enterprise-grade security, and easy manageability on which application-specific software can run. Connectivity up to the WISE Cloud and down to sensors and existing controllers embedded in the system. Pre-process filtering and gathering of selected data for delivery.

**Sensor Hub (SenHub)**: A sensor hub is capable of performing some processing, gathering sensory information and communicating with other connected hubs in the network. A sensor hub should be small in size, consume extremely low energy, operate in high volumetric densities, be autonomous and operate unattended, and be adaptive to the environment. As sensor hubs are typically very small electronic devices, they can only be equipped with a limited power source.

**Sensor (Sen)**: Sensor is hardware device that produce a measurable response to a change in a physical condition like temperature or pressure. Sensor measures physical data of the parameter to be monitored.

**Resource Type(rt)**: Each root resource of a function set is assigned a Resource Type parameter, therefore making it possible to discover it. The Resource Type parameter defines the value that MUST be included in the rt= field of the sensor.

**1.3 IPSO Application Framework**

The IPSO Application Framework makes use of IETF standards as building blocks for a simple and efficient RESTful design model for IP smart objects. The framework may be used over either HTTP or CoAP web transfer protocols.

HTTP, REST, XML, JSON, COAP and other key components of web technology are powerful mechanisms in an Internet of Things application.

***Note: The objective of the Alliance is not to define technologies, but to document the use of IP-based technologies defined at the standard organizations such as IETF with focus on support by the Alliance of various use cases.***

**Function Sets**: The framework is organized into groups of resource types called Function Sets. A Function Set has a recommended root path, under which its sub-resources are organized. Each Function Set is assigned a Resource Type parameter, therefore making it possible to discover it.

|  |  |  |
| --- | --- | --- |
| Function Set | Root Path | Resource Type |
| General Purpose IO | /gpio | gpio |
| Power | /pwr | pwr |
| Load Control | /load | load |
| Sensors | /sen | sen |
| Light Control | /lt | lt |
| Message | /msg | msg |
| Location | /loc | loc |
| Configuration | /cfg | cfg |

**1.4 IPSO Smart Object**

IPSO Smart Object[2] Guidelines provide a common design pattern, an object model, that can effectively use the IETF CoAP protocol to provide high level interoperability between Smart Object devices and connected software applications on other devices and services

The common object model is based on the Lightweight M2M (LWM2M 1.0) specification from the Open Mobile Alliance. OMA LWM2M is a device management and service architecture specification based on IETF CoAP, and provides a simple and flexible object template (object model) for constrained device management.

The object model from OMA LWM2M is reused to define application level IPSO Smart Objects. This enables the OMA Name Authority (OMNA) to be used to register new objects, and enables existing LWM2M compliant device libraries and server software to be used as an infrastructure for IPSO Smart Objects

This first IPSO Smart Object Guideline describes 18 Smart Object types **[Appendix A]**, including a temperature sensor, a light controller, an accelerometer, a presence sensor, and other common sensor and actuator types representing a variety of use case domains. It is intended as a “starter pack” and example of how IPSO Smart Objects can be built to address some application specific use cases

**1.5 Media Types for Sensor Markup Language (SENML)**

**Semantics**

|  |  |  |  |
| --- | --- | --- | --- |
| SenML | JSON | Type | Description |
| Base Name | bn | String | This is a string that is prepended to the names found in the entries |
| Base Time | bt | Integer | A base time that is added to the time found in an entry |
| Base Units | bu | String | A base unit that is assumed for all entries, unless otherwise indicated |
| Version | ver | Number | Version number of media type format |
| Measurement or Parameters | e | Array | Array of values for sensor measurements or other generic parameters |
| Name | n | String | Name of the sensor or parameter |
| Units | u | String | Units for a measurement value |
| Value | v | Float | Value of the entry |
| String Value | sv | String |  |
| Boolean Value | bv | Boolean |  |
| Value Sum | s | Float | Integrated sum of the values over time |
| Time | t | Integer | Time when value was recorded |
| Update Time | ut | Integer | Update time. A time in seconds that represents the maximum time before this sensor will provide an updated reading for a measurement. |

**The Data Type of Sensor Data Type**

|  |  |
| --- | --- |
| Data Type (type) | SenML Field |
| b (boolean) | bv |
| s (string) | s |
| e (enum) | e |
| i (integer) | v |
| d (decimal) | v |
| h(hexadecimal) | s |
| o(octet-stream) | s |

**Advantech Sensor Semantics**

|  |  |  |  |
| --- | --- | --- | --- |
| SenML | JSON | Type | Description |
| Min Range Value | min | Float | The minimum value that can be measured by the sensor |
| Max Range Value | max | Float | The maximum value that can be measured by the sensor |
| Access Mode | asm | String | The access mode of the resource. Ex: read (r), write (w), read/write (rw) |
| Standard Format | st | String | The sensor format is which standard format |
| Health Status | Health | Integer | The health status of network or device. Range: -1 ~ 100  Good: > 80, Average: 60 ~ 80, Below Average: 40~60, Bad:0~40, -1: Off line or Fault |
|  |  |  |  |

**Example:**

{"n":"Room Temp","u":"Cel","v":0,"min":0,"max":100,"asm":"r","type":"d","rt":"ucum.Cel","st":"ipso","exten":"sid=3303"}

**Chapter 2**

**Application Program Interface (API)**

**2.1 Event of Callback**

typedef enum {

// Interface: 1000 ~ 1999

SN\_Inf\_UpdateInterface\_Data = 1000, // Update Interface data

// Sensor Hub: 2000 ~ 2999

SN\_SenHub\_Register = 2000, // A SenHub connected

SN\_SenHub\_SendInfoSpec = 2001, // Report the resource information of SenHub

SN\_SenHub\_AutoReportData = 2002, // Auto report data value of SenHub

SN\_SenHub\_Disconnect = 2003, // A SenHub disconnected

SN\_SetResult = 3000, // Return the result of setting value

} SN\_EVENT\_ID;

**2.2 Command ID**

typedef enum {

SN\_Set\_ReportSNManagerDataCbf = 3000, // Set Event Callback function point

}SN\_CTL\_ID;

**2.3 Return Code Definition**

typedef enum {

SN\_ER\_NOT\_IMPLEMENT = -13, // Dose Not Support this command (501)

SN\_ER\_TIMEOUT = -12 // Request Timeout (408)

SN\_ER\_SYS\_BUSY = -11 // System is busy (503)

SN\_ER\_VALUE\_OUT\_OF\_RANGE = -10 // Value is out of range (416)

SN\_ER\_SYNTAX\_ERROR = -9 // Format is correct but syntax error (422)

SN\_ER\_FORMAT\_ERROR = -8 // Format error (415)

SN\_ER\_REQUEST\_ERROR = -7 // Request error (400)

SN\_ER\_RESOURCE\_LOSE = -6 // Resource is lose (410)

SN\_ER\_RESOURCE\_LOCKED = -5, // Resource is in setting (426)

SN\_ER\_NOT\_FOUND = -4, // Resource Not Found (404)

SN\_ER\_WRITE\_ONLY = -3, // Write Only (405)

SN\_ER\_READ\_ONLY = -2, // Read Only (405)

SN\_ER\_FAILED = -1, // Failed (500)

SN\_OK = 0, // Successfully (200)

SN\_INITILIZED = 1, // Library had initilized

} SN\_CODE;

**2.4 Structure Definition**

typedef struct \_SenHubInfo

{

char sMAC[MAX\_SN\_MAC]; // MAC: remove '-' and ':')

char sHostName[MAX\_SN\_HOSTNAME]; // WISE1020-MAC(4)

char sSN[MAX\_SN\_SNO]; // MAC: remove '-' and ':')

char sProduct[MAX\_SN\_PRODUCT\_NAME]; // WISE1020

void \*pExtened;

}SenHubInfo;

**2.5 Sensor Network Manager API Definition:**

**Syntax**

**SNMCODE SN\_Manager\_Initialize( );**

**Description**

This function initializes the Sensor Network Manager API Library. You must call this function before using this module.

**Parameters**

None

**Return Values**

SNM\_\_OK: Successfully to initialize the Sensor Network Manager module.

SNM\_FAILED: Failed to initialize the Sensor Network Manager module.

**Syntax**

**SNMCODE SN\_Manager\_Uninitialize();**

**Description**

This function uninitialized the Sensor Network Manager Module. You must call this function before closing this module to release resources.

**Parameters**

None

**Return Values**

SNM\_\_OK: Successfully to uninitialized the Sensor Network Manager module.

SNM\_FAILED : Failed to uninitialized the Sensor Network Manager module.

**Syntax**

**SNMCODE SN\_Manager\_GetCapability(char \*\*** ppzDataBuffer**, int \*pnLen );**

**Description**

This function is used to get all interfaces information of this IoT Gateway in JSON Format.

**Parameters**

\*ppzDataBuffer

[out] the point to the Sensor Network Interfaces’ information data with JSON format

\*pnLen

[out] the length of the data in ppzDataBuffer

**Return Values**

SNM\_\_OK: Get the IoTGW interface information successfully.

SNM\_FAILED: Failed to get IoTGW interface information.

**Example:**

**Condition**

Communication Type: WSN

Number of WSN Interface: 2

Name of Interfaces: WSN0, WSN1

MAC of Interfaces: 0000852CF4B7B0E8, 0000852CF4B7B0E7

"IoTGW": {

"WSN": {

"WSN0": {

"Info": {

"e": [{"n":"SenHubList","sv":"","asm":"r"},

{"n":"Topology","sv":"","asm":"r"}, {"n":"Health","v":100,"asm":"r"},

{"n":"Name","sv":"WSN0","asm":"r"}],

"bn":"Info"

},

"bn":"0000852CF4B7B0E8",

"ver":1

},

"WSN1": {

"Info": {

"e": [{"n":"SenHubList","sv":"","asm":"r"},

{"n":"Topology","sv":"","asm":"r"}, {"n":"Health","v":100,"asm":"r"},

{"n":"Name","sv":"WSN1","asm":"r"}],

"bn":"Info"

},

"bn":"0000852CF4B7B0E7",

"ver":1

},

"bn":"WSN"

},

"ver": 1

}

**Syntax**

**SNMCODE SN\_Manager\_ActionProc**( const int nCtlId,

void \*pParam1,

void \*pParam2,

void \*pRev1,

void \*pRev2 );

**Description**

This function is used to set callback function or others parameters depended on nCtlId.

**Parameters**

nCtlId

[in] control id. Refer to SN\_CTL\_ID

\*pParam1

[in] the point to parameter 1 ( data type depend on SN\_CTL\_ID)

\*pParam2

[in] the point to parameter 2 ( data type depend on SN\_CTL\_ID)

\*pPev1

[in] reserved for later use. Must be “NULL”

\*pPev2

[in] reserved for later use. Must be “NULL”

**Return Values**

SNM\_\_OK: Successfully to call this function.

>0 : Failed to call this function.

**Example:**

nCtlId: SN\_Set\_ReportSNManagerDataCbf

\*pParam1: ReportSNManagerDataCbf

the ReportSNManagerDataCbf that specify the callback function to be used to notify event.

**Syntax**

typedef int (\***ReportSNManagerDataCbf**) ( const int nEventID,

const char \*pzDataBuffer,

const int nDataLen,

void \*\*ppOutParam,

void \*pParam1,

void \*pRev1 );

**Description**

This function is used to receive events that trigger by Sensor Network Manager module. (ex: A New SenHub Connected/Disconnected/Auto Report Sensor Data/Auto Report Interface Data).

**Parameters**

nEventID

[in] event id. Refer to SN\_EVENT\_ID

\*pzDataBuffer

[in] the point to data with JSON format

nDataLen

[in] the length of the data in pzDataBuffer

\*\*ppOutParam

[out] User can specify the address or context value when “SN\_SenHub\_Register” event called,

It would be sent back when callback function gets called with this SenHub’s

events (SN\_SenHub\_SendInfoSpec, SN\_SenHub\_AutoReportData,

,SN\_SenHub\_Disconnect)

\* pParam1

[in] the point of data. The data type depends on “SN Event ID”.

\*pRev1

[in] reserved for later use.

**Return Values**

SNM\_\_OK: Process the callback event successfully.

SNM\_FAILED: Failed to process the callback event

**Example of Event Code:**

1. **SN\_Inf\_UpdateInterface\_Data:**

**Condition**

Communication Type: WSN

Name of Interface: WSN0

MAC of Interface: 0000852CF4B7B0E8

Sensor Hubs connected: 0000000EC6F0F830 ,0000000EC6F0F831

Network Health Status: 60

Neighbor: 0000000EC6F0F830

pzDataBuffer : Update Interface Data

"IoTGW": {

"WSN": {

"WSN0"{

"Info": {

"e": [{"n":"SenHubList","sv":"0000000EC6F0F830 ,0000000EC6F0F831"},

{"n":"Neighbor","sv":"0000000EC6F0F830"},

{"n":"Health","v":60}],

"bn":"Info"

}

"bn":"0000852CF4B7B0E8",

"ver":1

},

"bn":"WSN"

},

"ver": 1

1. **SN\_SenHub\_Register**

**Condition**

Sensor Hub: 0000000EC6F0F830

Name of Sensor Hub: Room

pzDataBuffer

[in] Empty char buffer point

ppOutParam

[out] the point that storage the user specify address or context value.

pParam1

[in] the point of data structure. “SenHubInfo”

1. **SN\_SenHub\_SendInfoSpec:**

**Condition**

Sensor Hub: 0000000EC6F0F830

Name of Sensor Hub: Room

Sensors of SenHub: temp1, co2, digital output

pzDataBuffer : Send SenHub Information

{

"SenHub":{

"SenData": {

"e": [{"n":" temp1", "u":"Cel","v":0,"min":0,"max":100,"asm":"r","type":"d","rt":"ucum.Cel","st":"ipso","exten":""},

{"n":"co2","u":"ppm","v":0,"min":0,"max":100000,"asm":"r","type":"d","rt":"ucum.ppm","st":"ipso","exten":""},

{"n":"digital output", "u":"","bv":0,"asm":"rw","type":"b","rt":"gpio.dout","st":"ipso","exten":""}],

"bn":"SenData"

},

"Info": {

"e": [ {"n":"Name","sv":"Room","asm":"rw"},

{"n":"sw","sv":"1.0.0.2","asm":"r"},

{"n":"reset","bv":0,"asm":"rw"}],

"bn":"Info"

},

“Net”:{

"e": [ {"n":"Neighbor","sv":"0000000EC6F0F831","asm":"r"},

{"n":"Health","v":88,"asm":"r"},

{"n":"sw","sv":”1.0.0.1”,"asm":"r"}],

"bn":"Net"

},

"ver": 1

}

}

ppOutParam:

[in] the point of address or context value that pass by user at SN\_SenHub\_Register event.

1. **SN\_SenHub\_AutoReportData**

**Condition**

Sensor Hub: 0000000EC6F0F830

Name of Sensor Hub: Room

Sensors of SenHub: temp1 (v:17), co2(v:1001), digital output(bv:0)

pzDataBuffer : Update SenHub Data

{

"SenHub":{

"SenData": {

"e": [{"n":"temp1","v":17},

{"n":"co2", "v":1001},

{"n":"digital output", "bv":0}],

"bn":"SenData",

"ver":1

},

"Info": {

"e": [{"n":"Health","v":100,"asm":"r"},

{"n":"Name","sv":"Room","asm":"rw"},

{"n":"IoTGW","sv":"0000852CF4B7B0E8","asm":"r"}],

"bn":"Info"

},

"ver": 1

}

}

ppOutParam:

[in] the point of address or context value that pass by user at SN\_SenHub\_Register event.

1. **SN\_SenHub\_Disconnect**

Condition

Sensor Hub: 0000000EC6F0F830

pzDataBuffer : Empty char buffer point

ppOutParam:

[in] the point of address or context value that pass by user at SN\_SenHub\_Register event.

**Syntax**

**char\* SN\_Manager\_GetData(const char \*pszInURI, const ACTION\_MODE mode);**

**Description**

This function is used to get values of resource of sensor network manager

( ex: network interface, sensor hub, or sensor ).

**Parameters**

pszInURI

[in] uri path of resource.

Example:

SenHub: SenHub’s UID/bn/bn/n or SenHub’s UUID/bn/bn

Interface: bn/bn/Interface’s MAC/bn/n

mode

[in] mode of getting ( cache or direct mode)

**Return Values**

{“StatusCode”:200, “Result”:{JSON Object}}

StatusCode : Please refer [Appendix C] for status code.

Result : Result of getting in JSON format

**Example:**

Get the SenHub (0000000EC6F0F830)’s temp1 value

pzInURI: 0000000EC6F0F830/SenHub/SenData/temp1 mode: cache

**Result:**

{“StatusCode”:200, “Result”: {"n":"temp1","v":17}}

**Syntax**

**char\* SN\_Manager\_SetData(const char \*pszInURI, const char \*pszValue );**

**Description**

This function is used to get value of resource of sensor network manager in asynchronous.

( ex: network interface, sensor hub, or sensor ). This function would check the set resource and value is correct and settable. If all parameters are correct and returns accept this request immediately. Then it will set value to resource and return the result in asynchronous.

**Parameters**

pszInURI

[in] path of resource. (Note: Need to specify to “n” resource not “bn” )

pszValue

[in] the set value in JSON format

**Return Values**

{“StatusCode”:200, “Result”:{JSON Object}}

StatusCode : Please refer [Appendix C] for status code.

Result : Result of setting in JSON format

**2.5 Application Sample Code**

Please refer to the distribution package for the sample code.

**Appendix A:**

Smart Objects defined by this Technical Guideline

|  |  |  |
| --- | --- | --- |
| **Object** | **Object ID** | **Multiple Instances?** |
| **IPSO Digital Input** | 3200 | Yes |
| **IPSO Digital Output** | 3201 | Yes |
| **IPSO Analogue Input** | 3202 | Yes |
| **IPSO Analogue Output** | 3203 | Yes |
| **IPSO Generic Sensor** | 3300 | Yes |
| **IPSO Illuminance Sensor** | 3301 | Yes |
| **IPSO Presence Sensor** | 3302 | Yes |
| **IPSO Temperature Sensor** | 3303 | Yes |
| **IPSO Humidity Sensor** | 3304 | Yes |
| **IPSO Power Measurement** | 3305 | Yes |
| **IPSO Actuation** | 3306 | Yes |
| **IPSO Set Point** | 3308 | Yes |
| **IPSO Load Control** | 3310 | Yes |
| **IPSO Light Control** | 3311 | Yes |
| **IPSO Accelerometer** | 3313 | Yes |
| **IPSO Magnetometer** | 3314 | Yes |
| **IPSO Barometer** | 3315 | Yes |

**Table 1**

**Appendix B:**

**The Unified Code for Units of Measure (UCUM)**

[**http://unitsofmeasure.org/ucum.html**](http://unitsofmeasure.org/ucum.html)

|  |  |
| --- | --- |
| Symbol | Description |
| m | meter |
| kg | kilogram |
| s | second |
| A | ampere |
| K | kelvin |
| cd | candela |
| mol | mole |
| Hz | hertz |
| rad | radian |
| sr | steradian |
| N | newton |
| Pa | pascal |
| J | joule |
| W | watt |
| C | coulomb |
| V | volt |
| F | farad |
| Ohm | ohm |
| S | siemens |
| Wb | weber |
| T | tesla |
| H | henry |
| Cel | degrees Celsius |
| lm | lumen |
| lx | lux |
| Bq | becquerel |
| Gy | gray |
| Sv | sievert |
| kat | katal |
| pH | pH acidity |
| % | Value of a switch. A value of 0.0 indicates the switch is off while 100.0 indicates on. |
| count | counter value |
| %RH | Relative Humidity |
| m2 | area |
| l | volume in liters |
| m/s | velocity |
| m/s2 | acceleration |
| l/s | flow rate in liters per second |
| W/m2 | irradiance |
| cd/m2 | luminance |
| Bspl | bel sound pressure level |
| bit/s | bits per second |
| lat | degrees latitude. Assumed to be in WGS84 unless another reference frame is known for the sensor. |
| lon | degrees longitude. Assumed to be in WGS84 unless another reference frame is known for the sensor. |
| %EL | remaining battery energy level in percents |
| EL | remaining battery energy level in seconds |
| beet/m | Heart rate in beets per minute |
| beets | Cumulative number of heart beats |
| h | Hour. |

**Table 2**

**Appendix C:**

Status Code

|  |  |
| --- | --- |
| Status Code | Description |
| 200 | OK |
| 201 | Created |
| 202 | Accepted |
| 203 | Non-Authoritative Information |
| 204 | No Content |
| 400 | Bad Request |
| 401 | Unauthorized |
| 403 | Forbidden |
| 404 | Not Found |
| 405 | Method Not Allowed |
| 406 | Not Acceptable |
| 408 | Request Timeout |
| 409 | Conflict |
| 410 | Resource is Gone |
| 415 | Unsupported Media Type |
| 416 | Requested Range Not Satisfiable |
| 426 | Locked |
| 500 | Internal Server Error |
| 501 | Not Implemented |
| 503 | Service Unavailable |

**Table 3**

**Reference**

**[1] IPSO Alliance:** [**http://www.ipso-alliance.org/**](http://www.ipso-alliance.org/)

**[2] IPSO Smart Object: IPSO-Smart-Objects-1.0.pdf**

**[3] SenML:** [**https://datatracker.ietf.org/doc/draft-jennings-senml/**](https://datatracker.ietf.org/doc/draft-jennings-senml/)

**[4] UCUM:** [**http://unitsofmeasure.org/trac/**](http://unitsofmeasure.org/trac/)

**[5] JavaScript Object Notation (JSON)** [**http://json.org/**](http://json.org/)