

Data Interface Instruction

Model: G1

Version: V2.0.0

Contents

Preface.....	2
1. Data Collection.....	3
1.1 Json Format.....	3
1.2 Binary Format.....	5
2. Service Access.....	9
2.1 MQTT Access.....	10
2.2 HTTP Access.....	16
3. Remote Control (only MQTT).....	19
3.1 Reboot Gateway.....	21
3.2 Heartbeat Package.....	21
3.3 Parameter Configuration.....	22
3.4 Get Version.....	28
3.5 Upgrade Firmware.....	29
4. U-disk Storage.....	30

Preface

This specification applies to G1-B,G1-C gateway.

Revision	Date	Author	Description
v1.3.0	2018.01.12	Yancy	Initial draft
v1.4.0	2018.08.25	Yancy	1.Add <u>Get Version</u> in Remote Control chapter 2.Add <u>Upgrade Firmware</u> in Remote Control chapter 3.Improve the <u>Parameter configuration</u> in Remote Control chapter
v2.0.0	2019.6.22	Yancy	1.Add keep alive interval parameter in <u>MQTT Access</u> 2.Add raw data filter 3.Add <u>U-disk Storage</u> function

1. Data Collection

After the G1 gateway is started, BLE broadcast data will be continuously collected by gateway. If the network is available, the data will be sent to the accessed server by default once every 1 second.

There are two type data formats for G1 gateway uploading to server, one is the Json array data format (including Long and Short data format), the other is Binary's original hex data format (including Long and Short data format).

1.1 Json Format

The following four data types are available in the format of the Json array that G1 gateway uploads:

Type	Description
<i>iBeacon</i>	Belong to BLE data type, iBeacon protocol proposed by Apple. https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf
<i>S1</i>	Belong to BLE data type, Minew's S1 temperature & humidity sensor.
<i>Unknown</i>	Belong to BLE data type, unrecognized BLE broadcast data.
<i>Gateway</i>	Does not belong to the BLE data type, which indicates the current status of the gateway.

Currently the BLE devices can be parsed by G1 gateway is such as a standard ***iBeacon*** device and a S1 sensor. In addition to the data of these two BLE devices, the raw data of the BLE is uploaded for the server to parse itself.

In addition, a data type of ***Gateway*** is added to show that the data of the JSON array is uploaded by the gateway.

Json Long Data Format Sample

```
[
  {
    "timestamp": "2017-04-28T08:16:13Z",
    "type": "iBeacon",
    "mac": "CC0101000011",
    "bleName": "MiniBeacon_00012",
    "ibeaconUuid": "FDA50693A4E24FB1AFCFC6EB07647825",
    "ibeaconMajor": 10001,
    "ibeaconMinor": 19641,
    "ibeaconTxPower": -58,
    "rssi": -76,
    "battery": 100,
  },
  {
    "timestamp": "2017-04-28T08:16:14Z",
    "type": "S1",
    "mac": "CC3101000034",
    "bleName": "S1",
    "rssi": -76,
    "battery": 100,
    "temperature": 21.23,
    "humidity": 43.23
  },
  {
    "timestamp": "2017-04-28T08:16:14Z",
    "type": "Unknown",
    "mac": "EF3101000034",
    "bleName": "MI",
    "rssi": -76,
    "rawData": "3A4E24FB1AFCFC6EB07647825FDA50693A4E24FB1AFCFC6EB07647825"
  },
  {
    "timestamp": "2017-04-28T08:16:14Z",
    "type": "Gateway",
    "mac": "EF3101000034",
    "gatewayFree": 78,
    "gatewayLoad": 0.5
  }
]
```

Note: Json data format is divided into two formats. The long data format contains all the above fields. In short data format, except for Gateway data type, the BLE data type contains only two fields, MAC and RSSI.

The Json Long Data Format fields is described in detail as follows:

Field	Description
timestamp	ISO 8601 time format
type	Optional value: <i>iBeacon/S1/Unknown/Gateway</i>
mac	MAC's hexadecimal uppercase character form, when type is Gateway , the MAC address is the gateway's MAC address, and the other is the MAC address of the BLE.
bleName	Field that exist when device belong to BLE type. The string is empty when parsing is not available.
ibeaconUuid	Field that exist when device belong to iBeacon BLE type.

ibeaconMajor	Field that exist when device belong to iBeacon BLE type.
ibeaconMinor	Field that exist when device belong to iBeacon BLE type.
rssi	Field that exist when device belong to BLE type.
ibeaconTxPower	Field that exist when device belong to iBeacon BLE type.
battery	Field that exist when device belong to S1 or iBeacon BLE type. The integer is zero when parsing is not available.
temperature	Field that exist when device belong to S1 BLE type.
humidity	Field that exist when device belong to S1 BLE type.
rawData	rawData is hexadecimal uppercase character form of BLE broadcast data. Field that exist when device belong to Unknown BLE type.
gatewayFree	Field that exist when device belong to Gateway type,unit is MBytes.
gatewayLoad	Field that exist when device belong to Gateway type.

1.2 Binary Format

The Binary data format is divided into long and short data formats, and the format is defined as follows:

	Frame Header				BLE Frame	Frame Tail
	Protocol Header	Data Length	Gateway MAC Address	Number of BLE	BLE Frame=Body*Number of BLE	Protocol Tail
Long	0xBB01	Include	Include	Include	Include	0xDD
Short	0xBB00	Include	Include	Include	Include	0xDD
Desc.	2 bytes Fixed protocol header	4 bytes The byte length of the whole data frame	6 bytes MAC address of gateway	2 bytes Number of BLE device	* bytes BLE Frame(* bytes)=BLE numbers*Body	1 byte Fixed protocol tail

The data format of the body in the BLE frame is defined as follows:

	Body						
	BLE MAC Address	Number of BLE Data	BLE Data				...
			Raw Data Length	Raw Data	Response	rss	
	Long	Include	Include	Include	Include	include	
Short	Include	Include	exclude	exclude	exclude	include	
Desc.	6 bytes Mac address of BLE	2 bytes The number of times the BLE Data is received	1 byte Length of Raw Data which may be 0	Raw Data Length bytes Raw data of BLE,specific format refer to its specification	1 byte Response packet or not 0x00: Advertising packet 0x01: Scan Response packet	1 byte Rssi value	(Number of BLE Data - 1)*BLE Data

Binary Long Data Format Sample:

```

BB01 0000 0329 AC23 3FC0 000D 000F 1234 5678 A600 0001 1E02 0106 1AFF
4C00 0215 0112 2334 4556 7889 900A ABBC CDDE EFF0 03E8 07D0 C500 C618
6590 DF4F B200 010B 0201 0607 FF4C 0010 020B 0000 BE20 1700 0113 0100
0114 0201 0603 03E1 FF0C 16E1 FFA1 0E00 0113 0100 1720 00B7 2017 1218
1103 0001 1E02 0106 1AFF 4C00 0215 0112 2334 4556 6778 8990 0AAB BCCD
DEEF 03E8 05D2 C500 CF23 CE63 0D41 DE00 011F 1EFF 0600 0109 2000 F184
2846 888D A6EC 08E3 1892 101A BA3A F36B 45FB 6DD1 1200 C166 6666 6669
6900 011D 0201 0603 03AA FE15 16AA FE00 E800 0000 0000 0000 0000 FF00
0000 0000 9000 C999 8877 6612 3400 041E 0201 061A FF4C 0002 15AB 8106
93A4 E24F B1AF CFC6 EB07 6478 2500 0100 01C5 00D1 1E02 0A00 0816 F0FF
6400 0100 0111 094D 696E 6962 6561 636F 6E5F 3730 3330 3201 D11E 0201
061A FF4C 0002 15AB 8106 93A4 E24F B1AF CFC6 EB07 6478 2500 0100 01C5
00D0 1E02 0A00 0816 F0FF 6400 0100 0111 094D 696E 6962 6561 636F 6E5F
3730 3330 3201 D0AB ABAB AB22 2200 011D 0201 0603 03AA FE15 16AA FE00
E800 0000 0000 0000 0000 0100 0000 0000 9100 C5AB CDEF 2300 1500 0214
0201 0603 03E1 FF0C FFE1 FF52 4201 0001 1200 6401 00BC 0001 BCAB CDEF
2300 2B00 0114 0201 0603 03E1 FF0C FFE1 FF52 4201 0001 1100 6401 00BC
AC23 3F23 C311 0001 1902 0106 0303 AAFE 1116 AAFE 2000 0BF4 1200 0065
F1D0 011C 4E76 00B7 C0CC 90F7 465D 0002 1E02 0106 1AFF 4C00 0215 FDA5
0693 A4E2 4FB1 AFCF C6EB 0764 7825 2711 4CB9 C500 D81E 020A 0008 16F0
FF64 2711 4CB9 1109 4D69 6E69 4265 6163 6F6E 5F30 3134 3332 01D8 C100
B100 000D 0002 1E02 0106 1AFF 4C00 0215 E2C5 6DB5 DFFB 48D2 B060 D0F5
A710 96E0 0000 0000 C500 D61E 020A 0008 16F0 FF64 0000 0000 1109 4D69
6E69 4265 6163 6F6E 5F30 3030 3133 01D5 C100 B100 0017 0003 1E02 0106
1AFF 4C00 0215 E2C5 6DB5 DFFB 48D2 B060 D0F5 A710 96E0 0000 0000 C500
CA1E 0201 061A FF4C 0002 15E2 C56D B5DF FB48 D2B0 60D0 F5A7 1096 E000
0000 00C5 00D2 1E02 0A00 0816 F0FF 6400 0000 0011 094D 696E 6942 6561
636F 6E5F 3030 3032 3301 D2C1 00B1 0000 2600 011E 020A 0008 16F0 FF64
0000 0000 1109 4D69 6E69 4265 6163 6F6E 5F30 3030 3338 01D1 DD

```

The frame header format is as follows:

	Frame header			
	Protocol Header	Data Length	Gateway MAC Address	Number of BLE
Data value	0xBB01	0x0000 0329	0xAC23 3FC0 000D	0x000F
Desc.	Shows that it is a Binary Long Format data	4 bytes The length of the entire data frame is 809 bytes	6 bytes GW Mac: AC:23:3F:C0:00:0D	2 bytes The number of BLE devices is 15.

The BLE Frame is as follows:

BLE NO.	Body					
	BLE MAC Address	Number of BLE Data	BLE Data			
			Raw Data Length	Raw Data	Response	rsi
1	1234 5678 A600	0001	1E	02 0106 1AFF 4C00 0215 0112 2334 4556 7889 900A ABBC CDDE EFF0 03E8 07D0 C5	00	C6
2	18 6590 DF4F B2	0001	0B	0201 0607 FF4C 0010 020B 00	00	BE
3	20 1700 0113 01	0001	14	0201 0603 03E1 FF0C 16E1 FFA1 0E00 0113 0100 1720	00	B7
4	2017 1218 1103	0001	1E	02 0106 1AFF 4C00 0215 0112 2334 4556 6778 8990 0AAB BCCD DEEF 03E8 05D2 C5	00	CF
5	23 CE63 0D41 DE	0001	1F	1EFF 0600 0109 2000 F184 2846 888D A6EC 08E3 1892 101A BA3A F36B 45FB 6DD1 12	00	C1
6	66 6666 6669 69	0001	1D	0201 0603 03AA FE15 16AA FE00 E800 0000 0000 0000 FF00 0000 0000 90	00	C9
7	99 8877 6612 34	0004	1E	0201 061A FF4C 0002 15AB 8106 93A4 E24F B1AF CFC6 EB07 6478 2500 0100 01C5	00	D1
			1E	02 0A00 0816 F0FF 6400 0100 0111 094D 696E 6962 6561 636F 6E5F 3730 3330 32	01	D1
			1E	0201 061A FF4C 0002 15AB 8106 93A4 E24F B1AF CFC6 EB07 6478 2500 0100 01C5	00	D0
			1E	02 0A00 0816 F0FF 6400 0100 0111 094D 696E 6962 6561 636F 6E5F 3730 3330 32	01	D0
9	AB CDEF 2300 15	0002	14	0201 0603 03E1 FF0C FFE1 FF52 4201 0001 1200 6401	00	BC
			00	NULL	01	BC
10	AB CDEF 2300 2B	0001	14	0201 0603 03E1 FF0C FFE1 FF52 4201 0001 1100 6401	00	BC

11	AC23 3F23 C311	0001	19	02 0106 0303 AAFE 1116 AAFE 2000 0BF4 1200 0065 F1D0 011C 4E76	00	B7
12	C0CC 90F7 465D	0002	1E	02 0106 1AFF 4C00 0215 FDA5 0693 A4E2 4FB1 AFCF C6EB 0764 7825 2711 4CB9 C5	00	D8
			1E	020A 0008 16F0 FF64 2711 4CB9 1109 4D69 6E69 4265 6163 6F6E 5F30 3134 3332	01	D8
13	C100 B100 000D	0002	1E	02 0106 1AFF 4C00 0215 E2C5 6DB5 DFFB 48D2 B060 D0F5 A710 96E0 0000 0000 C5	00	D6
			1E	020A 0008 16F0 FF64 0000 0000 1109 4D69 6E69 4265 6163 6F6E 5F30 3030 3133	01	D5
14	C100 B100 0017	0003	1E	02 0106 1AFF 4C00 0215 E2C5 6DB5 DFFB 48D2 B060 D0F5 A710 96E0 0000 0000 C5	00	CA
			1E	0201 061A FF4C 0002 15E2 C56D B5DF FB48 D2B0 60D0 F5A7 1096 E000 0000 00C5	00	D2
			1E	02 0A00 0816 F0FF 6400 0000 0011 094D 696E 6942 6561 636F 6E5F 3030 3032 33	01	D2
15	C1 00B1 0000 26	0001	1E	020A 0008 16F0 FF64 0000 0000 1109 4D69 6E69 4265 6163 6F6E 5F30 3030 3338	01	D1

The Frame tail is 0xDD.

Binary Short Data Format Sample:

```
BB00 0000 00C5 AC23 3FC0 000D 0013 ABCD EF06 0006 0002 A4A4
C0D4 AD37 09CA 0001 A5C1 00B1 0000 0600 03AE AFAF C100 B100
000D 0001 ABC1 00B1 0000 1700 01AD C100 B100 0025 0001 A5C1
00B1 0000 5A00 01AE C1EB 07D2 E9AA 0002 ABAB C989 CE35 C7B6
0002 A5A8 CE56 CAFB 285B 0001 C0D3 7C4F 0B2B 5200 02AB ACE0
D2B3 7E96 9C00 01A2 E54D 425F 7DB6 0001 A6E6 F5D3 6082 A200
02B4 B5EA D25C 1986 FC00 02BF C0F0 A4D2 9EBF 1B00 03C2 C3C2
F56A C925 CF6C 0001 A5FD 1DD7 AB03 F500 01AD FDFC D4FB 5D98
0002 A9AA DD
```

The frame header format is as follows:

	Frame Header			
	Protocol Header	Data Length	Gateway MAC Address	Number of BLE
Data	0xBB00	0x0000 00C5	0xAC23 3FC0 000D	0x0013
Desc.	Shows that it is a Binary Short Format	4 bytes The length of the entire data frame is 197 bytes	6 bytes GW's Mac:AC:23:3F:C0:00:0D	2 bytes The number of BLE devices is 19.

The BLE Frame is as follows:

BLE NO.	Body		
	BLE MAC Address	Number of BLE Data	BLE Data
			rss
1	ABCD EF06 0006	0002	A4
			A4
2	C0D4 AD37 09CA	0001	A5
3	C100 B100 0006	0003	AE
			AF
			AF
4	C100 B100 000D	0001	AB
5	C100 B100 0017	0001	AD
6	C100 B100 0025	0001	A5
7	C100 B100 005A	0001	AE
8	C1EB 07D2 E9AA	0002	AB
			AB
9	C989 CE35 C7B6	0002	A5
			A8
10	CE56 CAFB 285B	0001	C0
11	D37C 4F0B 2B52	0002	AB
			AC
12	E0D2 B37E 969C	0001	A2
13	E54D 425F 7DB6	0001	A6
14	E6 F5D3 6082 A2	0002	B4
			B5
15	EA D25C 1986 FC	0002	BF
			C0
16	F0 A4D2 9EBF 1B	0003	C2
			C3
			C2
17	F56A C925 CF6C	0001	A5
18	FD 1DD7 AB03 F5	0001	AD
19	FDFC D4FB 5D98	0002	A9
			AA

The Frame tail is 0xDD.

2. Service Access

The G1 gateway currently supports the use of MQTT or HTTP network protocols to communicate with cloud servers.

Users can choose the appropriate network protocols according to their own server architecture, and recommend the use of MQTT protocols.

The service access of the G1 gateway needs MQTT or HTTP service to support at the back end. When the G1 gateway is out of the factory, the default access is the BeaconYun IoT service in the cloud.

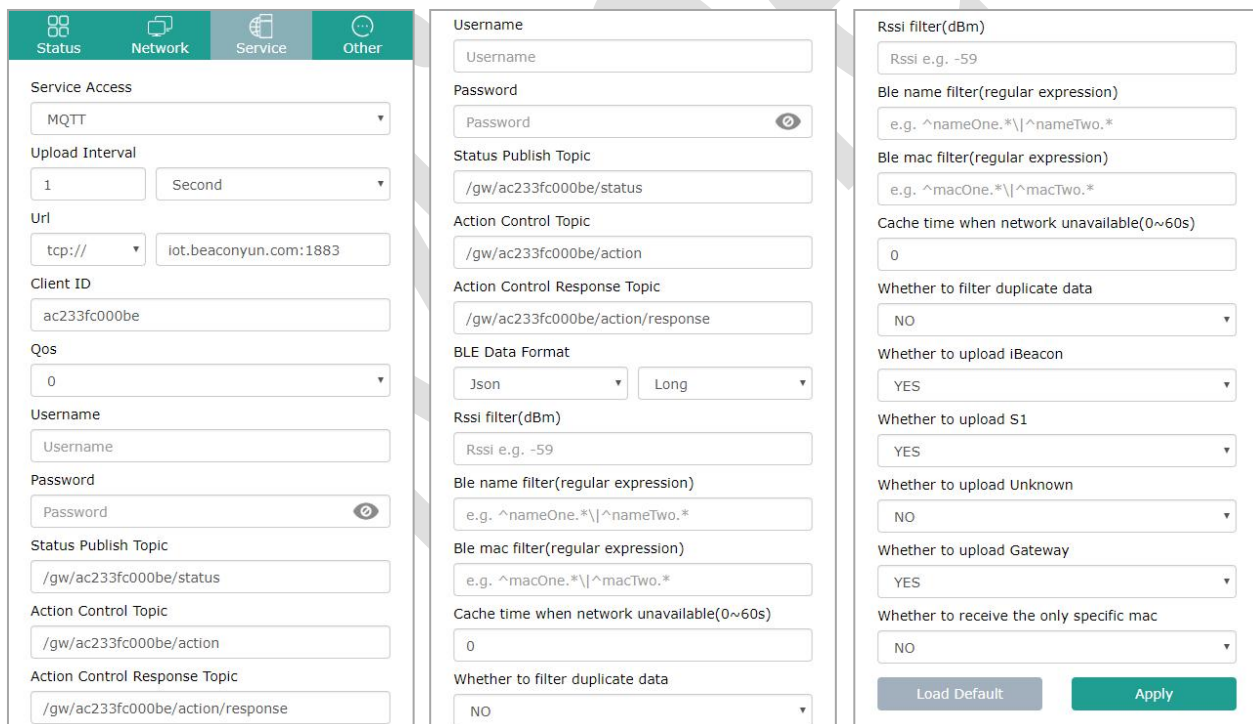
The G1 gateway acquiescence with the BeaconYun IoT service and console in the cloud as a demonstration, and you can quickly experience the application of the G1 gateway.

For the detailed BeaconYun IoT service console, see the [Thingoo G1 Dashboard User Guide document](#).

2.1 MQTT Access

When using MQTT access, the G1 gateway supports the timing of uploading BLE data and remote command control functions.

The G1 gateway can support the MQTT 3.1.1 protocol standard server. The specific protocol refer to the MQTT 3.1.1 protocol document in <http://mqtt.org>. The MQTT related libraries refer to <https://github.com/mqtt/mqtt.github.io/wiki/libraries>



Field	Name	Remark
-------	------	--------

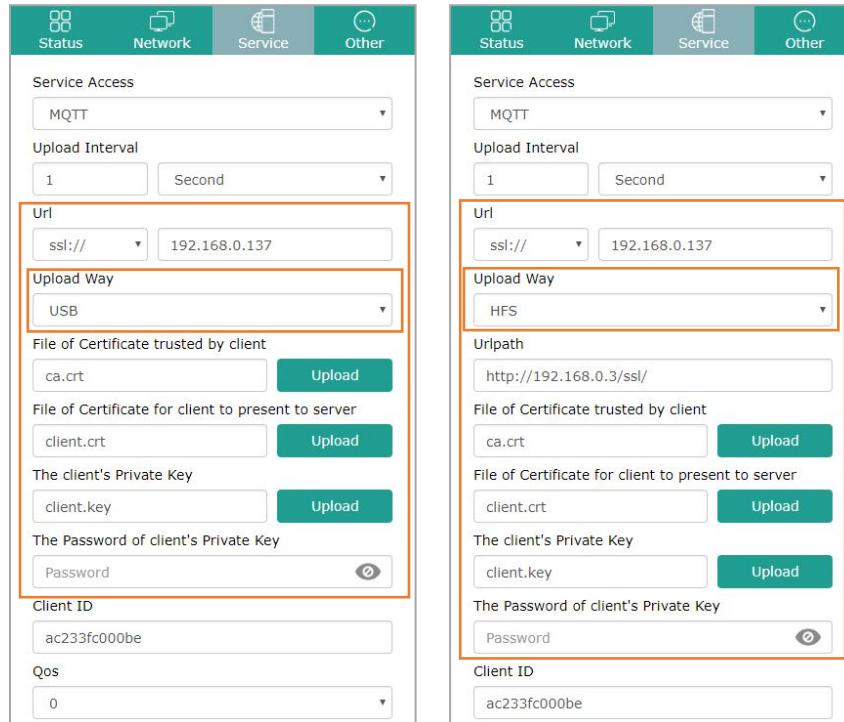
Service Access	Service Access selection	Choose MQTT protocol.
Upload Interval	Timing upload interval	Optional unit, default value is 1 second.
Url	Url of MQTT broker	<p>Format: scheme+domain name/IP Address+port scheme:Optional value is tcp://, ssl:// port: When port is not filled in,default value is 1883 in tcp:// scheme, 8883 in ssl:// scheme. Default value is tcp://iot.beaconyun.com:1883 Note: You can use the MQTT.fx desktop software to subscribe the gateway's Status Publish Topic to fetch gateway's data.</p>
Client ID	MQTT Client ID	<p>The Client ID defined by the MQTT protocol as the identity of the client. Default value: \${gatewayMAC} \${gatewayMAC}for the gateway's Mac in the hexadecimal lowercase character form, such as aabbccddeeff</p>
Qos	MQTT QoS level	<p>The Qos level defined by the MQTT protocol. QoS=0: at most times, it is possible to repeat or lose. QoS=1: at least once, it's possible to repeat it. Default value: 0</p>
Keep alive interval	Keep alive interval	<p>The keep alive interval defined by the MQTT protocol. 10 seconds by default. Some MQTT IoT Hub, such as Aliyun, do not allow such a frequent heartbeat interval, and need to set this parameter to 60 seconds to connect.</p>
Username	MQTT User Name	The Username defined by the MQTT protocol
Password	MQTT Password	The password defined by the MQTT protocol
Status Publish Topic	Topic to publish the BLE data	<p>Topic used by gateway to publish BLE data Default value: /gw/\${gatewayMac}/status \${gatewayMAC} for the gateway's Mac in the hexadecimal lowercase character form, such as /gw/aabbccddeeff/status.</p>
Action Control Topic	Topic to receive action control data	<p>For the gateway receiving control command, the message is JSON format. Default value: /gw/\${gatewayMac}/action \${gatewayMAC} for the gateway's Mac in the hexadecimal lowercase character form, such as /gw/aabbccddeeff/action.</p>
Action Control Response Topic	Topic to response action control result data	<p>For the gateway response action control command result,message is JSON format. Default value: /gw/\${gatewayMac}/action/response \${gatewayMAC} for the gateway's Mac in the hexadecimal lowercase character form, such as /gw/aabbccddeeff/action/response.</p>
BLE Data Format	BLE data format	Json array format or Binary format, each format is divided

	selection	<p>into long and short data formats.</p> <p>Note: When the bandwidth of a gateway's deployed network environment is small, it is recommended to use Binary data format.</p>
Rssi filter	Filtering by rssi value	<p>Only upload the rssi data that the signal intensity is greater than that, unit is dBm.</p> <p>When not filled, it is not filtered, and the value of -100 < RSSI < 0 can be filled.</p> <p>For example, when -67 is filled in, the -68 and the below weak signals are not uploaded to the server.</p>
Ble name filter	filtering BLE name via regular expression	<p>Ble name's regular expression</p> <p>Example:</p> <ol style="list-style-type: none"> 1. Just want to receive the BLE name which start with MiniBeacon,the regular expression can be <code>^MiniBeacon.*</code> 2. Just want to receive the BLE name which start with MiniBeacon or start with HelloBeacon,the regular expression can be <code>^MiniBeacon.*\ ^HelloBeacon.*</code> <p>Note: If the BLE type is <i>S1</i> or <i>iBeacon</i>,this filter option works well,beacause GW have already merged the advertising packet and the scan response packet.But if the BLE is <i>Unknown</i>, please use this filter option carefully,beacause if the <u>Complete Local Name</u>(parsed as the <i>bleName</i>) only exist in the scan response packet, you will miss the all advertising packet.</p>
Ble mac filter	filtering BLE mac via regular expression	<p>The regular expression of BLE mac address you wanna recevice.</p> <p>Example:</p> <ol style="list-style-type: none"> 1. Just want to receive the BLE mac which start with AC233FC,the regular expression can be <code>^AC233FC.*</code> 2. Just want to receive the BLE name which start with AC233FC or start with D678, the regular expression can be <code>^AC233FC.*\ ^D678.*</code> <p>Note: Only the uppercase MAC address without any delimiters can be recognized by gateway.</p>
Raw data filter	filtering raw data through regular expression	<p>The regular expression of Raw Data you wanna recevice.</p> <ol style="list-style-type: none"> 1.This filtering condition applies to all data types, including Json/Binary-Long/Short. 2.For example, just only want to receive the iBeacon adv packets with UUID FDA50693A4E24FB1AFCFC6EB07647825 <p>The expression of iBeacon broadcast package can be written as <code>^.*FDA50693A4E24FB1AFCFC6EB07647825.*</code></p> <p>Note: Only uppercase Raw Data strings can be recognized by gateway.</p>
Cache Time	Time of gateway caching	Available values: 0 to 60 seconds, positive integers

When Network unavailable	BLE data after Network unavailable	Default value is 0 Note: It is the RAM where gateway caching data. When firmware version is greater than or equal to v2.0.0, this parameter no longer works.
Whether to filter duplicate data	Duplicate data within the time interval be filtered to upload or not	Default value is NO,Optional Value:YES,NO Note: Gateway determines duplicate data based on BLE MAC address. If BLE broadcasts multiple packets, such as Minew's MiniBeaconPlus firmware, Please use this filter option carefully.
Whether to upload iBeacon	Whether to upload data of the ibeacon type	Default value is YES,Optional value:YES,NO
Whether to upload S1	Whether to upload data of the S1 type	Default value is YES,Optional value:YES,NO
Whether to upload Gateway	Whether to upload data of the Gateway type	Default value is YES,Optional value:YES,NO
Whether to upload Unknown	Whether to upload data of the Unknown type	Default value is NO,Optional value:YES,NO Note: if your BLE is not S1 or iBeacon , please enable this option to upload your own BLE tag.
Whether to receive the only specific mac	Whether only the data of a specific Mac address is accepted	Default value is NO,Optional value:YES,NO If you select YES option,macList parameter must be submitted at least one mac address. Note: Not recommended to use this filter option. Recommended to use Ble Mac filter instead.
macList	A list of Mac addresses that only be uploaded	Input in the form of AABBCDDDEEFF. After entering an MAC address, check the validity of MAC address according to the space or carriage return. MAC address will only be submitted to the macList parameter if the validity is passed.

MQTT with SSL/TLS protocol

When using MQTT access, it supports SSL/TLS protocol. After selecting "ssl://" in the drop-down list of Url parameters, the parameters displayed are used to configure loading certificates to gateway.



The fields for configuration of the SSL/TLS certificate is as follows:

Field	Name	Remark
Upload Way	The way to upload a certificate	Optional value:USB or HFS(Http File Server) USB: Uploading certificates using USB HFS: Uploading certificates using HFS
Urlpath	URL PATH of HFS	The Url path is for the gateway to download certificate files. Default value:http://192.168.0.3/ssl/
Username of basic authentication in HFS	/	If HFS does not have basic authentication, it is not necessary to fill in this parameter.
Password of basic authentication in	/	If HFS does not have basic authentication, it is not necessary to fill in this parameter.

HFS		
File of Certificate trusted by client	File of Certificate trusted by Gateway	The file name of the CA Certificate in the USB-Stick /HFS. USB: the gateway will download the certificate files directly from the root directory of USB-Stick. HFS: the gateway will download the certificate files directly from the \${Urlpath} path.
File of Certificate for client to present to server	File of Certificate for gateway to present to server if server needs it	The file name of the Certificate in the USB-Stick /HFS. USB: the gateway will download the certificate files directly from the root directory of USB-Stick. HFS: the gateway will download the certificate file directly from the \${Urlpath} path.
The client's Private Key	Private key file of the gateway's certificate	The file name of the private key file in the USB-Stick /HFS. USB: the gateway will download the private key file directly from the root directory of USB-Stick. HFS: the gateway will download the private key file directly from the \${Urlpath} path.
The Password of client's Private Key	The Password of client's Private Key	If the private key of the gateway certificate is encrypted, the password must be provided.

USB Upload Certificate Tutorial

The method of uploading certificate files in USB mode is as follows:

- (1) select **USB** in the drop-down list of "**Upload Way**" in the web page configuration page;
- (2) name the certificate that needs to be uploaded into the file name of the corresponding input box;
- (3) copy the named certificate files to the root directory of the USB-Stick.
- (4) insert the USB-Stick into the gateway's USB port;
- (5) the corresponding certificate files can be uploaded by clicking the corresponding Upload key of the configuration page.

HFS Upload Certificate Tutorial

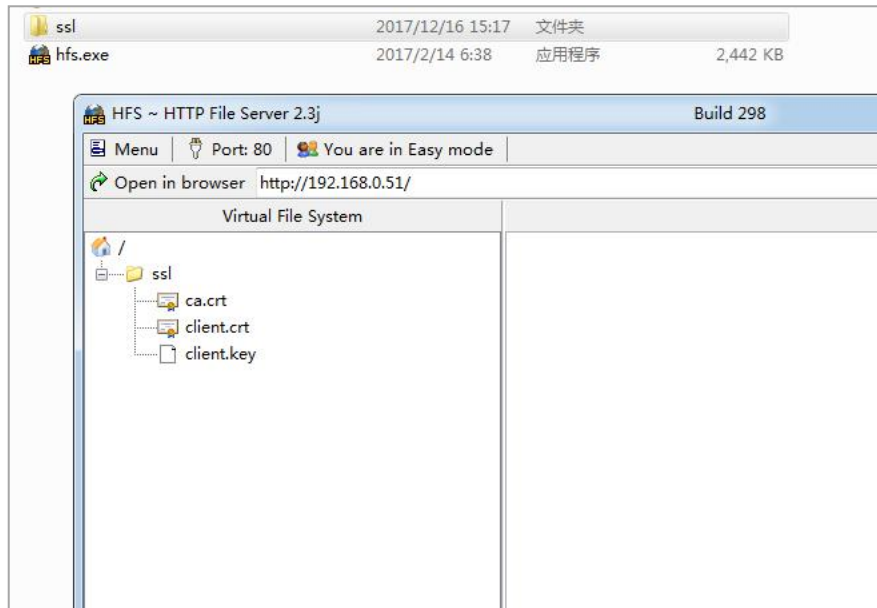
The method of uploading certificate files in HFS mode is as follows:

Preparation: HFS software link: <https://sourceforge.net/projects/hfs/>

Click open hfs.exe software to open the service of HFS, and the gateway's network need to be configured to communicate with HFS.

- (1) select **HFS** in the drop-down list of "**Upload Way**" in the web page configuration page;
- (2) name the certificate that needs to be uploaded into the file name of the corresponding input box;

(3) add the named certificate to the path of the HFS, as shown in the following figure;



(4) fill in the path of the certificate in the **Urlpath** input box, for example: `http://192.168.0.51/ssl`;

(5) the corresponding certificate files can be uploaded by clicking the corresponding Upload key of the configuration page.

2.2 HTTP Access

When HTTP access is setted, the G1 gateway only supports the timing of uploading collected data functions.

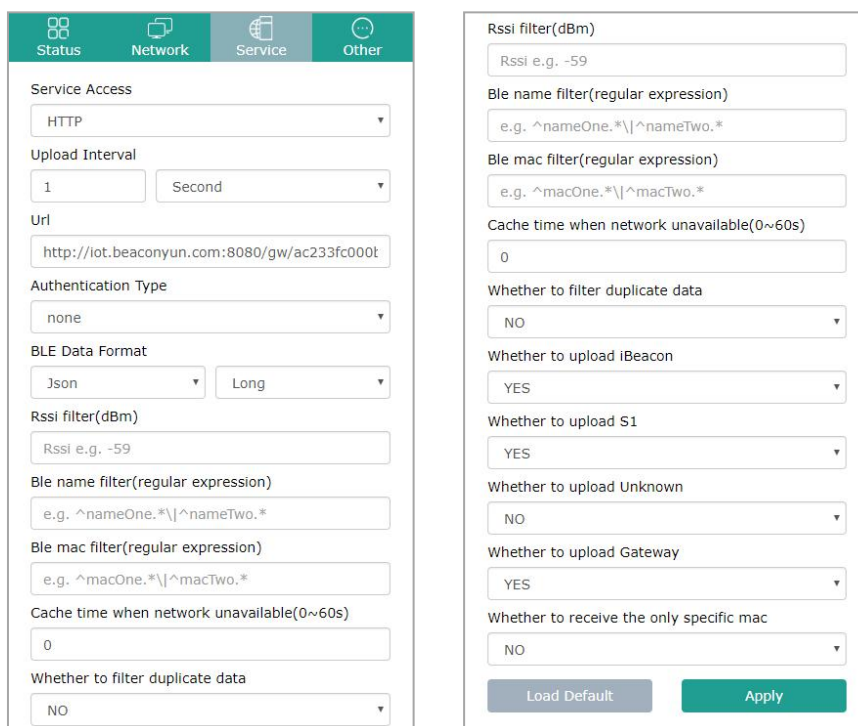
In the service connection settings section of the gateway WEB configuration interface, select HTTP and fill in the URL information of the HTTP server, which will take effect after apply.

HTTP server development note:

1.The HTTP Server must can receive the HTTP POST Method's request.

2.The HTTP server should receive any data packets uploaded by gateway, including empty packets(It is the PING REQUEST packet for gateway to test the HTTP Server online or not). Only HTTP servers return

HTTP status code 200 can gateway continue to send the next packet.



Field	Name	Remark
Service Access	Service Access selection	Choose HTTP protocol
Upload Interval	Timing upload interval	Optional unit, default value is 1 second.
Url	Url of HTTP server	<p>The value of Url begins with http:// or https://.</p> <p>G1 gateway uses the HTTP's POST method to upload the data to the HTTP server.</p> <p><u>http://iot.beaconyun.com:8080/gw/\${gatewayMac}/status</u></p> <p><u>http://iot.beaconyun.com:8080/gw/aabbccddeeff/status</u>.</p> <p>Note: You can't fetch data from gateway via type this url in browser. Because our default url does not offer HTTP's GET method. You can try another URL: <u>http://iot.beaconyun.com</u> to access the our demonstration website.</p>
Authentication Type	HTTP authentication type	Optional value: none,basic Default value: none
Username	Authentication user	Http basic authentication's user name

	name	
Password	Authentication Password	Http basic authentication's password
BLE Data Format	BLE data format selection	<p>Json array format or Binary format, each format is divided into long and short data formats.</p> <p>Note: When the bandwidth of a gateway's deployed network environment is small, it is recommended to use Binary data format.</p>
Rssi filter	Filtering by rssi value	<p>Only upload the rssi data that the signal intensity is greater than that, unit is dBm.</p> <p>When not filled, it is not filtered, and the value of $-100 < RSSI < 0$ can be filled.</p> <p>For example, when -67 is filled in, the -68 and the below weak signals are not uploaded to the server.</p>
Ble name filter	Through regular expression filtering	<p>Ble name's regular expression</p> <p>Example:</p> <ol style="list-style-type: none"> 1. Just want to receive the BLE name which start with MiniBeacon, the regular expression can be <code>^MiniBeacon.*</code> 2. Just want to receive the BLE name which start with MiniBeacon or start with HelloBeacon, the regular expression can be <code>^MiniBeacon.* ^HelloBeacon.*</code> <p>Note: If the BLE type is <i>S1</i> or <i>iBeacon</i>, this filter option works well, because GW have already merged the advertising packet and the scan response packet. But if the BLE is <i>Unknown</i>, please use this filter option carefully, because if the <u>Complete Local Name</u> (parsed as the <i>bleName</i>) only exist in the scan response packet, you will miss the all advertising packet.</p>
Ble mac filter	filtering BLE mac via regular expression	<p>The regular expression of BLE mac address you wanna receive.</p> <p>Example:</p> <ol style="list-style-type: none"> 1. Just want to receive the BLE mac which start with AC233FC, the regular expression can be <code>^AC233FC.*</code> 2. Just want to receive the BLE name which start with AC233FC or start with D678, the regular expression can be <code>^AC233FC.* ^D678.*</code> <p>Note: Only the uppercase MAC address without any delimiters can be recognized by Gateway.</p>
Raw data filter	filtering raw data through regular expression	<p>The regular expression of Raw Data you wanna receive.</p> <ol style="list-style-type: none"> 1. This filtering condition applies to all data types, including Json/Binary-Long/Short. 2. For example, just only want to receive the iBeacon adv packets with UUID FDA50693A4E24FB1AFCFC6EB07647825 <p>The expression of iBeacon broadcast package can be</p>

		written as <code>^.*FDA50693A4E24FB1AFCFC6EB07647825.*</code> Note: Only uppercase Raw Data strings can be recognized by gateway.
Cache Time When Network unavailable	Time of gateway caching BLE data after Network unavailable	Available values: 0 to 60 seconds, positive integers Default value is 0 Note: It is the RAM where gateway caching data.
Whether to filter duplicate data	Duplicate data within the time interval be filtered to upload or not	Default value is NO,Optional Value:YES,NO Note: Gateway determines duplicate data based on BLE MAC address. If BLE broadcasts multiple type packets, such as Minew's MiniBeaconPlus firmware, Please use this filter option carefully.
Whether to upload iBeacon	Whether to upload data of the iBeacon type	Default value is YES,Optional value:YES,NO
Whether to upload S1	Whether to upload data of the S1 type	Default value is YES,Optional value:YES,NO
Whether to upload Gateway	Whether to upload data of the Gateway type	Default value is YES,Optional value:YES,NO
Whether to upload Unknown	Whether to upload data of the Unknown type	Default value is NO,Optional value:YES,NO Note: if your BLE is not S1 or iBeacon, please enable this option to upload your own BLE tag.
Whether to receive the only specific mac	Whether only the data of a specific Mac address is accepted	Default value is NO,Optional value:YES,NO If you select YES option,macList parameter must be submitted at least one mac address. Note: Not recommended to use this filter option. Recommended to use Ble Mac filter instead.
macList	A list of Mac addresses that only be uploaded	Input in the form of AABBCCDDEEFF. After entering an MAC address, check the validity of MAC address according to the space or carriage return. MAC address will only be submitted to the macList parameter if the validity is passed.

3. Remote Control (only MQTT)

G1 gateway can receive control commands through MQTT Access, so as to realize remote control of server to gateway. After receiving instructions, G1 gateway will response action control result message through another Topic, and data format is JSON format. The HTTP Access does not support this function. In MQTT, the gateway will subscribe to the Topic specified by the **Action Control Topic**: `/gw/${gatewayMac}/action` is the default value, receive control instructions, and send a control instruction response message with **Action Control Response Topic**: `/gw/${gatewayMac}/action/response` is the default value.

Different MQTT lot hub providers, Topic is often different, and need to refer to the document of the service provider. For the self - developed MQTT server, we can refer to our design. Taking BeaconYun IoT services as an example, the definition of the topic is as follows:

Topic	Action	Description
/gw/\${gatewayMac}/action	Subscribe	The topic for gateway receives the action control instructions.
/gw/\${gatewayMac}/action/response	Publish	The topic for gateway response action control result.

Where \${gatewayMac} is the hexadecimal lowercase character of the gateway's Mac. That is to say, the server sends commands json format to /gw/\${gatewayMac}/action topic. After receiving instructions, the gateway will return the result to /gw/\${gatewayMac}/action/response topic.

The request response message sampe is as follows:

```
{
  "code": 200,
  "message": "success",
  "requestId": "b2c3d4e5-3424-4dca-32dc-12b73290cfed"
}
```

The response message data fields is described in detail as follows:

Detail Field	Type	Code	Message	Description
Sucess		200	Success	Request operation success.
Error json format		300	Error json format	The received instruction is the wrong Json format.
Unrecognized action		400	Unrecognized action	Undefined action.

Parameter error	500	Parameter error	The parameters of the request are incomplete and exist in Parameter Configuration , Upgrade Firmware .
Network error	600	Network error	Network error, fail to download SSL certificate files, fetch upgrading firmware or get available version list, exist in Parameter Configuration , Get Version and Upgrade Firmware .

3.1 Reboot Gateway

This command is used to reboot the gateway remotely.
The request message sample is as follows:

```
{
  "action": "reboot",
  "requestId": "b2c3d4e5-3424-4dca-32dc-12b73290cfed"
}
```

The message data field is described in detail as follows.

Field	Name	Required	Description
action	action	Yes	reboot
requestId	Request id	No	UUID format or the other format you defined, when the request message contain a requestId, the response message will returns to this field too, which is used to match the request and response messages.

3.2 Heartbeat Package

This command is used to test whether the gateway is online or not remotely.
The request message sample is as follows:

```
{  
  "action": "heartBeat",  
  "requestId": "b2c3d4e5-3424-4dca-32dc-12b73290cfed"  
}
```

The message data field is described in detail as follows:

Field	Name	Required	Value
action	action	Yes	heartBeat
Request ID	Request ID	No	UUID format or the other format you defined, when the request message contain a request Id, the response message will returns to this field too, which is used to match the request and response messages.

The server sends a heartbeat package to the gateway to determine whether the gateway is online, and if the gateway receives the control instruction, it returns the successful message directly. If the gateway is not online, it does not return.

3.3 Parameter Configuration

The request message sample is as follows:

```
{
  "action": "config",
  "takeEffectImmediately": "YES",
  "common": {
    "proto": "default",
    "uploadInterval": "default",
    "isFilterDupData": "default",
    "isOnlySpecialMac": "default",
    "macList": "default",
    "disableLED": "default",
    "isLongBright": "default",
    "isUploadUnkown": "default",
    "isUploadBeacon": "default",
    "isUploadS1": "default",
    "isUploadGateway": "default",
    "isauto": "default",
    "timeout": "default",
    "schedule": {
      "istiming": "default",
      "min": "default",
      "hour": "default",
      "week": "default"
    },
    "rssi": "default",
    "regex": "default",
    "macReg": "default",
    "cacheTime": "default",
    "isLongFormat": "default",
    "isJsonFormat": "default"
  },
  "mqtt": {
    "mUrl": "default",
    "urlpath": "default",
    "cafile": "default",
    "certfile": "default",
    "keyfile": "default",
    "keypass": "default",
    "publishTopic": "default",
    "subscribeTopic": "default",
    "responseTopic": "default",
    "qos": "default",
    "userName": "default",
    "passWord": "default",
    "clientId": "default"
  },
  "http": {
    "hUrl": "default",
    "auth": "default",
    "httpUser": "default",
    "httpPass": "default"
  },
  "requestId": "b2c3d4e5-3424-4dca-32dc-12b73290cfed"
}
```

All fields are string types, and as shown above, when the field value is default, it will be restored to the default value. When the field is doesn't contain, the original parameters of the field will not be modified.

The message data field is described in detail as follows:

Field	Name	required	Description
action	action	YES	config
takeEffectImmediately	Whether to take effect immediately	NO	Optional value:YES,NO When the field is not included,the default value is NO. You need to reboot the gateway manually to take effect.
proto	Service access	NO	Optional value:MQTT,HTTP. Default value is MQTT.
uploadInterval	Upload Interval	NO	The unit of time is milliseconds. Default value is 1000.
isFilterDupData	Duplicate data within the time interval be filtered to upload or not	NO	Optional value:YES,NO. Default value is NO.
isOnlySpecialMac	Whether only the data of a specific Mac address is accepted	NO	Default value is NO, Optional value:YES,NO If you select YES option,macList parameter must be submitted at least one mac address.
macList	A list of Mac addresses that only be uploaded	NO	Input in the form of AABBCCDDEEFF, separated by a comma. example:AABBCCDDEEFF,BBCCDDEEFFAA,CCDDEEFFAABB Default value is NULL
disableLED	Whether to disable leds permanently	NO	Optional value:Yes,NO
isLongBright	Whether to enable the top LED strip long brightly	NO	Optional value: YES,NO Default value is NO.
isUploadUnkown	Whether to upload data of the Unknown type	NO	Optional value :YES,NO Default value is NO
isUploadIBeacon	Whether to upload data of the ibeacon type	NO	Optional value : YES,NO Default value is YES
isUploadS1	Whether to upload data of the S1 type	NO	Optional value : YES, NO Default value is YES

isUploadGateway	Whether to upload data of the Gateway type	NO	Optional value : YES, NO Default value is YES
isauto	Whether to enable Automatic Reboot function	NO	Optional value: YES,NO Default value is YES
timeout	watch dog timeout	NO	Optional value: 5~65s Default value is 30 seconds
istiming	Whether to enable the Timing Reboot function	NO	Optional value: YES,NO Default value is NO
min	minutes	NO	Optional value: 00~59 Default value is 00
hour	hour	NO	Optional value: 00~23 Default value is 00
week	week	NO	Optional value: 0,1,2,3,4,5,6 Default value is 0,1,2,3,4,5,6 Where 0 means Sunday,and so on. Separated by commas. For example, if you wan the gateway to be rebooted on Sunday and Wednesday, the week field value can be "0,3".
rss	Filtering by rss value	NO	Only upload the rss data that the signal intensity is greater than that, unit is dBm. When not filled, it is not filtered, and the value of $-100 < RSSI < 0$ can be filled.
regex	filtering ble name through regular expression	NO	Ble name's regular expression Example: 1. Just want to receive the BLE name which start with MiniBeacon,the regular expression can be <i>^MiniBeacon.*</i> 2. Just want to receive the BLE name which start with MiniBeacon or start with HelloBeacon,the regular expression can be <i>^MiniBeacon.* ^HelloBeacon.*</i>
macReg	filtering ble mac through regular expression	NO	The regular expression of BLE mac address you wanna receive. Example: 1.Just want to receive the BLE mac which start

			<p>with AC233FC,the regular expression can be <code>^AC233FC.*</code></p> <p>2.Just want to receive the BLE name which start with AC233FC or start with D678, the regular expression can be <code>^AC233FC.*\ ^D678.*</code></p> <p>Note: Only the uppercase MAC address without any delimiters can be recognized by Gateway.</p>
cacheTime	Time of gateway caching BLE data after Network unavailable	NO	<p>Available values: 0 to 60 seconds, positive integers</p> <p>Default value is 0</p>
isLongFormat	Long or Short BLE data type	NO	<p>Optional Value:Long,Short</p> <p>Default value is Long</p>
isJsonFormat	Json or Binary BLE data type	NO	<p>Optional Value:Json,Binary</p> <p>Default value is Json</p>
mUrl	Url of mqtt broker	NO	<p>Format is : tcp://iot.beaconyun.com:1883</p> <p>ssl://192.168.0.3:8883</p> <p>Default value is</p> <p>tcp://iot.beaconyun.com:1883</p>
urlpath	URL path of HFS	NO	<p>The Url path is for the gateway to download certificate files.</p> <p>Default value:http://192.168.0.3/ssl/</p>
cafile	File of Certificate trusted by Gateway	NO	<p>The file name of the CA Certificate in the HFS.</p> <p>The gateway will download the certificate files directly from the <code>\${urlpath}/\${cafile}</code> path.</p>
certfile	File of Certificate for gateway to present to server if server needs it	NO	<p>The file name of the Certificate in the HFS.</p> <p>The gateway will download the certificate files directly from the <code>\${urlpath}/\${certfile}</code> path.</p>
keyfile	Private key file of the gateway's certificate	NO	<p>The file name of the private key file in the HFS.</p> <p>The gateway will download the certificate files directly from the <code>\${urlpath}/\${keyfile}</code> path.</p>

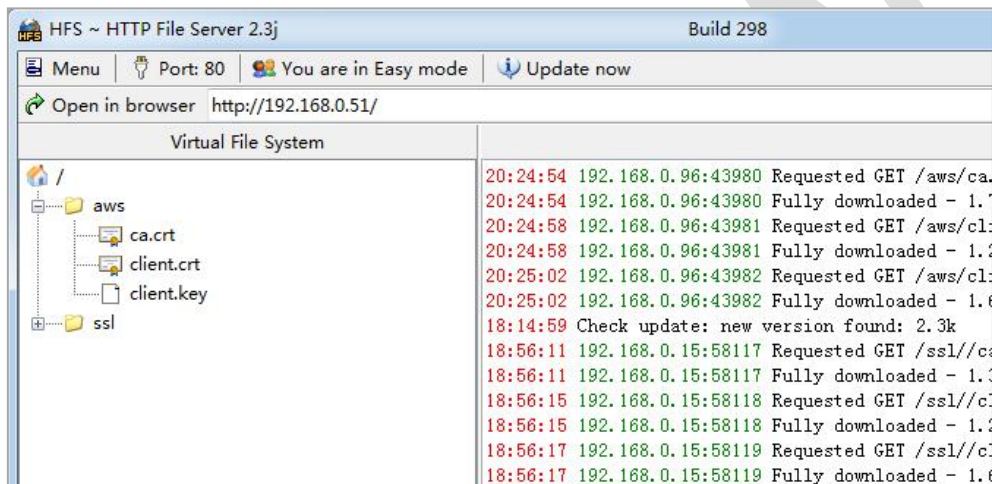
keypass	The Password of client's Private Key	NO	If the private key of the gateway certificate is encrypted, the password must be provided.
publishTopic	Topic to publish the BLE data	NO	Topic used by gateway to publish BLE data Default value: /gw/\${gatewayMac}/status
subscribeTopic	Topic to receive action control data	NO	For the gateway receiving control command, the message is JSON format. Default value: /gw/\${gatewayMac}/action
responseTopic	Topic to response action control result data	NO	For the gateway response action control command result, message is JSON format. Default value: /gw/\${gatewayMac}/action/response
qos	MQTT QoS level	NO	The QoS level defined by the MQTT protocol. QoS=0: at most times, it is possible to repeat or lose. QoS=1: at least once, it's possible to repeat it. Default value: 0
userName	MQTT User Name	NO	The Username defined by the MQTT protocol
passWord	MQTT Password	NO	The password defined by the MQTT protocol
clientId	MQTT client ID	NO	The Client ID defined by the MQTT protocol as the identity of the client. Default value: \${gatewayMAC}
hUrl	Url of HTTP server	NO	The value of Url begins with http:// or https://. G1 gateway uses the HTTP's POST method to upload the data to the HTTP server. http://iot.beaconyun.com:8080/gw/\${gatewayMac}/status
auth	HTTP authentication type	NO	Optional value:none,basic Default value is none
httpUser	username of HTTP basic authentication	NO	username of HTTP basic authentication
httpPass	Password of HTTP basic authentication	NO	Password of HTTP basic authentication
requestId	Request id	NO	UUID format or the other format you defined, when the request message contain a requestId, the response message will returns to this field too, which is used to match the request and response messages.

Configuration Example:

- If you want to enable the topic LED strip long brightly and take effect immediately after the setting is successful. Just send the following message to the topic: /gw/\${gatewayMac}/action.

```
{
  "action": "config",
  "takeEffectImmediately": "YES",
  "common": {
    "isLongBright": "YES"
  }
}
```

- Configure the parameters of gateway to a specific mqtt broker, and if the mqtt broker needs to configure the certificate, the premise needs to build a HFS.



Take the configuration to Amazon AWS IoT Hub as an example, download the AWS certificate and put it in HFS, if you want to take effect immediately after set up successfully, just send the following message to topic: /gw/\${gatewayMac}/action.

```
{
  "action": "config",
  "takeEffectImmediately": "YES",
  "mqtt": {
    "mUrl": "ssl://a2c2geiksinfkf.iot.us-west-2.amazonaws.com",
    "urlpath": "http://192.168.0.51/aws",
    "cafile": "ca.crt",
    "certfile": "client.crt",
    "keyfile": "client.key"
  }
}
```

3.4 Get Version

This command is used to get the current version of the gateway and a list of version numbers available

for upgrading.

The message data field is described in detail as follows:

Field	Name	Required	Value
action	action	YES	getVersions
requestId	request id	NO	UUID format or the other format you defined, when the request message contain a requestId, the response message will returns to this field too, which is used to match the request and response messages.

The request message sample is as follows:

```
{
  "action": "getVersions",
  "requestId": "b2c3d4e5-3424-4dca-32dc-12b73290cfed"
}
```

If everything is ok, will response the current version and the available version list.

```
{
  "code" : 200,
  "message" : "success",
  "currentVersion" : "v1.4.0",
  "versions" : [ "v1.3.0", "v1.3.1", "v1.3.2", "v1.3.3", "v1.4.1" ],
  "requestId": "b2c3d4e5-3424-4dca-32dc-12b73290cfed"
}
```

3.5 Upgrade Firmware

The gateway supports remote sending commands to upgrade the gateway's firmware to the specified version.

There are two ways to upgrade: **public** type (the upgrade firmware downloaded from our company's HTTP server, which is equivalent to the OTA Upgrade on the Web configuration page), and **self** type (the upgrade firmware downloaded from the custom HTTP server, which is equivalent to the LAN Upgrade on the Web configuration page).

The message data field is described in detail as follows:

Field	Name	Required	Value
action	action	YES	upgrade
type	Upgrade type	NO	Optional value: public , self Default value is public

isSave	Whether to save the configuration	NO	Optional value: YES,NO Default value is NO
version	The version you wanna upgrade to	NO(Required if the type is public)	Upgrade to the specified version. The available version can be fetched via Get Version .
urlpath	URL path of HFS	NO(Required if the type is self)	the URL path of the HTTP server where the firmware is located at . The gateway will download the firmware from {urlpath}/{filename}.
filename	file name of upgrading firmware in HFS	NO(Required if the type is self)	The gateway will download the firmware from {urlpath}/{filename}.
requestId	request id	NO	UUID format or the other format you defined, when the request message contain a requestId, the response message will returns to this field too, which is used to match the request and response messages.

The **public** type upgrade command message sample is as follows:

```
{
  "action": "upgrade",
  "type": "public",
  "isSave": "YES",
  "version": "v1.4.0",
  "requestId": "xxx"
}
```

The **self** type upgrade command message sample is as follows:

```
{
  "action": "upgrade",
  "type": "self",
  "urlpath": "http://192.168.0.3/firmware/",
  "filename": "thingoo-upgrade.bin",
  "isSave": "YES",
  "requestId": "xxx"
}
```

4. U-disk Storage

The U-disk storage function is introduced since v2.0.0, which supports storing data on external U-disk when server is unavailable. And after the server is available, the gateway will first upload the historical data of the

U-disk to the server.

Without any settings, just inserting the U-disk will take effect directly. As long as the gateway can identify and mount the U-disk, the gateway will store the data in the U-disk when server is unavailable. When server is available, the gateway will upload the historical data of the U-disk in time sequence.

Note:

1. The file system format of U disk can use FAT32 or NTFS, but it recommended to use FAT32 format for better compatibility. Because the native Linux system does not support NTFS writing, it needs a third-party library to support NTFS writing. However, this third-party library is not stable and may damage the file system.
2. Although the gateway will not actively delete private files on the U-disk, it should not store private important files on the U-disk.
3. After inserting the U-disk, only after seeing the green flashing state light within 15 seconds can the U-disk be recognized by the gateway. If the **Disable leds permanently** parameter is NO, and no green flash can be seen in 15 seconds, indicating that the U-disk format is incompatible or that the U disk is not in good contact with the gateway USB port, please try another USB port.
4. When the gateway loses connection with the server, if you see the yellow flashing status light, it means that the gateway is writing or reading data to the U disk.

<END>