SolidSense Edge Gateway

Experience The Edge of The Internet of Things

SolidSense MQTT Gateway

MQTT interface definition

Version 2.0

Scope of the document

The document defines the MQTT interface for the SolidSense MQTT gateway. This software replaces the former BLE MQTT gateway including all the existing features from it.

The document does not include the Bluetooth standard and is assumed to be known by the reader.

The version 2.0 includes the possibility to also manage the GPS with the MQTT interface as well as the possibility to interface vehicle On Board Diagnostic service.

1. General structure of the MQTT interface

Identifiers

The BLE devices (tags/sensors/beacons/….) are uniquely identified by their MAC address in the form of 5 pairs of hexadecimal digits separated by colon.

Ex: 22:54:EF:6D:0C

BLE services and characteristics UUID are represented by their canonical hexadecimal strings.

By defaults the gateway ID is its hostname (serial number)

Dynamically typed values on Bluetooth

The gateway is translating basic types from and into their Bluetooth transport representation. Here is the enumeration defining the types on the interface

Keyword	Numeric value	Associated type
BTRAW	0	Hexadecimal string representing binary values. Little endian on 16 bits words assumed. Length shall always be even.
INT	1	Integer, can be signed
FLOAT	2	Float, can be signed
STRING	3	UTF 8 string assumed
UUID	4	Bluetooth UUID as Hexadecimal string following the standard
BYTES	5	Binary data in hexadecimal string. This is not strictly equivalent to BTRAW as byte ordering can change

High level topics summary and MQTT structure

Topic name	Publish	Subscribe	Purpose
solidsense	N	Y	Global status and control of the MQTT gateway
solidsense_resp	Y	N	
scan	N	Y	Control the gateway BLE scanner
filter	N	Y	Control the BLE scan filters
scan_result	Y	N	Send back BLE scanner status
advertisement	Y	N	Send advertisement from a BLE device
gatt	N	Y	Top level topic for all BLE GATT transactions from the broker
gatt_result	Y	N	Topic to send back BLE GATT results
modem	N	Y	Topic to send request to the cellular modem
modem_result	Y	N	Modem status
gps	N	Y	Topic to send request to the GNSS (GPS)
gps_result	Y	N	GPS publish
vehicle	*N*	*y*	*Topic to send request to vehicle (OBD) interface*
vehicle_result	*Y*	*n*	*Vehicle publish*

General syntax of topics

/<high level topic>/<gateway ID>/[<device address>/<sub-topic>]

Sub-topics are optional and linked to a measurement or status that can be published individually like a temperature for instance. Device address is not used for the Scan and Filter topics. Sub-topics are optional (see corresponding paragraph)

Gateways will subscribe only for their own ID

Payload encoding

Payload is JSON encoded and the content and structure are explained for each high-level topic.

Time stamps

Each message published by the gateway is time stamped. By default, Bluetooth related messages are timestamped with the epoch time is seconds (floating point) while all other messages are time stamped with the iso time format. The time stamping configuration will be introduced in a future release.

Solidsense topic

This topic is used to send global command to the gateway, in this version this limited, but will expand in future releases

Keyword		**Type**	**Value**	**Action/signification**
command	M	String	status	Query the status of active micro services in the MQTT gateway

Solidsense_resp topic

This topic is used by the gateway to send responses to ‘solidsense’ commands. One message is systematically sent once the gateway is connected to the broker. This is always the first message sent.

Keyword		**Type**	**Value**	**Action/signification**
status	M	Array of pair		Service/enable flag
Timestamp	M	String		ISO time

The list of available services is:
1. Modem_gps: access to the modem and GPS (if installed)
2. Ble: Bluetooth Low Energy
3. Vehicle: Access to On Board Diagnostic data (Future)

2. Bluetooth Low Energy micro service

Scan topic

The role of the topic is to control the gateway behavior on the BLE interface. There is only a payload that define how the gateway will listen for BLE device and report the results.

These parameters can also be set via configuration file to allow the BLE gateway to start operating as soon as the system starts.

Keyword		**Type**	**Value**	**Action/signification**
command	M	String	start	Start listening on BLE and reporting advertising from the devices. The command is not time bound and will remain in effect until a stop command is received
			stop	Stop listening for device advertisement. This is occurring at the of the minimum listen period or after timeout for periodic scan. For indefinite scan started by “start” a scan_result is sent upon the actual end of scan.
			time_scan	Start listening for a time bound period if no timeout is defined the default timeout in the configuration is applied
The following fields are used to configure the scan what is sent at then of the scan (topic scan_result)				
timeout	O	Float		Scan time out in seconds (default=10s)
period	O	Float		Repeat period for timed scan. If 0 or omitted, no repeat. If superior to timeout then a pause time is set between scans. If inferior or equal to scan timeout, then scan restarts immediately
result	O	String	none	Nothing reported in timeout scan
			summary	Publish a summary report (see scan_result) – this is the default
			devices	Publish an array of devices detected and selected by filters during scan (see scan results)
gps	O	String	position	Add the GPS position to the scan result payload
The following define when and what will be published in the advertisement topic. Mode details to be given in the advertisement topic payload description,				
advertisement	O	String	none	No advertisement reported. Useful with time_scan to have the devices reported only at the end
			min	Minimum set of data (default)
			full	Full set of data of the advertisement frame
sub_topics		Boolean	false	No sub-topics (default)
			true	Beacons or Service data as separated sub-topics (see dedicated paragraph)
adv_interval	O	Float		Minimum reporting interval of advertising in seconds. Useful to avoid flooding the broker when some devices are advertising at high rate. If omitted each advertisement received will be pushed towards the broker.
gps	O	String	None	Default – nothing reported
			Pos	Only the position (lat, long) is reported
			Full	Lat,long, heading, speed

Example JSON payload

```
{"command":"time_scan","timeout":20,"advertisement":"min","period":30,"sub_topics":true}
```

Start a periodic scan every 20 seconds for 20 seconds. Reports minimum info on advertisement and publish sub-topics when corresponding data are sent by the devices

Filter topic

This topic allows to set filters to avoid having advertisement messages for beacons or sensors that are not part of the application. Without filter, all BLE devices detected by the gateway are reported. Filters can be set statically via configuration file. They are always passed as a JSON array.

Keyword		Type	Value	Action/signification
type	M	String	rssi	Keep devices with a RSSI above or equal to the minimum defined (negative integer)
			white_list	Keep devices whose addresses are specified
			connectable	Keep devices that have the connectable flag (True or False) equal to the one specified
			starts_with	Keep the devices whose name starts with the string specified. Devices with no name are ignored
			mfg_id_eq	Keep the devices with the specifies Manufactured ID (4 hex digits). Devices with no manufacturer ID defined are ignored
			none	Do not create a filter. Used to clear the filter list and shall be the only item
min_rssi	M	Integer	Negative value between -30 and -99	Only for RSSI filter
match_string	M	String		Only for the starts_with filter
addresses	M	Array of string		Only for the white_list filter
connectable_flag	M	Boolean		Only for the connectable filter
mfg_id	M	Integer (4 hex digits)		Only for the Manufacturer ID filter

To combine filters, just pass an element to the array. In that case a AND condition will be applied.

Example JSON payload

```
[{"type":"rssi","min_rssi":-80},{"type":"connectable","connectable_flag":true}]
```

Select the devices with a RSSI above -80 that are connectable.

Note on RSSI filter: as the RSSI value can fluctuate rapidly, the filter is re-evaluated for each advertisement packet received and only when the threshold is reached a MQTT message is sent. However, if during the scan period the filter is validated once, then the device is kept as valid device and sent in the scan result message. It becomes therefore eligible for GATT transactions.

Scan result topic

This topic is published at the end of a timed scan when the Result keyword in the Scan parameters is not None

If the Summary is selected, then the following JSON structure is sent

Keyword		Type	Value	Action/signification
typestamp	M	Float		Time is seconds from the Epoch of scan end
error	M	Integer		Nonzero if an error occurred
dev_detected	M	Integer		Total number detected during the scan
dev_selected	M	Integer		Total number of devices selected after filtering

gps	O	Data structure		Present when the gps position option is set in the scan directive

When the Devices option is selected the payload also includes an array of the following JSON structure for each selected device

Keyword		**Type**	**Value**	**Action/signification**
address	M	String		MAC address of the device
name	M	String		Local Name of the device, if no name is defined then a zero-length string is sent
rssi	M	Integer		Negative value corresponding to the Received Signal Strength Indicator

Advertisement topic

This topic is published by the gateway each time an advertisement frame is received and meet the following conditions:

a) The device is not filtered out
b) The advertisement frame is not within the advertisement interval set in the scan parameters

The device MAC address is part of the topic structure

/advertisement/<gateway ID>/<Device MAC>/

The payload structure has 2 options min or full. In the JSON structure table, the **min** fields are referred as Mandatory while those from the Full option only as marked as **full**, with a possible Optionality indicator meaning that the field is not present in the advertisement frame from the device

Keyword		**Type**	**Value**	**Action/signification**
local_name	M	String		Local Name of the device, if no name is defined then a zero-length string is sent
timestamp	M	Float		Time is seconds from the Epoch
rssi	M	Integer		Negative value corresponding to the Received Signal Strength Indicator
flags	M	Integer		Bitwise indicator (1 byte)
connectable	M	Boolean		True if the device is connectable
service_data	F/M	Integer		Number of service data values. If nonzero, then an array of service data value
service_data_array	F/O	Array		
service_uuid	M	Integer		16 bits UUID of the service
type	M	Enum		Value data type (see table)
value	M	Variable		Value of the service data, if type is Raw, then the raw hexadecimal byte string is sent
mfg_id	F/O	Integer		Manufacturer ID (16bits)
mfg_data	F/O	Hex_string		Hexadecimal string of the manufacturer data less the Manufacturer ID
tx_power	F/O	Integer		Value of Tx Power when sent from the device

GATT topic

The GATT topic allows to interact with the GATT protocol with devices that are GATT server. The gateway is always a GATT client.

Only the devices that have been selected during a scan can handle GATT request. If a request is sent to a device that has not been seen (and selected) during a scan, the request is rejected.

The topic is structured the following way:

/gatt/<gateway ID>/<device MAC>   and the description of the request is in the payload

Keyword		**Type**	**Value**	**Action/signification**
command	M	String	read	Read a characteristic
			write	Write a characteristic
			allow_notifications	Allow receiving notifications for that characteristic. If a value is present in the request, it is written to the characteristic just after allowing the notifications for that device. (see details below)
			discover	Discover the device and send back the list of services and characteristics
transac_id	O	Integer		If set, that id will be present in the corresponding GATT result frame(s)
service	O	UUID		If present the request will look only for the service with that UUID, if not present this will raise an error an nothing is returned. For devices with many services this is speeding up the transactions
properties	O	Boolean		This parameter is used only for discover and if true, the properties of the characteristics will be reported as well.
bond	O	Boolean		If True, then a bond (pair) request is made after the connection
keep	O	Float		If present and non-zero, keep the connection open for the number of seconds after a message has been received from the device. A default value is applied for allow_notifications
The following fields needs to be passed for read/write/allow_notifications Single characteristic action or array are mutually exclusive				
characteristic	O	UUID		UUID of the Characteristic to be written or read or to allow notifications upon. UUID can be short form (4bytes hexadecimal string) or long form.
type	O	Enum		Type of the value to be written or expected for read. If omitted Raw Hex string is assumed
value	O			Value to be written
action_set	O	Array of tuple (char,type) or (char,type,value)		For optimization, an array of the following tuple can be passed and will

				be processed in one connection to the device

Note on allowing notifications.
To allow notifications, the process is writing the GATT Descriptor corresponding to the Characteristic.
If a type & value are indicated, then they are written to the Characteristic. To allow notifications on one Characteristic and start notifications by writing to same or another Characteristic, this is supported by using an action set. The first action shall contain on the Characteristic UUID of the one that shall send the notifications and the second shall include the full set= Characteristic UUID, type and value.

Example JSON payload

```
{"command":"read","keep":10.0,"action_set": [
{"characteristic":"2A00","type":3},
{"characteristic": "2A19","type":1},
{"characteristic":"F000AA01-0451-4000-B000-000000000000","type":5}
]}
```

Perform reading 3 characteristics and keep the connection alive for 10 seconds

```
{"command":"discover","keep":10.0,"properties":true,"service":"1800"}
```

Discover the service 0x1800 (Generic Access), return the characteristics and properties and keep the connection alive for 10 seconds

```
{"command":"allow_notifications","keep":10.0,"action_set": [
{"characteristic":"6e400003-b5a3-f393-e0a9-e50e24dcca9e","type":3},
{"characteristic": "6e400002-b5a3-f393-e0a9-e50e24dcca9e","type":3,"value":"LED_OFF"}
```

Trigger activity on Nordic serial service (6e400001-b5a3-f393-e0a9-e50e24dcca9e)
Allow notification on the first characteristic then write the string “LED_OFF” on the second.

GATT result topic

This topic is used by the gateway to publish the result of the corresponding GATT request

/gatt_result/<gateway ID>/<Device MAC>

The structure of the payload is depending from the nature of the request.

For read/write/notification

Keyword		Type	Value	Action/signification
command	M	String		Nature of the result (read/write/notification/discover)
error	M	Integer		If nonzero, then an error occurred
transac_id	O	Integer		Present if a transaction ID was set for the request
result	O	JSON structure		One of the following
For read/write/notify				
characteristic	M	UUID		For each valid write
For read and notify only				
type	M	Enum		
value	M			

For discover
An array of service descriptors is published with the GATT_Description tag.

Keyword		Type	Value	Action/signification
service_uuid	M	UUID		UUID of the service up to 128 bits
characteristics	M	Array of UUID		Set of characteristics UUID supported
If the properties parameter is set to true then characteristics are reported as a array of				
uuid	O	UUID		UUID of the characteristics
properties	O	String		List of properties in a single string separated by space

Advertisement sub-topics

In addition to the standard advertisement message and with the same rule the gateway can be configured to publish a specific message for the service data or beacons (Eddystone or iBeacon)

/advertisement/<gateway ID>/<MAC ADDRESS>/<Service data name or uuid>
Or
/advertisement/<gateway ID>/<MAC ADDRESS>/eddystone
Or
/advertisement/<gateway ID>/<MAC ADDRESS>/ibeacon

The service data name is the one from the Bluetooth standard. The value field will be converted according to standard. Not all service data Characteristics are implemented.

The payload for service data includes

Keyword		Type	Value	Action/signification
timestamp	M	float		Time in seconds of the Epoch
type	M	Enum		Type of the value
value	M			Converted following Bluetooth standard

For Eddystone

Keyword		Type	Value	Action/signification
timestamp	M	float		Time in seconds of the Epoch
type	M	Int		Type of Eddystone beacon
For UID Beacons				
txpower	M	int		
beacon_id	M	String		HEX digits
For URL Beacons				
txpower	M	Int		
url	M	String		
For other beacons				
frame	M	String		Hex digits

For iBeacon

Keyword		Type	Value	Action/signification
timestamp	M	float		Time in seconds of the Epoch
uuid	M	UUID		UUID of the Beacon
majmin	M	String		Major-Minor values
txpower	M	Int		

To avoid duplicate the advertisement messages, the standard advertisement messages can be turned off in the scan configuration command option.

General behavior and limitations

1. Scan and GATT operations are mutually exclusive. This is controlled by the gateway. All GATT connections are terminated before a Scan start. GATT transaction requests are refused while a Scan is running
2. In the current version (0.5), time_scan is more reliable and for permanent scanning it is preferred to use periodic scan instead of start/stop.
3. By default, the service runs on hci0, but when several Bluetooth interface are existing this can be changed in the configuration file. Currently only one interface can be managed at a given time.

3. Modem / GNSS Topics

Gps topic

The gps topic allows to query the system gps and publish the result as well as configure the MQTT client to publish the GPS information regularly.
All responses are published on the gps_result/<gateway ID> topic

/gps/<gateway ID>

Keyword		**Type**	**Value**	**Action/signification**
command	M	String	status	Get the GPS status
			start	Start periodic publish. Only the period is considered
			stop	Stop periodic/streaming publish
			read	Publish the position once
			stream	Start publishing continuously following the parameters
period	O	Float	seconds	Publishing period in seconds, default=60 seconds
fix_interval	O	Float	seconds	Maximum publish interval for fixed GPS
nofix_interval	O	Float	seconds	Publish interval when the GPS is not fixed
distance	O	Float	meters	Maximum distance between 2 publish for a moving system. If the system moves slowly, then the ‘fix_interval’ parameter is used for publishing

Gps_result topic

Keyword		**Type**	**Value**	**Action/signification**
command	M	String	status	The result is a status
			positiion	The result is a position
			gps	The result is a global gps error
timestamp	M	String		ISO date and time of publish
error	M	Int		0 = No error 1= GPS not enabled in settings 3 = Communication error with GPS
result	M			JSON structure

Status result

Keyword		Type	Value	Action/signification
fix	M	Boolean		True if GPS is fixed
gps_time	O	String		UTC time HH.MM.SS.mmmmm
nbsat	O	Int		Number of visible satellites
date	O	String		DD/MM/YY date of the last fix
sat_num	O	Int array		Present only if nbsat > 0. Array of the satellite's id (sat number)

Position result

Keyword		Type	Value	Action/signification
fix	M	Boolean		True if GPS is fixed. If false, no other value is transmitted
gps_time	O	String		UTC time HH.MM.SS.mmmmm
latitude	O	Float		Decimal degree negative => south
longitude	O	Float		Decimal degree negative => east
laltitude	O	Float		Altitude in meters
SOG	O	Float		Speed on Ground in Knots (1.852km/h)
COG	O	Float		Heading (Cape on Ground) in degree

Modem topic

The modem topic allows to query the status of the cellular modem. This can be done once or periodically. All messages from the gateway are published on topic /modem_result/<gateway ID>

Keyword		Type	Value	Action/signification
command	M	String	status	publish the modem status
			operators	Publish the list of visible operators
			stop	Stop periodic publish
period	O	Float	seconds	Publishing period in seconds. If 0 or omitted the status is published only once

Modem_result topic

Keyword		Type	Value	Action/signification
command	M	String	status	The result is a status
timestamp	M	String		ISO date and time of publish
error	M	Int		0 = No error
result	M			JSON structure

Status result

Keyword		Type	Value	Action/signification
Model	M	String		Name of Modem model
IMEI	M	String		
gps_on	M	Boolean		True if the GPS is turned on
SIM_Status	M	String		
IMSI	O	String		Only in a SIM is READY
ICCID	O	String		Only if a SIM is ready

Registered	O	Boolean		True if attached
Network_reg	O	String		NONE/HOME/ROAMING
PLMNID	O	Integer		Only if attached
Network_name	O	String		
Network	O	String		
LAC	O	Integer		Location area ID
CI	O	Integer		Cell ID
rat	O	String		Radio technology
band	O	String		
Rssi	O	Integer	-30 to -113	RSSI in dBm
Operators	O	String		Operator view (one operator / line)

4. Vehicle On Board Diagnostic (OBD2) micro service (experimental)

This micro service connects to an OBD2 BLE dongle and publish on MQTT the data (OBD commands) that are requested either via the command or via the parameter file.

Here is the list of the OBD commands supported: https://python-obd.readthedocs.io/en/latest/Command%20Tables/. Only the Mode 1 commands are considered for the reporting.

It should be noted that the available commands are vehicle dependent and operating mode dependent (hybrid vehicle are sending a different set of commends while running on electricity versus fuel).

OBD dongle must be Bluetooth 4.0 compatible. There several existing on the market, but the best performer is the Vgate Icar pro bluetooth 4.0. It is recommended to check that the dongle is visible with a BLE scanner before starting the service.

Vehicle topic

/vehicle/<gateway-id>. Results are published via vehicle_result topic

Keyword		**Type**	**Value**	**Action/signification**
command	M	String	connect	Request the connection to the vehicle OBD system. Not needed if the OBD service is in autoconnect mode.
			read	Start periodic publish
			stop	Stop periodic publish
			status	Publish the OBD service status
device	O	String		This parameter is used for the connect command only, itis either the MAC address of the dongle of a string used to detect it via BLE scan. The string shall contain the first characters used to detect the dongle via its name
on_period	O	Integer		Reporting period is seconds when the engine is on (for read command only)

off_period	O	Integer		Reporting period is seconds when the engine is off (for read command only)
option	O	String	min/vehicle_cmds/actual_cmds	Publish the status of the service (min) and optionally the list of the commands that are supported by the vehicle or the list of the commands that are currently configured to be reported

Vehicle_result topic

Section to be added

5. Configuration via Kura plugin

A configuration service in Kura (and accessible via Kapua) allows the configuration of the following parameters:

1. Activation/Deactivation of the BLE/MQTT gateway
2. Device ID (default is hostname)
3. MQTT broker URL and port
4. MQTT broker credentials
5. MQTT secure connection

6. Configuration files

Global configuration

Here is the list of the configuration parameters

```
optional arguments:

  -h, --help          show this help message and exit

file_settings:
  --settings SETTINGS   A yaml file with argument parameters (see help for
                    options). (default: None)
--autostart  [true/false]  If true, the autostart file will be executed when the MQTT client is starting (see
corresponding paragraph)

mqtt:
  --mqtt_hostname MQTT_HOSTNAME
                    MQTT broker hostname. (default: None)
  --mqtt_username MQTT_USERNAME
                    MQTT broker username. (default: None)
  --mqtt_password MQTT_PASSWORD
                    MQTT broker password. (default: None)
  --mqtt_port MQTT_PORT
```

```
                MQTT broker port. (default: 8883)
  --mqtt_ca_certs MQTT_CA_CERTS
                A string path to the Certificate Authority certificate
                files that are to be treated as trusted by this
                client. (default: None)
  --mqtt_certfile MQTT_CERTFILE
                Strings pointing to the PEM encoded client
                certificate. (default: None)
  --mqtt_keyfile MQTT_KEYFILE
                Strings pointing to the PEM encoded client private
                keys respectively. (default: None)
  --mqtt_cert_reqs MQTT_CERT_REQS
                Defines the certificate requirements that the client
                imposes on the broker. (default:
                VerifyMode.CERT_REQUIRED)
  --mqtt_tls_version MQTT_TLS_VERSION
                Specifies the version of the SSL / TLS protocol to be
                used. (default: _SSLMethod.PROTOCOL_TLSv1_2)
  --mqtt_ciphers MQTT_CIPHERS
                A string specifying which encryption ciphers are
                allowable for this connection. (default: None)
  --mqtt_persist_session
                When False the broker will buffer session packets
                between reconnection. (default: False)
  --mqtt_force_unsecure
                When True the broker will skip the TLS handshake.
                (default: False)
  --mqtt_allow_untrusted
                When true the client will skip the TLS check.
                (default: False)
  --mqtt_timestamp MQTT_TIMESTAMP
                Format of the time stap to be used in messages
                published by the gatreway (default: iso)

gateway:
  --gateway_id GATEWAY_ID
                Id of the gateway. It must be unique on same broker.
                (default: None)

ble:
  --ble           True if a BLE scanner/GATT client to be attached to
                the MQTT client (default: False)
--ble_interface  List of all hci interfaces managed by the service. The current version supports only one
interface at a at time.

Other services:
  --modem_gps       True if a Modem/GPS is to be attached to the MQTT
                client (default: False)
  --modem_gps_addr MODEM_GPS_ADDR
                Address and port of the gps micro service if not the
                default one (default: 127.0.0.1:20231)
  --obd_device     service to be used to connect OBD! Device address or Name
                (default: None)
  --obd_addr OBD_ADDR   Address and port of the OBD micro service if not the
```

default one (default: 127.0.0.1:20232)

By default, the gateway is started by systemd with only the –settings option and the configuration file is /data/solidsense/mqtt/solidsense-mqtt.service.cfg
When using the Kura plugin, it is not recommended to modify directly that file as it is overwritten by Kura.

MQTT client configuration

Bluetooth configuration

For Bluetooth there is also a general parameters file in a JSON file: parameters.json in /data/solidsense/ble_gateway. This file is read by the service upon start only and include the following fields:
Interface: name of the interface to be used hci0/hc1/hci2
Notif_MTU: maximum length of a notification message
Trace: level of tracing (info by default) info/debug/error
Debug_bluez: allow low level tracing in the bluez interface and stack. Only for troubleshooting
Max_connect: maximum number of simultaneous GATT connection

The parameters.json file is automatically generated by the Provisioning system and any modification may lead to a nonfunctional system.

Modem configuration

There is an internal configuraytion file the modem and gps (parameters.json) in /data/solidsense/modem_gps. This file is read by the service upon start only and include the following fields:

Address:	listening IP address for the service
Port:	20231
Modem_ctrl	modem control port /dev/ttyUSB2
Nmea_tty:	nmea port
Trace:	error/info/debug default info
PIN:	PIN of the SIM card if any
Roaming	Allow roaming
OperatorsDB	name of the file with all operator names/PLMNID
Start_gps:	start the GPS
Timer:	period for checking the modem by the service
Nb_retry	Number of periods without attachment before a reset is performed
Start_gps_service	If False, the service is not started, only the modem initialization sequence is perfomed

The parameters.json file is automatically generated by the Provisioning system and any modification (except the PIN) may lead to a nonfunctional system.

Autostart file

The autostart file allows to simulate MQTT messages that the gateway is receiving to start operating. The file is located in /data/solidsense/mqtt/autostart

The structure of the file is simple and reuse the payload structure described in this document.
General syntax:
<topic>:<corresponding MQTT payload>
# comment
Example:
# initial BLE scan topic
scan:{"command":"time_scan","timeout":30,"period":40,"advertisement":"none","result":"summary","gps":"position"}
# initial GPS topic
gps:{“command”:”stream”,”fix_interval”:10.0,”nofix_interval”:300.0,”distance”:100}

Vehicle OBD configuration

The configuration resides in /data/solidsense/vehicle with 2 files:

parameter.json take includes the following parameters:

- trace: level of service traces (error/**info**/debug)
- address: registration address of the service (local 127.0.0.1 or global 0.0.0.0)
- port: IP port 20232 by default
- connect_retry: number of retries of the connection to the OBD dongle before falling into error
- Interface: Bluetooth interface (hci0)
- obd_trace: level of trace of the OBD layer (**error**/info/debug)
- autoconnect: true/false. If true, the service automatically performs a connection to the OBD system using the MAC specified in the parameters. If false, then an explicit connect via MQTT needs to be performed before accessing the data.
- MAC: MAC address of the OBD dongle

std_obd_cmd.json contains the list of OBD commands that are used by default if no more restrictive list is passed as parameter

The configuration is read only upon service start