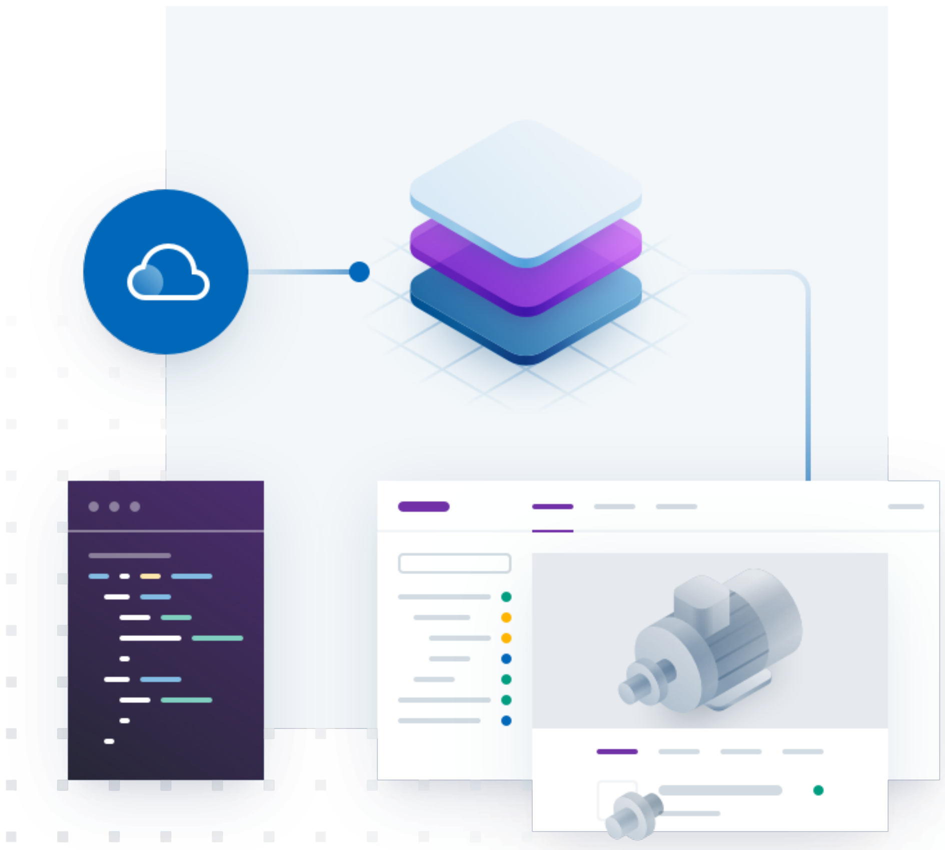
ADI OtoSense™ Smart Motor Sensor Commissioning Guide



Introduction

Before an *ADI OtoSense™ Smart Motor Sensor (SMS)* can connect to the cloud, it must be commissioned.

Broadly, the commissioning process involves three steps:

1. Establishing communication between a mobile or desktop device and the *ADI OtoSense™ SMS*.
2. Adding a network profile to the *ADI OtoSense™ SMS* so that the SMS can connect to a Factory Wi-Fi access point.
3. Provision the *ADI OtoSense™ SMS* in the *ADI OtoSense™ Cloud Application*.

This guide is designed for two types of users:

1. Desktop and mobile developers who want to create a discrete application for commissioning *ADI OtoSense™ Smart Motor Sensors*.
2. Desktop and mobile developers who want to include an *ADI OtoSense™ SMS* commissioning function within an application that has a broader range of use cases.

Before you begin

ADI OtoSense™ SMS Account

Before you start developing a commissioning application, you will need *ADI OtoSense™ SMS* cloud account credentials.

> *Note: ADI OtoSense™ SMS accounts credentials are available by invitation only. If you require access, visit the ADI OtoSense™ website at https://otosense.analog.com/ and fill in the "Contact Us" form.*

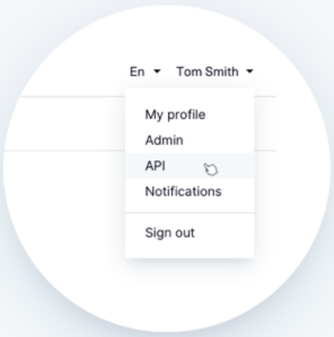
After obtaining *ADI OtoSense™ SMS* account credentials, you should log in to the *ADI OtoSense™ SMS* Developer Portal, located here: https://adi.otosensesms.com/

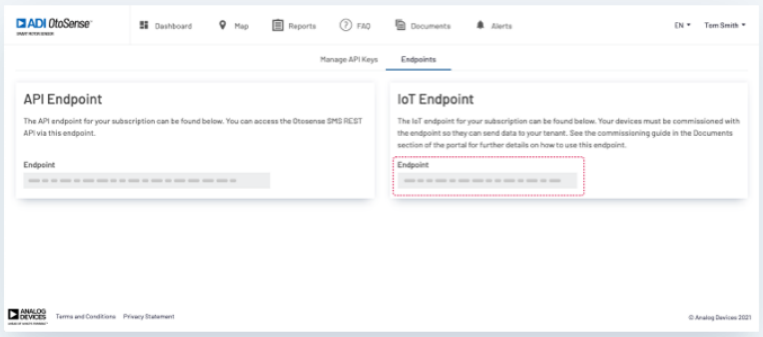
After you have logged in, click on the name associated with your account in the top right corner of the screen. Then, click 'API'. Here, you can generate API Keys and access your API Endpoint and IoT Endpoints.

See Figure 1 for a step-by-step graphical overview of the process.

> *Warning: The API Endpoint, IoT Endpoint, and the API Key are extremely sensitive and should be kept secure. Do not embed the API Endpoint, the IoT Endpoint, or the API Key directly into your code. Do not store the API Key or IoT Endpoint client-side.*

Figure 1: Generating API Keys and locating the API and IoT Endpoints in the *ADI OtoSense™ SMS* Developer Portal





API Endpoint

The API endpoint for your subscription can be found below. You can access the Otosense SMS REST API via this endpoint.

Endpoint

IoT Endpoint

The IoT endpoint for your subscription can be found below. Your devices must be commissioned with the endpoint so they can send data to your tenant. See the commissioning guide in the Documents section of the portal for further details on how to use this endpoint.

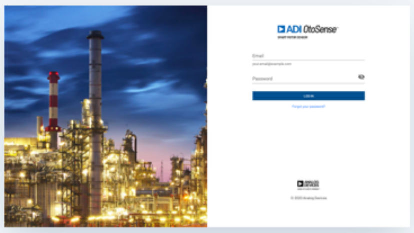
Endpoint

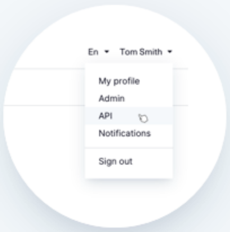
Navigate to the menu in the top right of the portal and select API

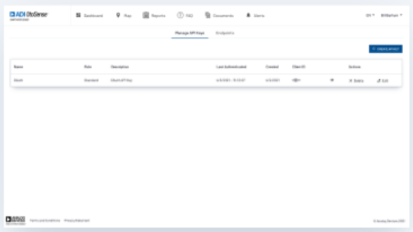
1

On the next screen, select 'Endpoints' in the sub navigation. Your unique IoT Endpoint will be displayed here.

2







Log in to the ADI OtoSense™ Dashboard

1

Navigate to the menu in the top right of the portal and select API

2

From here you can manage API keys and access your Endpoints

3

Network requirements

ADI OtoSense™ Smart Motor Sensors require internet connectivity to communicate with the *ADI OtoSense™ Cloud Application*.

ADI OtoSense™ Smart Motor Sensors must be connected to a 2.4GHz Wi-Fi network that uses the WPA2/3 security protocol.

For detailed network requirements, consult the Network Requirements Guide, available here:
https://docs.otosensesms.com/1.0/en/topic/networking-requirements

> ***Warning:***
>
> - *Smart Motor Sensors do not support 5GHz Wi-Fi networks.*
> - *Smart Motor Sensors do not support WPA2/3 Enterprise security protocol.*
> - *Smart Motor Sensors use MQTT to communicate with the ADI OtoSense™ Cloud Application via port 8883. Ensure this port is open.*

External resources

ADI OtoSense™ Smart Motor Sensors (SMS) are equipped with a Texas Instruments SimpleLink™ Microcontroller. Texas Instruments has published documentation detailing the operation of the SimpleLink™ Microcontroller. For links to Texas Instruments documentation, see Appendix A.

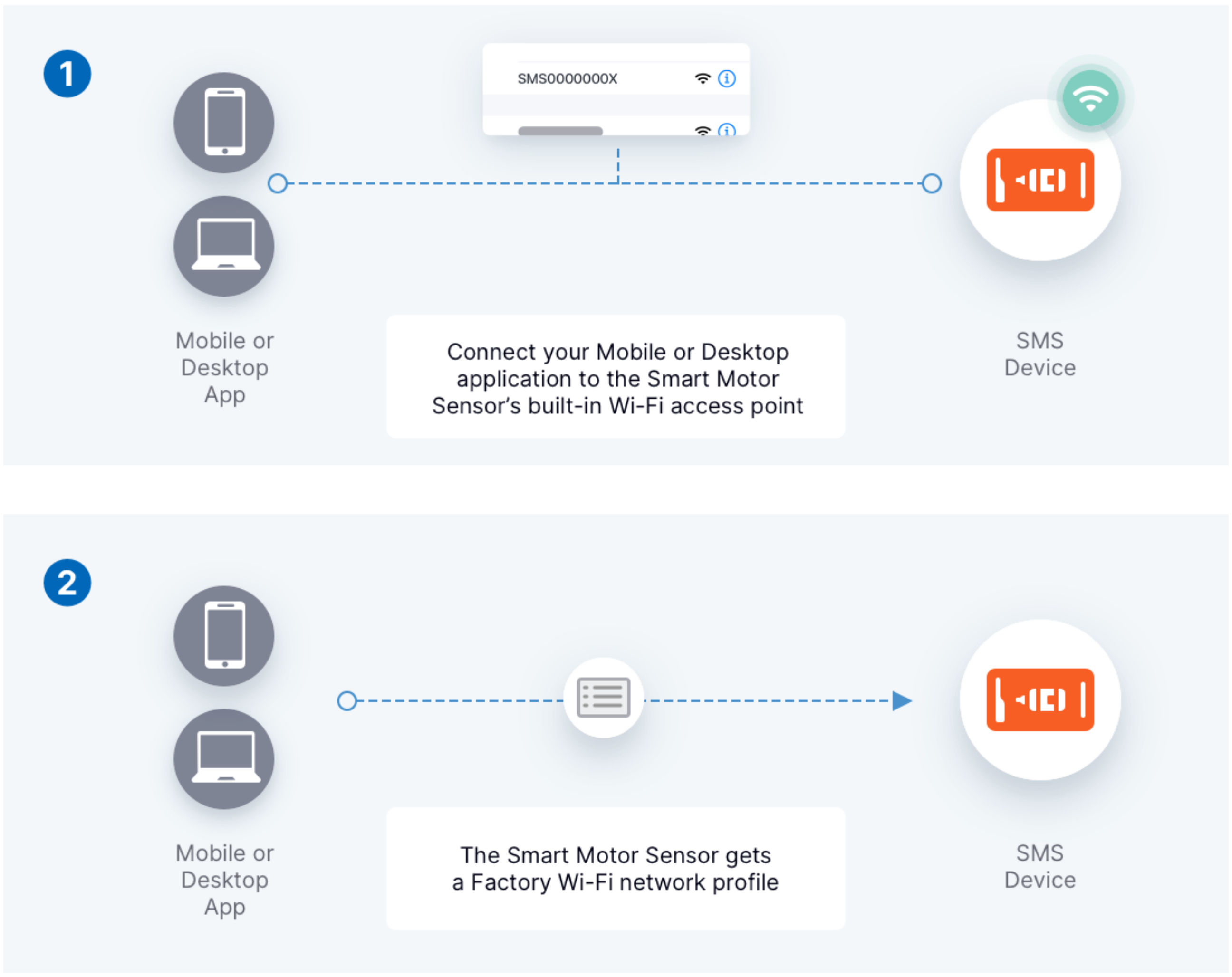
Commissioning a sensor

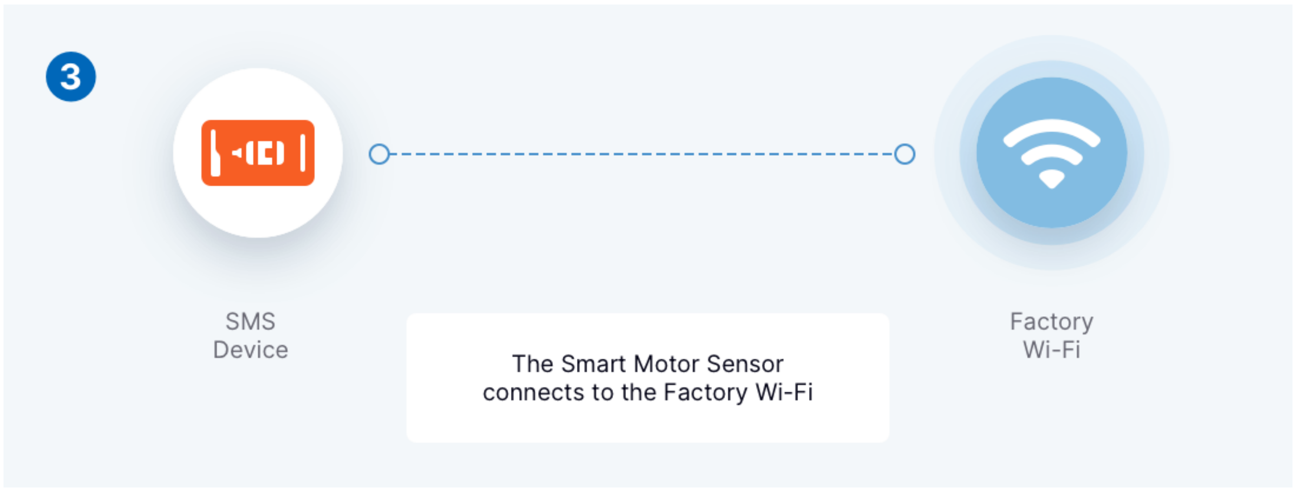
Overview

In order to commission a Smart Motor Sensor (SMS), a mobile or desktop application will first need to connect to the wireless access point (AP) broadcast by the SMS. After the connection has been established, mobile and desktop applications can interface with the Smart Motor Sensor Server API via REST calls.

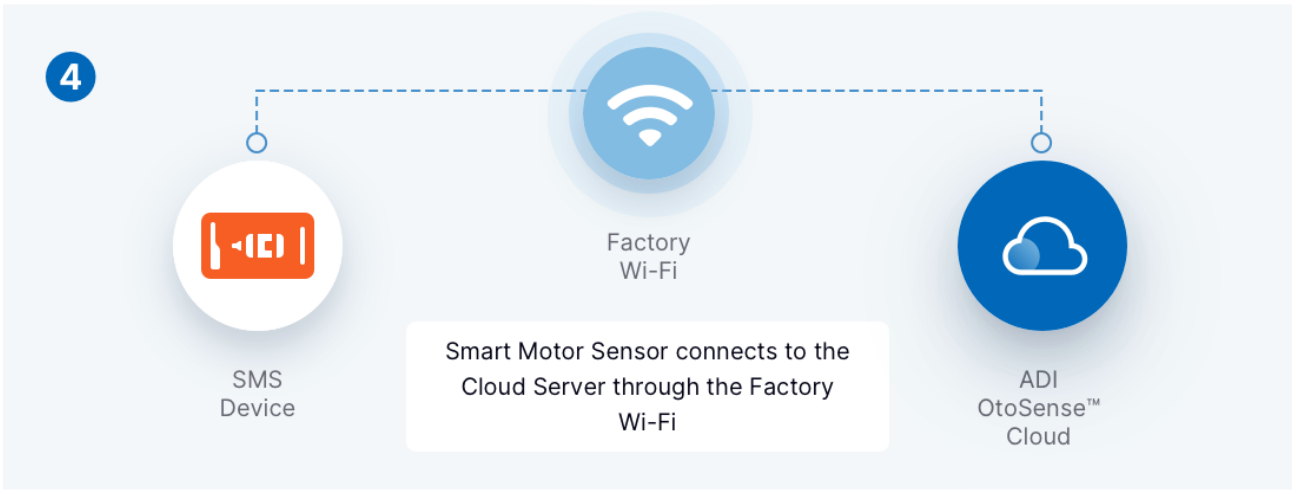
The SMS will need to be provided with a network profile, which it will then use to connect to the Factory Wi-Fi network. Simultaneously, the mobile or desktop application will interface with the *ADI OtoSense™ Cloud API* via REST calls, provisioning the sensor in the cloud.

Figure 2: A high-level overview of the commissioning process


1
SMS0000000X
Mobile or Desktop App
Connect your Mobile or Desktop application to the Smart Motor Sensor's built-in Wi-Fi access point
SMS Device
2
Mobile or Desktop App
The Smart Motor Sensor gets a Factory Wi-Fi network profile
SMS Device



The Smart Motor Sensor connects to the Factory Wi-Fi



Smart Motor Sensor connects to the Cloud Server through the Factory Wi-Fi

Connecting to the sensor’s wireless access point

ADI OtoSense™ Smart Motor Sensor (SMS) devices are equipped with a Texas Instruments SimpleLink™ Microcontroller Unit (MCU). When batteries are inserted into the sensor, the SimpleLink™ MCU enters access point (AP) mode and broadcasts a Wi-Fi access point in the format ‘SMS’ + the sensor’s serial number (e.g., ‘SMS000000X’) for 5 minutes.

If a mobile or desktop device does not connect to the SMS within 5 minutes, the SMS will stop broadcasting its AP, and one of the batteries will need to be removed from the sensor for 30 seconds. After 30 seconds have passed, the battery should be reinserted. The sensor will then return to AP mode, and the mobile or desktop device can once again attempt to connect to it.

> ***Warning:*** *iOS® does not allow applications to scan for, connect to, or disconnect from Wi-Fi access points. Therefore, iOS® commissioning applications must instruct iOS® users to manually scan for, connect to, and disconnect from SMS devices’ Wi-Fi access points via their mobile device settings application.*

> *Information: Need help? The ADI OtoSense™ support team can answer questions about the development process. Reach us by email at: otosensesms@analog.com*

Interfacing with RESTful APIs

Overview

This section provides general information about interfacing with both the *ADI OtoSense™ Smart Motor Sensor* Server API (which is integrated into the Smart Motor Sensor's firmware) and the *ADI OtoSense™ Cloud API*.

> *Warning: REST calls made to the Smart Motor Sensor Server API and those made to the ADI OtoSense™ Cloud API involve different base paths. See Table 2 for additional details.*

Request and response format

When interfacing with the Smart Motor Sensor Server API, mobile and desktop applications should use a combination of JSON ('application/json') and URL ('application/x-www-form-urlencoded') encoded requests and responses. To determine the appropriate format for a particular request, see the "Format" column in Table 3.

When interfacing with the *ADI OtoSense™ Cloud API*, mobile and desktop applications should only use JSON ('application/json') encoded requests and responses.

> *Note: When making POST requests, set the Content-Type header to the appropriate format (i.e., application/json or application/x-www-form-urlencoded).*

HTTP response status codes

Table 1: HTTP response status codes

Code	Description	Result
200	OK	Successful
204	No Content	Successful
400	Bad Request	Unsuccessful
404	Not Found	Unsuccessful

REST Calls

Table 2: Base URLs for Rest Endpoints

Location	Base URLs for REST Endpoints	Notes
Sensor	http://10.123.45.1 †	†The hostname, `http://mysimplelink.net` can also be used but it often fails to resolve. Thus it is recommended to use the IP address instead.
Cloud	Your API Endpoint* (ex. https://api-adi.otosensesms.com/)	*See 'Before you begin' for more details.

Table 3: REST Calls to the Smart Motor Sensor Server API

Description	Method	Path	Format*	Parameters / Notes
Get Sensor Name	GET	/__SL_G_DNP	TEXT	The prefix on the name, 'SMS', should be removed from the response.
Get Sensor Firmware and Hardware Version Numbers	GET	/api/1/host/version/1/sms_version	JSON	Optional. Can be used to determine if a connection has been made to the sensor.
Get Sensor Certificate	GET	/api/1/host/certs/1/clientcert	JSON	Any instances of the string '%02' should be removed from the response and replaced with a space (' ').
Set Sensor IoT Endpoint	POST	/api/1/host/cloud/1/config	JSON	url: The IoT Endpoint
Delete Sensor Network Profiles	POST	/api/1/wlan/profile_del_all	FORM	

Start AP Scan	POST	/api/1/wlan/en_ap_scan	FORM	Optional. Parameters should be set as follows:
				
				__SL_P_SC1=10
				__SL_P_SC2=1

Get Sensor Scan Result	GET	/netlist.txt	TEXT	Optional. Should be called after 'Start AP Scan'. Results will be ';' separated strings with AP names prefixed with security type integer (ex. '2MyAP;3EntAP')

Set Sensor Network Profile	POST	/api/1/wlan/profile_add	FORM	__SL_P_P.A= SSID
				__SL_P_P.B= Security Type
				__SL_P_P.C= Password
				__SL_P_P.D= Priority

Confirm Sensor Network Profile	POST	/api/1/wlan/confirm_req	FORM	Once this request is made, the sensor will stop broadcasting its AP and attempt to connect to the *ADI OtoSense™* cloud.
Get Sensor Profile Error	GET	/api/1/host/info/1/error_info	JSON	If the sensor cannot successfully connect to the internet, this can be used to determine the reason why a profile did not work.

*JSON refers to 'application/json' encoding and FORM refers to 'application/x-www-form-urlencoded' encoding.

Table 4: REST Calls to the *ADI OtoSense™ Cloud API*

Description	Method	Path	Format	Parameters / Notes
Get Oauth Token	POST	/oauth/token	JSON	see https://adi.otosensesms.com/api-reference for more details
Provision Sensor	POST	/sensors	JSON	see https://adi.otosensesms.com/api-reference for more details
Confirm Sensor is Connected to the Cloud	GET	/sensors/{sensorId}	JSON	see https://adi.otosensesms.com/api-reference for more details

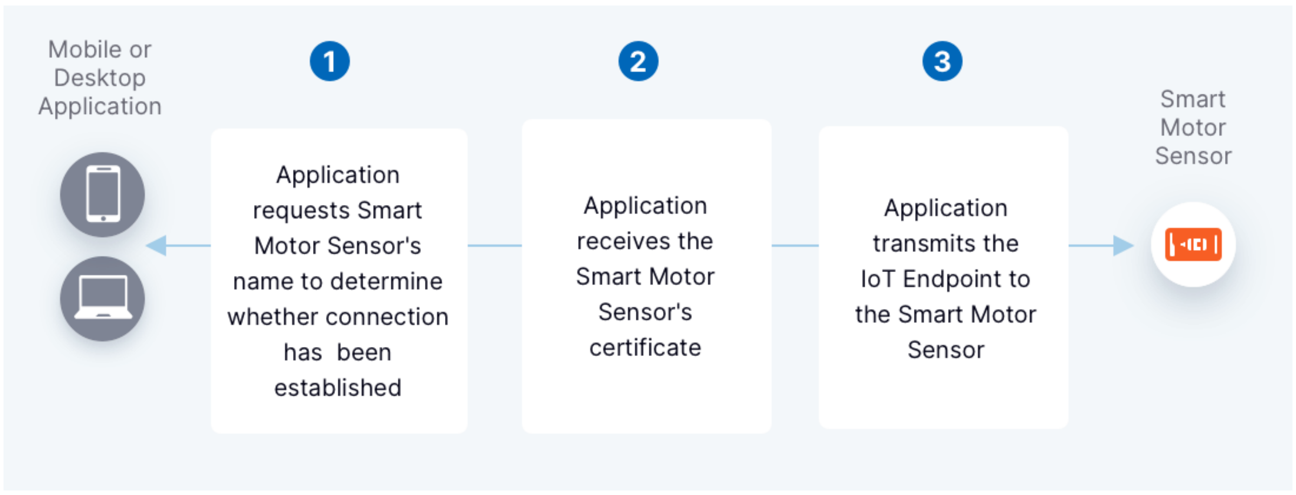
Step-by-step guide with example code

Detecting the connection

It can take up to a minute before a connection between a mobile or desktop device and the *ADI OtoSense™ Smart Motor Sensor* is established.

To determine whether this connection has been successfully established, you can use HTTP GET requests to check for the sensor’s name and/or firmware and hardware version numbers. If the sensor returns these values, the connection has been established.

Figure 3: An example of an initial connection flow


Mobile or Desktop Application
1
Application requests Smart Motor Sensor's name to determine whether connection has been established
2
Application receives the Smart Motor Sensor's certificate
3
Application transmits the IoT Endpoint to the Smart Motor Sensor
Smart Motor Sensor

Getting the sensor name

The sensor name will be used later to identify the sensor in the *ADI OtoSense™* cloud.

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `GET`
Path: `/__SL_G_DNP`
Response Content Type: `text/plain`
Sample Response:

```
SMS0000S111A
```

> ***Warning:*** *The prefix 'SMS' should be removed from the response. This prefix is added to enable users to more easily identify and find the SMS AP SSID.*

Getting the sensor firmware and hardware version numbers (optional)

This is an optional step. Retrieves the firmware and hardware version.

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `GET`
Path: `/api/1/host/version/1/sms_version`
Response Content Type: `application/json`
Sample Response:

```
{
  "Firmware Version": "1.1.0",
  "Hardware Version": "1.0"
}
```

Getting the sensor certificate

The mobile or desktop application should read the *ADI OtoSense™ Smart Motor Sensor* certificate via an HTTP GET request. The certificate is found in the `certInfo` property and is required to provision the SMS in the cloud.

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `GET`
Path: `/api/1/host/certs/1/clientcert`
Response Content Type: `application/json`
Sample Response:

```
{
    "certType": "clientcert.crt",
    "certInfo": "-----BEGIN%02CERTIFICATE-----\nbase64encodeddata .. base64encodeddata=\n-----END%02CERTIFICATE-----\n-----BEGIN%02CERTIFICATE-----\nbase64encodeddata .. base64encodeddata=\n-----END%02CERTIFICATE-----"
}
```

> ***Warning:*** *The certInfo string will contain instances of the string '%02'. These are artifacts of the transfer and should be removed from the response and replaced with a space (' ').*

Setting the IoT endpoint

The Smart Motor Sensor will communicate with the cloud using MQTT packets that are sent and received via the IoT endpoint. The IoT Endpoint needs to be added to the sensor’s configuration file. This can be accomplished via an HTTP POST request.

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `POST`
Path: `/api/1/host/cloud/1/config`
Request Content Type: `application/json`
Request Schema:

Parameter	Description
url	The IoT Endpoint [see Before you begin for details]

Sample Request:

```
{
  "url": "xxxxxxxxxxxx-xxx.iot.xx-xxxx-x.amazonaws.com"
}
```

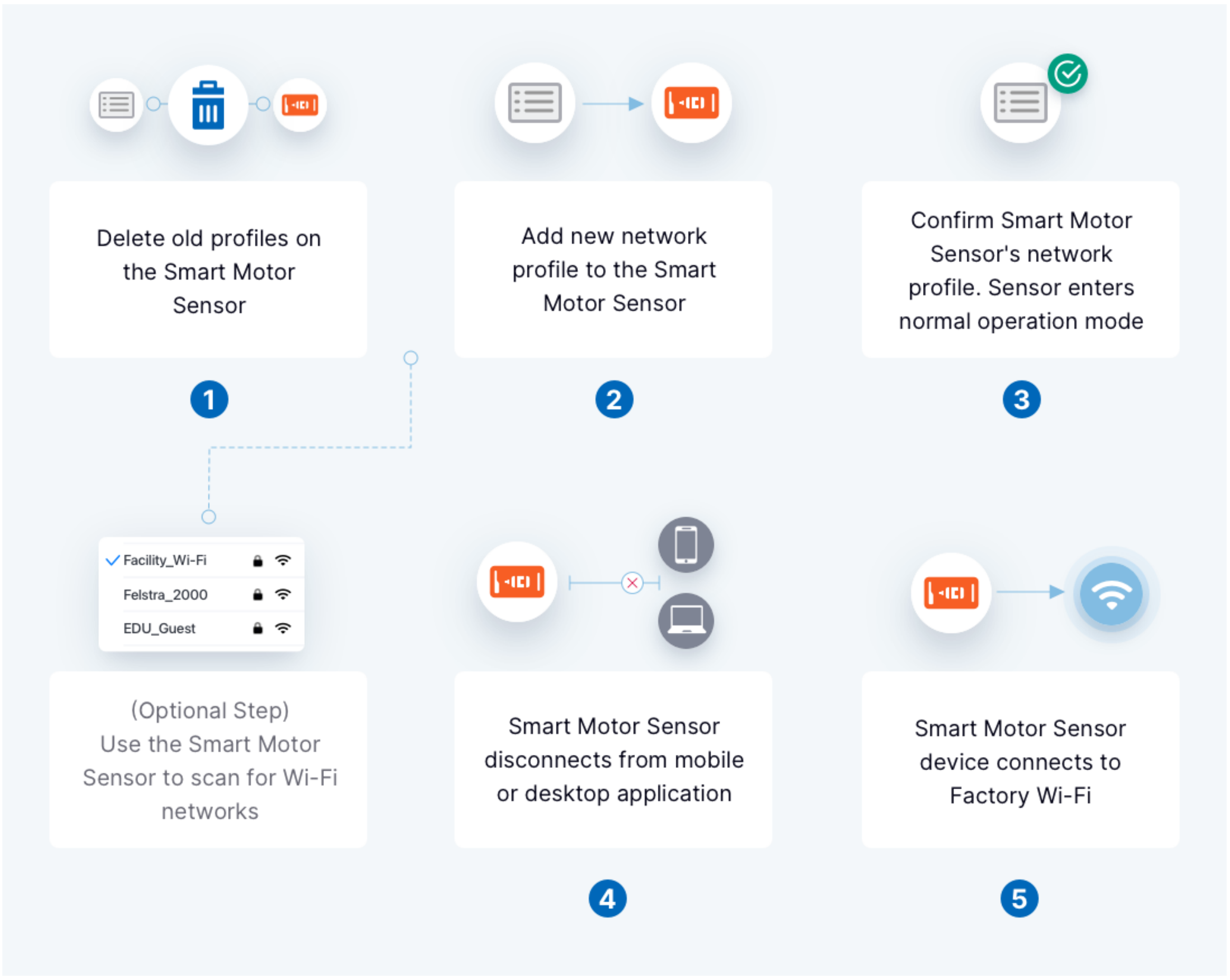
Configuring the SMS network profile

In order to connect to a Wi-Fi access point, the *ADI OtoSense™ Smart Motor Sensor* needs to be provided with a network profile. Once the sensor has received a network profile, it will stop broadcasting its access point / SSID, and the mobile or desktop device should disconnect from it.

After the sensor stops broadcasting its access point, it will connect to the internet via the network specified in the network profile. At this point, the mobile or desktop device should also connect to the internet, typically via a Wi-Fi access point.

If the mobile device does not disconnect from the sensor after 30 seconds, then it can be assumed that an error has occurred.

Figure 4: An overview of the network configuration process


Delete old profiles on the Smart Motor Sensor
1
Add new network profile to the Smart Motor Sensor
2
Confirm Smart Motor Sensor's network profile. Sensor enters normal operation mode
3
Facility_Wi-Fi
Felstra_2000
EDU_Guest
(Optional Step) Use the Smart Motor Sensor to scan for Wi-Fi networks
Smart Motor Sensor disconnects from mobile or desktop application
4
Smart Motor Sensor device connects to Factory Wi-Fi
5

Deleting old network profiles on the sensor

Old network profiles need to be deleted before a new network profile can be added. This can be accomplished via an HTTP POST request.

> *Note: Even if an ADI OtoSense™ Smart Motor Sensor has not been previously commissioned, this step should still be performed.*

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `POST`
Path: `/api/1/wlan/profile_del_all`

> *There is no body required in this request.*

Using the sensor to scan for Factory Wi-Fi access points (optional)

Have the SMS perform an AP scan. The results can be retrieved using the 'Getting sensor scan results (optional)' detailed below.

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `POST`
Path: `/api/1/wlan/en_ap_scan`
Request Content Type: `application/x-www-form-urlencoded`
Request Schema:

Parameter	Description
__SL_P_SC1	Time between scan cycles. Should be set to ‘0’.
__SL_P_SC2	The number of scan cycles. Should be set to ‘10’.

Sample Request:

```
__SL_P_SC2=10&__SL_P_SC1=0
```

Getting sensor scan results (optional)

Should only be called after ‘Using the sensor to scan for Factory Wi-Fi access points (optional)'. The response will be a semi-colon ';' separated list of APs, each prefixed with the AP security type (see table).

Security Type Prefix	Description
0	Open
1	WEP
2	WPA
3	WPA/WPA2
5	WPA3

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `GET`
Path: `/netlist.txt`
Response Content Type: `text/plain`
Sample Response:

```
2MyAP;2CoffeeShop;3EntAP;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;X;
```

Setting the sensor’s network profile

In order to connect to a Wi-Fi network, the SMS device needs a network profile. Use this POST request to set the sensor profile.

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `POST`
Path: `/api/1/wlan/profile_add`
Request Content Type: `application/x-www-form-urlencoded`
Request Schema:

Parameter	Description
__SL_P_P.A	SSID. The name of the Factory Wi-Fi access point the sensor should connect to. No more than 32 characters are allowed. If the SSID includes special characters (e.g., ‘%’ or ‘+’), they must be URL encoded (also known as “percent encoded”). Semicolons (‘;’) cannot be used in the SSID. **Warning**: The encoding of the SSID is *independent* of the standard URL encoding of x-www-form-urlencoded payloads. Therefore, special characters will need to be double encoded before transmission - (ex. `%25` will need to be encoded again as `%2525`)
__SL_P_P.B	Network security type. Security type options are as follows: 0-Open, 1-WEP, 2-WPA, 3-WPA2/WPA2+PMF, 5- WPA3, 6-WPS/Push-button, 7-WPS/Pin Keypad, 8-WPS/Pin Display
__SL_P_P.C	Factory Wi-Fi password. Must not exceed 64 characters. If the router password includes special characters (e.g., ‘%’ or ‘+’), they must be URL encoded (also known as “percent encoded”). Semicolons (‘;’) cannot be used in the router password. **Warning**: The encoding of the password is *independent* of the standard URL encoding of x-www-form-urlencoded payloads. Therefore, special characters will need to be double encoded before transmission - (ex. `%25` will need to be encoded again as `%2525`)
__SL_P_P.D	Profile priority. Should be set to ‘0’.

Sample Request:

```
 __SL_P_P.A=FactoryWiFiRouter& __SL_P_P.B=3& __SL_P_P.C=dEs%24%2F%5E%7ER0vDQ50o%22%3F%27UA&
__SL_P_P.D=0
```

Confirming the network profile

After the sensor’s network profile has been set, it needs to be confirmed via an HTTP POST request. After this request is made, the sensor should stop broadcasting its AP and attempt to connect to the Factory Wi-Fi.

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `POST`
Path: `/api/1/wlan/confirm_req`

> *There is no body required in this request.*

Getting sensor profile error

Once the profile is confirmed the sensor will attempt to obtain an IP address and connect to the internet. If this is unsuccessful, the sensor will return to AP mode. The provisioning application can connect to the sensor AP and call this request to gain some insight into why the profile did not work. Possible reasons include bad profile, bad profile password, or no internet connectivity.

> *Note: This request is only useful after a sensor is unable to use the profile to connect to the internet. No other error conditions are covered by this request.*

Protocol: `HTTP`
Host: `10.123.45.1`
Method: `GET`
Path: `/api/1/host/info/1/error_info`
Response Content Type: `application/json`
Response Schema:

Parameter	Description
profileError	0=no error, 1=error
internetError	0=no error, 1=error

Sample Response:

```
{
  "profileError": 0,
  "internetError": 1
}
```

Connecting to the cloud

After the network profile has been successfully added to the Smart Motor Sensor, it will stop broadcasting its AP, and both the sensor and the mobile or desktop device should connect to the internet.

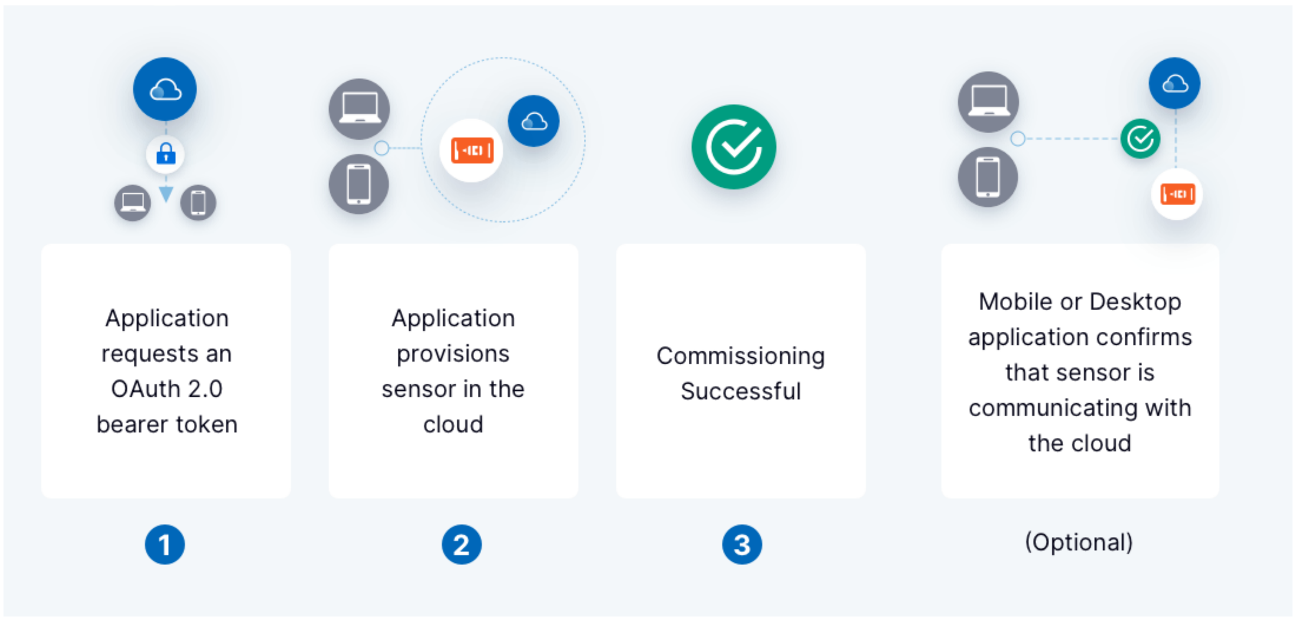
The Smart Motor Sensor will connect to the internet via the Wi-Fi network specified in the network profile. The mobile or desktop device may connect to the internet via the same Wi-Fi network, another Wi-Fi network, or, less commonly, another access mode (e.g. wired LAN).

Once the sensor and the mobile or desktop device have connected to the internet, the sensor will attempt to connect to the cloud via the IoT endpoint. Before the sensor can connect to the cloud, however, the mobile or desktop application must provision it using the sensor certificate. Provisioning the sensor with the cloud ensures that the data from the sensor will flow into the user's tenant and can be processed by the *ADI OtoSense™ Smart Motor Sensor* analytics.

The sensor will then enter normal operation mode and begin communicating with the cloud.

> ***Warning:*** *The sensor will make 3 attempts (at 30 second intervals) to connect to the cloud. If you are unable to provision the sensor within this time frame, the sensor will enter sleep mode, and you will need to wait 20 minutes before you will be able to confirm that the sensor is connecting to the cloud [see Confirming the sensor's cloud connection (optional)].*

Figure 5: An overview of the cloud connection phase


Application requests an OAuth 2.0 bearer token
Application provisions sensor in the cloud
Commissioning Successful
Mobile or Desktop application confirms that sensor is communicating with the cloud
1
2
3
(Optional)

Getting the OAuth bearer token

The *ADI OtoSense™ Smart Motor Sensor REST API* implements the OAuth 2.0 Client Credentials flow. The Client Credentials flow is recommended for use in machine-to-machine authentication. Your application will need to securely store its Client ID and Secret and pass those to the token service in exchange for an access token. At a high-level, the flow only has two steps:

1. Your application passes its client credentials in a token request.
2. If the credentials are accurate, OtoSense SMS responds with an access token.

> *Refer to https://adi.otosensesms.com/api-reference for more details on how to obtain a client id and secret.*

Protocol: `HTTPS`
Host: `your.apiendpoint.otosensesms.com`
Method: `POST`
Path: `/oauth/token`
Request Headers:

Parameter	Description
Authorization	HTTP Basic Auth. Create a basic auth header from your `client_id` and `client_secret` using the rules of HTTP basic auth. The header field should be in the form of `Authorization: Basic {credentials}` where credentials is the Base64 encoding of client_id and client_secret joined by a single colon `:`.

Request Content Type: `application/x-www-form-urlencoded`
Request Body:

```
grant_type=client_credentials
```

Response Content Type: `application/json`
Sample Response:

```
{
  "token_type": "bearer",
  "access_token": "eyJ0eXBlIXN0In0=.OTMMtNUxZGI2.Py6iJNUM3CEy2S.GTKP/92lYch4B/c=.MjAyMDBa",
  "expires_in": 3600,
  "scope": "service.sms tenant.tid.abc123 role.standard"
}
```

Provisioning the sensor in the cloud

Once this step is finished, the commissioning process is complete. Using the sensor certificate, provision the SMS in the cloud to claim it for the user's tenant.

Protocol: `HTTPS`
Host: `your.apiendpoint.otosensesms.com`
Method: `POST`
Path: `/sensors`
Request Headers:

Parameter	Description
Authorization	Bearer Token. This bearer token obtained from the 'Getting the OAuth bearer token' request.

Request Content Type: `application/json`
Request Schema:

Parameter	Description
cert	X509 certificate PEM for this sensor obtained from 'Getting the sensor certificate'.

Sample Request:

```
{
  "cert": "-----BEGIN CERTIFICATE-----\nbase64encodeddata .. base64encodeddata=\n-----END CERTIFICATE-----\n-----BEGIN CERTIFICATE-----\nbase64encodeddata .. base64encodeddata=\n-----END CERTIFICATE-----"
}
```

Response Content Type: `application/json`
Sample Response:

```
{
  "sensorId": "0000S111A",
  "provisioned": "2020-08-18T18:43:35.713Z",
  "provisionedBy": "APIKEYID"
}
```

Confirming the sensor’s cloud connection (optional)

If you want to confirm that data is flowing from the device to the cloud, you can use an HTTP GET request to check the sensor’s `lastConnected` property in the cloud.

> *Note: The sensor may take 1-20 minute(s) to begin communicating with the cloud, depending on whether the device is in hibernation mode.*

Protocol: `HTTPS`
Host: `your.apiendpoint.otosensesms.com`
Method: `GET`
Path: `/sensors/{sensorId}`
Path Parameters:

Parameter	Description
sensorId	The senosr ID. (ex. 0000S111A)

Request Headers:

Parameter	Description
Authorization	Bearer Token. This bearer token obtained from the 'Getting the OAuth bearer token' request.

Response Content Type: `application/json`
Sample Response:

```
{
  "sensorId": "0000S111A",
  "provisioned": "2020-08-18T18:43:35.713Z",
  "provisionedBy": "APIKEYID",
  "lastConnected": "2021-03-02T17:21:03.314Z"
}
```

Appendix A: External Documentation

1. The Texas Instruments SimpleLink™ Wi-Fi® CC3x20, CC3x3x Network Processor User's Guide

Available here: https://www.ti.com/lit/ug/swru455m/swru455m.pdf

Key reference points are 9.4.4 (HTTP Server/RESTful API Processing/WLAN Profiles) and Chapter 16 (Provisioning).

2. The Texas Instruments SimpleLink™ Wi-Fi® CC3x20, CC3x3x Device Provisioning Guide

Available here: https://www.ti.com/lit/an/swra513c/swra513c.pdf?ts=1614601600584

Key reference points are table 8.5 (Successful AP Provisioning With Cloud Confirmation) and table 9 (Provisioning HTTP/HTTPS Server APIs).