Tuya uses a private cluster (0xEF00 or 61184) which they use to set and report data point values. These datapoints (DPs) values are defined each particular device: I've attached an Excel file Thomas has sent me earlier, which contains the configured datapoints for this device and their values. Accessing them requires you to use their privately defined commands and payloads, which are defined in their documentation:

1. Zigbee Connection Standard-TuyaOS-Tuya Developer

    a. TY_DATA_ REQUEST command is used to write data point values
    b. TY_DATA_REPORT command is used by the thermostat to report a data point value to the gateway (marked as ` directionToClient`).
    c. The “DP Data Format” is important, note that all number are transferred as Big-Endian.
    d. TUYA_MCU_SYNC_TIME to set the thermostat time. Needs to have both UTC Unix timestamp in seconds and that same timestamp but then adjusted to the local timezone.

2. Serial Communication Protocol-Documentation-Tuya Developer

    a. This mostly defines the Data Unit (DU) format which becomes part of the Zigbee command payload.
    b. In specific, the different data types are interesting, such as boolean, values & enum.

In short, you can set a datapoint value by using the `TY_DATA_REQUEST` command (0x0 or 0), with as payload a sequence number and a data unit. And yes, I agree that the command name is confusing. Below is how I defined the `TY_DATA_REQUEST` command for the Homey platform. Note that this is a TS abstraction, but the defined `args` are serialized into their bitwise representation and pasted after each other.

```
dataRequest: {
  id: 0x0,
  frameControl: ['clusterSpecific', 'disableDefaultResponse'],
  args: {
    sequenceNumber: ZCLDataTypes.uint16,
    dataPoint: ZCLDataTypes.uint8,
    dataType: ZCLDataTypes.uint8,
    length: ZCLDataTypes.data16,
    data: ZCLDataTypes.buffer,
  },
},
```

As an example, this is the helper function I use on the Homey platform to read/write a value data type, which is basically a 32 bit unsigned integer:

```
public readData32(data: Buffer): number {

  return data.readUInt32BE();

}
```

```
public async writeData32(dataPoint: number, value: number): Promise<void> {
```

```
    const data = Buffer.alloc(4);
    data.writeUInt32BE(value, 0);
    this.debugger('Writing uint32 value', data);

    return this.cluster.dataRequest({
      sequenceNumber: this.transactionId++,
      dataPoint,
      dataType: TuyaDataType.Value,
      length: 4,
      data,
    });
  }
```

The final part is the part I struggled the most with: you want to receive reports from the thermostat so you get the changes made on the thermostat self in your smarthome gateway. It appears that the thermostat does not start reporting these data points values by itself, but there is trick to do so. You simply need to make a “magic” request to the basic cluster (0x0 or 0), and request a couple of its properties in a single query. I haven’t confirmed whether the attribute order is important, but let us assume it is.

- manufacturerName (0x4)
- zclVersion (0x0)
- appVersion (0x1)
- modelId (0x5)
- powerSource (0x7)
- attributeReportingStatus (0xFFFE)

The thermostat now believes it is connected with the Tuya gateway and it will now start reporting all data point values on the Tuya private cluster (0xEF00 or 61184) regularly (at least once a minute or so, or promptly after a change has been made). The reports will come in a `TY_DATA_REPORT` command (0x2 or 2), but not the `directionToClient` flag: it might need a different approach for your integration

- Følgende datapunkter

DP ID	Data Point (DP)	Identifier	Data Transfer Type	Data Type	Properties
1	Switch	switch	Send and Report	bool	

2	Mode	mode	Send and Report	enum	Enum Value: 0, 1, 2, 3, 4
16	Set temperature	temp_set	Send and Report	value	Value Range: 5-35, Pitch: 1, Scale: 0, Unit: ℃
24	Room temperature	temp_current	Report Only	value	Value Range: 0-99, Pitch: 1, Scale: 0, Unit: ℃
28	Temperature correction	temp_correction	Send and Report	value	Value Range: -9-9, Pitch: 1, Scale: 0, Unit: ℃
30	Child lock	child_lock	Send and Report	bool	
101	地面温度	TempFloor	Report Only	value	Value Range: 0-99, Pitch: 1, Scale: 0, Unit: ℃
102	传感器类型	SensorType	Send and Report	enum	Enum Value: 0, 1, 2
103	启动温差	TempActivate	Send and Report	value	Value Range: 1-9, Pitch: 1, Scale: 0, Unit: ℃
104	负载	LoadStatus	Report Only	bool	
105	编程	TempProgram	Send and Report	raw	
106	防开窗	OpenWindow	Send and Report	bool	
107	高温保护	MaxProtectTemper	Send and Report	value	Value Range: 20-95, Pitch: 1, Scale: 0, Unit: ℃
199	产品序列号	SN	Report Only	string	

- Zigbee Connection Standard-TuyaOS-Tuya Developer.
  - TY_DATA_ REQUEST command is used to write data point values
  - TY_DATA_REPORT command is used by the thermostat to report a data point value to the gateway (marked as ` directionToClient`).
  - The “DP Data Format” is important, note that all number are transferred as Big-Endian.
  - TUYA_MCU_SYNC_TIME to set the thermostat time. Needs to have both UTC Unix timestamp in seconds and that same timestamp but then adjusted to the local timezone.
- The “magic” attribute read to enable the automatic data point reporting by the thermostat. The gateway should request the following properties (possibly in this order, but in a single request) from the basic cluster on device initialisation
  - `manufacturerName`
  - `zclVersion`
  - `appVersion`
  - `modelId`
  - `powerSource`
  - `attributeReportingStatus`

Basic cluster:

NWK Key Sequence Number: 0
NWK Payload: (23 bytes)
APS Header: 0xB80101040000FF40
Frame Control: 0x40
Destination Endpoint: 0xFF
Cluster ID: [0x0000] General: Basic
Profile ID: [0x0104] ZigBee Home Automation
Source Endpoint: 0x01
APS Counter: 184
APS Payload: (15 bytes)
ZCL Header: 0x001C10
Frame Control: 0x10
Transaction Sequence Number: 28
General Command Frame: [0x00] Read Attributes
ZCL Payload: (12 bytes)
Attribute ID: [0x0004] Manufacturer Name
Attribute ID: [0x0000] ZCL Version
Attribute ID: [0x0001] Application Version
Attribute ID: [0x0005] Mode ID
Attribute ID: [0x0007] Power Source
Attribute ID: [0xFFFE] Reserved