Using a RESTful API to Connect to Remote I/Os

Moxa Technical Support Team
support@moxa.com

Contents

Released on June 13, 2017

About Moxa

Moxa is a leading provider of edge connectivity, industrial networking, and network infrastructure solutions for enabling connectivity for the Industrial Internet of Things. With over 30 years of industry experience, Moxa has connected more than 50 million devices worldwide and has a distribution and service network that reaches customers in more than 70 countries. Moxa delivers lasting business value by empowering industry with reliable networks and sincere service for industrial communications infrastructures. Information about Moxa's solutions is available at www.moxa.com.

How to Contact Moxa

Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

MOXA
Reliable Networks ▲ Sincere Service

1. Introduction

The latest firmware versions of ioLogik E1200 series products support a RESTful API protocol. In this tech note, we explain what a RESTful API is, why using a RESTful API is easy for web service development, and show how to implement a RESTful API for IIoT applications.

RESTful API functionality can only be implemented if the latest firmware and utility versions have been installed on your ioLogik E1200 product. Refer to the information listed below for details:

Firmware

- ioLogik E1210 V2.5 (std. version)
- ioLogik E1211 V2.4 (std. version)
- ioLogik E1212 V2.5 (std. version)
- ioLogik E1213 V2.6 (std. version)
- ioLogik E1214 V2.5 (std. version)
- ioLogik E1240 V2.4 (std. version)
- ioLogik E1241 V2.5 (std. version)
- ioLogik E1242 V2.5 (std. version)
- ioLogik E1260 V2.5 (std. version)
- ioLogik E1262 V2.5 (std. version)

Utility

- ioSearch V1.15 (std. version)

2. What is a RESTful API?

A RESTful API is an API that is based on REST (REpresentational State Transfer) principles. A RESTful API provides programmers with convenient programming functions for transmitting data back and forth between web resources residing at various network locations.

3. Why using a RESTful API is easy for web service development

Since RESTful APIs communicate via HTTP (HyperText Transfer Protocol) and are a standard means of web communication, they play an important role in IIoT (Industrial Internet of Things) applications. RESTful APIs are suitable for cross-application and cross-device topologies due to the large number of frameworks, drivers, and other resources that leverage the HTTP protocol. With the high degree of accessibility and convenience RESTful APIs provide to web connectivity, they have become one of the best options for IIoT device-to-cloud communications.

A RESTful API can be used to turn an ioLogik E1200 series product into an IoT device. Users will be able to monitor production lines, check machine status, and receive active alarm messages quickly—anytime, anywhere. Furthermore, it is easy to connect ioLogik E1200 series products to the Microsoft Azure cloud with the aid of Moxa's MX-AOPC UA Server and MX-AOPC UA Logger.

A RESTful API can reduce the need for specialized gateway hardware, protocol converters, and middleware. Not only are RESTful APIs easy for IT experts to use since they are already familiar with HTTP, RESTful APIs can also be used by automation engineers, who are generally more familiar with OT protocols. From the automation engineer's point of view, RESTful APIs are an ideal tool for connecting legacy industrial devices to the Internet.

4. Using a RESTful API with Remote I/O devices

How to use a RESTful API with remote I/O devices

Since RESTful APIs are based on HTTP, they are highly compatible with a variety of programming languages (including C# and JavaScript). Consequently, you do not need to learn a new programming language to communicate with Moxa's ioLogik E1200 products. The RESTful API's GET method makes it easy for IT professionals to get data, and the PUT method allows you to easily change the configuration and/or status of your devices.

The following examples give step-by-step instructions on how to use a web service or Postman to PUT and GET data.

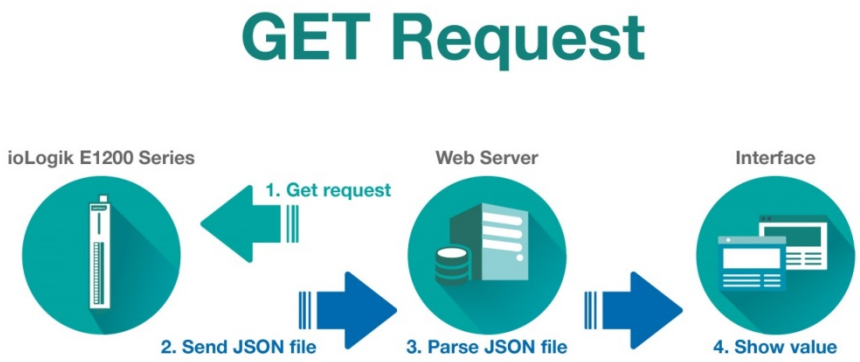
Using RESTful API via web service

The example uses the Windows 7 OS, with the server configured as follows:

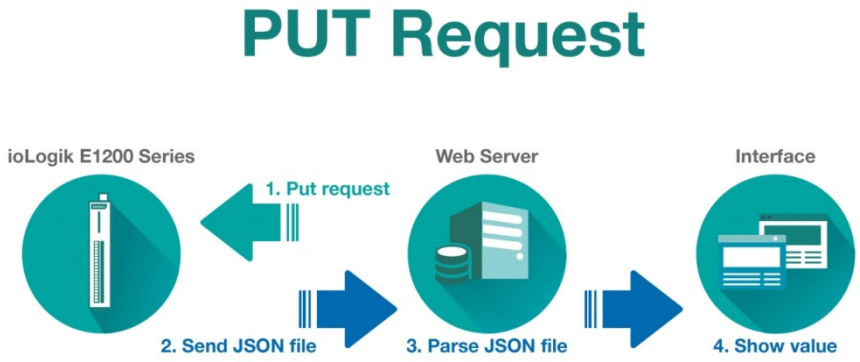
- Note.js: v5.3.0
- Express: v4.15.0

The following flowcharts illustrate how only four steps are needed to implement the GET and PUT methods.

GET: 4 steps (status updated at regular intervals)



PUT: 4 steps (triggered by command)



1. GET request: The server sends RESTful API GET requests to the ioLogik E1200 to get information.

```
//RESTful Demo: GET requset with HTTP Header info
var getIOInfo = function(req, res) {
    var get_request = require('request');
    get_request({
        url: 'http://192.168.127.254/api/slot/0/io/di',
        headers: {
            'Content-Type': 'application/json',
            'Accept': 'vdn.dac.v1'
        }
    }, reqGetCallback);
};
```

PUT request: The server sends RESTful API PUT requests to the ioLogik E1200 to change information.

```
//RESTful Demo: PUT requset with HTTP Header info
exports.setDiTable = function(req, res) {
    //Set DI data to json format
    var formData = {};
    formData = req.body;
    jsonData = JSON.stringify(formData);

    //PUT request
    var put_request = require('request');
    put_request({
        method: 'PUT',
        url: 'http://192.168.127.254/api/slot/0/io/di',
        headers: {
            'Accept': 'vdn.dac.v1',
            'Content-Type': 'application/json',
            'Content-Length': Buffer.byteLength(jsonData)
        },
        body: jsonData  //set data to PUT request content
    }, reqPutCallback);

};
```

Note: When sending requests, headers should be included to ensure that the RESTful API is implemented successfully. Headers should include the following information:

> Accept: vdn.dac.v1

> Content-Type: application/json

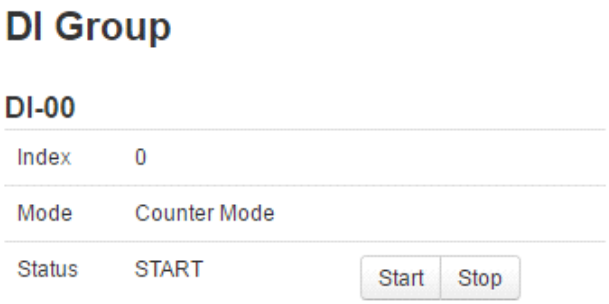
2. Send JSON file: The ioLogik E1200 sends a response to the server in JSON format.

3. Parse JSON file: The content is parsed and saved by the server.

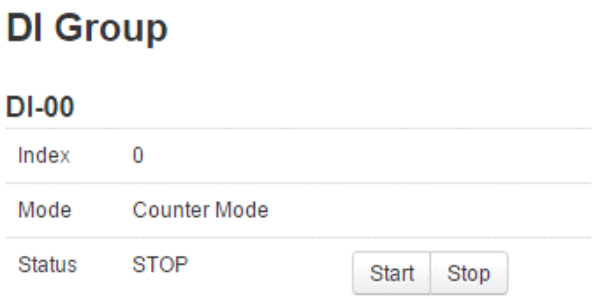
```
//Get & parse DI json table
function reqGetCallback(error, resp, body) {
    if(!error && resp.statusCode == 200) {  //request success

        restTable = {};
        diInfoList = [];
        var info = JSON.parse(body); //parse HTTP content to json
        restTable = info;
        diInfoList= info.io.di; //save IO-DI info to server
    }
    else
    {
        diInfoList = [];
        diInfoList = defaultDiList;
    }
};
```

4. Show value: The information can be displayed in various interfaces. Here, we take a web browser as an example:

GET: Show current I/O status



PUT: The counter status will be changed from STOP to START.



Using RESTful API via Postman

As an alternative, test tools offer another, easier solution. Postman, for example, is an application that helps users build, test, and document APIs. Step-by-step procedures on how to use Postman are described below:

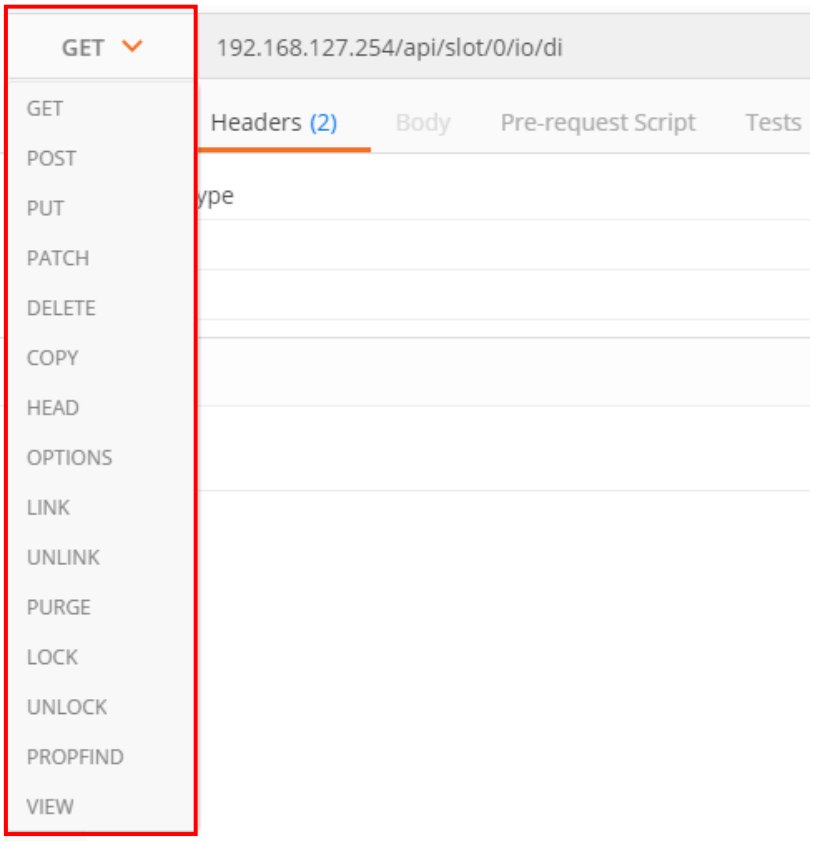
Introduction to Postman

In the example illustrated below, Postman is the master and the ioLogik E1200 is the slave. We first explain how to set up Postman, and then explain how to deploy GET and PUT methods to get data from, and change data in, the ioLogik E1200.

How to set up Postman

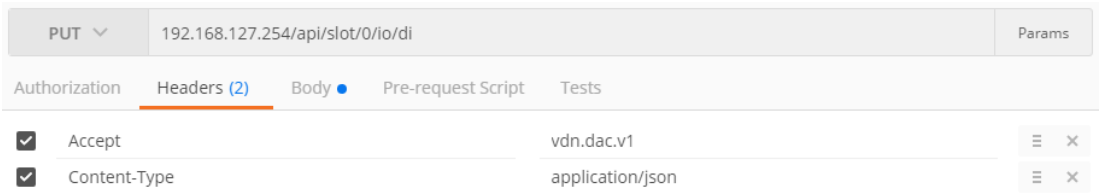
1. First, check the type of data your product is transmitting (e.g., DI, relay, or RTD).
2. Refer to Chapter 3 of the ioLogik E1200 series user's manual to enable the RESTful API.
3. Open Postman, and use the tabs near the top to create a **GET example** and **PUT example**.

4. Select the GET method on the GET example tab, and select the PUT method on the PUT example tab.



5. Click **Headers** on the menu bar and then create the following content:

   - Content-Type: application/json
   - Accept: vdn.dac.v1

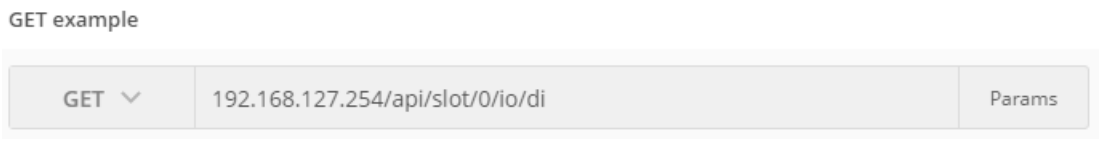


Note: When typing commands (e.g., headers) keep in mind that the commands are case-sensitive.

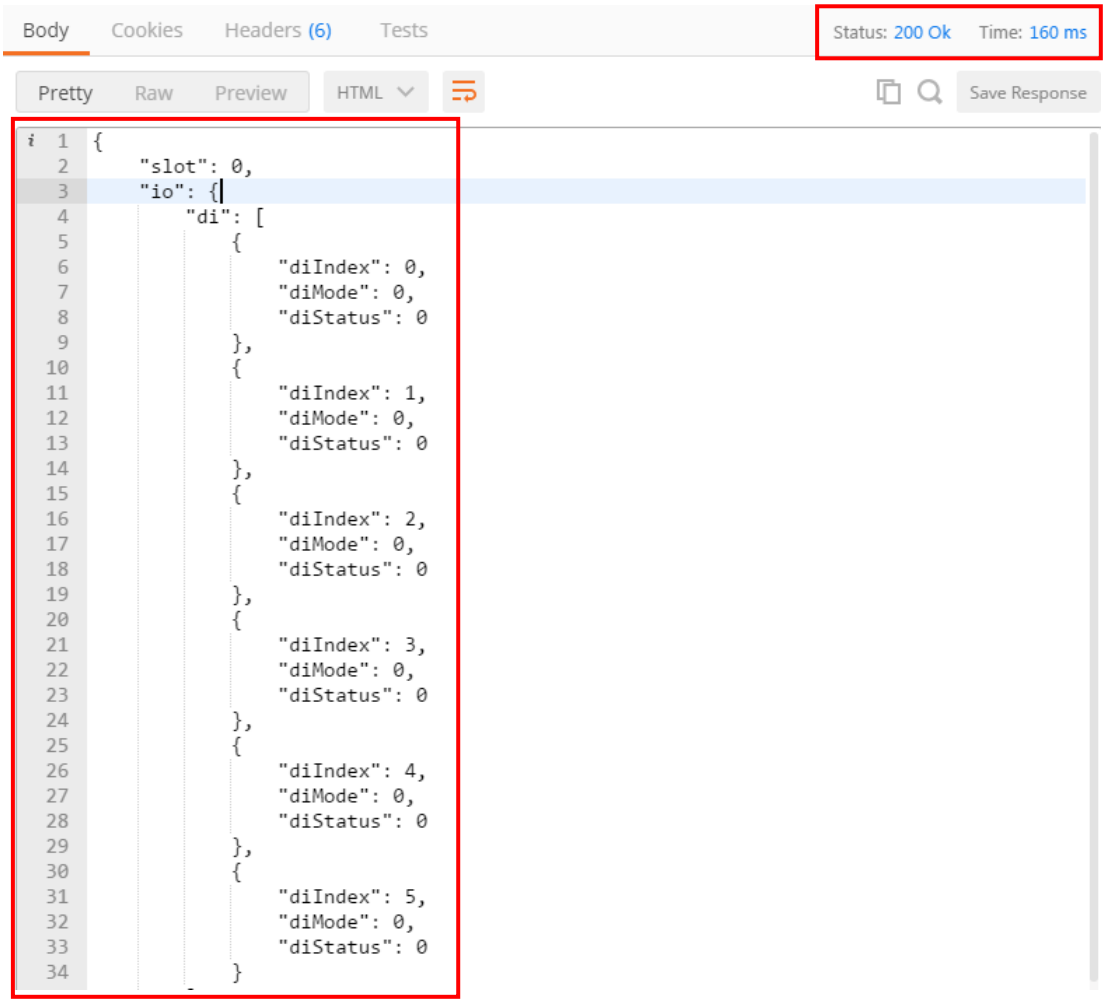
How to deploy GET and PUT methods with the ioLogik E1200

1. Refer to **Appendix C of the E1200 series user's manual: RESTful API Default Address Mappings** for API map details. The **GET** and **PUT** methods are supported.
   (We use the E1212 module with 8 DIs and 8 DIOs to illustrate.)

2. Enter the request URL based on the IP address, file type, and API map. For this example, the request URL should be written as follows:
   192.168.127.254/api/slot/0/io/di

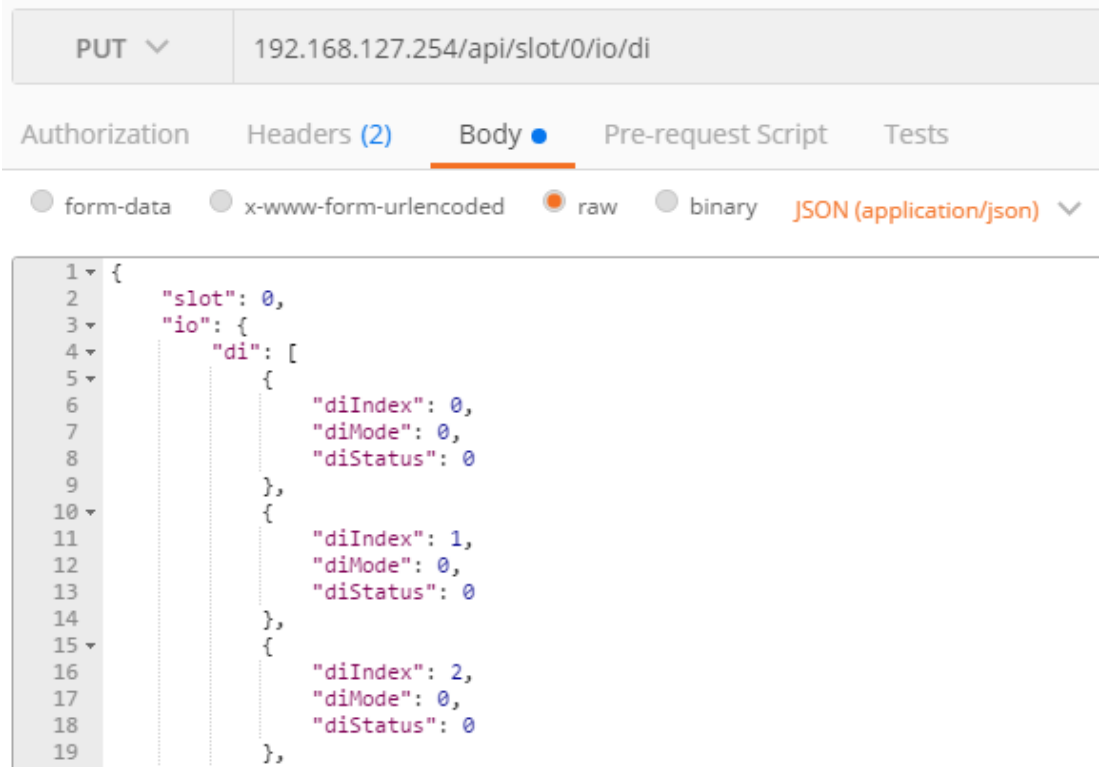


3. To use the GET method, click the **Send** button. You should see the status code, response time, and results.



4. To use the PUT method, first locate the **Body** tag and select **raw** to edit the contents. The format is the same as for the GET results.

Note: You must perform a GET first before performing a PUT.

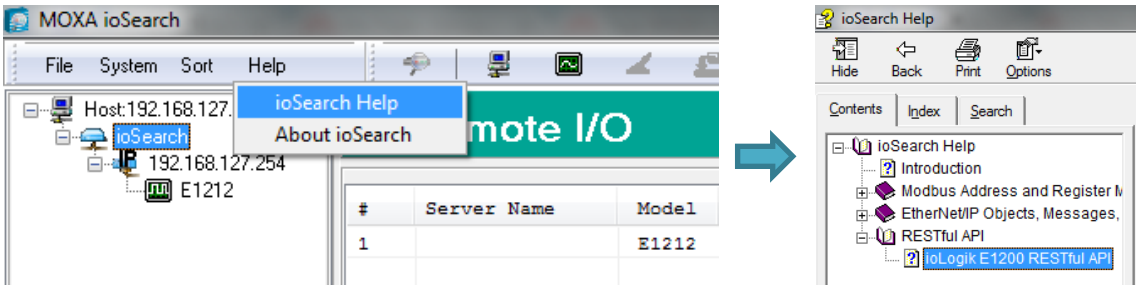


5. Click **Send**. You should see the status code, response time, and results.

Troubleshooting reference

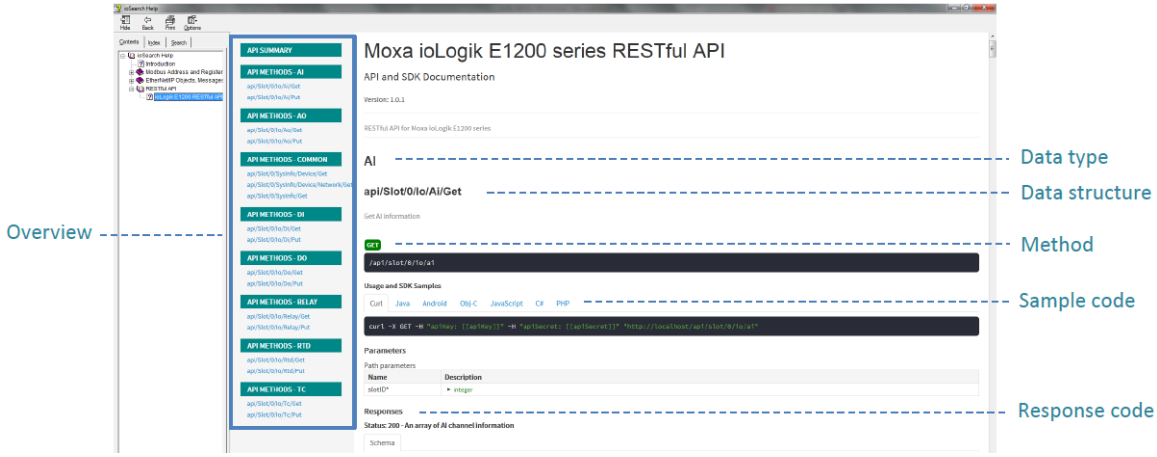
The ioLogik E1200 series supports various combinations of I/O points. To get detailed data structure information for troubleshooting purposes, use the ioSearch utility's built-in help tool.

Path: **Help → ioSearch Help → RESTful API → ioLogik E1200 RESTful API**



The help file includes information related to data type, data structure, supported methods, sample code (Curl, JavaScript, etc.), and response codes.

Note: Some of the sample code does not apply to the ioLogik E1200 series.

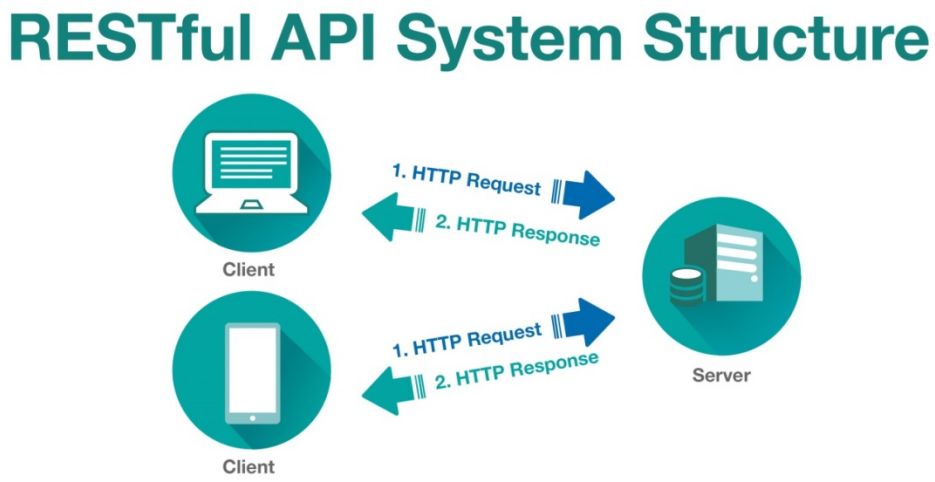


5. The difference between RESTful APIs and MQTT

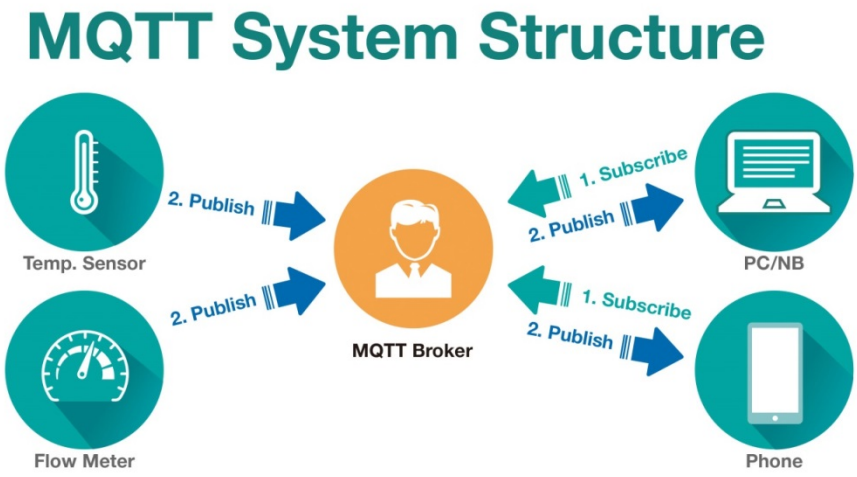
MQTT (message queue telemetry transport) is another protocol that could be used for some IIoT applications. MQTT is appropriate for low power, small bandwidth applications that use lightweight data packets. MQTT differs from RESTful APIs in that MQTT requires using a network packet broker to parse publish/subscribe requests, whereas RESTful API packets can get data directly from web services via HTTP commands. The characteristics of MQTT make it easy for programming and management. However, MQTT is not particularly suitable for local networks or small-scale communication between devices because of the network packet broker requirement. Network packet brokers have the disadvantage of being relatively costly and difficult to integrate. The simple application diagrams shown below provide a clear illustration of how implementing

MQTT requires a higher degree of complexity compared with a RESTful API implementation.

RESTful API system structure



MQTT system structure



6. Conclusion

Currently, a number of different protocols are available for implementing IIoT applications. However, before implementing your own IIoT application, research the various protocols to determine which protocol best suits your needs. Moxa's ioLogik E1200 series remote I/O products not only collect data from sensors and I/O points, they also support multiple protocols, including RESTful APIs, EtherNet/IP, SNMP, and Modbus, offering a great solution for both OT and IT experts. For more information about Moxa's durable industrial-grade products, please contact a Moxa sales representative or visit our global website at http://www.moxa.com/.