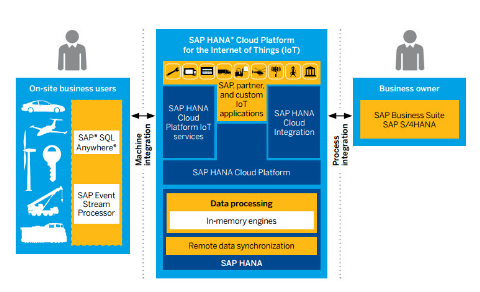
IoT1C06 – Simulate a Device with Node.js – On Premise

|  |  |
| --- | --- |
| **Product**  HANA Cloud Platform IoT Services  **Level**  Undergraduate/Graduate  Beginner  **Focus**  HANA Cloud Platform  **Author** Ross Hightower | MOTIVATION  This case describes registering for a developer trial accounts on SAP cloud services.  **PREREQUISITES**  None |



# Simulate a Device with Node.js

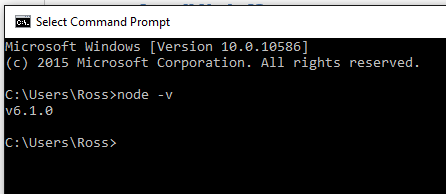
Node.js is a platform for creating network applications using JavaScript. Node makes it easy to create applications that can create and consume RESTful services so it is ideal to interacting with HANA OData services. In this case, you will create a node.js application that simulates a device. A later case uses node.js to interface between an actual device and IoT services.

## Install Node.JS

To install Node.JS navigate to the URL below:

[http:// nodejs.org/](http://nodejs.org/)

Click **Install** and follow the installation instructions**.** To test the installation open a command prompt (on Windows) or a terminal (on a Mac) and enter the command node –v.



## Create the Node.js App

Node applications are text files so, to create the application, you can use any text editor including Notepad. If you want a more advanced, free editor you can use and editor like the open source Atom (<https://atom.io/>).

Create a folder that will contain your application file and create a file called index.js. Copy the code shown below into the file and save it.

|  |
| --- |
| var http = require('http');  var host = **'db1.hana2.ucc.uwm.edu'**;  var port = **8002**;  var path = **'/gbi-student-366/iot/data/iot.xsodata/DATA'**;  var username = "**GBI\_366**";  var password = "**Password1**";  var token = new Buffer(username + ":" + password).toString("base64");  var currentTemp = 34.699999999999996;  var currentHum = 58.99999999999999;  var jsonData = {  "ID": 1,  "TEMPERATURE": currentTemp.toString(),  "HUMIDITY": currentHum.toString()  }  var strData = JSON.stringify(jsonData)  var options = {  host: host,  port: port,  path: path,  agent: false,  headers: {  'Authorization': 'Basic ' + token,  'Content-Type': 'application/json;charset=utf-8',  'Content-Length' : Buffer.byteLength(strData)  },  method: 'POST'  };  options.agent = new http.Agent(options);  callback = function(response) {  var body = '';  response.on('data', function (data) {  body += data;  console.log(body);  });  response.on('end', function () {  console.log("From HANA:", response.statusCode, body);  });  response.on('error', function(e) {  console.error(e);  });  }  var req = http.request(options, callback);  req.on('error', function(e) {  console.error(e);  });  req.shouldKeepAlive = true;  console.log(strData)  req.write(strData); |

Listing

Update the highlighted portions of the code for your circumstances.

## Send a Message

The message is the same format as in the previous case. You will find the message near the top of the application code.

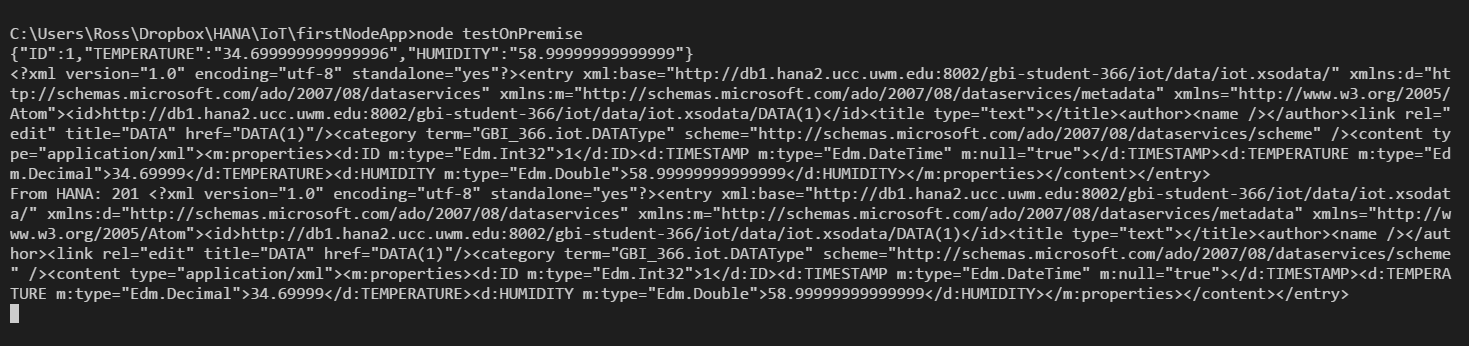
|  |
| --- |
| var currentTemp = 34.699999999999996;  var currentHum = 58.99999999999999;  var jsonData = {  "ID": 1,  "TEMPERATURE": currentTemp.toString(),  "HUMIDITY": currentHum.toString()  } |

To send the message, open a command prompt in the folder where you saved the application file.

|  |
| --- |
| On Windows: Find the folder in Windows Explorer then hold the shift key while you right-click the folder. Select Open Command Window Here  On a Mac: Open System Preferences and select Keyboard > Shortcuts > Services. Find "New Terminal at Folder" in the settings and click the box. Now, when you're in Finder, just right-click a folder and you're shown the option to open Terminal. When you do, it'll start right in the folder you're in. |

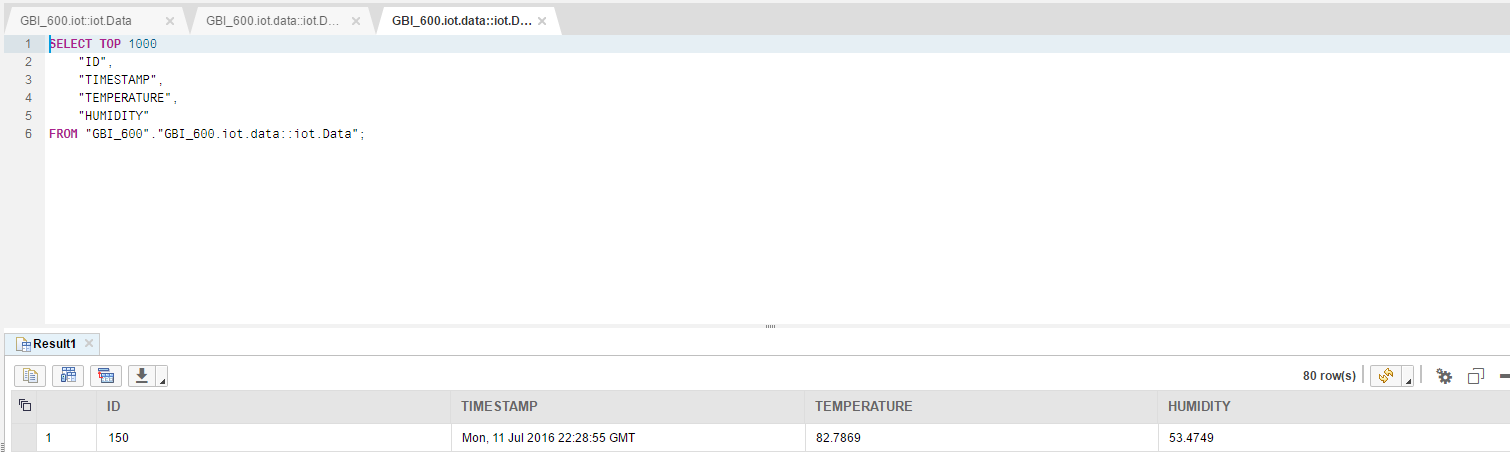
Enter the following command:

node index



The response is original message sent followed by the xml returned by HANA which is the record in the table in OData xml format.

Open the table in the Catalog editor and click **Open Content** to see the data.



You can also use the OData service to retrieve the data in a browser.



## Simulate a Continuous Feed

Often, a device would send a continuous stream of data so in this section we will simulate that. Create a new application file called deviceSim.js and paste the following code into it.

|  |
| --- |
| var http = require('http');  var host = **'db1.hana2.ucc.uwm.edu'**;  var port = **8002**;  var path = '**/gbi-student-366/iot/data/iot.xsodata/DATA**';  var username = "**GBI\_366**";  var password = "**Password1**";  var token = new Buffer(username + ":" + password).toString("base64");  var options = {  host: host,  port: port,  path: path,  agent: false,  headers: {  'Authorization': 'Basic ' + token,  'Content-Type': 'application/json;charset=utf-8',  'Content-Length' : 0  },  method: 'POST'  };  options.agent = new http.Agent(options);  callback = function(response) {  var body = '';  response.on('data', function (data) {  body += data;  console.log("Success...");  });  response.on('end', function () {  console.log("From HANA:", response.statusCode);  });  response.on('error', function(e) {  console.error(e);  });  }  //Seed values for temperature and humidity trends  var sequence = 0;  var currentTemp = 20;  var tempTrend = 2;  var currentHum = 20;  var humTrend = 5;  //Function to send a message  sendMessage = function(){  //Compute temperatue and humidity values  currentTemp += tempTrend + Math.floor(Math.random()\*2-2)/10;  currentHum += humTrend + Math.floor(Math.random()\*2-2)/10;  var jsonData = {  "ID": 1,  "TEMPERATURE": currentTemp.toString(),  "HUMIDITY": currentHum.toString()  }  var strData = JSON.stringify(jsonData);  console.log("Sending " + strData);  options.agent.options.headers['Content-Length'] = Buffer.byteLength(strData);  var req = http.request(options, callback);  req.on('error', function(e) {  console.error(e);  });  req.shouldKeepAlive = true;  req.write(strData);  }  //Send a message every 1 minute  setInterval(sendMessage, 10000); |

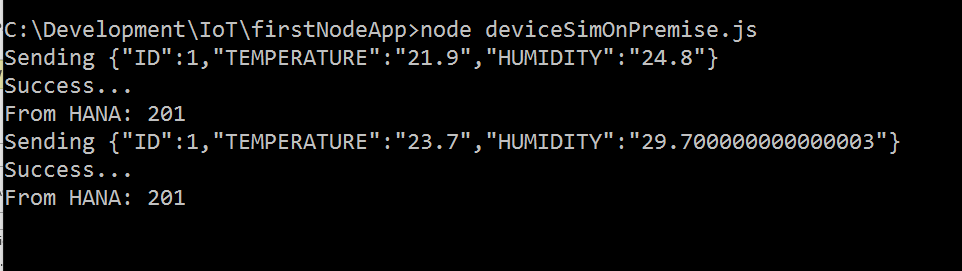
Listing 2

The setInterval() function at the bottom of the code will call the sendMessage function once a minute. In the sendMessage() function, a time interval, current temperature and current humidity are calculated and the message is sent. The interval is in milliseconds. This will provide some data for the next case in which we create an application to view the data.

Execute the application using the following command at the command prompt:

node deviceSim

The program sends a message to HANA every minute. To end the application use control-C (on Windows) or command-C (on a Mac).



Let the program run so that you have 10 – 20 messages.