**IoT C SDK Compiling and Practicing on Linux**

* **Preconditions to compile IOT Client SDK on Linux**

There are 3 preconditions to complie IOT Client SDK on Linux:

* Add Dependencies: build-essential curl libcurl4-openssl-dev libssl-dev uuid-dev
* gcc comlier version is higher than V4.9
* CMake version is higher than V3.2

Add Dependencies:

sudo apt-get install -y build-essential curl libcurl4-openssl-dev libssl-dev uuid-dev

To verify the version of gcc,you could use gcc --version，if the version is lower than V4.9,you should upgrade gcc version with the following ways:

sudo add-apt-repository ppa:ubuntu-toolchain-r/test  
sudo apt-get update  
sudo apt-get install gcc-4.9 g++-4.9  
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.9 60 --slave /usr/bin/g++ g++ /usr/bin/g++-4.9

As well, check CMake version by cmake –versio. If lower than V3.2, pls upgrade with the version with the following ways:

sudo add-apt-repository ppa:george-edison55/cmake-3.x  
sudo apt-get update

* **Compiling IOT C SDK on Linux devices**

When conpiling SDK through ./build.sh scripts, you could control the result by different inout parameters as bellow implications:

| **parameter** | **implication** |
| --- | --- |
| --run-e2e-tests | End to end test |
| --run-unittests | Unit test |
| --run-longhaul-tests | Longhaul test |
| --no-amqp | Don’t build AMQP protocol parts |
| --no-http | Don’t build HTTP protocol parts |
| --no-mqtt | Don’t build MQTT protocol parts |
| --no\_uploadtoblob | Data will not upload to Blob storage |
| --no-make | No Make and only generating codes without compiling |
| --use-websockets | Open AMQP over WebSockets option |
| --toolchain-file | Toolchain folder for cross compiling |
| --install-path-prefix | make install prefix |
| --build-python python | Python C package (need boost) |
| --build-javawrapper | Java C wrapper |
| -rv, --run\_valgrind | Loading valgrind to run ctest unit test |
| --no-logging | No logging |

In project folders, you could find the relevant samples files in amqp,mqtt and seializer folders,which include the sample source codes for testing in relevant scenarios. Before using these codes,IoT device connection string is needed.

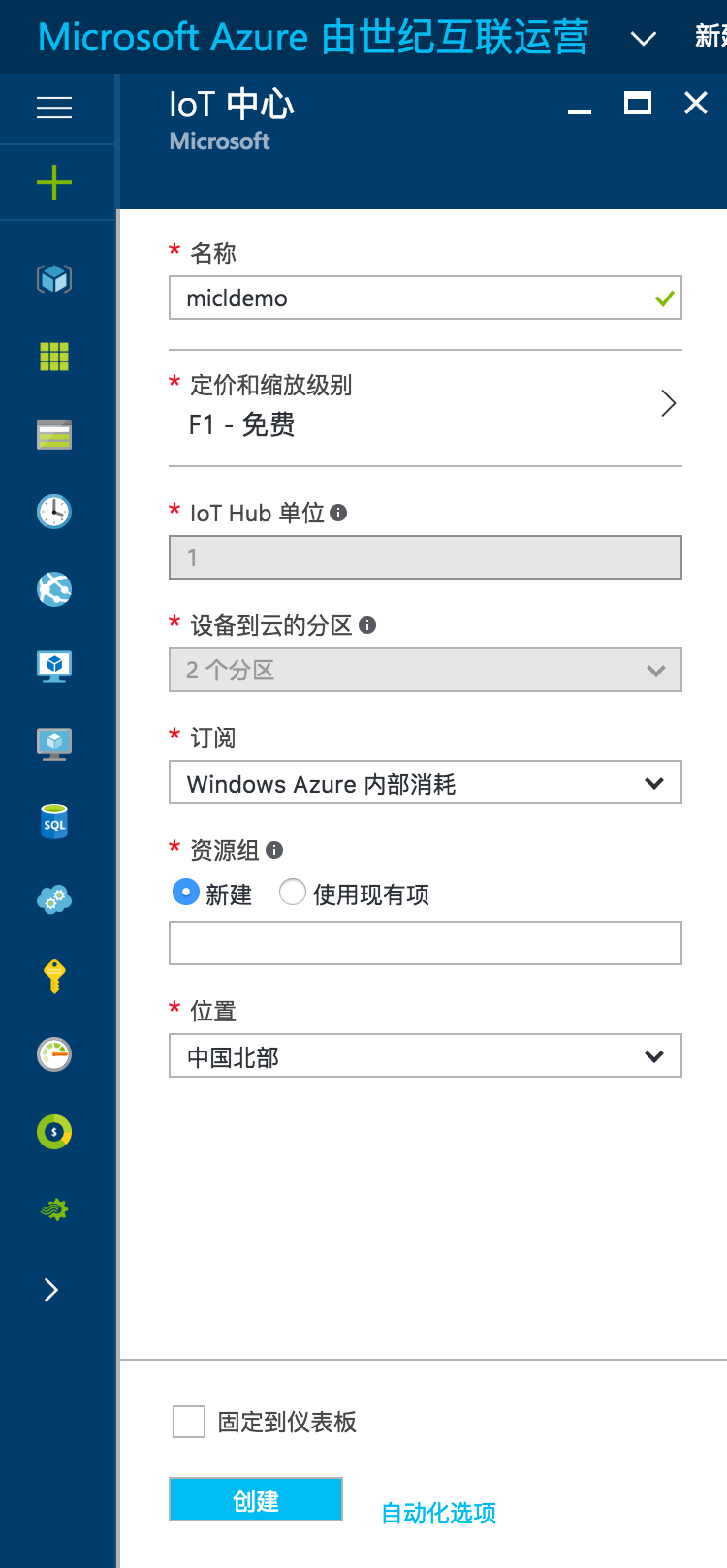
Here, the sample is samples/simplesample\_amqp under serializer folder. In the source code,there is no device connection string and need to update a valid connect code manually in .c source code as below:

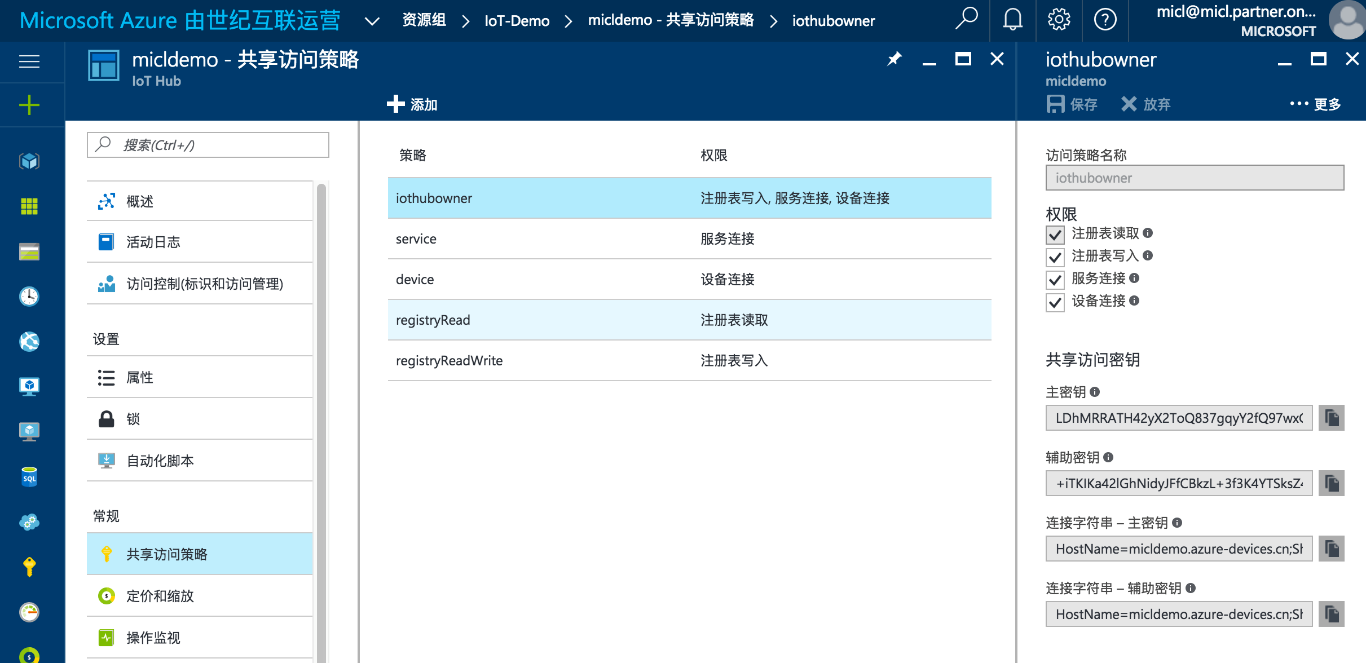
[](https://github.com/micli/learning/blob/master/images/IoT-C-SDK/device-connstr.png)

Before running, we must assign a valid data to connectionString and pls ref to the next sector to know how to create a valid device connection string.

**Get IoT Hub Device Connection String**

Firstly, we need to create a IoT Hub Service in Azure management portal as below by click”+” and choose IoT-> IoT Hub:

[](https://github.com/micli/learning/blob/master/images/IoT-C-SDK/IoTHub-Create.png)

When the service is ready, copy the primary key under “Shared access policie” ->"iothubowner"->"Connection string—primary key” in created IoT hub service, as shown in the below figure： [](https://github.com/micli/learning/blob/master/images/IoT-C-SDK/IoTHub-Copy-String.png)

Now, we could use command Prompt to create the device connection string. In IoT SDK, there is a Node.js tool to support developers to manage the Azure IoT Hub services, which is named iothub-explorer. We could install it by:

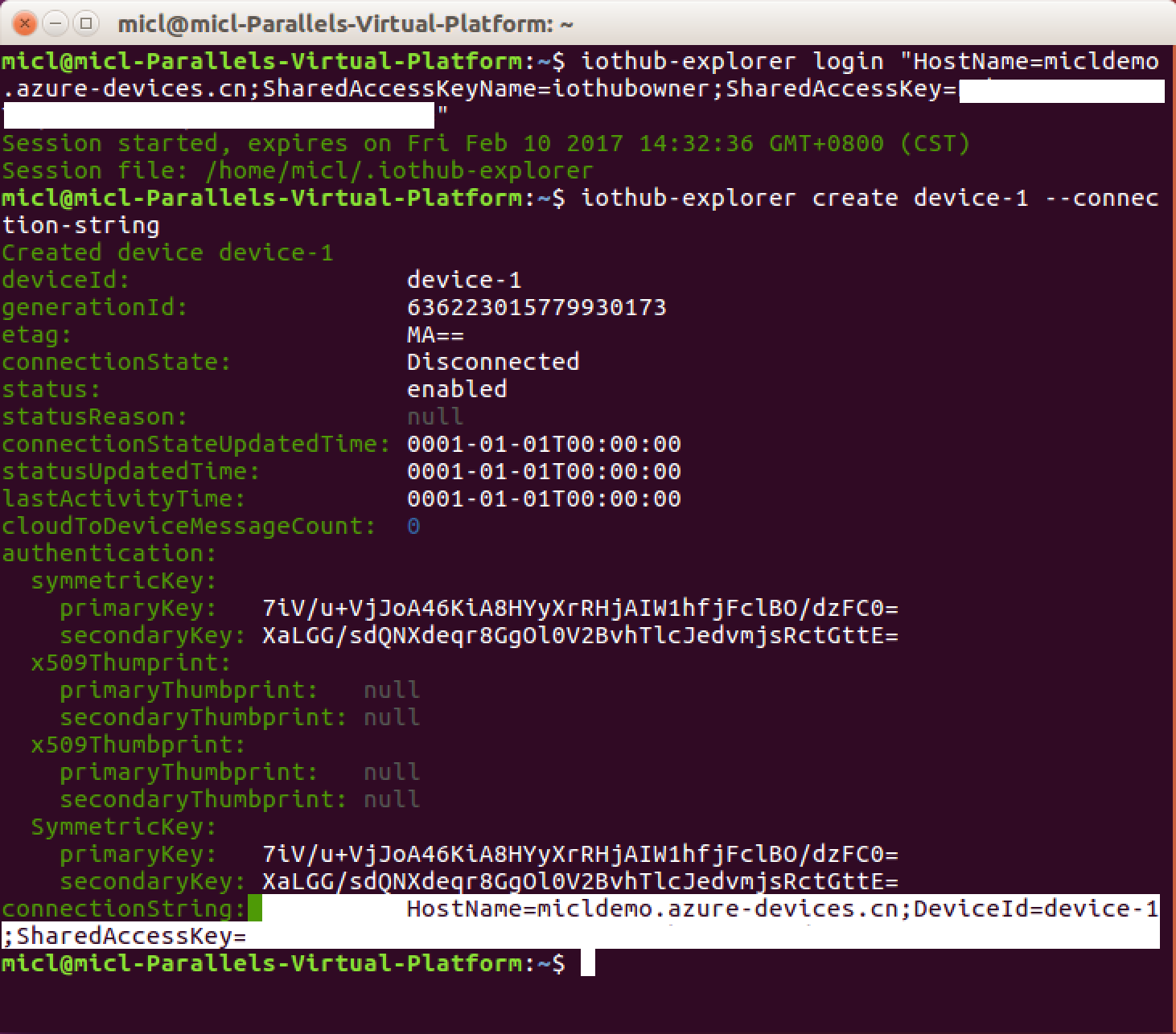
sudo npm install -g iothub-explorer

if meet an error "/usr/bin/env: node: No such file or directory", pls run the following command,because Node.js have a updated folder names in latest version:

sudo ln -s /usr/bin/nodejs /usr/bin/node

Then, we could use the following steps to create IoT device connection string:

iothub-explorer login "[IoT Hub service connection string]"  
iothub-explorer create <device\_ID> --connection-string

[](https://github.com/micli/learning/blob/master/images/IoT-C-SDK/iothub-explorer-create.png)

* **Initializing IOT Hub Client on Linux**

// 初始化IoT Hub Client

IOTHUB\_CLIENT\_HANDLE InitIoTHubClient(const char\* connectionString)

{

printf("Starting the IoTHub client simple sample...\r\n");

if(NULL == connectionString)

{

printf("Invalid connection string...\r\n");

return NULL;

}

IOTHUB\_CLIENT\_HANDLE iotHub = NULL;

if (0 != platform\_init())

{

printf("Failed to initialize the platform.\r\n");

}

else if(NULL == (iotHub = IoTHubClient\_CreateFromConnectionString(

connectionString, AMQP\_Protocol)))

{

printf("Failed to creating the IoT Hub on Azure, please check connection string.\r\n");

}

serializer\_init(NULL);

return iotHub;

}

* **Sending Data to IOT Hub**

Before sending data to IOT Hub through IOT C SDK on a Linux device,we need to serialize the message with Macro SERIALIZE in IOT C SDK,which will convert the data and object in device memory to JASON byte stream. Pls note: we nned to release the buffer when finished the sending to avoid the memory leak.

// 序列化消息数据

unsigned char\* destination = NULL; // 序列化结果缓冲区指针

size\_t size; // 序列化结果的大小

if (SERIALIZE(&destination, &size, msg->DeviceId, msg->Data) != CODEFIRST\_OK)

{

printf("Serialize message failed.\r\n");

continue;

}

// 发送数据到IoT Hub 服务端

SendMessageToIoTHub(handle, destination, size);

// 发送数据到IoT Hub

void SendMessageToIoTHub(IOTHUB\_CLIENT\_HANDLE iothubClient, unsigned char\* buffer, size\_t size)

{

if(NULL == iothubClient)

{

printf("Invalid IoT Hub handle...\r\n");

return ;

}

if(NULL == buffer)

{

printf("Invalid message buffer...\r\n");

return ;

}

IOTHUB\_MESSAGE\_HANDLE messageHandle = IoTHubMessage\_CreateFromByteArray(buffer, size);

if (messageHandle == NULL)

{

printf("unable to create a new IoTHubMessage\r\n");

}

// 利用SDK API 异步发送消息

IoTHubClient\_SendEventAsync(iothubClient, messageHandle, MessageSentCompleted, messageHandle);

printf("The message:\r\n %s \r\n has been sent.\r\n", buffer);

// 释放缓冲区内存

free(buffer);

buffer = NULL;

}