



**Software Version: 2.0.0**

---

## **Installation Guide v2.0.0**

Document Release Date: July 2017

Software Release Date: July 2017



# Copyright and Disclaimer of Liability

This document may contain technical inaccuracy or type errors, and the author does not have any responsibility on this matter.

The contents of this document can be changed or added regularly, and the relevant corrected version will be added to the document under the title named “New Edition” in consecutive order. The product or program mentioned in this document may be changed or modified without any prior notice.

The source code of Mobius is distributed according to the license policy below.

- The open source code shared by OCEAN (Open alliance for iot stANdard) is distributed based on the 3-clause BSD-style license. While maintaining copyright header in the source code file, the open source code can be used freely in the purpose of commercial or non-commercial systems.
- License of OCEAN does not force users to share the developed source code with others. The ownership of the developed source code belongs to the developer and (s)he has no obligation to share it.
- Anyone can contribute to improvement of the open source environment of OCEAN. If so, the developed source code should follow the license policy of OCEAN.

Copyright (c) 2017, OCEAN  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# Table of Contents

1. Mobius.....	5
1.1. Introduction .....	5
1.2. Mobius Platform .....	6
1.2.1. Introduction.....	6
1.2.2. Mobius Platform Functionalities.....	7
1.2.3. Mobius Platform Components.....	7
1.2.4. Mobius Platform S/W Architecture.....	8
1.2.5. Mobius Platform Source Code Directory .....	9
2. Mobius Server Platform Installation.....	11
2.1. Introduction .....	11
2.2. Pre-requisites Installation (Windows) .....	12
2.2.1. MySQL Installation.....	12
2.2.2. MQTT Server Installation.....	18
2.2.3. Node JS Installation.....	20
2.3. Mobius Installation .....	23
3. Running Mobius Server.....	25
3.1. Runtime Environment Configuration .....	25
3.2. Test .....	26
3.2.1. oneM2MBrowser .....	26
3.2.2. Postman.....	26

# 1. Mobius

## 1.1. Introduction

Mobius platform is a oneM2M compliant IoT server platform designed to offer diverse IoT services and functionalities such as smart devices control and management, and user management etc. Users can experience the IoT services through user IoT applications. Figure 1 shows the architecture of Mobius platform structured with oneM2M entities and reference points through which any two entities can communicate with each other.

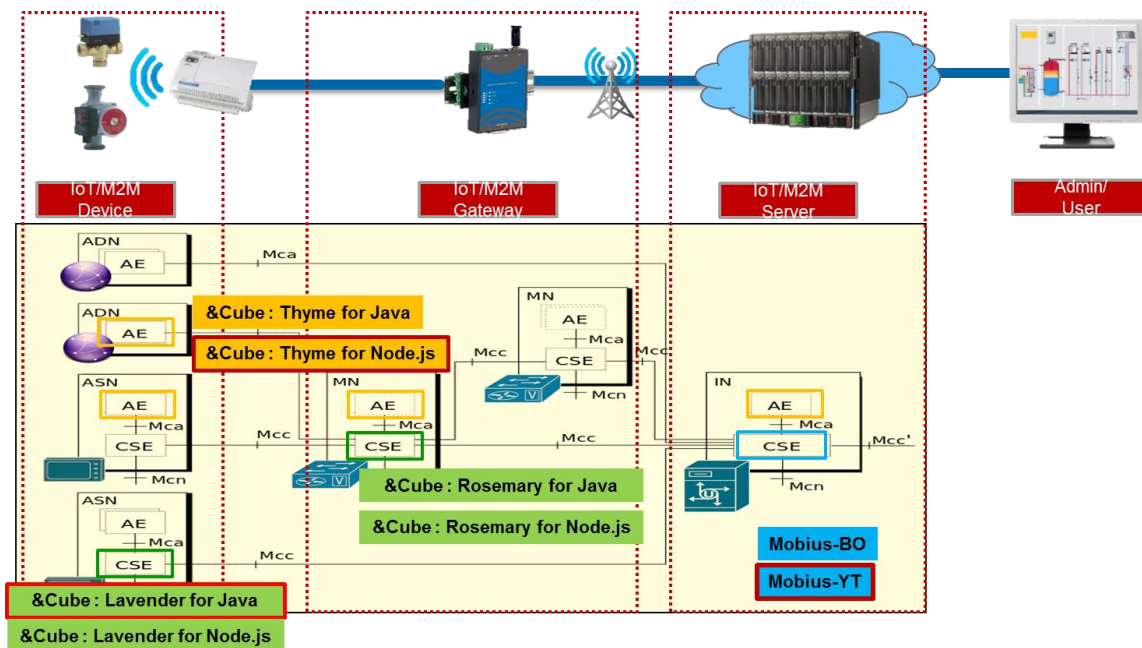


Figure 1 oneM2M compliant Mobius platform architecture

Mobius platform provides a series of REST APIs for protocol HTTP, MQTT, CoAP and WebSocket to connect IoT devices together and implements interaction between each other through Mobius as shown as 오류! 참조 원본을 찾을 수 없습니다.. APIs for creation and retrieval of oneM2M resources are also provided to simplify the procedures of data management.

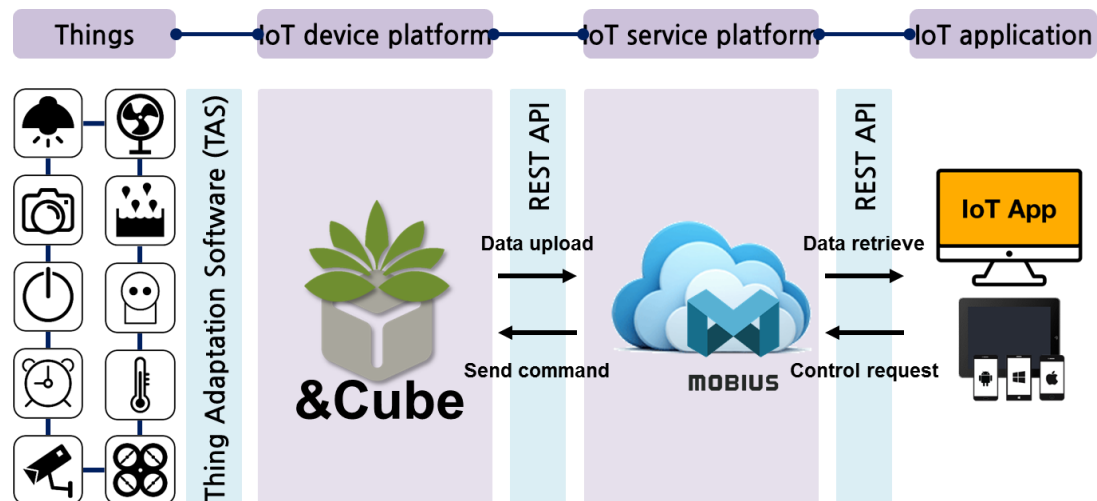


Figure 2 Interconnection between Mobius and IoT devices

## 1.2. Mobius Platform

### 1.2.1. Introduction

Mobius is a middleware server platform that connects diverse IoT devices through physical communication medias and creates virtual representations (oneM2M resources) for each IoT device to enable the interactions between each other as well as the communication between devices and IoT applications. In this way, Mobius provides an open environment and APIs for users to interconnect their own devices together and develop user specific IoT services to build an IoT ecosystem. 오류! 참조 원본을 찾을 수 없습니다. shows the components of Mobius platform i.e. IoT devices, Mobius server and IoT applications.

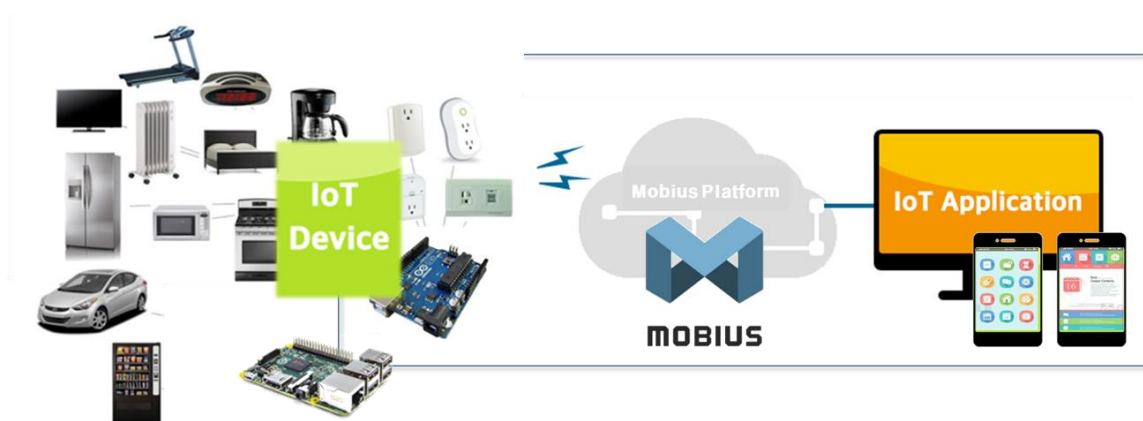


Figure 3 Mobius platform components

The IoT server platform can be implemented using diverse programming languages and the Mobius server platform is developed using Node JS. In addition, Mobius server platform uses Node JS express modules which provides diverse modules for developers including HTTP,

XML etc. instead of Node JS express framework.

The Mobius server platform is compliant to oneM2M standards and supports HTTP, CoAP, MQTT and WebSocket bindings specified in oneM2M standards. The Mobius server platform implements oneM2M Infrastructure Node Common Services Entity (IN-CSE) with structured resource architectures and provides IoT services through RESTful APIs. The Mobius server platform uses MySQL DBMS for resources storage.

### 1.2.2. Mobius Platform Functionalities

Mobius Platform works as the middle bridge to enable the communications and interactions between IoT devices and IoT applications as following:

- ① The Mobius server accepts the uploaded data from authenticated IoT devices and stores the data into the MySQL DB;
- ② The authenticated IoT applications can retrieve the data stored in Mobius server using Mobius open APIs for monitoring purpose;
- ③ The authenticated IoT applications are able to control remotely IoT devices that have already registered with Mobius server by sending control commands included in *<contentInstance>* resource. The Mobius server then executes control commands to the target devices.

The call flows as shown as Figure 4 indicates the communications and interactions among devices, IoT applications and Mobius server.



Figure 4 Mobius interaction with IoT devices and applications

### 1.2.3. Mobius Platform Components

Mobius server consists of HTTP, CoAP, MQTT, WebSocket server and MySQL DBMS while IoT applications implement HTTP, CoAP and/or MQTT clients in order to communicate with Mobius server.

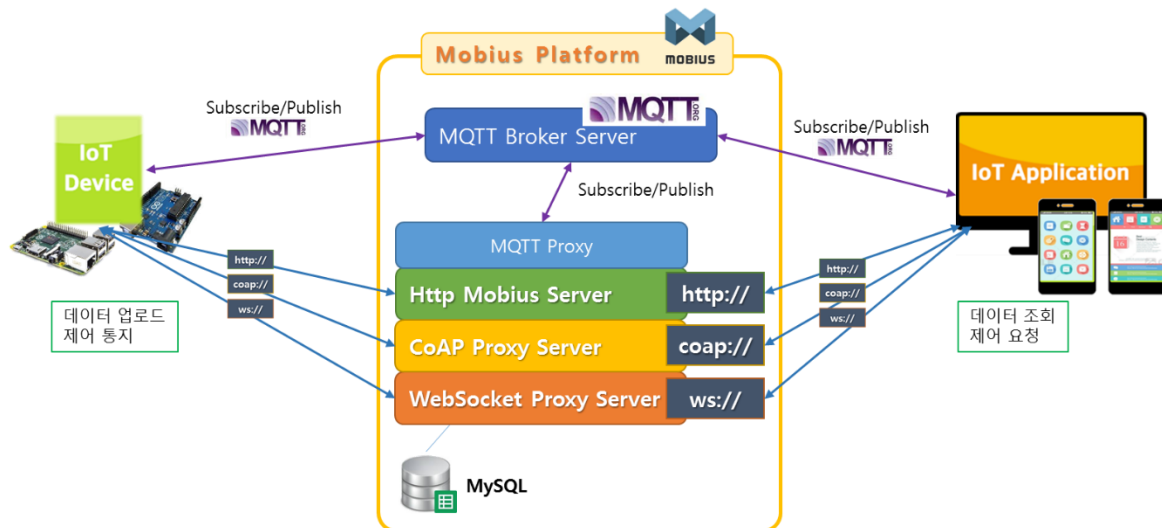


Figure 5 Mobius platform components

### 1.2.4. Mobius Platform S/W Architecture

For protocol binding support, Mobius has MQTT broker, CoAP server, WebSocket server that is bound to HTTP server internally. For example, MQTT protocol binding is supported by implementing MQTT to HTTP proxy. CoAP and WebSocket are designed in the same way. Mainly it consists of requester and responder. The requester contains the DB access component. Every HTTP request is go through requester component, parser, actor and then create SQL query to data access (e.g. retrieval, discovery) with DB connector. When it gets access result, the responder creates the response in XML, JSON or CBOR serialization.

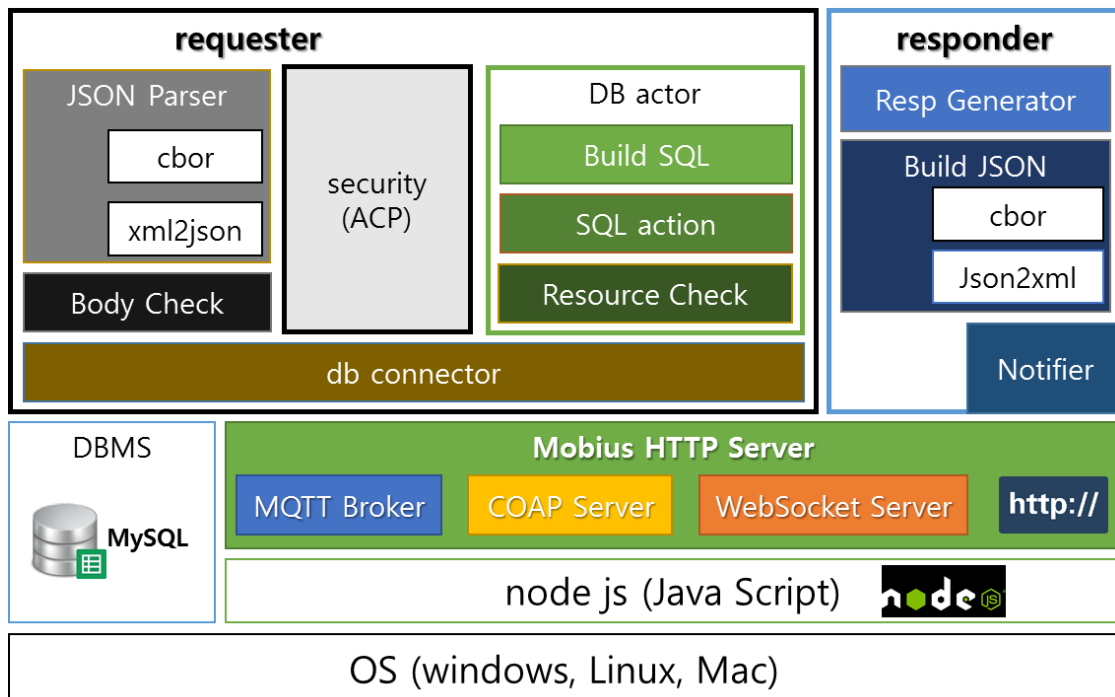


Figure 6 Mobius platform S/W architecture



### 1.2.5. Mobius Platform Source Code Directory

The figure below shows the Mobius Node JS source directory. For the detailed functions and roles for each Node JS file, please refer to the Table 1.

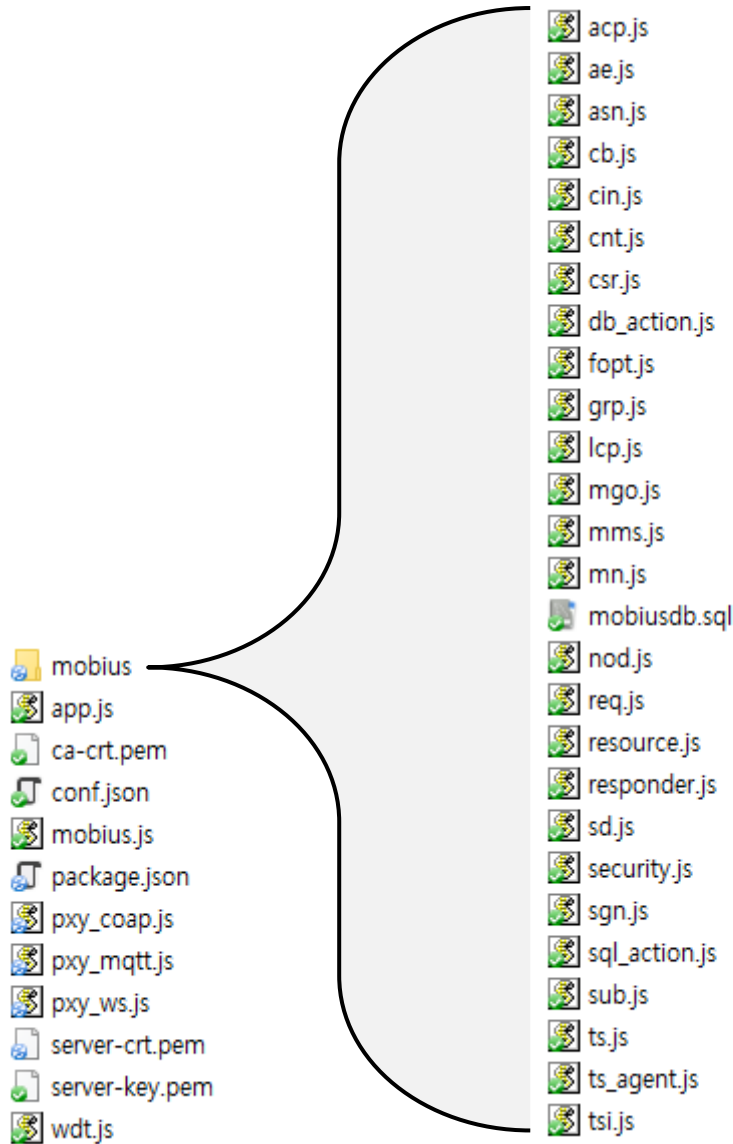


Figure 7 Mobius Node JS source code directory

Table 1 Function Reference Table for Node JS Files

Source File	Role and Function
<b>mobius.js</b>	This file initiates Mobius server and helps loading main Node JS files. It also contains configuration parameters for Mobius server such as <i>defaultbodytype</i> indicating the serialization, <i>usecsetype</i> indicating CSE type (either IN-CSE, MN-CSE or ASN-CSE), <i>usecsebase</i>

	indicating CSEBase name, <i>usecseid</i> indicating CSEID, <i>usedbhost</i> indicating the host address, and <i>usedbpass</i> indicating the password for MySQL etc. Users can modify those configuration parameters.
<b>app.js</b>	<p>This file acts as role of flow router and it is the main code running Mobius-YT server.</p> <ul style="list-style-type: none"> <li>① It handles initial processing of received packets.</li> <li>② It initiates HTTP server with 'listening' mode to wait for HTTP requests target to the Mobius HTTP server.</li> <li>③ It handles the parsing of URL of packets and evaluate the correctness of the request body resulted of parsing. It then sends the request to resource.js to continue the processing if the request is valid one, otherwise throws exceptions.</li> </ul> <p>This file also implements the server clustering algorithms to improve the performance of HTTP server.</p>
<b>mobius/resource.js</b>	<p>It is core file to process the CREATE, RETRIEVE, UPDATE, DELETE, NOTIFY and DISCOVERY operations for oneM2M resource primitives.</p> <p>This file undertakes the processing of parsed request URI and request body received from app.js according to corresponding operation. It converts the data into a format to process the data and connect to mysql database.</p> <p>The mysql database is initialized and handled by db_action.js and sql_action.js module.</p>
<b>mobius/responder.js</b>	<p>It is responsible for handling the response process.</p> <p>It receives processing results from app.js and resource.js modules and generates responses from the processing results following format requested by originator either XML or JSON serialization.</p>
<b>mobius/db_action.js</b>	This file contains parameters used to connect and access to the database and parameters for returning response results from the database.
<b>mobius/sql_action.js</b>	This file contains functions to receive data and parameters required for a series of database operations and functions to call db_action.js module to return data from database.
<b>mobius/sgn.js</b>	<p>This module file is responsible for checking the existence of &lt;subscription&gt; child resource under the requested target resource.</p> <p>If it exists, the module checks the event type and retrieves the field value of attribute <i>notificationUri</i>. Then it generates and sends a notification message to the address indicated by field value of attribute <i>notificationUri</i>.</p>
<b>mobius/security.js</b>	<p>This file contains functions to check the access privileges of originators for a requested resource when the resource applies with &lt;accessControlPolicy&gt; resource.</p> <p>The originator ID is abstracted from request header field of X-M2M-Origin. The process checks the access right of current originator ID to a target resource and responds with ACCESS_DENIED error when the originator ID has no privileges to access to the resource.</p>
<b>mobius/fopt.js</b>	<p>This file contains function to handle operations target to &lt;fanOutPoint&gt; virtual resource of a &lt;group&gt; resource.</p> <p>It transmits operation request to all group members contained in &lt;group&gt; resource and aggregates responses from all group members into an aggregated response.</p>

<b>mobius/ts_agent.js</b>	This file contains functions to manage <i>&lt;timeSeriesInstance&gt;</i> resource. It monitors the missing data in the <i>&lt;timeSeriesInstance&gt;</i> resource and then stores the missing data.
<b>pxymqtt.js</b>	<p>This file contains functions implementing mqtt proxying function to handle mqtt protocol messages in http protocol manner as following procedures:</p> <ul style="list-style-type: none"> <li>④ It creates a oneM2M mqtt topic with configuration information of Mobius server and then subscribe to the topic to receive mqtt requests targeted to this topic later;</li> <li>⑤ Whenever receiving mqtt protocol messages, it generates and sends a http request wrapping mqtt request message to Mobius server and waits for http response from Mobius server. It then abstracts http response and generates a mqtt response message correspondingly.</li> <li>⑥ It responds with mqtt protocol message.</li> </ul>
<b>mobius/acp.js</b>	It contains functions to parse accessControlPolicy request. The accessControlPolicy resource is used to manage the access control privileges to resources that apply with access control policy
<b>mobius/ae.js</b>	It contains functions to parse AE request
<b>mobius/cb.js</b>	It contains functions to parse CSEBase request. The CSEBase resource contains configuration information of Mobius server
<b>mobius/cin.js</b>	It contains functions to parse contentInstance request
<b>mobius/cnt.js</b>	It contains functions to parse container request
<b>mobius/csr.js</b>	It contains functions to parse remoteCSE request
<b>mobius/grp.js</b>	It contains functions to parse group request
<b>mobius/lcp.js</b>	It contains functions to parse locationPolicy request
<b>mobius/mms.js</b>	It contains functions to parse multimediaSession request
<b>mobius/mn.js</b>	It contains functions to parse MN-CSE resource request including implementation of registration function with IN-CSE. It is used to drive nCube:rosemary.
<b>mobius/sd.js</b>	It contains functions to parse semanticDescriptor request
<b>mobius/sub.js</b>	It contains functions to parse subscription request.
<b>mobius/ts.js</b>	It contains functions to parse timeSeries request.
<b>mobius/tsi.js</b>	It contains functions to parse timeSeriesInstance request.

## 2. Mobius Server Platform Installation

### 2.1. Introduction

Mobius Yellow Turtle server platform deploys MySQL DBMS as a database so MySQL is required to be installed as a basic and then MQTT server and Mobius server have to install one by one. Note that it is unnecessary to install MQTT server if MQTT protocol and MQTT related functions are not supported.



Figure 7 Mobius server installation procedures

## 2.2. Pre-requisites Installation (Windows)

### 2.2.1. MySQL Installation



Figure 8 MySQL Introduction

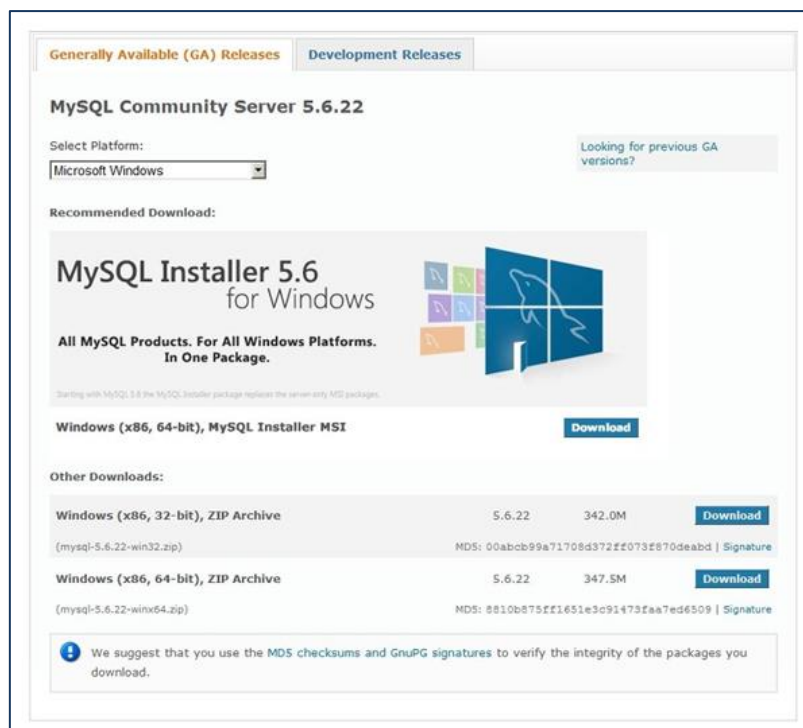
Mobius server platform stores the data uploaded from devices into MySQL database. MySQL is an open source RDBM and the installation procedures are introduced as following:



**Figure 9 MySQL community server**

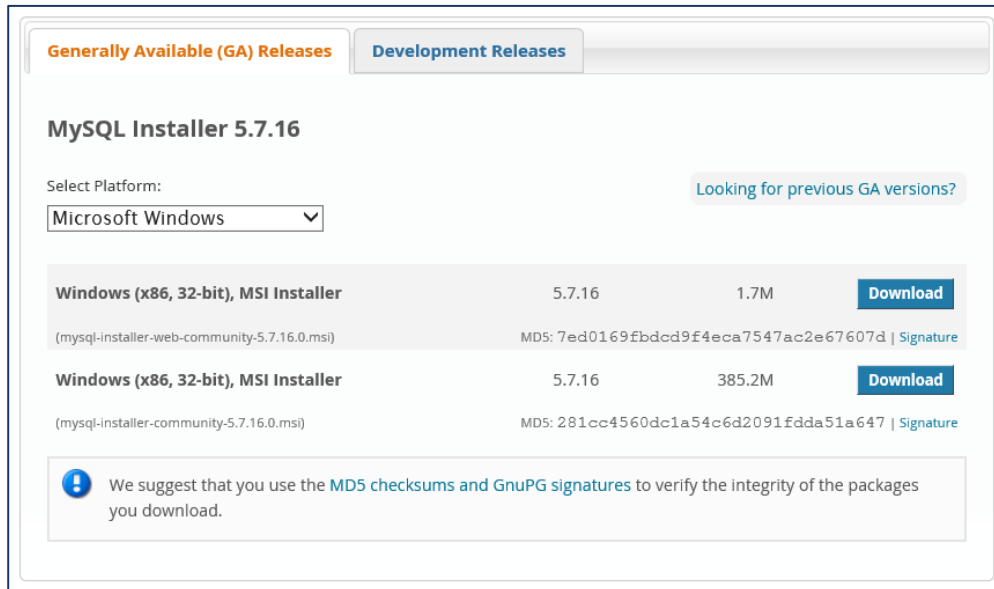
Different versions of MySQL are provided for download at below link:

<http://dev.mysql.com/downloads/mysql>.



**Figure 10 Download MySQL Installer**

There is a MySQL community server version for free downloading. Users could also select to download MySQL Enterprise Edition or MySQL Cluster CGE version which need to buy licenses before starting use. User can choose to install a light MSI installer which installs all required MySQL packages online or a standalone version that contains all required packages. If users prefer to install using zip archive, you can download and install as shown as Figure 11.



**Figure 11 Select MySQL Installer for downloading**

Here we demonstrate downloading the free version MySQL community server and you can find it from the top 'Downloads' menu tab and we select to install a Windows 64bit version MySQL through MySQL installer as shown as Figure 12.

The MySQL can be easily installed following the procedures as depicted as Figure 13.

- ① Read and accept the license agreement, then
- ② Choose a Setup type: A list of setup types "Developer Default, Server only, Client only, Full or Custom" are available to install. Here we recommend to select "Custom" and then select only two packages "MySQL Server" and "MySQL Workbench" to install.
- ③ Install automatically the selected packages and after successful installation, then
- ④ Configure the MySQL runtime environment shown as Figure 14.
  - i. Configure the machine type to either 'Development Machine', 'Server Machine', or 'Dedicated Machine'. Here we select option 'Development Machine' to configure.
  - ii. Set 'root' account passwords and optionally create users with different privileges to access to the MySQL.
  - iii. Configure MySQL server as a Window Service and finally apply the user configurations to MySQL server.
  - iv. Input the root password and check the connection status to the configured MySQL as shown as Figure 15

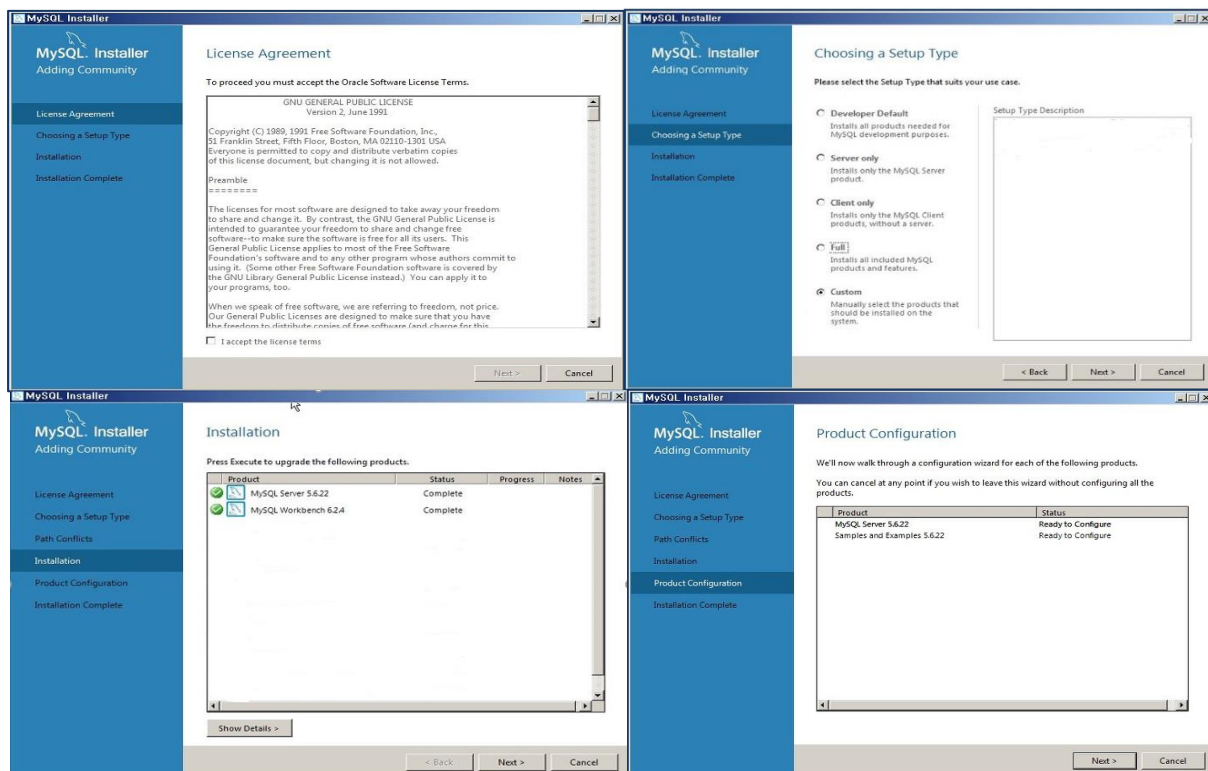


Figure 12 MySQL installation procedures

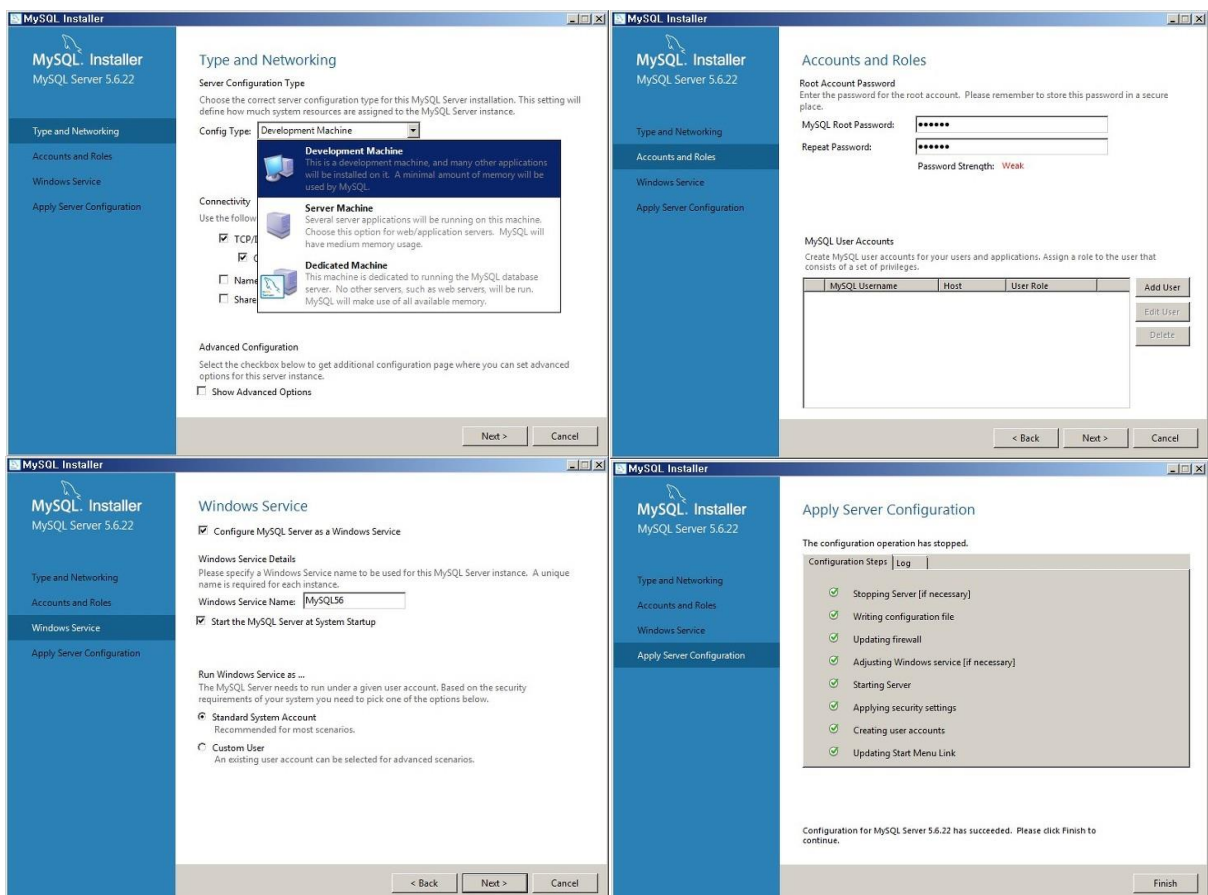
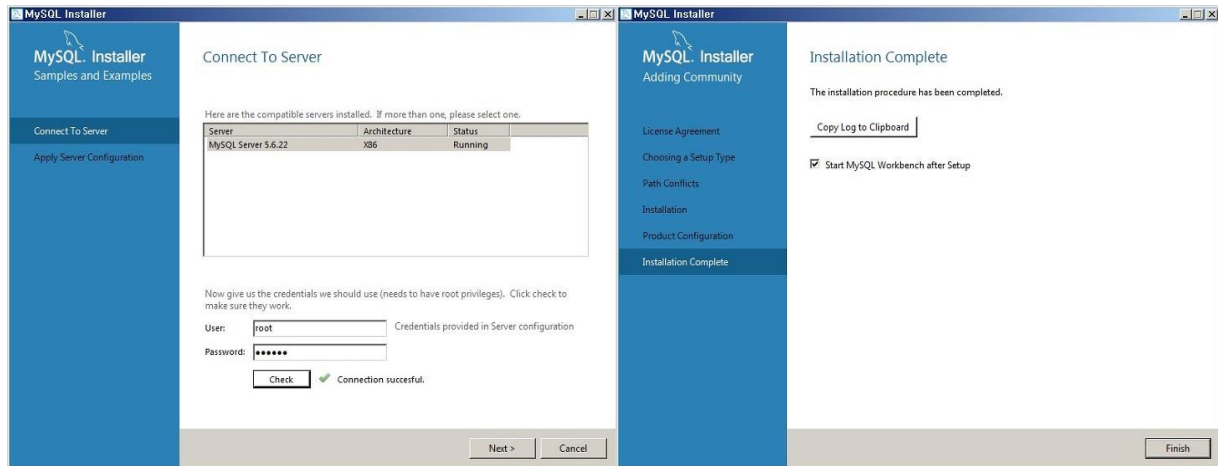


Figure 13 MySQL Runtime configurations





**Figure 14 Connect to MySQL server for test**

Run MySQL workbench and create a MySQL connection and then connect to MySQL server with root password as shown as Figure 16.



**Figure 15 Create a MySQL connection and connect to the server**

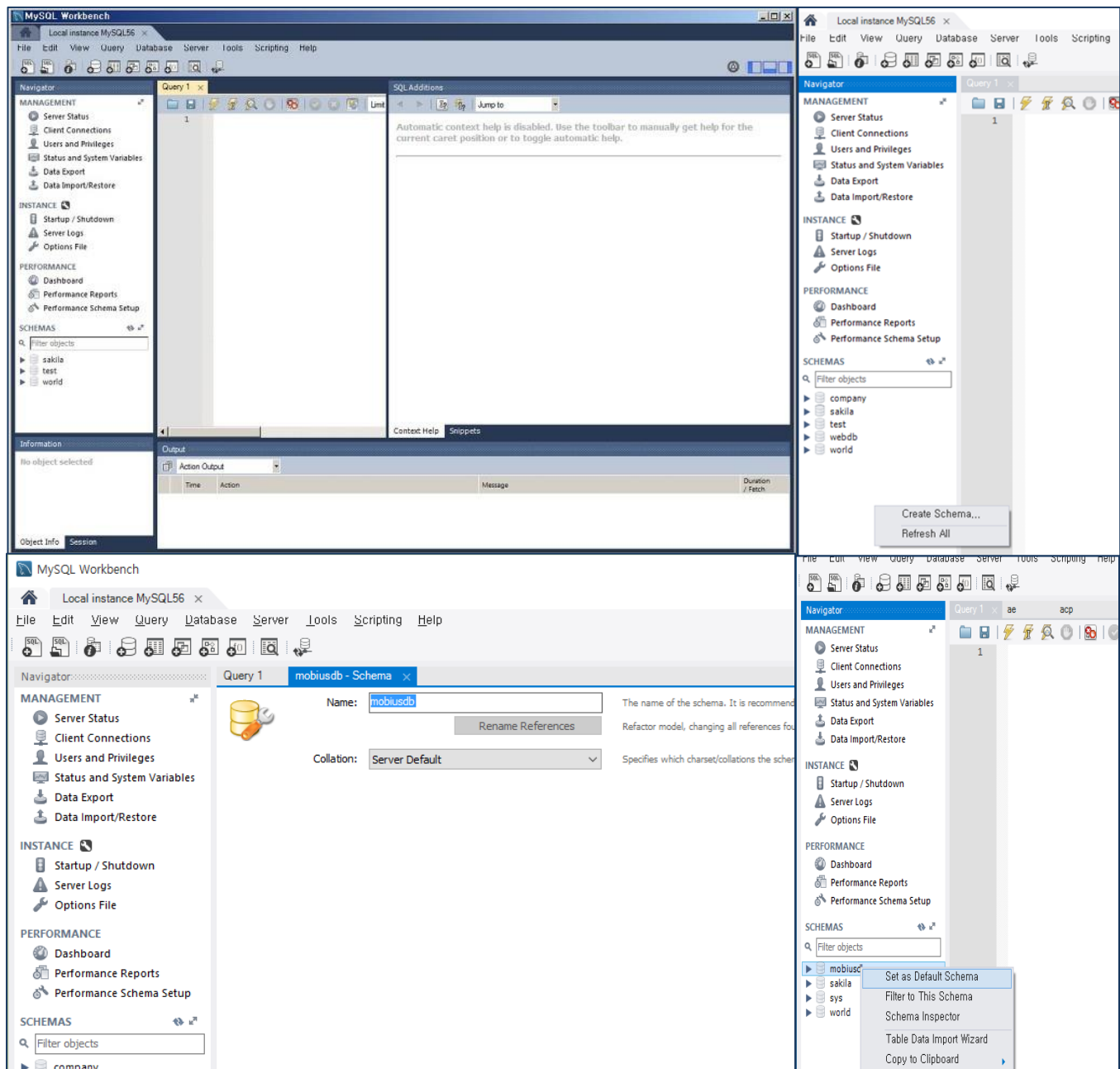
MySQL workbench provides both a graphical user interface (GUI) and command line to manage the database, data tables, import and output existing database as well as execution of queries. If users are familiar with MySQL commands, you can choose command line to input commands to control MySQL database. Here we demonstrate how to configure MySQL database through workbench GUI.

Until this step, we have installed MySQL successfully and then we have to configure the MySQL in terms of creation of a database for Mobius. In [OCEAN alliance](#) website download page, a MySQL database file can be downloaded to use. How to download and import that file will be explained later.

#### Step 1: Create a Mobiusdb Schema

After the MySQL is connected properly, right click the mouse at the workbench GUI as depicted as Figure 17 and create a schema using popped tab. Name the new schema as 'mobiusdb' and apply the schema creation. Then you can see the new created 'mobiusdb' schema at the SCHEMA column at the left down side of GUI, select the mobiusdb schema and right click to select 'Set as Default Schema' to set the current schema as default.





**Figure 16 Create a MySQL connection and connect to the server**

## Step 2: Import Mobiusdb database

All we need regarding for configuring Mobius database are bundled into a sql script file (mobiusdb.sql). Users can download freely the .sql script file from [OCEAN alliance](#) website Mobius page.

As depicted as Figure 18, import the downloaded .sql script file through 'Data Import' popped window. In 'Import from Disk tab', select option 'Import from Self-Contained File' and browse to the downloaded .sql script and select 'the default schema to be imported to' as created 'mobiusdb' schema. Finally click the 'Start Import' to apply the change.

After the .sql file is imported, refresh the mobiusdb schema and you can see the mobiusdb schema is updated with empty tables are updated.

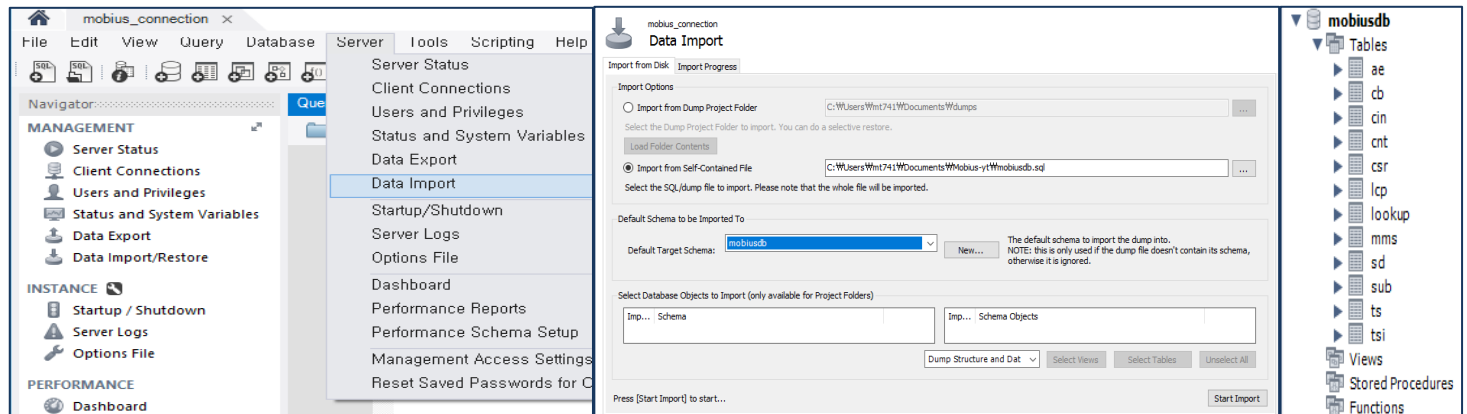


Figure 17 Import Mobius sql script

## 2.2.2. MQTT Server Installation

Mobius enables IoT devices to communicate with MQTT server through MQTT protocol. To this end, a MQTT broker needs to be installed. We prefer to use open source MQTT broker Mosquitto server. Note: MQTT protocol can optionally be supported, if users will not implement MQTT functionalities, step “installation of MQTT server” can be ignored.

The Mosquitto server can be freely downloaded from [Mosquitto Downloads](#) page as shown as Figure . You can see the source and binary versions of Mosquitto with latest and older versions in the download page. Here we prefer to download an older version as highlighted in Figure . If users prefer to use latest functionalities of Mosquitto broker, latest version is recommended.

We download a Mosquitto from <http://mosquitto.org/files/> as shown as Figure 23 and install it just following set up wizard as shown as Figure 20.

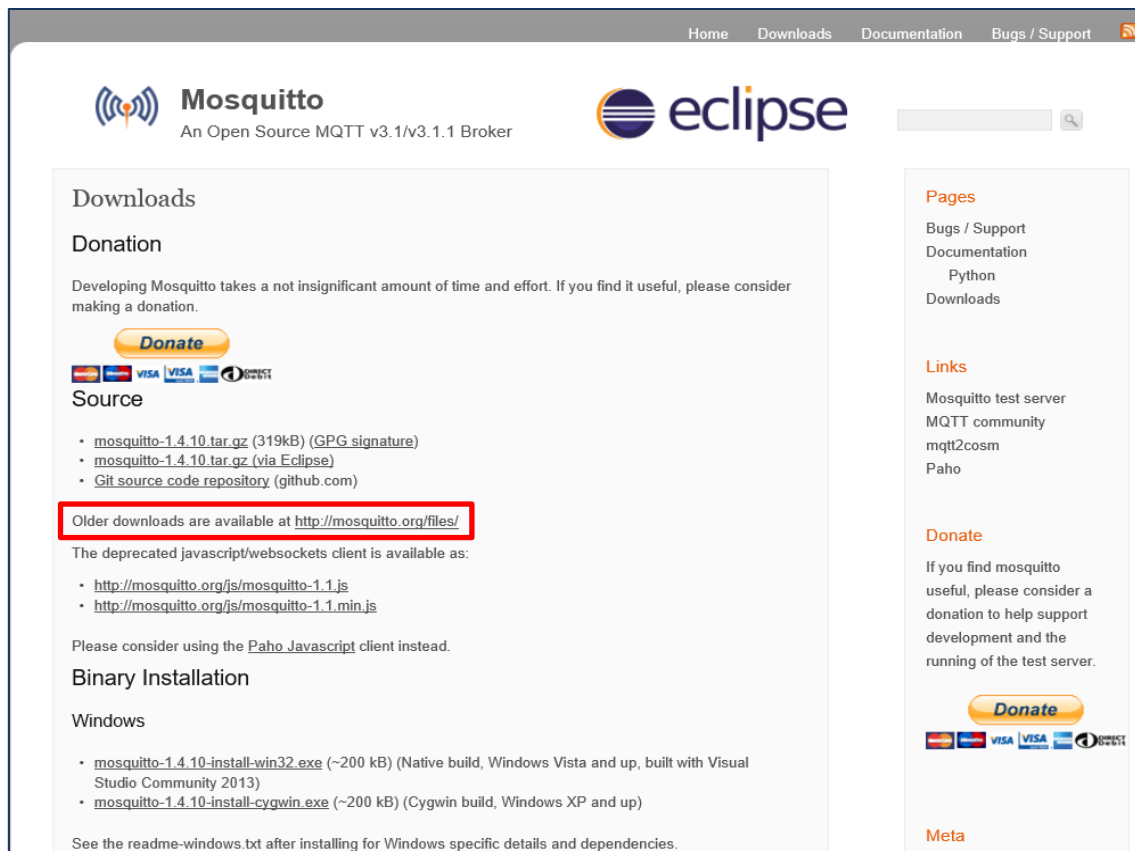


Figure 19 Download MQTT broker Mosquitto

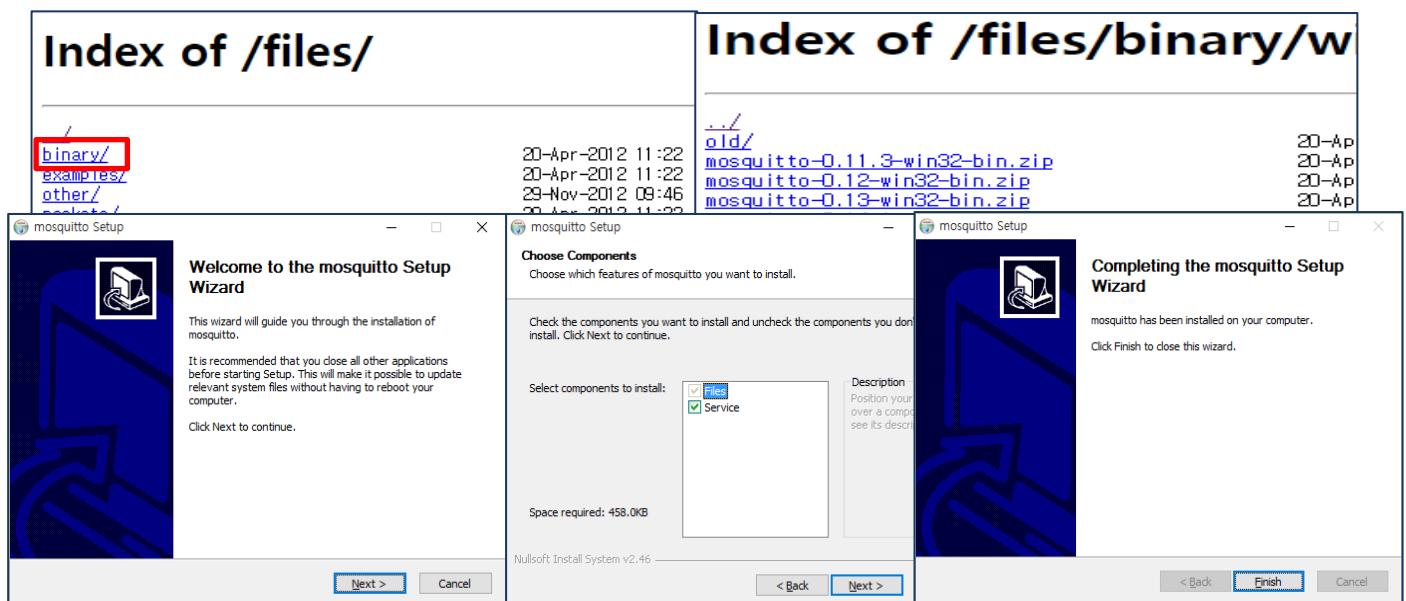


Figure 18 Installation of Mosquitto

After installed Mosquitto broker, we can test whether the Mosquitto broker works properly or not as following:

Step -1: Subscription test

- ⑤ Open a Windows Command Prompt (window-A), go to Mosquitto installation directory (default installation directory is C:\Program Files (x86)\mosquitto) and input command as below

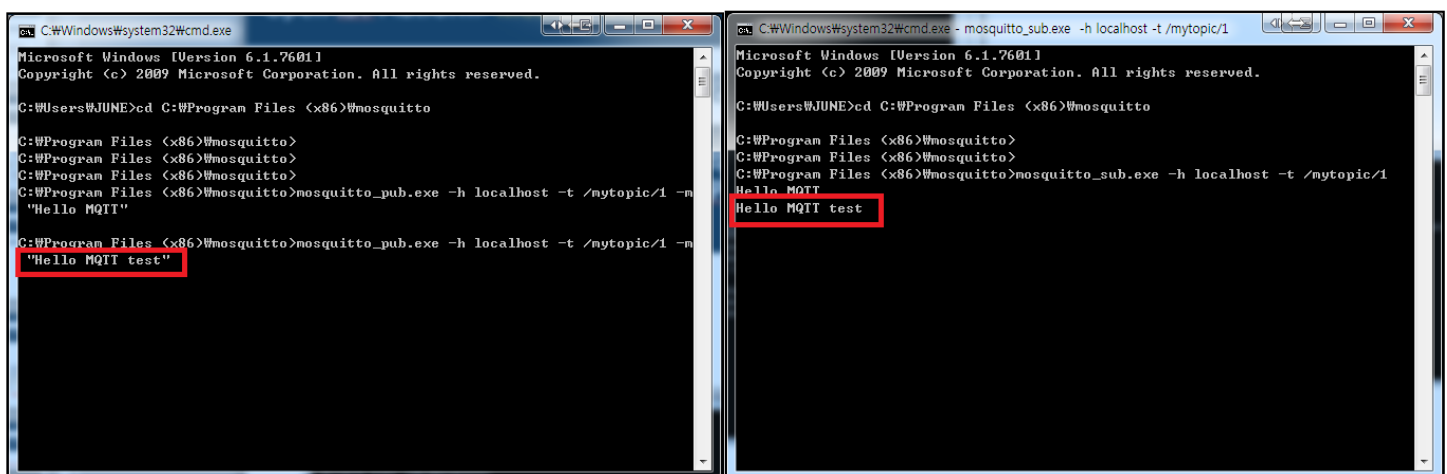
```
mosquitto_sub.exe -h localhost -t /mytopic/1
```

Step -2: Notification test

- ⑥ Open a new Windows Command Prompt (window-B), go to Mosquitto installation directory (default installation directory is C:\Program Files (x86)\mosquitto) and input command as below

```
mosquitto_pub.exe -h localhost -t /mytopic/1 -m "Hello MQTT test"
```

Step -3: Check whether there is 'Hello MQTT test' received in window-B as shown as Figure 22.



Window-A

Window-B

Figure 20 Mosquitto broker test window

## 2.2.3. Node JS Installation

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. In contrast to multi-thread based servers, node.js operating on a single thread using non-blocking I/O calls allows it to support tens of thousands of concurrent connections. With node.js, users don't need worry about the dead-locking of processes because almost no function in node.js directly performs I/O so there is no locks problem at all and therefore, scalable systems can be

reasonably developed through node.js.

Node.js was originally written by developer Ryan Dahl in 2009 and distributed as an open source and currently a diverse of modules developed from contributors are included covering file system I/O, networking, binary data (buffers), cryptography and data streams etc.

For more details about Node.js, please visit [Node.js homepage](https://nodejs.org/) as show as Figure 23. A

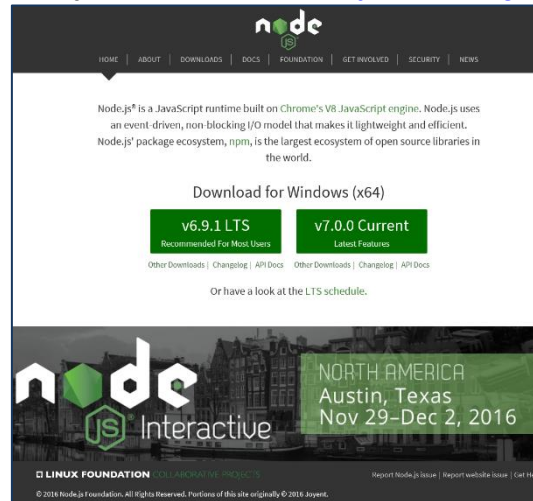


Figure 21 Introduction of Node.Js

number of versions (binary/source) support for different platforms are provided in NodeJs Downloads page as shown as Figure 24. Users can download a preferred version from here and install it following set up wizard.

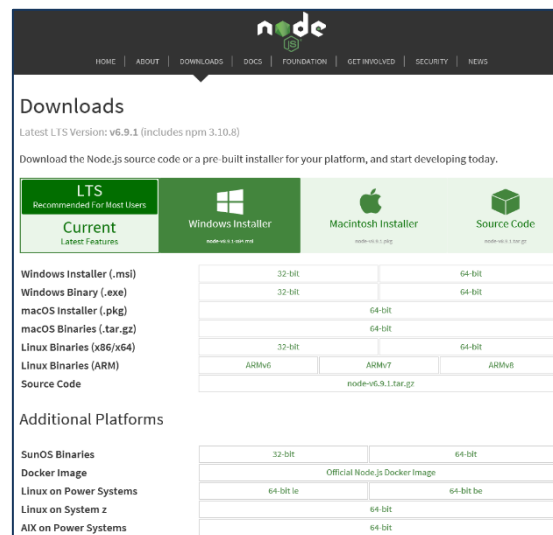
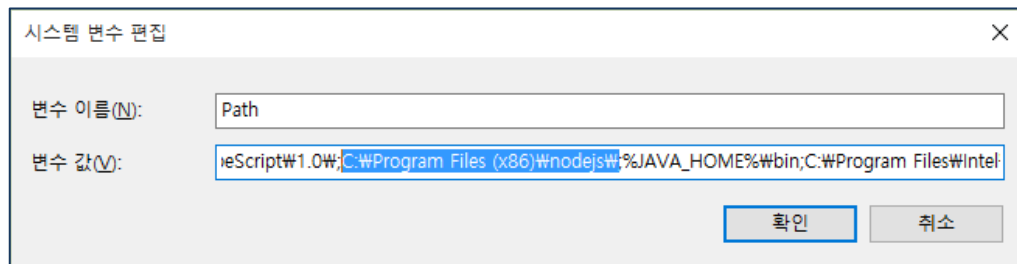


Figure 22 Download Node.js package

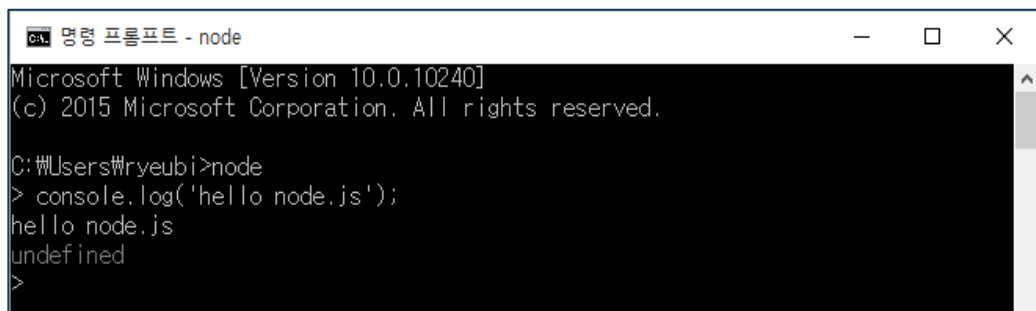
After installation of Node.js, check the system environment variables table (here taking Windows system as an example) to ensure that the installation path of node.js (default is C:\Program File\nodejs) is already added automatically in PATH environment variable and if there is no node.js path, update the PATH with node.js path as shown as Figure 25.

To test whether Node.js is installed properly, open a Windows Command Prompt window and under the user directory, run `node` command to enable node.js mode. In the node.js

mode, input `console.log("hello node.js");` if receives response `hello node.js` then it indicates the Node.JS are installed and works properly.



**Figure 23 Update PATH environment variable with Node.js installation path5**

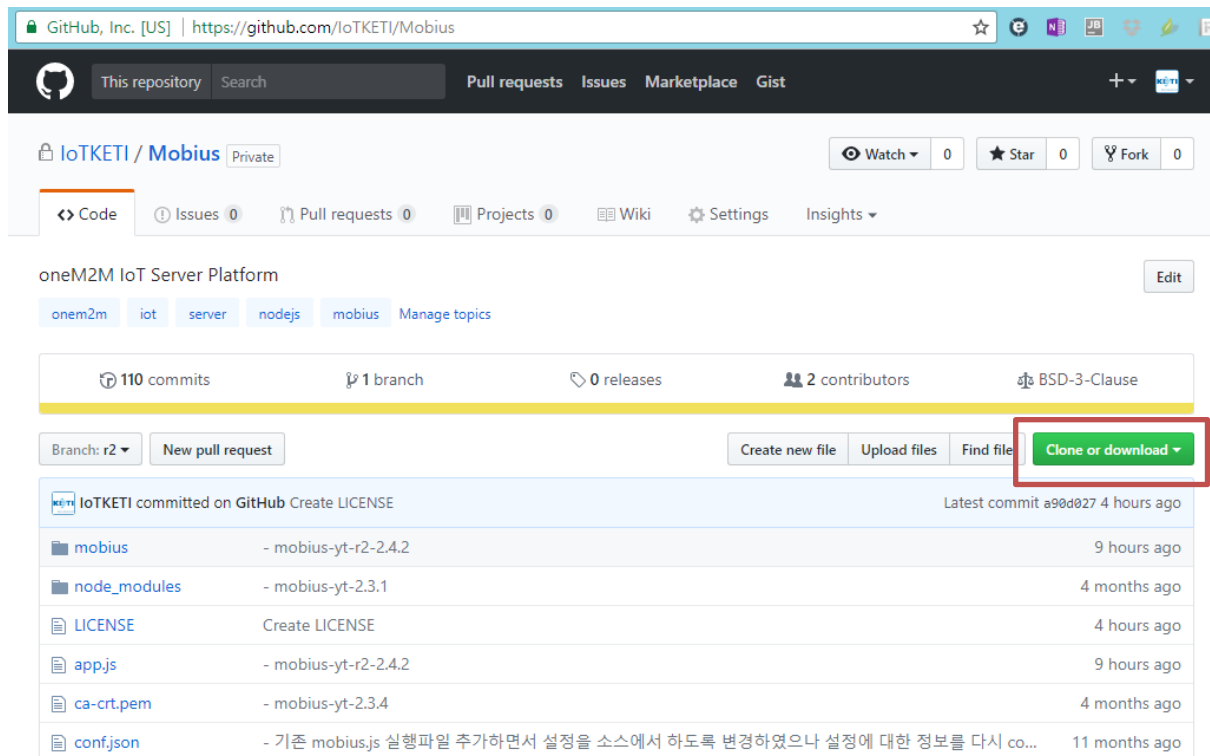


**Figure 24 Test Node.JS**

## 2.3. Mobius Installation

Mobius server platform is distributed with binary and source package in OCEAN alliance website (<http://www.iotocean.org>) or the Mobius GitHub (<https://github.com/loTKETI/Mobius>).

User can download the Mobius server zipped package and configure the Mobius server in terms of communication port and CSE name etc. following this Mobius user guide.



**Figure 25 Download Mobius from GitHub**

After downloading, unzip the package and you will see the included folders and files as Figure . The roles of files are introduced in Table 1.

mobius	2016-10-31 오후...	파일 폴더	
app.js	2016-09-23 오전...	JavaScript 파일	65KB
conf.json	2016-10-11 오후...	JSON File	1KB
mobius.js	2016-09-23 오전...	JavaScript 파일	3KB
package.json	2016-09-23 오전...	JSON File	1KB
pxymqtt.js	2016-09-23 오전...	JavaScript 파일	41KB

**Figure 28 Unzipped Mobius server package files**

The unzipped node.js files can be directly executed without any additional compiling operation. But there are several node.js modules are not yet installed. To installed the required additional node.js modules, open a Windows command prompt window and go to the directory where unzipped the mobius package and run `npm install` command as shown as Figure 29 Run `npm install` command to install additional node.js modules.

```
C:\Windows\system32\cmd.exe
C:\Users\ryeubi\Dropbox\Downloads\mobius-yt-2.1.2>npm install
```

Figure 29 Run `npm install` command to install additional node.js modules

After running the `npm install` command, all additional required node.js modules will be generated and stores in `node_modules` folder as shown as Figure 26 Installation of additional node.js modules You can see there is a `node_modules` folder in the unzipped mobius source folder as shown as Figure 27.

```
C:\Windows\system32\cmd.exe
+-- end-of-stream@1.0.0
+-- readable-stream@2.1.4
|   +-- buffer-shims@1.0.0
|   |   +-- isarray@1.0.0
|   |   +-- stream-shift@1.0.0
|   +-- through2@2.0.1
|   |   +-- readable-stream@2.0.6
|   |   +-- isarray@1.0.0
|   +-- ws@1.1.1
|   +-- options@0.0.6
|   +-- ultron@1.0.2
|   +-- xtend@4.0.1
+-- mysql@2.11.1
+-- bignumber.js@2.3.0
+-- readable-stream@1.1.14
|   +-- sqlstring@2.0.1
+-- url@0.10.3
+-- punycode@1.3.2
+-- querystring@0.2.0
+-- util@0.10.3
+-- xml2js@0.4.17
|   +-- sax@1.2.1
|   |   +-- xmlbuilder@4.2.1
|   |   |   +-- lodash@4.15.0
|   |   +-- xmlbuilder@2.6.5
|   |   |   +-- lodash@3.10.1
+-- npm WARN mobius-yt@0.9.2 No repository field.
C:\Users\ryeubi\Dropbox\Downloads\mobius-yt-2.1.2>
```

Figure 26 Installation of additional node.js modules

mobius	2016-10-11 오전...	파일 폴더	
node_modules	2016-10-11 오후...	파일 폴더	
app.js	2016-09-23 오전...	JavaScript 파일	65KB
conf.json	2016-10-11 오후...	JSON File	1KB
mobius.js	2016-09-23 오전...	JavaScript 파일	3KB
package.json	2016-09-23 오전...	JSON File	1KB
pxymqtt.js	2016-09-23 오전...	JavaScript 파일	41KB

Figure 27 A `node_modules` folder created in unzipped source folder

Until this step, we have installed all required software and modules that enable the running of Mobius server. Users can run `node mobius.js` in the command line to activate the Mobius server. The installation of Mobius server is almost done and still some configurations left.



## 3. Running Mobius Server

Mobius 서버 설치까지 정상적으로 설치된 후 Mobius를 구동하기 전에 환경 설정을 해야 정상적으로 구동할 수 있다.

### 3.1. Runtime Environment Configuration

A configuration file `conf.json` is included in the unzipped folder of Mobius package. User can customize the communication port `csebaseport` for CSEBase which is the root of all resources to be created in hierarchical structure in Mobius server, and the MySQL database connection password `dbpass` as shown as Figure 28. These configuration data are required to guarantee the Mobius server is accessible from the CSEBase port and the Mobius server can access to the MySQL database with the configured password. In addition, there are configuration data included in `mobius.js` module file as shown as Figure 29.

```
{
  "csebaseport": "7579",
  "dbpass": "<mysql db connect password>"
}
```

Figure 28 Mobius-YT server configuration file

```
15 var fs = require('fs');
16
17 var data = fs.readFileSync('conf.json', 'utf-8');
18 var conf = JSON.parse(data);
19
20 global.defaultnmtype = 'short';
21 global.defaultbodytype = 'json';
22
23
24 // my CSE information
25 global.usecsetype = 'in'; // select 'in' or 'mn' or 'asn'
26 global.usecsebase = 'mobius-yt';
27 global.usecseid = '/mobius-yt';
28 global.usecsebaseport = conf.csebaseport;
29
30 global.usedbhost = 'localhost';
31 global.usedbpass = conf.dbpass;
32
33
34 global.usepxymqttport = '7580';
35 global.usesagentport = '7581';
36
37 global.usemqttbroker = 'localhost'; // mqttbroker for mobius
38
39
40 // CSE core
41 require('./app');
```

Figure 29 Other configuration data in main code file

Until this step, we have done all installation and configurations for the Mobius server platform. Now users can run `node mobius.js` command to activate Mobius server and it runs as shown as

```

C:\Windows\system32\cmd.exe - node mobius.js
C:\Users\Wryeubi\Dropbox\Downloads\mobius-yt-2.1.2>node mobius.js
Production Mode
select_ri_lookup: 28.471ms
update_cb_poa_csi: 7.455ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
CPU Count: 8
pymatt server (192.168.1.9) running at 7580 port
Production Mode
Production Mode
Production Mode
mobius server (192.168.1.9) running at 7579 port
mobius server (192.168.1.9) running at 7579 port
mobius server (192.168.1.9) running at 7579 port
select_ri_lookup: 31.378ms
update_cb_poa_csi: 8.172ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
select_ri_lookup: 35.592ms
update_cb_poa_csi: 6.664ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
select_ri_lookup: 27.257ms
Production Mode
update_cb_poa_csi: 6.111ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
mobius server (192.168.1.9) running at 7579 port
Production Mode
mobius server (192.168.1.9) running at 7579 port
select_ri_lookup: 36.145ms
update_cb_poa_csi: 6.031ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
select_ri_lookup: 20.215ms
update_cb_poa_csi: 6.081ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
Production Mode
mobius server (192.168.1.9) running at 7579 port
Production Mode
Production Mode
mobius server (192.168.1.9) running at 7579 port
mobius server (192.168.1.9) running at 7579 port
select_ri_lookup: 21.119ms
update_cb_poa_csi: 5.955ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
select_ri_lookup: 19.553ms
update_cb_poa_csi: 6.019ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
select_ri_lookup: 20.090ms
update_cb_poa_csi: 6.198ms
{"rsc":"2004","ri":"/mobius-yt","sts":""}
select_direct: 3.282ms
resource_retrieve: 1.301ms
{"rsc":"2000","ri":"/GET-/mobius-yt-{"fu#":2,"rcn#":1},"sts":""}
subscribe req_topic as /oneM2M/req/+/:/mobius-yt/#
subscribe req_topic as /req/+/:/mobius-yt/#
subscribe req_req_topic as /oneM2M/req_req/+/:/mobius-yt/#
subscribe resp_topic as /oneM2M/resp/:/mobius-yt/#
subscribe resp_topic as resp:/mobius-yt/#
ts_missing agent server (192.168.1.9) running at 7581 port
select_direct: 1.009ms
search_parents_lookup: 6.394ms
search_lookup: 291.309ms
{"rsc":"4004","ri":"/GET-/mobius-yt-{"fu#":1,"ty#":25,"rcn#":1},"sts":"resource do not exist"}
init_TS - 4004

```

Figure 30. Running capture of Mobius server platform

## 3.2. Test

### 3.2.1. oneM2MBrowser

Refer to the oneM2MBrowser manual from OCEAN webpage.

<http://iotocean.org/archives/module/onem2mbrowser>

### 3.2.2. Postman

The Mobius server accepts valid HTTP request in terms of POST, GET, PUT and DELETE request following oneM2M service primitives. In this session, we provide several examples to test against the installed Mobius server platform using Postman Rest Client (hereafter short for Postman). Postman is a Google Chrome extension which you can use to easily test Web

API methods. It can also be used against 3rd party APIs. We use Postman to test Mobius APIs.

Users can install Postman through Google Chrome web browser. Go to the Chrome Settings option and it brings you to the Chrome Extension (plugins) page where there is a 'Get more extensions' option through which more extensions can be downloaded as shown as Figure 31.

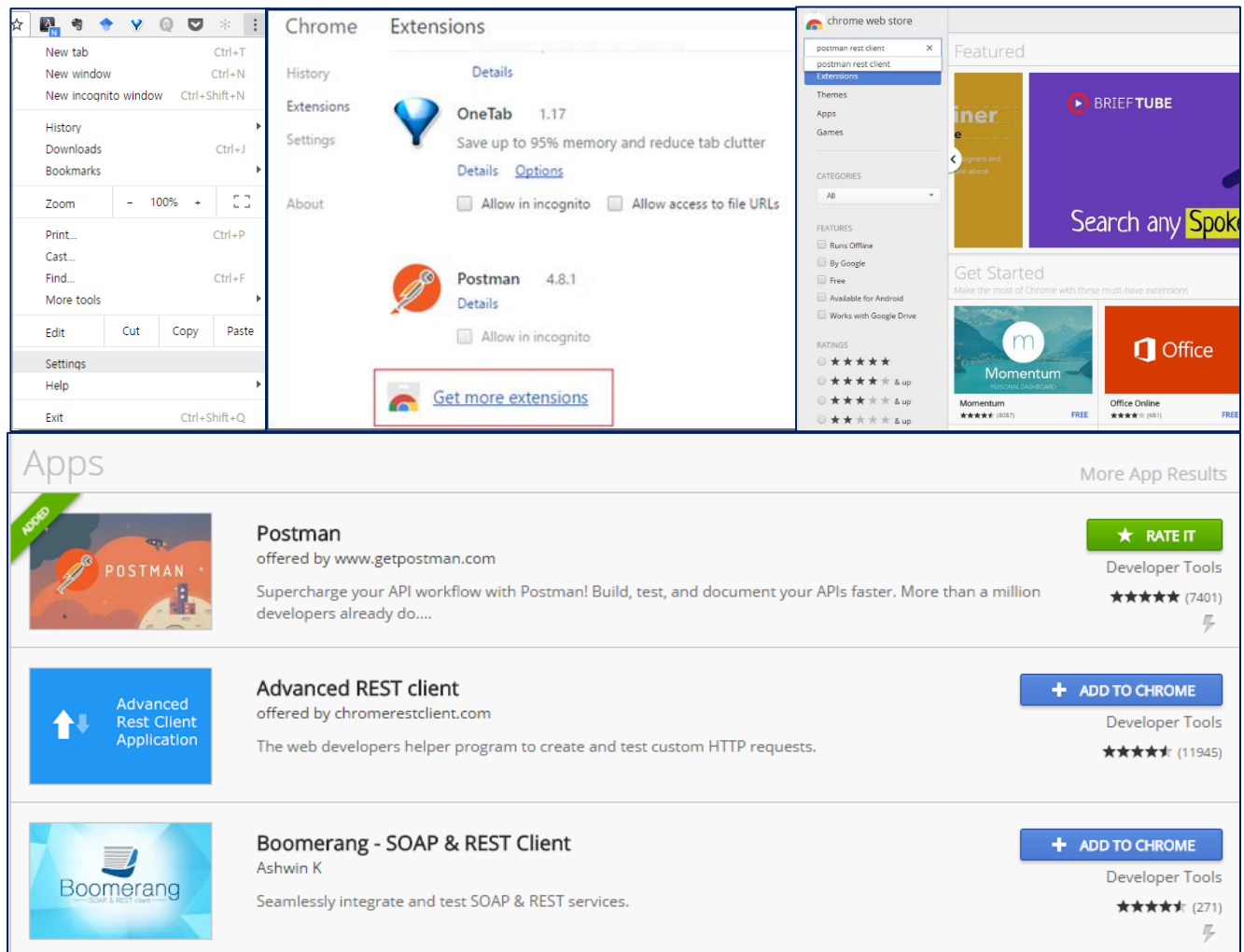


Figure 31 Download Postman REST Client

Postman provides an user interface to specify request headers and request body (if any). To simplify the test of Mobius API, we provide a collection of Postman script written for testing Mobius API. The Postman script is represented as a json file can be downloaded from OCEAN or GitHub. After download the Postman script, open the Postman and import the script as shown as Figure 32.

- ① In 'Collections' view, a group of Postman scripts are listed
- ② Click 'Import' button and
- ③ browse the downloaded script from the saved directory then the Postman script can be automatically added into the collections list.

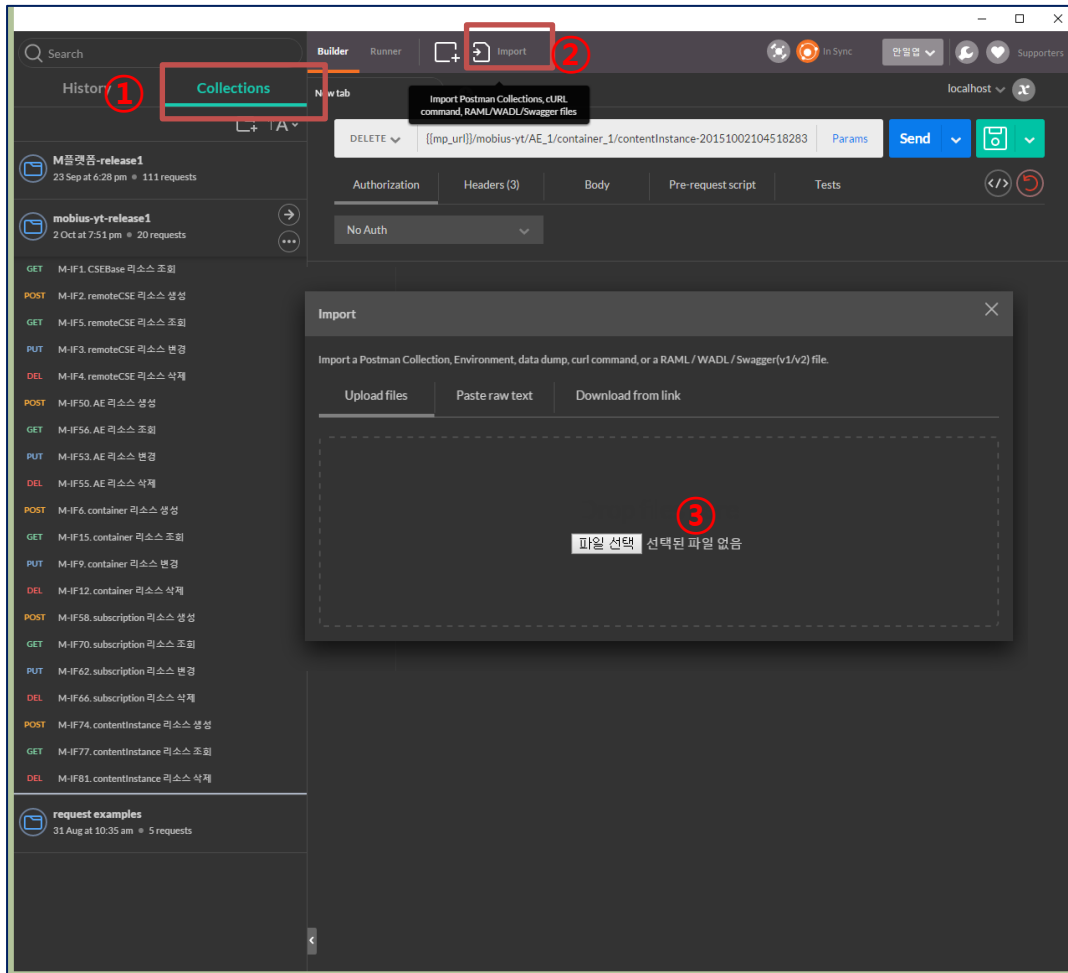



Figure 32 Import Postman Script

If the Postman script is imported successfully, a new collection will be displayed at the top of collections list shown as Figure 33. The script is designed to test Mobius-YT API in terms of oneM2M CREATE, RETRIEVE, UPDATE, and DELETE operations for oneM2M primitives represented in XML and JSON serializations.



Figure 33 A list of imported Postman Scripts

To access to the Mobius-YT server, the access configurations in terms of host and port as well as the CSEBase name for the Mobius-YT server need to be configured in Postman environment. To this end,

go to the setting  and select the 'Manage Environments' option in the option list then it pops a window where lists all existing configured environments as shown as Figure . User can edit existing or add a new environment by clicking 'Add' at the right bottom and in the popped window input Mobius server access configurations in terms of host and CSEBase name as shown as Figure .

- ① Name the new environment, e.g. Mobius\_config
- ② {{mp\_url}}: specify the host and port number of Mobius server
- ③ {{cb}}: specify CSEBase name of Mobius server
- ④ Apply the changes

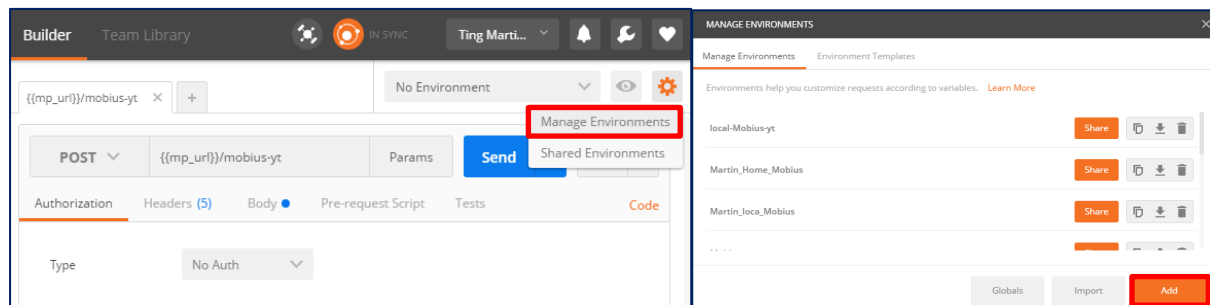


Figure 38 Manage Postman Environment

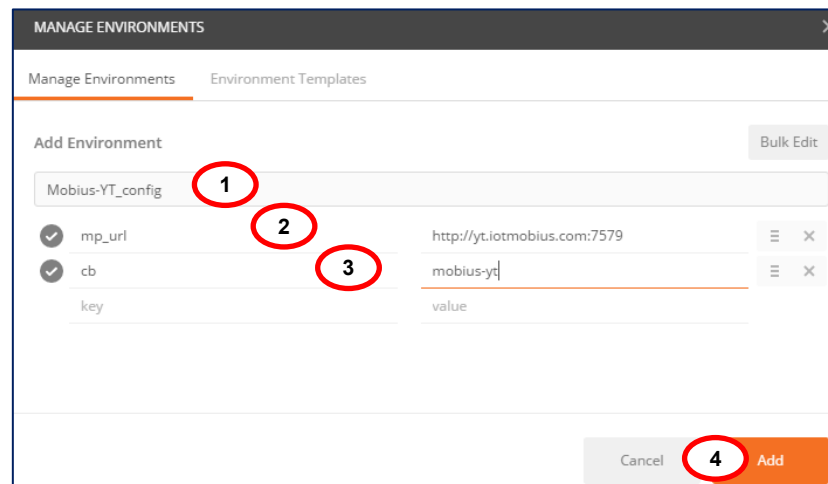


Figure 39 Add a new Environment with host and CSEBase information

Besides, there is another way to get the Postman environment, i.e. import directly an environment (POSTMAN Environment Script) that can be downloaded from OCEAN or GitHub.

After downloading the Postman environment script, import into Postman following steps as shown as Figure 34.

- ① Click 'Import' and then pops up a window to choose files
- ② Browse the downloaded Postman environment script from the saved directory
- ③ After imported successfully, you can see a new environment is added in the Manage Environments list

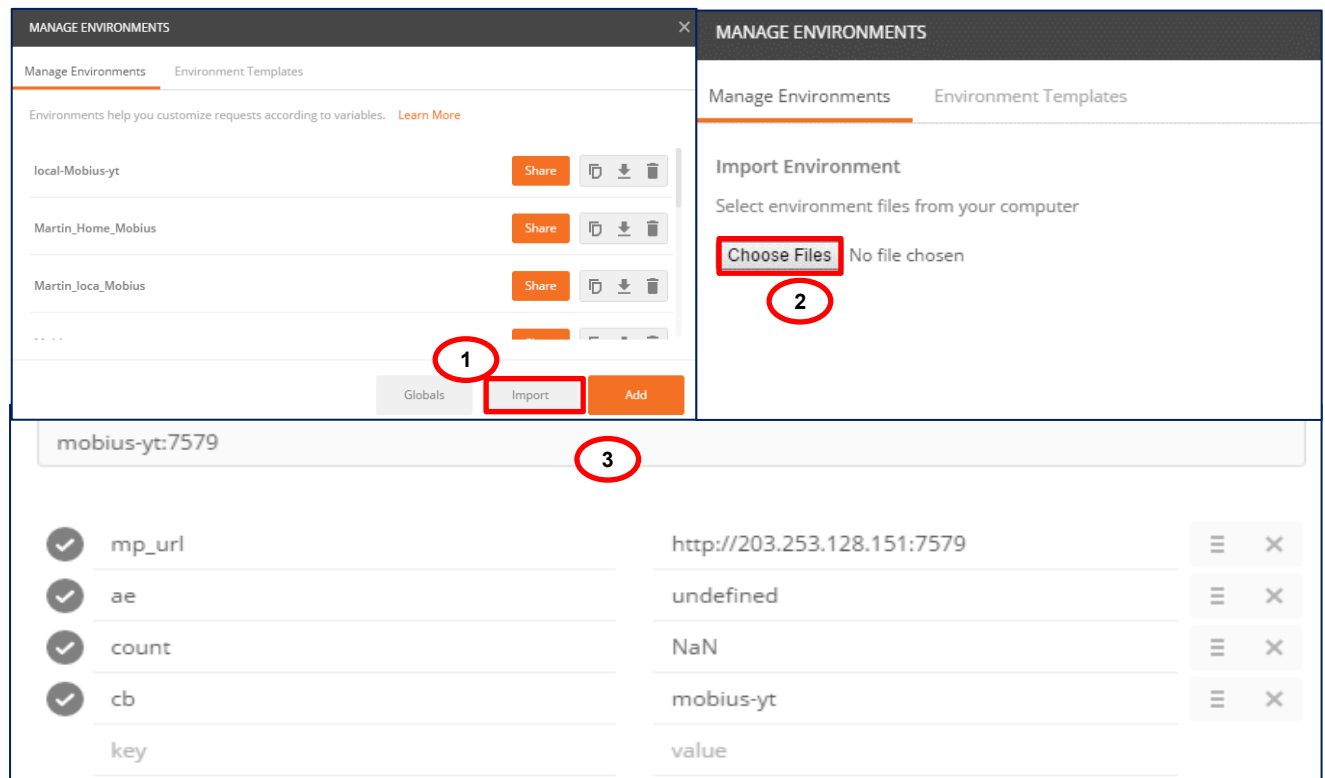


Figure 34 Import Postman Environment Script

To demonstrate how to use Postman to test Mobius API in case users don't know how to use Postman. If you are familiar with Postman, just skip this.

Here we demonstrate CSEBase RETRIEVE request which try to retrieve the CSEBase information from Mobius server. Mapping to REST request,

- ① To use HTTP GET method
- ② Specify the Mobius server access URL
- ③ Specify HTTP Headers (at least mandatory headers)
- ④ Send request to Mobius server
- ⑤ Check the response status code: 200 OK indicates request was accepted and response is returned successfully
- ⑥ Check the returned CSEBase information

The screenshot displays a REST client interface for a test named "TD\_M2M\_NH\_01. CSEBase resource retrieve".

**Request Section:**

- 1** Method: GET
- 2** URL: `{{mp_url}}/{{cb}}`
- 4** Action: Send

**Headers Section:**

key	value
Accept	application/xml
X-M2M-RI	12345 <b>3</b>
X-M2M-Origin	50.2.481.1.1.232466

**Response Section:**

- 5** Status: 200 OK
- 6** Time: 72 ms

**Response Body (XML):**

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <m2m:cb xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001
3 /XMLSchema-instance" rn="mobius-yt">
4   <ty>5</ty>
5   <ct>20160419T120301</ct>
6   <ri>/mobius-yt</ri>
7   <lt>20160419T120301</lt>
8   <lbl>mobius-yt</lbl>
9   <cst>1</cst>
10  <csi>/mobius-yt</csi>
11  <srt>1 2 3 4 10 16 23 23 24 25 26</srt>
12  <poa>http://203.253.128.151:7579 mqtt://203.253.128.151:/mobius-yt</poa>
13 </m2m:cb>
  
```

Figure 35 Demo capture of CSEBase RETRIEVE Test

Users can test other scripts to deep understand the Mobius server API and we recommend to refer to Mobius REST Reference API User Guide to use and extend Mobius to apply with user-customized services.