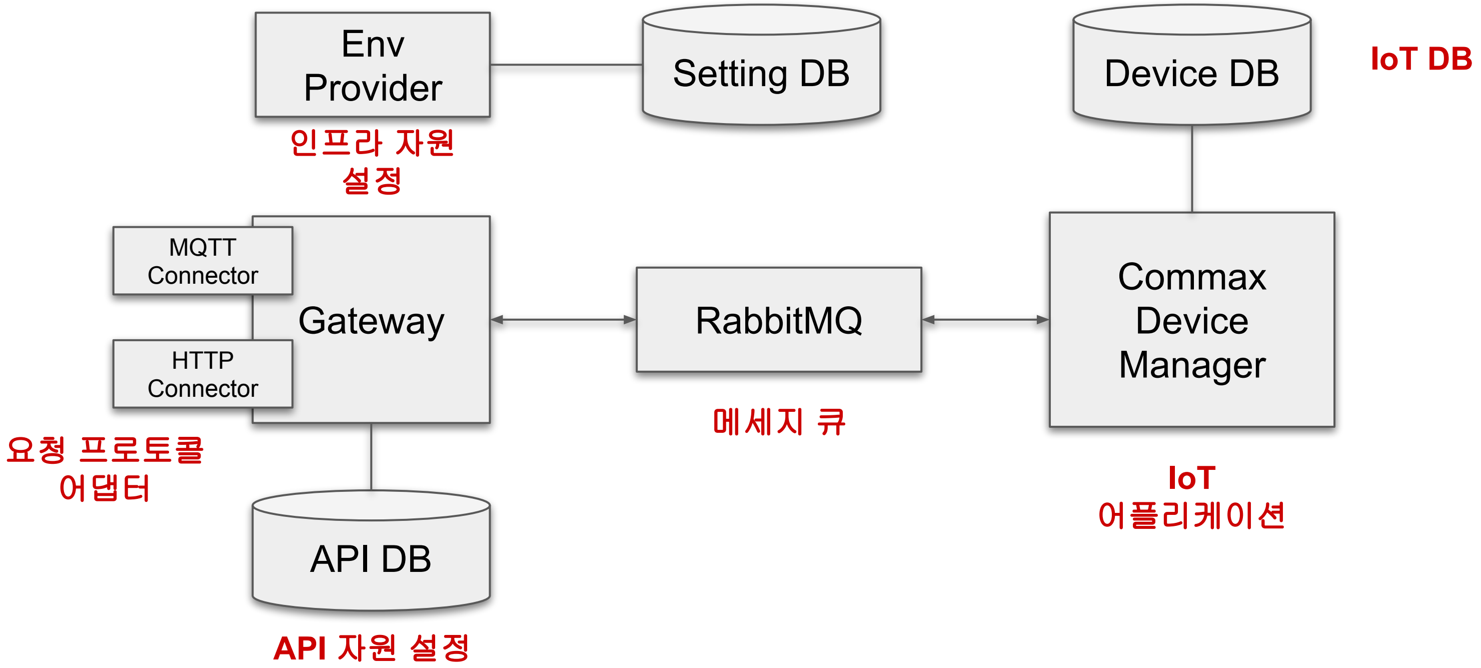
POCAT 0.1 Design

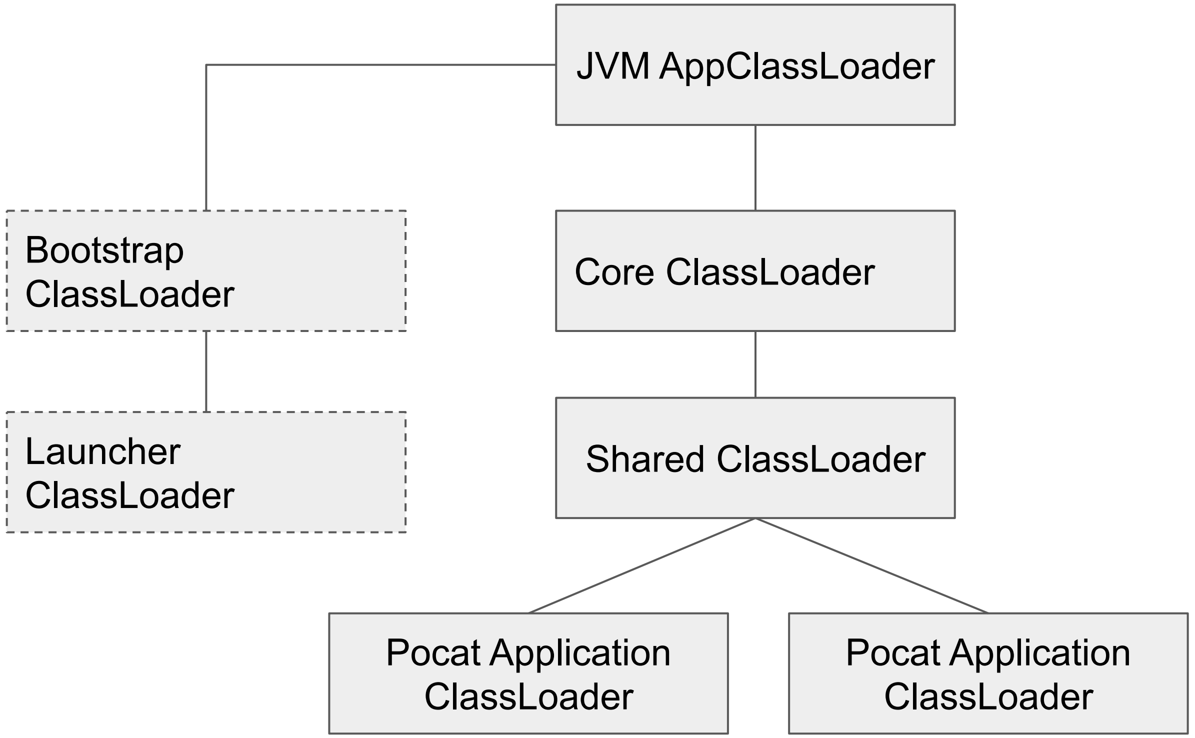
동고비 소프트

Overview

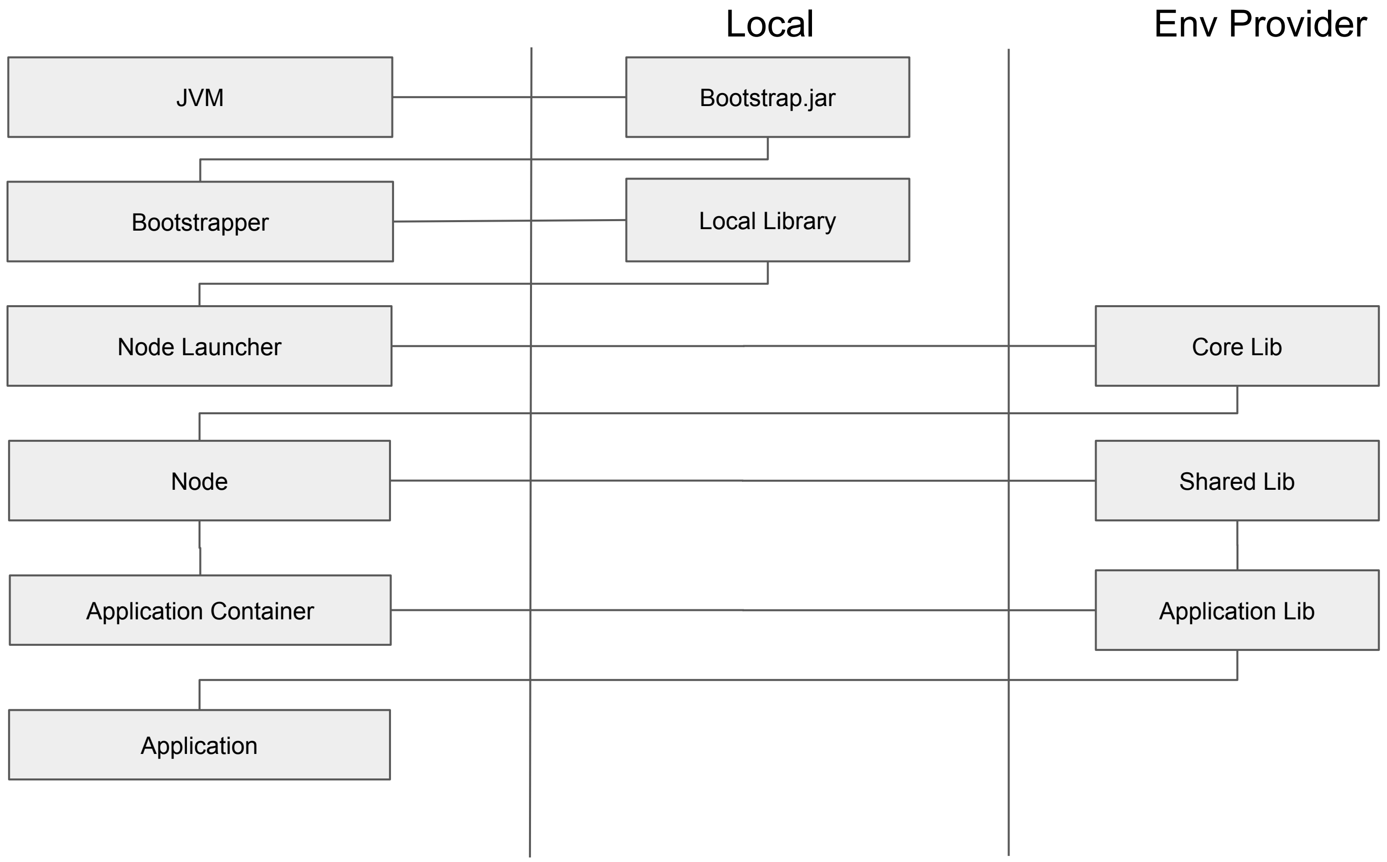
Architecture



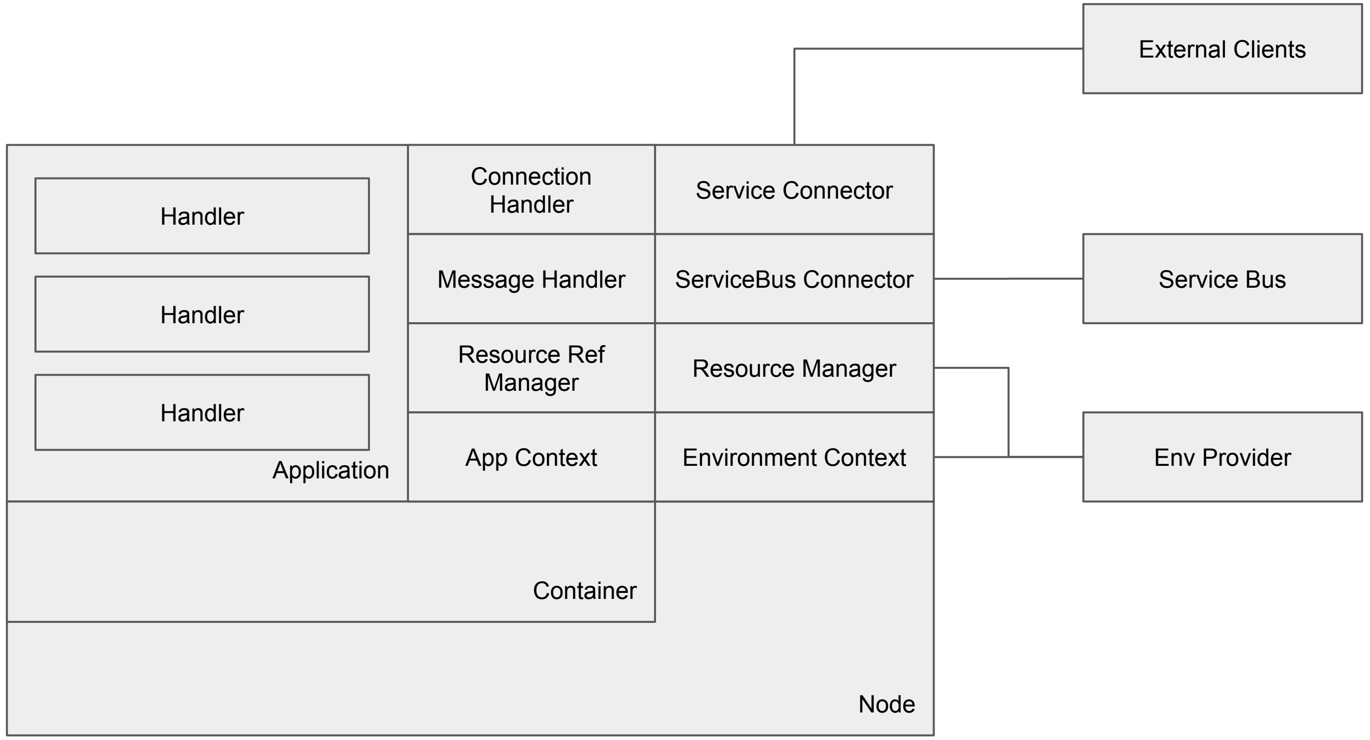
Class Loader Hierarchy



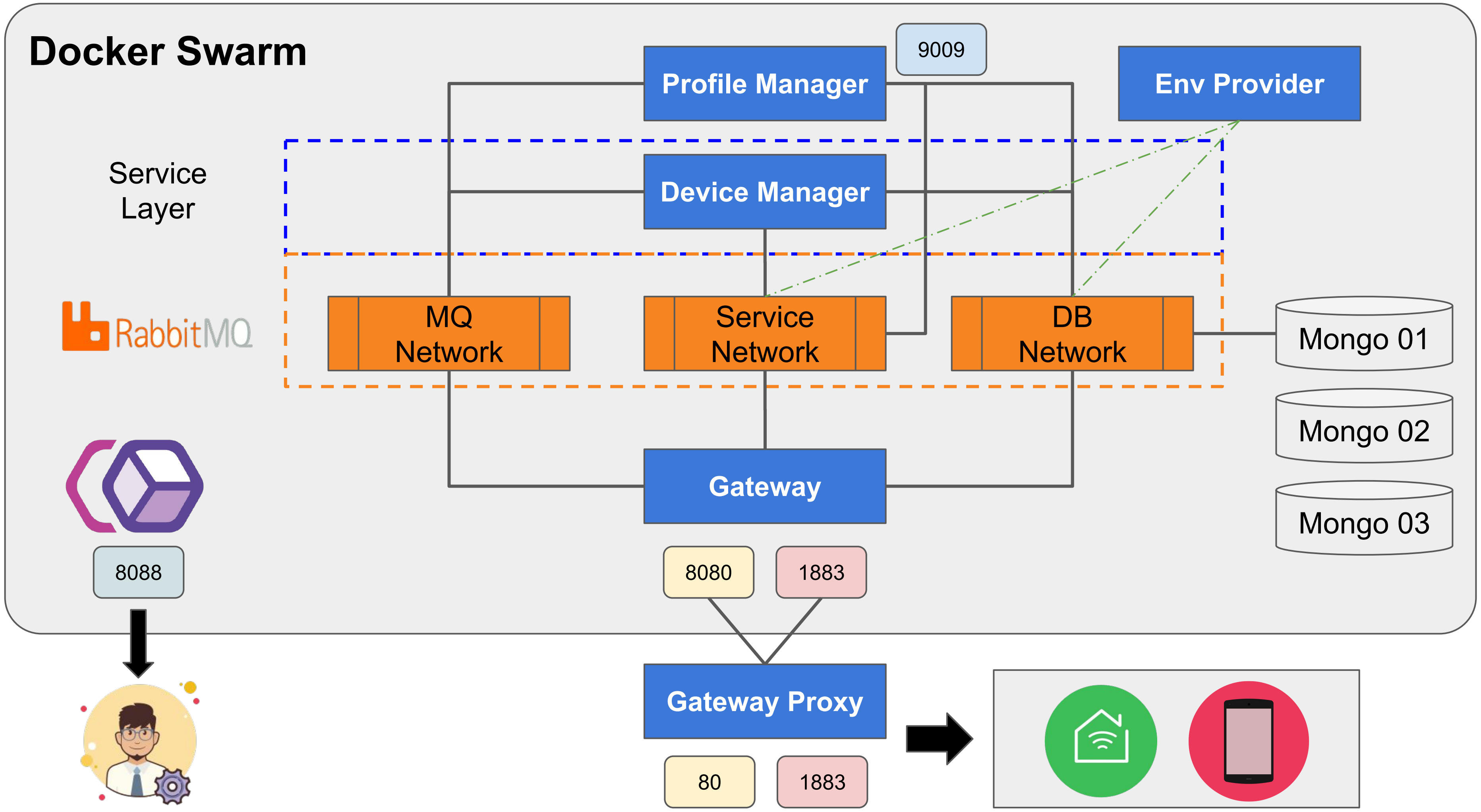
Class Loading 관계도



Pocat Node Service Architecture



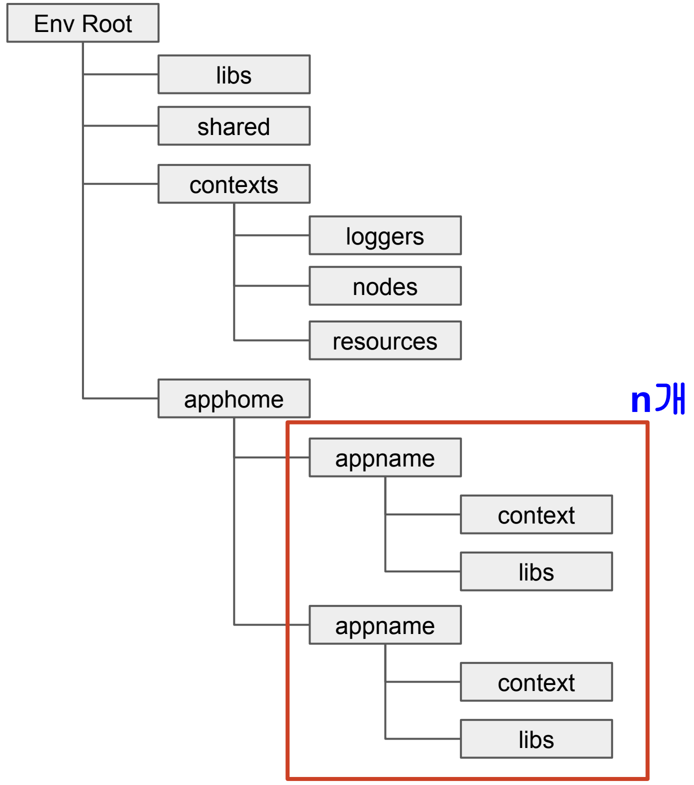
각각의
모듈간의 구조
및 연결관계


Docker Swarm
Profile Manager
9009
Env Provider
Service Layer
Device Manager
RabbitMQ
MQ Network
Service Network
DB Network
Mongo 01
Mongo 02
Mongo 03
Gateway
8088
8080
1883
Gateway Proxy
80
1883

Environment Provider

Env Provider Data Structure

Environment File Tree

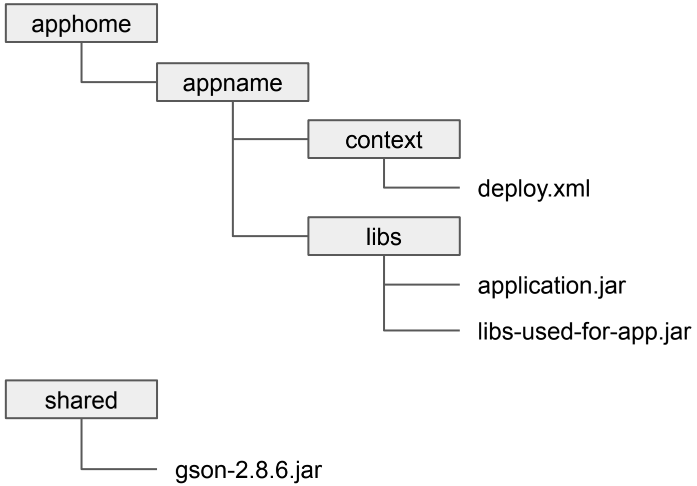


Environment Properties

key-value properties

플랫폼에서의 리소스와 어플리케이션의
리소스의 구조를 정의하는 파일 트리

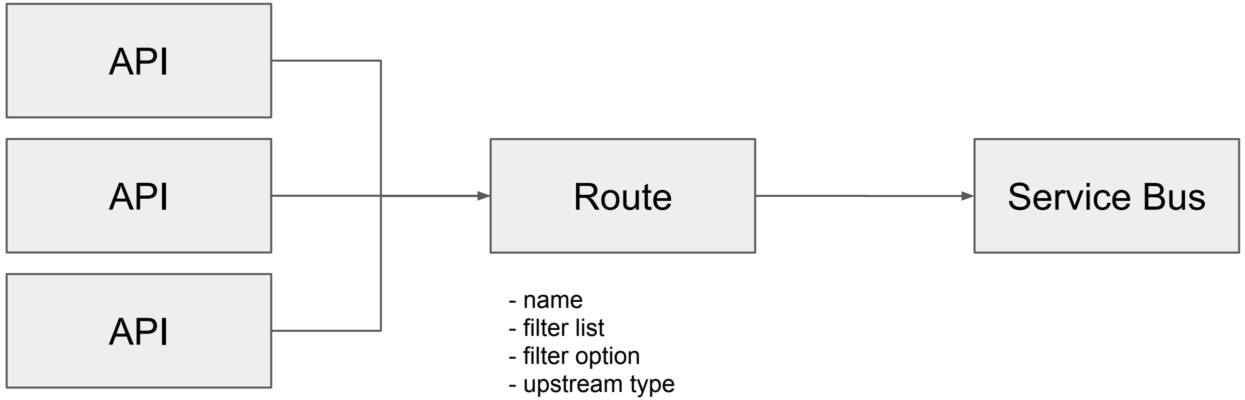
AppHome Data Structure



어플리케이션
데이터 구조

Gateway

API Composition



- name
- filter list
- filter option
- upstream type
- upstream destination

- name
- api group
- uri pattern
- method
- description
- parameter
- route
- response

API Config

Gateway Detail API

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<api>
   <key>CMX GATEWAY DETAIL</key>
   <group>commax</group>

   <protocol>http</protocol>
   <method>GET</method>
   <path>/v1/gateways/${resource_no}</path>

   <description>List gateways</description>

   <route>DETAIL_GATEWAY</route>
   <parameters>
       <parameter in="header" name="Authorization"
required="true"/>
   </parameters>
   <scopes>
       <!--<role>SELF</role>-->
       <role>ANY</role>
   </scopes>
   <request-filters/>
   <response-filters/>
</api>
```

Get Token API

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<api>
   <key>CMX GET TOKEN</key>
   <group>commax</group>

   <protocol>http</protocol>
   <method>GET</method>
   <path>/oauth/authorize</path>

   <description>Login and get access token</description>

   <route>CREATE_ACCESS_TOKEN</route>
   <parameters>
       <parameter in="query" name="client_id" required="true"/>
       <parameter in="query" name="client_secret" required="true"/>
       <parameter in="query" name="grant_type" required="true"/>
       <parameter in="query" name="mac" required="true"/>
   </parameters>
   <scopes>
       <!--<role>SELF</role>-->
       <role>ANY</role>
   </scopes>
   <request-filters/>
   <response-filters/>
</api>
```

Route Config

Gateway Detail Route

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<route>
   <key>DETAIL_GATEWAY</key>

   <description>register mac and user</description>

   <request-filters/>
   <response-filters/>

   <upstream>
       <type>amqp</type>
       <destination>v1.gateway.*</destination>
       <respond>true</respond>
   </upstream>
</route>
```

Get Token Route

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<route>
   <key>CREATE_ACCESS_TOKEN</key>

   <description>register mac and user</description>

   <request-filters/>
   <response-filters/>

   <upstream>
       <type>amqp</type>
       <destination>oauth.authorize</destination>
       <respond>true</respond>
   </upstream>
</route>
```

Application

Application Example

Application.java

```java
package pocatx.application;

public interface Application {
   void init(ApplicationConfig config)
                             throws InitializeException;
   void onCommand(Command command)
                        throws CommandProcessException;
}
```

MessageHandler.java

```java
package pocatx.application.handler;

public interface MessageHandler extends Handler {
   void handleMessage(Request req, Response resp)
         throws HandlerException;
}
```

Application Example

DeviceManagerApp.java

```java
package com.dongobi.device.manager;

public class DeviceManagerApp extends
AbstractApplication {
   private DeviceRepository repo;
   private Gson gson = new Gson();

   @Override
   protected void init() {
       Object deviceDB =
getConfig().getResource("device-db");
       try {
           repo =
DeviceRepositoryFactory.createRepository(deviceDB);
       } catch (Exception e) {
           e.printStackTrace();
       }
   }
```

```java
   @Override
   public void onCommand(Command command) {
       switch(command.getType().toLowerCase()) {
           case "device.add.client":
               ClientInfo clientInfo =
buildClientInfo(command.getContents());
               repo.addClient(clientInfo);
               break;
           case "device.add.model":
               ModelInfo modelInfo =
buildModel(command.getContents());

               repo.addModel(modelInfo);
               break;
       }
   }
```

Application Example

DeviceHandler.java

```
public abstract class AbstractDeviceManagerHandler
extends AbstractMessageHandler {
   private static final Logger logger =
LoggerFactory.getLogger(AbstractDeviceManagerHandler clas
s);
   private DeviceManagerApp deviceManagerApp;
   protected static final Gson gson = new Gson();

   @Override
   protected void init() {
       deviceManagerApp = (DeviceManagerApp)
getApplication();
   }

   protected DeviceManagerApp getDeviceManager() {
       return deviceManagerApp;
   }
```

```
   @Override
   public void handleMessage(Request req, Response
resp) throws HandlerException {
       logger.info("Request with txId [{}]",
req.getTxId());
       String method = req.getMethod();
       switch(method.toUpperCase()) {
           case "GET":
               doGet(req, resp);
               break;
           case "POST":
               doPost(req, resp);
               break;
           default:
               throw new
HandlerException(StatusCode NOT_ALLOWED, "Not
allowed method [" + method + "]");
       }
   }
```

Application Deploy descriptor <IoT 어플리케이션>

```
<?xml version="1.0" encoding="UTF-8" ?>
<application>
   <deploy-node>default-service-node</deploy-node>

<app-class>com.dongobi.device.manager.DeviceManagerApp</app-c
lass>
   <init-params>
       <init-param name="gateway.api.group" value="commax"/>
   </init-params>

   <resource-refs>
       <ref name="device-db" resource-name="device-db"/>
   </resource-refs>

   <servicebus-handler-mapping>
       <queue-name>device-manager-queue</queue-name>
       <message-handler>

<handler-class>com.dongobi.device.manager.handler.ModelHandle
r</handler-class>
           <topic>cmx.model</topic>
       </message-handler>
       <message-handler>

<handler-class>com.dongobi.device.manager.handler.ClientHandl
er</handler-class>
           <topic>cmx.client</topic>
       </message-handler>
```

```
       <message-handler>

<handler-class>com.dongobi.device.manager.handler.UserHandler
</handler-class>
           <topic>cmx.user</topic>
       </message-handler>
       <message-handler>

<handler-class>com.dongobi.device.manager.handler.MacHandler<
/handler-class>
           <topic>cmx.mac</topic>
       </message-handler>
       <message-handler>

<handler-class>com.dongobi.device.manager.handler.RegisterHan
dler</handler-class>
           <topic>cmx.register</topic>
       </message-handler>
       <message-handler>

<handler-class>com.dongobi.device.manager.handler.TokenHandle
r</handler-class>
           <topic>oauth.authorize</topic>
       </message-handler>
   </servicebus-handler-mapping>
</application>
```

Conclusion

추가 개발 예정 사항

1. 관리용 UI 추가 개발
2. Log 정리 및 Rule 결정
3. 배치/설정 변경 간편화
4. API Protocol 추가
5. CORS 지원