What to do

- Connect Artik to your application.
- Sending data to Artik (as device).
- Getting data of specific device from Artik.
- Make your own device type and connect to Artik.
- Send action to device.
- Getting action and react to it.
Make your own device
(server, service, application⋅⋅⋅)
Device type in Artik developer page

ARTIK Cloud is open and device agnostic

Devices can upload data to ARTIK Cloud in any format that works for them.

• Any data, any message, sent and received in real-time
• Develop private device types with security built in
• Share device type capabilities with other developers

Add your device type

Tell ARTIK Cloud what your device type is called then start building a Manifest to describe it further.
Create device type

Fill it as you want

- **DEVICE DISPLAY NAME**: My_Device
- **UNIQUE NAME**: sky.com.mydevice
Make manifest

- Manifest
  - Manifest defines *what kind of data* will you use in a device.
  - For example, in arrival sensor, its manifest defines `arrive(Boolean)`, `battery capacity(float)` for device’s data.
Define data fields

- Define name
- Define type and unit
- Define the range of meaningful value
- Fill it as you want
- You can add other data fields
Define action
Define action

- Define parameter name
- Define the range of meaningful value
- Define type and unit
- You can add other actions
Activate manifest

**Device Fields**
Describe fields for each piece of data produced by this device.

**Device Actions**
Describe actions that this device is capable of receiving.

**Activate Manifest**
Publish this device manifest on the ARTIK Cloud platform.

Your manifest is ready to be activated and does not require approval before going live. Activating this manifest will not make your device type public.

<table>
<thead>
<tr>
<th>Fields</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>fine_dust_concentration</td>
<td>sleep</td>
</tr>
<tr>
<td>Double</td>
<td>Action</td>
</tr>
<tr>
<td>%</td>
<td>sleeptime</td>
</tr>
<tr>
<td></td>
<td>Parameter</td>
</tr>
<tr>
<td></td>
<td>Long</td>
</tr>
</tbody>
</table>
Add new device as my own device type

At my Artik page, we can add new device (see ARTIK 2 class)

We can fine our new device type.
Change device ID

- Use previous source in ARTIK 2 class
- Check device type is what you made.
- Change ‘Config.java’

```java
package cloud.artik.example.hellocloud;

class Config {
    static final String CLIENT_ID = "886f62c629e314da0b1329651a017b0c"; // AKA application ID
    static final String DEVICE_ID = "f620a7fa8f284a5586a0f825d23d6c96";

    // MUST be consistent with "AUTH REDIRECT URL" of your application set up at the developer.artik.cloud
    static final String REDIRECT_URI = "cloud.artik.example.hellocloud://oauth2callback";
}
```
Change posting data part

- Use your own device’s data field.
- Change ‘MessageActivity.java’

```java
private void postMsg() {
    final String tag = TAG + " sendMessageActionAsync";

    Message msg = new Message();
    msg.setSdid(Config.DEVICE_ID);
    msg.getData().put("fine_dust_concentration", 13.251);
}```
Send action and receive action
Action

• **Action** is a kind of command(명령) to your any devices.

• The format of actions can be defined at ‘device type’(8~9 page)

• When a device receive action, you can do something according to action’s contents.

• One device can **send action** to other device **manually**.

• Or you can set **rules** for **sending action automatically** at some condition.

• We already define action “sleep”(8~9 page)
Sending action

- Modify messageActivity.java

- We will use buttons of previous code.

- From now, we will make `postAction()`, `getLatestAction()`
Sending action

-postAction()

- Finally, we send Actions
- In one Actions, multiple and heterogeneous actions are can be included.
- You can set Destination and Source device.
  - Who send action?
  - Who receive action?

```java
private void postAction(){
    final String tag = TAG + "sendActionAsync";

    // Make action
    Action action = new Action();
    action.setName("sleep");
    action.getParameters().put("sleepTime", 10);

    // Add action to action array
    // You can add "multiple" and "hetero" actions in action array at one time.
    ActionArray actionArray = new ActionArray();
    actionArray.addActionItem(action);

    // Make action message. action message is sent by below function
    Actions actionsMsg = new Actions();
    actionsMsg.setData(actionArray);
    actionsMsg.setDid(Config.DEVICE_ID); // Destination device : who receive action?
    actionsMsg.setSdid(Config.DEVICE_ID); // Source device : who send action?
}
```

As we defined at 8~9page.
Sending action

postAction()

public void onSuccess(MessageEnvelope result, int statusCode, Map<String, List<String>> responseHeaders) {
    updateSendResponseOnUiThread(result.getData().toString());
}

public void onUploadProgress(Long bytesRead, Long contentLength, boolean done) {
}

public void onDownloadProgress(Long bytesRead, Long contentLength, boolean done) {
}

```java
// Make action message. action message is sent by below function
Actions actionsMsg = new Actions();
actionsMsg.setData(actionArray);
actionsMsg.setDid(Config.DEVICE_ID); // Destination device: who receive action?
actionsMsg.setDid(Config.DEVICE_ID); // Source device: who send action?

try{
    mMessagingApi.sendActionsAsync(actionsMsg, new ApiCallback<MessageEnvelope>() {
        @Override
        public void onFailure(ApiException e, int statusCode, Map<String, List<String>> responseHeaders) {
        }
        @Override
        public void onSuccess(MessageEnvelope result, int statusCode, Map<String, List<String>> responseHeaders) {
            updateSendResponseOnUiThread(result.getData().toString());
        }
        @Override
        public void onUploadProgress(Long bytesRead, Long contentLength, boolean done) {
        }
        @Override
        public void onDownloadProgress(Long bytesRead, Long contentLength, boolean done) {
        }
    });
}
catch (ApiException exc){
    processFailure(tag, exc);
}
Sending action

- use rule

- At Artik mypage
Sending action

- use rule

- At Artik mypage
- At right page, Artik automatically send action “sleep” when my_device send message with the higher value than 10 of ‘fine_dust_concentration’
- By this, you can automatically send action.
Receive action

- `getLatestAction()`

```java
private void getLatestAction()
{
    final String tag = TAG + " getNormalizedActionAsync";

    try{
        final int actionCount = 100;
        long ts = Calendar.getInstance().getTimeInMillis();

        MessagesApi.getNormalizedActionsAsync(null, Config.DEVICE_ID, null, null, actionCount,
                                               ts-3600000, ts, "desc", new ApiCallback<NormalizedActionsEnvelope>()
                                                   { @Override
                                                       public void onFailure(ApiException e, int statusCode, Map<String, List<String>> responseHeaders) {
                                                            // onFail
                                                            updateGetResponseOnUiThread("fail", "failure");
                                              }
    }
```

### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>가능한 값</th>
<th>설명</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserId</td>
<td>User id</td>
<td>특별한 경우 외 Null로 고정 할 것</td>
</tr>
<tr>
<td>Destination device id</td>
<td>Device id</td>
<td>어떤 디바이스로 보낸 액션을 가져 올 것인가</td>
</tr>
<tr>
<td>Mid</td>
<td>Msg id</td>
<td>특별한 경우 외 Null로 고정 할 것</td>
</tr>
<tr>
<td>offset</td>
<td>Int</td>
<td>특별한 경우 외 Null로 고정 할 것</td>
</tr>
<tr>
<td>Count</td>
<td>Int</td>
<td>몇 개 가져올 것인가</td>
</tr>
<tr>
<td>Start timestamp</td>
<td>Long</td>
<td>언제부터 발생한 액션을 가져올 것인가</td>
</tr>
<tr>
<td>End timestamp</td>
<td>Long</td>
<td>언제까지 발생한 액션을 가져올 것인가</td>
</tr>
<tr>
<td>order</td>
<td>&quot;asc&quot;,&quot;desc&quot;</td>
<td>오름차순, 내림차순</td>
</tr>
<tr>
<td>callback</td>
<td></td>
<td>Callback functions</td>
</tr>
</tbody>
</table>
Receive action

- getLatestAction()

As the actions are formed with Actions, ActionArray, we should parse it.

To get parameter in detail, see ‘getParameters()’ on the source code. It returns a map which contain parameters (ex – sleeptime in our device)
실행 화면
<table>
<thead>
<tr>
<th>DESTINATION DEVICE</th>
<th>SENT AT</th>
<th>SOURCE DEVICE</th>
<th>ACTION NAME</th>
<th>ACTION PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>My_Device</td>
<td>Jun 7 2017 14:08:46,327</td>
<td>My_Device</td>
<td>sleep</td>
<td>&quot;sleeptime&quot;:10</td>
</tr>
<tr>
<td>My_Device</td>
<td>Jun 7 2017 13:45:50,727</td>
<td>My_Device</td>
<td>sleep</td>
<td>&quot;sleeptime&quot;:5</td>
</tr>
<tr>
<td>My_Device</td>
<td>Jun 7 2017 13:41:08,671</td>
<td>My_Device</td>
<td>sleep</td>
<td>&quot;sleeptime&quot;:5</td>
</tr>
<tr>
<td>My_Device</td>
<td>Jun 7 2017 13:41:07,739</td>
<td>My_Device</td>
<td>sleep</td>
<td>&quot;sleeptime&quot;:5</td>
</tr>
</tbody>
</table>
THANK YOU!!!