

The logo features the text "NF 19" in a large, white, sans-serif font. The "NF" is positioned above the "19". This text is centered over a blue, wireframe-style globe. The globe is set against a dark background with a network of white dots and lines, resembling a digital or data network. A bright blue circular ring encircles the globe and the text.

NF
19

携手互
联无限可能



NF
19

携手无限
互联可能

Niagara与微信的 交互集成

黄维光 (Martin)

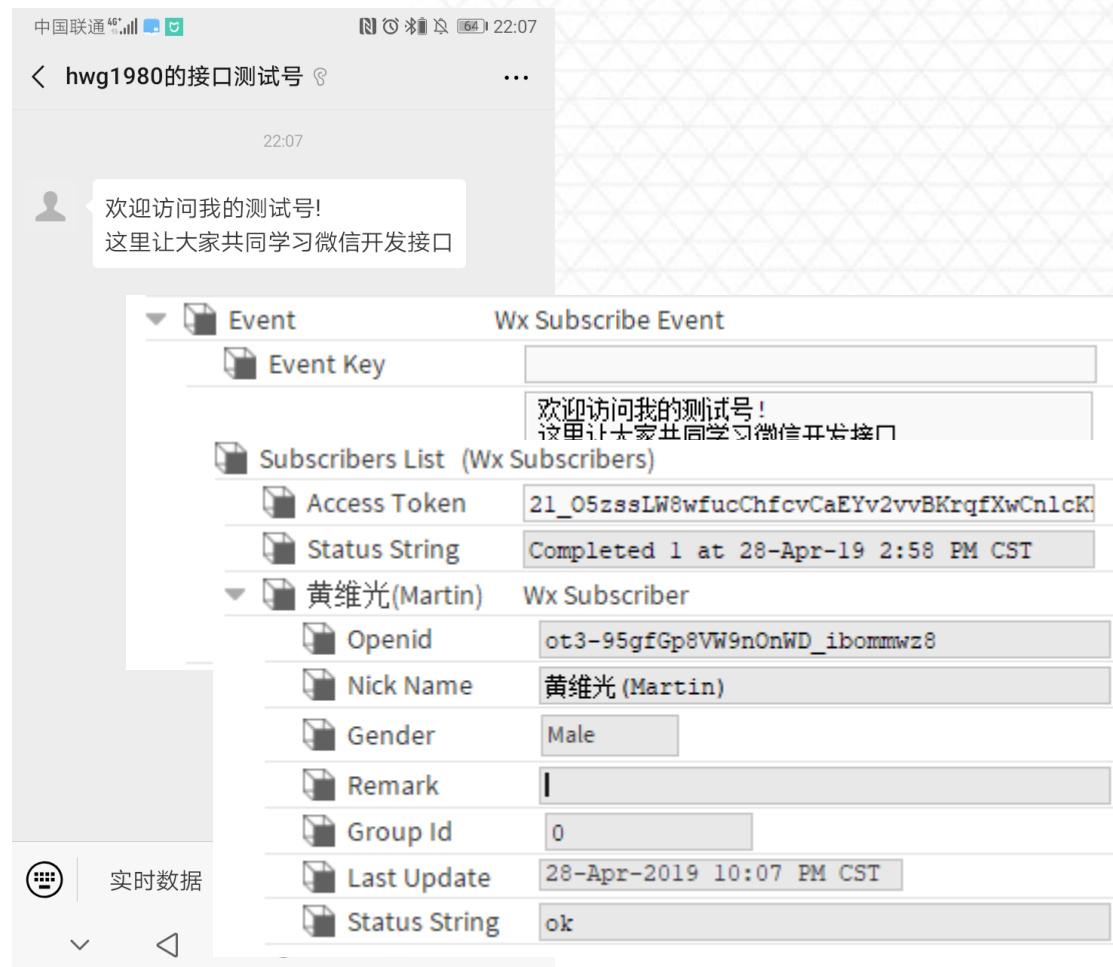
Tridium

- 微信 + Niagara 能做什么？
- 开发需要什么？
- 实战



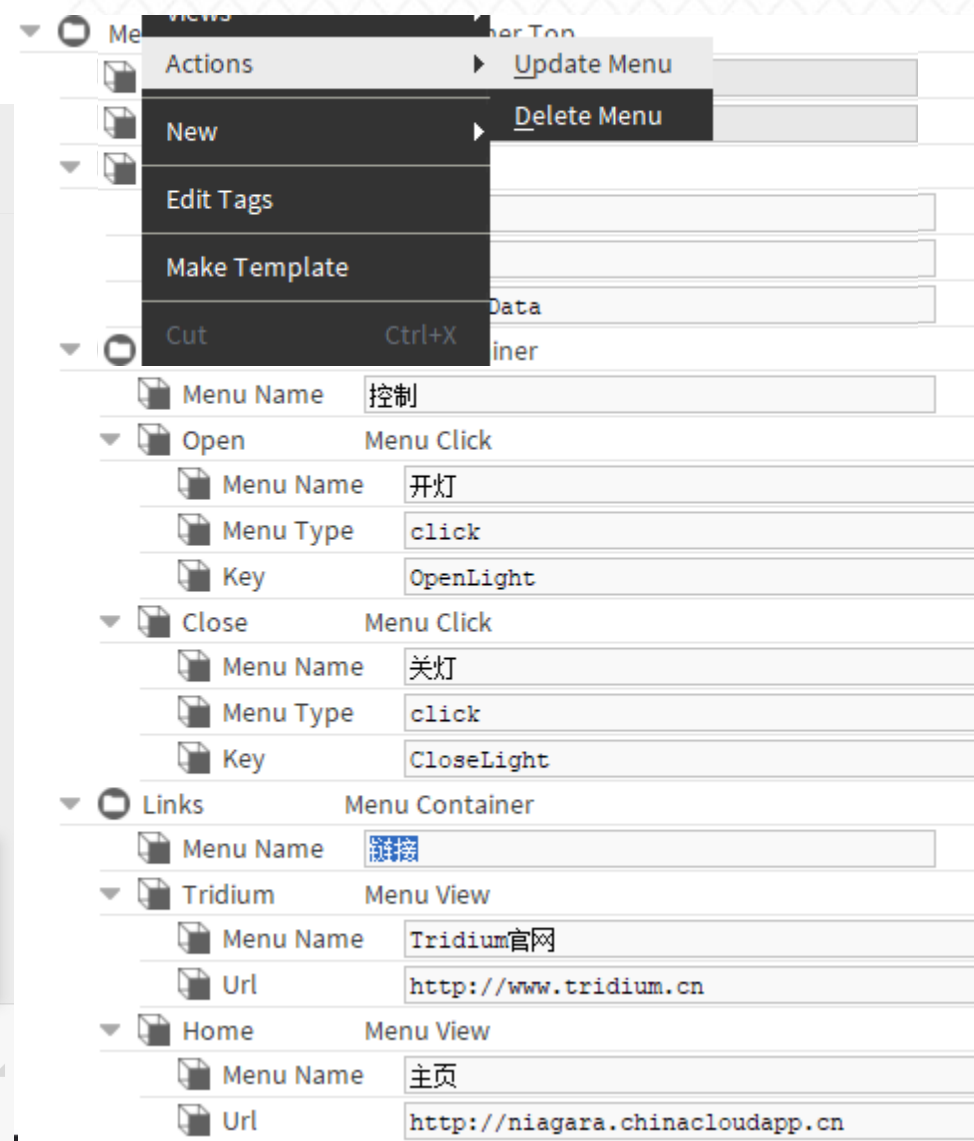
欢迎订阅

- 配置欢迎接口，内容可以定制
- 后台可以得到订阅者信息



定制菜单

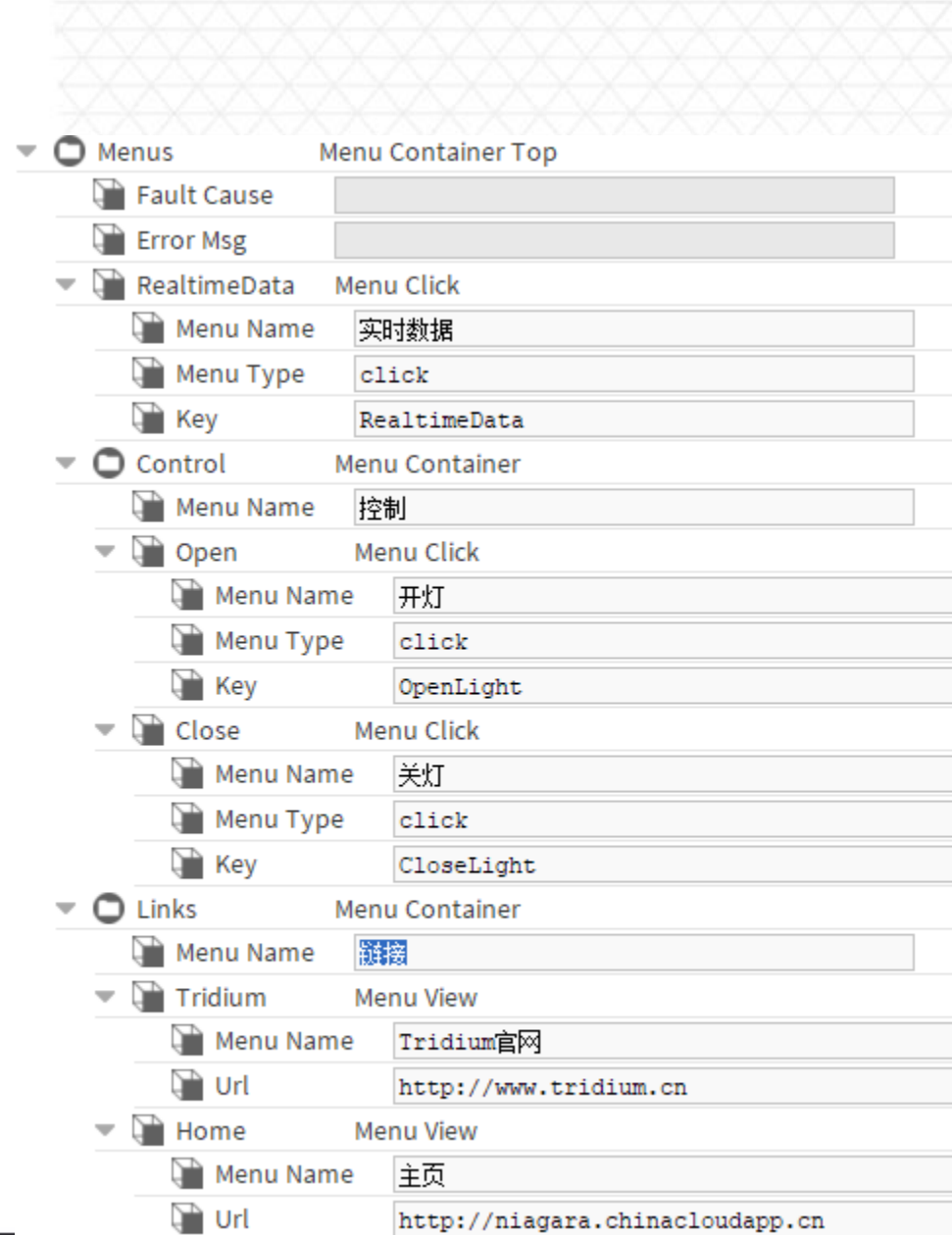
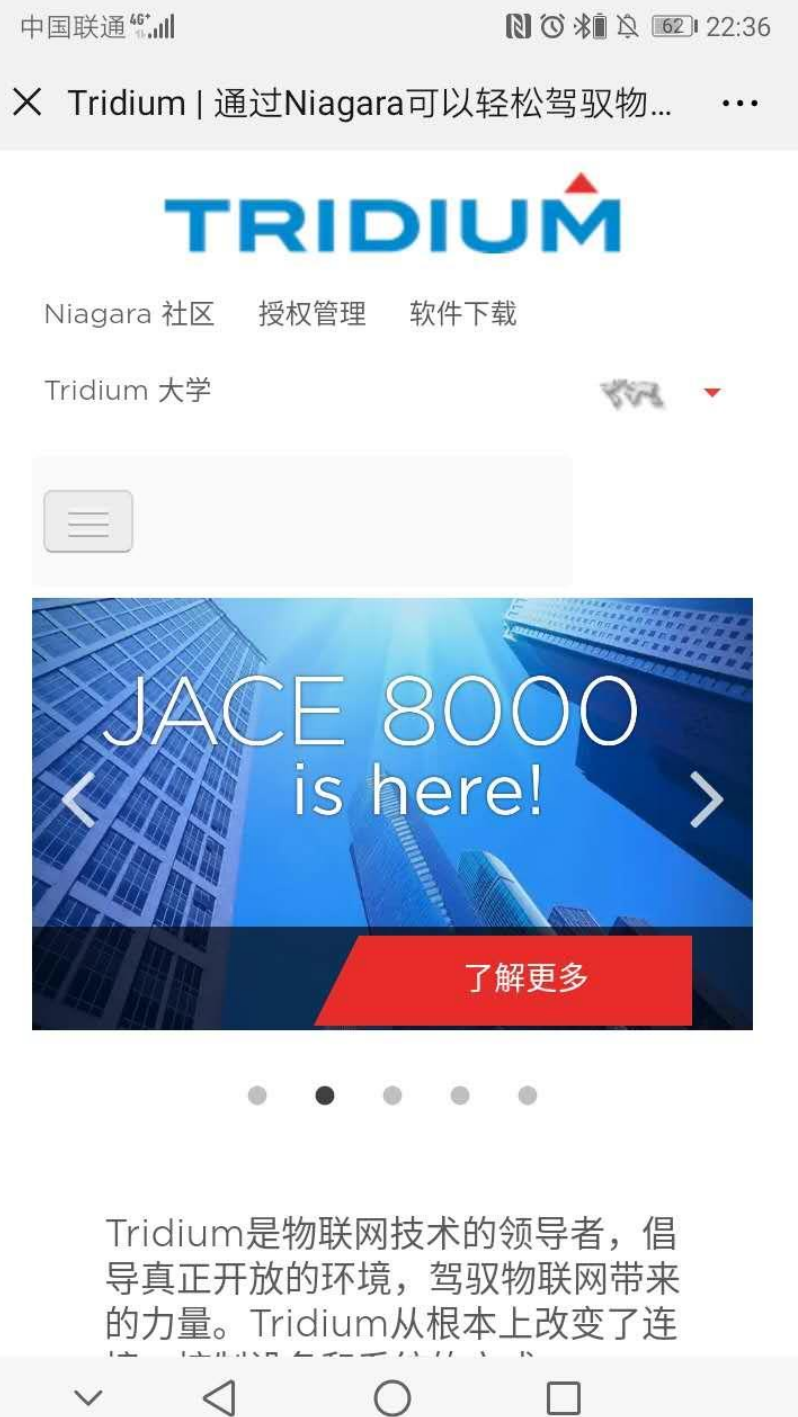
- Workbench中定制菜单



携手互联
无限可能

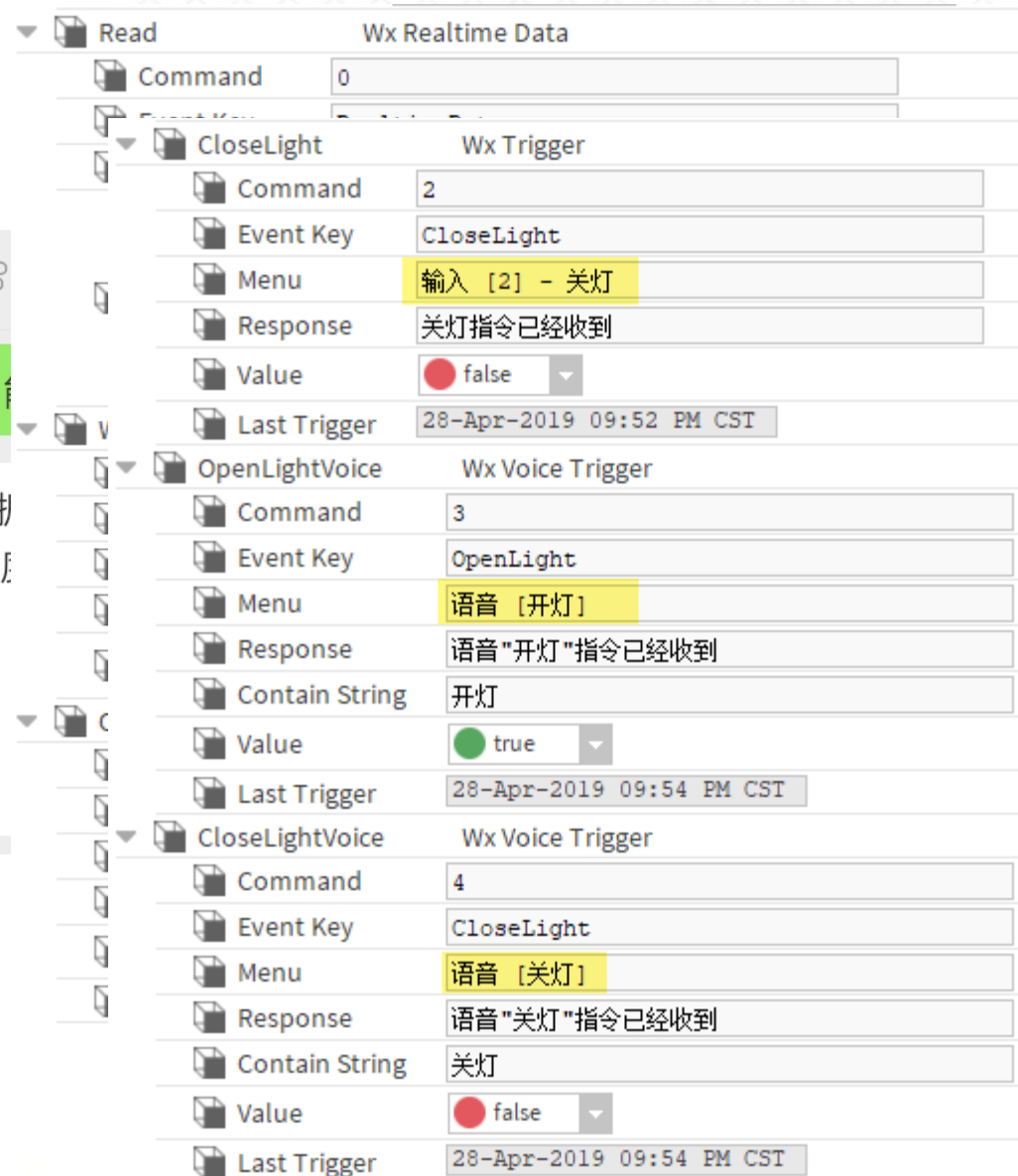
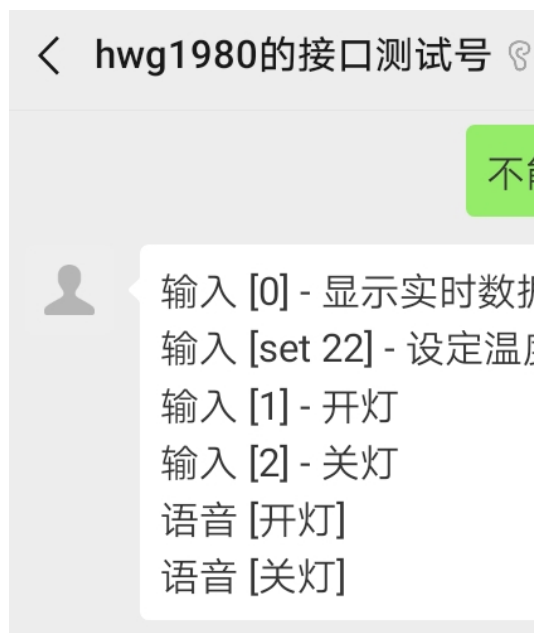
TRIDIUM

菜单交互



文字交互 - 提示菜单

- 用户输入一个未配置的信息时，回复提示



文字交互 - 实时数据

- 命令可配置
- 与菜单关联
- 数据可配置
- 数据订阅

中国联通 4G+ 21:55

< hwg1980的接口测试号 ...

不能识别的命令

输入 [0] - 显示实时数据
输入 [set 22] - 设定温度 22°C
输入 [1] - 开灯
输入 [2] - 关灯
语音 [开灯]
语音 [关灯]

0

Light:关 {ok} @ def
室温:84.6 °C {ok}
设定温度:23.0 °C {ok} @ def

Read Wx Realtime Data

Command	
Event Key	RealtimeData
Menu	输入 [0] - 显示实时数据
Data Ord List	station: slot:/wechat/Light station: slot:/wechat/Temp station: slot:/wechat/Setpoint

设定温度:23.0 °C {ok} @ def



携手互联
无限可能

MIUM

文字交互 - 数据设定

- 命令可配置
- 反馈信息可配置
- 数据可配置



Light:关 {ok} @ def
室温:13.4 °C {ok}
设定温度:23.0 °C {ok} @ def

set 25

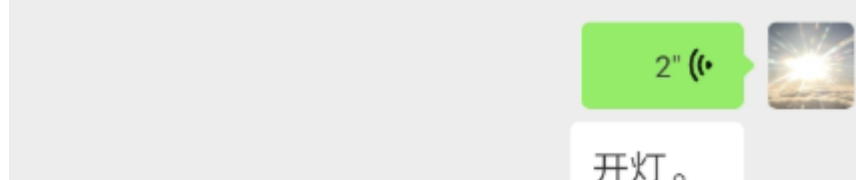
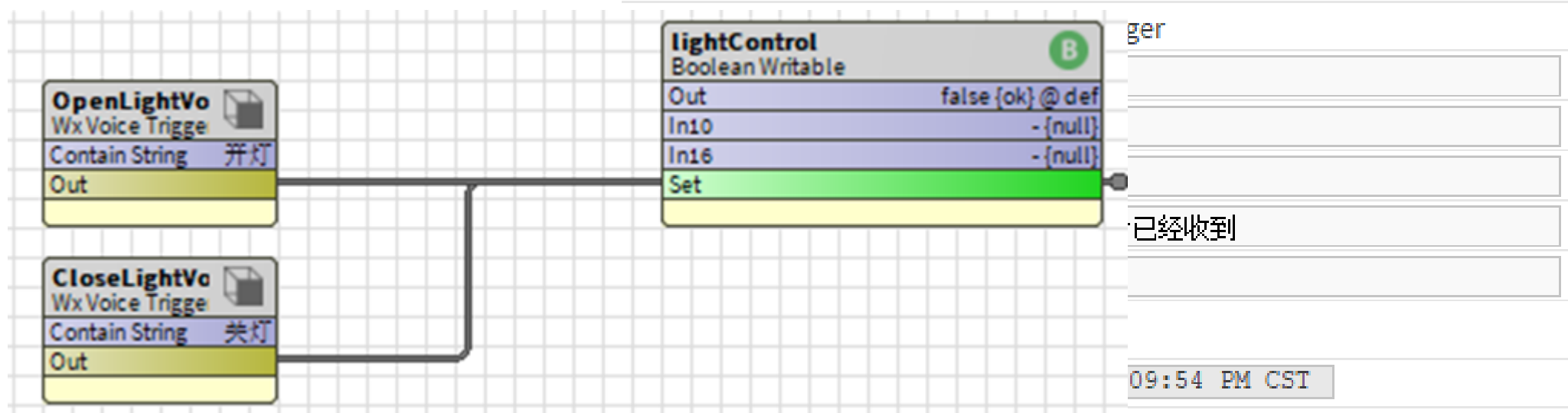
温度设定为: 25

Write	
Wx Write Data	
Command	set
Event Key	
Menu	输入 [set 22] - 设定温度22°C
Response	温度设定为:
Data Ord	station: slot:/wechat/Setpoint

Light:关 {ok} @ def
室温:0.8 °C {ok}
设定温度:55.0 °C {ok} @ def

语音控制

- 命令可配置
- 可与菜单关联
- 应答可配置
- 语音内容可配置



OpenLightVoice	Wx Voice Trigger
Command	3
Event Key	OpenLight
Menu	语音 [开灯]
Response	语音"开灯"指令已经收到
Contain String	开灯
Value	<input checked="" type="radio"/> true
Last Trigger	28-Apr-2019 09:54 PM CST

lightControl	Boolean Writable	B
Out	false {ok} @ def	
In10	- {null}	
In16	- {null}	
Set		
		已经收到
		09:54 PM CST

LightControl {ok} @ def	
室温:43.1 °C {ok}	
设定温度:55.0 °C {ok} @ def	

报警推送

- 报警格式可配置
- 报警模板可配置
- 接收人可选择

WechatAlarmRecipient (Wx Recipient)

Time Range

12:00 AM - 12:00 AM

Days Of Week

☒ Sun ☒ Mon ☒ Tue ☒ Wed ☒ Thu ☒ Fri ☒ Sat

Transitions

☒ toOffnormal ☒ toFault ☒ toNormal ☒ toAlert

Route Acks

☒ true

Access Token

21_05zssLW8wfucChfcvCaEYv2vvBKrqfXwCnlcKl

Subscribers Query

slot:/|bql:select * from wechat:WxSubscriber where remark='1

Message

{
 "touser": "{touser}",
 "template_id": "wgMDv9idScXEU-FQPgaAL",
 "url": "",
 "topcolor": "red",
 "data": {
 "first": {
 "value": "报警信息",
 "color": "#173177"
 }
 }
}

Response

报警时间: 28-Apr-19 6:03:04 PM CST
报警源: Fire001
信息: 火警
状态: Offnormal
测试

实时数据

控制

链接

- 微信 + Niagara 能做什么？
- 开发需要什么？
- 实战



了解 Niagara 应用与开发

- 熟悉 Niagara 基本应用
- 熟悉 JAVA 开发
- 熟悉 Niagara 面向组件的开发
- 了解 Servlet 编程
- 了解 REST 接口开发
- 了解 JSON 的编程
- 学习微信公众平台开发接口

https://mp.weixin.qq.com/wiki?t=resource/res_main&id=mp1421135319

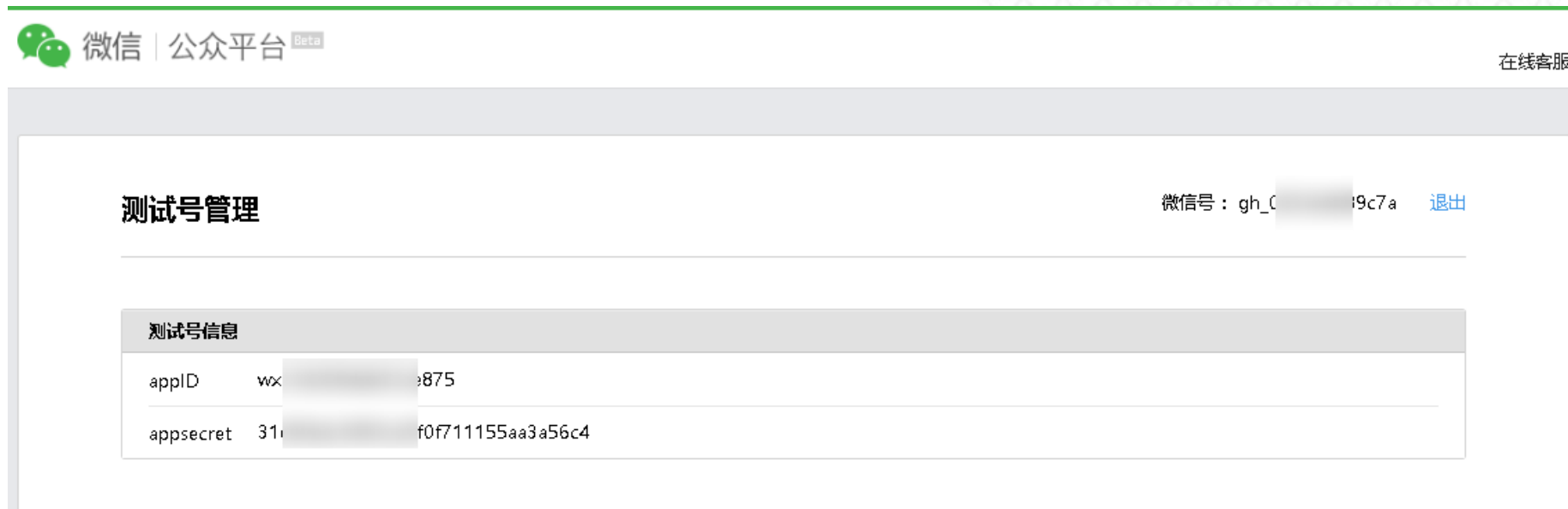


- 微信 + Niagara 能做什么？
- 开发需要什么？
- 实战



申请微信企业号

- 具体步骤网上搜索，这里截图是开发用的测试号
- 记住申请的appId和appsecret，用于获得动态密钥（Access Token），所有发给微信服务器的请求都需要校验动态密钥



申请微信企业号

- 指定Niagara服务器，开启80 web端口作为验证，/wechat/webchat为在Niagara上开发的免登陆Servlet
- Token需要和Niagara服务器上配置一致
- 注意：服务器需要先准备好，提交时就会进行验证

接口配置信息

请填写接口配置信息，此信息需要你有自己的服务器资源，填写的URL需要正确响应微信发送的Token验证，请阅读[消息接口使用指南](#)。

URL

Token

项目配置

- 设置项目依赖
- 增加Servlet配置

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <display-name>Wechat interface</display-name>
  <servlet>
    <servlet-name>wechat</servlet-name>
    <servlet-class>com.tridiumap.wechat.WxServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>wechat</servlet-name>
    <url-pattern>/wechat</url-pattern>
  </servlet-mapping>
</web-app>
```

```
dependencies {
  compile "Tridium:nre:4.4"
  compile "Tridium:baja:4.4"
  compile "Tridium:alarm-rt:4.4"
  compile "Tridium:control-rt:4.4"
  compile "Tridium:web-rt:4.4"
  compile "libraries:javafx.servlet-api-3.1.0:"
}

jar {
  from('src') {
    include 'WEB-INF/*.xml'
  }
}
```

权限申明

- 申明需要访问微信服务器
- 申请需要免登陆Servlet

```
<permissions>
  <niagara-permission-groups type="all">
    <!-- Insert any global permissions here. -->
  </niagara-permission-groups>
  <niagara-permission-groups type="workbench">
    <!-- Insert any workbench specific permissions here. -->
  </niagara-permission-groups>
  <niagara-permission-groups type="station">
    <req-permission>
      <name>NETWORK_COMMUNICATION</name>
      <purposeKey>Need communicate with WeChat servers.</purposeKey>
      <parameters>
        <parameter name="hosts" value="api.weixin.qq.com"/>
        <parameter name="ports" value="443"/>
        <parameter name="type" value="client"/>
      </parameters>
    </req-permission>
    <req-permission>
      <name>UNAUTHENTICATED_ACCESS</name>
      <purposeKey>WeChat interface needs unauthenticated servlet</purposeKey>
      <parameters/>
    </req-permission>
  </niagara-permission-groups>
</permissions>
```

创建无需登录Servlet

- 继承UnauthenticatedServlet
- 重写doGet
 - 处理微信服务器对Niagara认证
- 重写doPost
 - 处理订阅者发给公众号的请求



接口配置信息

请填写接口配置信息，此信息需要你有自己的服务器资源，填写的URL需要正确响应

URL

Token

提交

```
public class WxServlet extends UnauthenticatedServlet {  
    @Override  
    protected void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        // 处理微信认证  
    }  
  
    @Override  
    protected void doPost(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        // 处理订阅者发给公众号的请求  
        // 此处可能根据您的业务需求做得相当复杂  
    }  
}
```

< hwg1980的接口测试号

不能识别的命令

输入 [0] - 显示实时数据
输入 [set 22] - 设定温度 22°C
输入 [1] - 开灯
输入 [2] - 关灯
语音 [开灯]
语音 [关灯]

获取Access Token

- 微信服务器的Access Token是Niagara向微信服务器申请的验证码，每两个小时必须重新申请一次。

```
public void doExecute()
{
    try{
        String sUrl = "https://api.weixin.qq.com/cgi-bin/token?grant_type=client_credential&appid=" +
            getAppid() +
            "&secret=" + getSecret();
        String sRes = HttpClient.Send(HttpClient.ACTION_GET, sUrl, "", null, null);
        setResponse(sRes);

        parse(sRes);
    } catch (Exception e) {}
}
```

微信 | 公众平台

测试号管理

测试号信息

appid	wx	875
appsecret	31	f0f711155aa3a56c4

niagara⁴



```
{
  "access_token": "当前申请的ACCESS_TOKEN",
  "expires_in": 7200
}
```


推送报警示例

- Niagara发给微信服务器
- 微信服务器根据信息发给用户
- 该代码在自定义的报警接收器里被调用

```
public void doAsyncRoute(BAlarmRecord balarmrecord) {  
    String sMessage;  
    sMessage = getMessage().format(balarmrecord); // 使用format, 这样可以自由将报警记录中的信息组织成 JSON  
  
    BWxSubscriber[] array = getToArray();  
    setResponse("");  
    String sRes = "";  
    // 请求需要带上 access_token  
    String sUrl = "https://api.weixin.qq.com/cgi-bin/message/template/send?access_token=" + getAccessToken();  
    for(int i=0;i<array.length;i++){  
        String sSend = sMessage.replace("{touser}", array[i].getOpenid());  
        sRes += HttpClient.Send(HttpClient.ACTION_POST,sUrl,sSend, null, null);  
    }  
    setResponse(sRes);  
}
```

niagara⁴



更多高级功能

- 根据Session控制
- 微信人工智能接口
- 结合其他资源接口
 - 例如语音转发到智能音箱
 - 使用智能音箱回复微信请求
- 网站安全接口

开灯

开什么地方的灯?(三层走廊/Tridium办公室/Niagara会议室)

三层走廊

准备开启三层走廊的灯,是否确认?

确认

三层走廊的灯已经开启

更多高级功能

- 还有更多接口, 发挥您的想象, 结合Niagara的优势, 还有更多应用可以开发

<https://mp.weixin.qq.com/wiki>

自定义菜单	▼
消息管理	▼
微信网页开发	▼
素材管理	▼
图文消息留言管理	▼
用户管理	▼
帐号管理	▼
数据统计	▼
微信卡券	▼
微信门店	▼

微信小店	▼
智能接口	▼
微信设备功能	▼
新版客服功能	▼
微信摇一摇周边	▼
微信连Wi-Fi	▼
微信扫一扫	▼
微信发票	▼
其他文档	▼
微信商品	▼
微信非税缴费	▼

总结

开发要求
功能

微信→Niagara服务器：认证服务器、命令与控制

Niagara服务器→微信：服务请求、报警推送



携手互联
无限可能

TRIDIUM