# 前言

## 修改记录

| 版本编号 | *变化状态 | 变更内容和范围 | 变更日期 | 变更人 | 批准日期 | 批准人 |
|---|---|---|---|---|---|---|
| 1.0.0 | A | 新增 | 2022-7-8 | 高玉坤 | | 程揭章 |
| 1.0.1 | M | 修改 | 2023-3-21 | 高玉坤 | | 程揭章 |
| 1.1.0 | M | 修改 | 2023-8-21 | 马龙帮 | | 程揭章 |

> *变化状态：A——增加，M——修改，D——删除，N——正式发布

# 1 空间数据服务数据共享

## 1.1 WMS规范（网络地图服务）

**简要描述**

- Web Map Service（网络地图服务），简称WMS，由开放地理信息联盟（Open GeoSpatial Consortium，OGC）制定。该规范定义了Web客户端从网络地图服务器获取地图的接口标准。一个WMS可以动态地生成具有地理参考数据的地图，这些地图通常用GIF、JPEG或PNG等图像格式，或者SVG、KML、VML和WebCGM等矢量图形格式来表现。使用者通过指定的参数获取相应的地图图片。

## 1.2 WMTS规范（网络地图瓦片服务）

**简要描述**

- Web Map Tile Service（网络地图瓦片服务），简称WMTS，由开放地理信息联盟（Open GeoSpatial Consortium，OGC）制定，是和WMS并列的重要OGC规范之一。WMTS不同于WMS，它最重要的特征是采用缓存技术能够缓解WebGIS服务器端数据处理的压力，提高交互响应速度，大幅改善在线地图应用客户端的用户体验。WMTS是OGC主推的缓存技术规范，是目前各种缓存技术相互兼容的一种方法。 WMTS服务支持RESTful访问，其接口包括GetCapabilities、GetTile和GetFeatureInfo3个操作，这些操作允许用户访问切片地图。

## 1.3 WFS规范（网络要素服务）

**简要描述**

- Web Feature Service（网络要素服务），简称WFS，由开放地理信息联盟（Open GeoSpatial Consortium，OGC）制定。该规范主要对OpenGIS简单要素的数据编辑操作进行规范，从而使服务器端和客户端能够在要素层面进行“通讯”。其返回结果的XML格式的WFS服务元数据文档，通过该文档用户能够了解：WFS服务器支持的所有操作操作列表，GetFeature操作返回的数据格式，可用的坐标参照系统列表，操作异常信息的列表，WFS服务提供商的相关信息，WFS服务器的可用要素类列表等。
- WFS 通过 GML（Geography Markup Language，地理标记语言）传递地理空间数据，它支持在基于 HTTP 协议对地理要素进行插入（INSERT）、更新（UPDATE）、删除（DELETE）和发现（DISCOVERY）等操作，并且在这些操作的过程中保证了地理数据变化的一致性。

## 1.4 TMS规范（瓦片地图服务）

**简要描述**

- TMS与WMTS都是针对瓦片地图服务的推出的规范，两者的相似性大，主要差异如下：
- 由不同组织推进制定：WMTS是OGC的标准（开放地理信息联盟，Open GeoSpatial Consortium，其主要目标是制定地理标准），TMS是OSGEO的标准（OSGeo，开源空间信息基金会，是一个全球性非营利性组织，目标是支持全球性的合作,建立和推广高品质的空间信息开源软件。）
- 协议差异：TMS是纯RESTful的；而WMTS可以有三种：KVP SOAP RESTful。
  瓦片的组织方式：TMS瓦片是正方形 WMTS瓦片是矩形；在纵轴方向上面相反；WMTS中对应的不同比例尺瓦片可以尺寸不同。

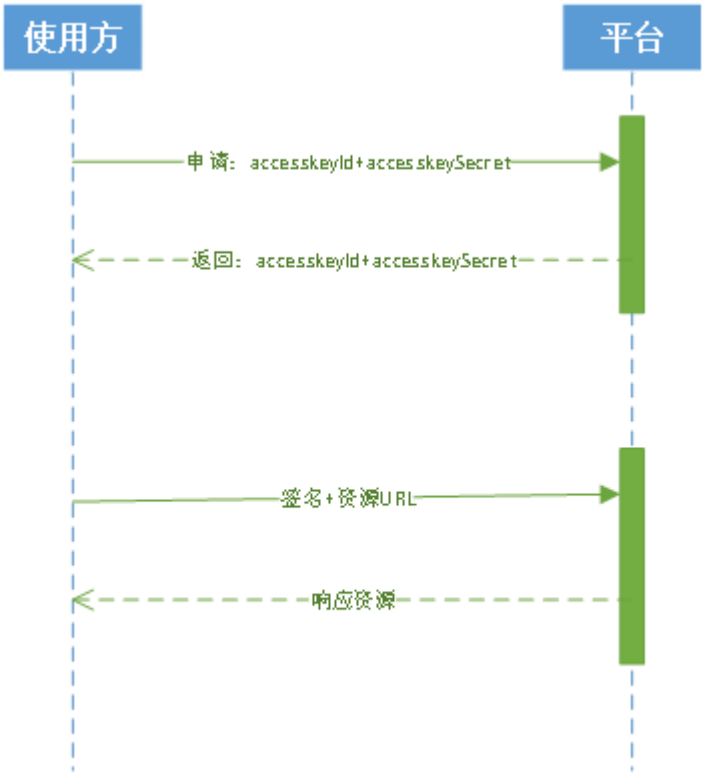
## 1.5 在线文档服务地址

http://icenter.geovis.online/static/icdocs/index.html#/api/geoserverplus/geoserverplus-guide/

# 2 非空间数据服务数据共享

## 2.1 公共说明

### 2.1.1 数据交互模型




注意：accessKeyId和accessKeySecret 平台申请

### 2.1.2 请求Header

- 应用与平台交互中，应用需要在每个请求Header中传入一些固定的参数；平台的每个响应中也会包含固定的响应码，具体如下：

> 输入参数,请求中需要传入

| 参数名 | 类型 | 位置 | 必填 | 说明 |
| --- | --- | --- | --- | --- |
| accessKeyId | String | Header | 是 | 授权ID，平台全局唯一。 |
| version | String | Header | 是 | 最新隐私协议版本，当前默认 1.0,写死即可。 |
| sequenceId | String | Header | 是 | 报文流水(客户端唯一)客户端交易流水号。6-32位。由客户端自行定义，自行生成。建议使用日期+顺序编号的方式。 |
| sign | String | Header | 是 | 对请求进行签名运算产生的签名,签名算法见2.1.4章节。 |
| timestamp | String | Header | 是 | Unix时间戳，精确到毫秒。 |
| Content-Type | String | Header | 是 | 必须为application/json;charset=UTF-8 |

### 2.1.3 响应参数

> 输出参数，平台响应中会包含

| 参数名 | 类型 | 位置 | 必填 | 说明 |
| --- | --- | --- | --- | --- |
| code | String | Body | 是 | 返回码（其中200代表请求成功,其它代表错误，错误码及描述见附录错误码表） |
| msg | String | Body | 是 | 用于调试的返回信息，不支持国际化，也不能直接显示在UI上 |
| data | Object | Body | 否 | 接口正常响应数据 |

### 2.1.4 签名认证

**说明**

- 调用方需要对发送到平台的请求进行签名，执行签名计算的签名值需要赋值到Header头中的sign属性（见公共部分说明），以便服务端进行签名验证。

**参数介绍**

- **待签名字符串为：**url字符串 +accesskeyId + accesskeySecret + timestamp；

- **url 字符串**：指请求的接口地址去除https://域名+端口 后剩余的路径部分；
  如：http://192.168.131.21:30280/api/view/findXData/7b277461626c654e616d65273a2763697479272c27646254797065273a276d7973716c272c2764624e616d65273a2767656f7669732d64632d62696764617461277d?pageNum=1&pageSize=30
  签名计算时
  取/view/findXData/7b277461626c654e616d65273a2763697479272c27646254797065273a276d7973716c272c2764624e616d65273a2767656f7669732d64632d62696764617461277d固定部分 (即/api后面部分)
- **accesskeyId**：Header头中的属性（见公共部分说明）；
- **accesskeySecret**：在平台申请的accesskeySecret，不能作为参数发送仅做签名使用；
- **timestamp**：Header头中的属性（见公共部分说明）；
- 注意：参数在url中的情况下 只去固定部分计算签名即可；

**签名算法示例**

```
public static String generate(String AccessKeyId, String AccessKeySecret, String
timestamp) {
        String sign = "";
        try {
        AccessKeySecret = AccessKeySecret.trim();
        AccessKeySecret = AccessKeySecret.replaceAll("\"", "");
        StringBuffer sb = new StringBuffer();
        sb.append(AccessKeyId).append(AccessKeySecret).append(timestamp);
        sign = Hex.encodeHexString(MessageDigest.getInstance("SHA-
256").digest(sb.toString().getBytes("utf-8")));
        } catch (NoSuchAlgorithmException | UnsupportedEncodingException e) {
        e.printStackTrace();
        }
        return sign;
        }
```

### 2.1.5 返回码列表公共错误码

| 错误码(code) | 描述(msg) |
| --- | --- |
| 200 | 操作成功 |
| 201 | 对象创建成功 |
| 202 | 请求已经被接受 |
| 204 | 操作已经执行成功，但是没有返回数据 |
| 301 | 资源已被移除 |
| 303 | 重定向 |
| 304 | 资源没有被修改 |
| 400 | 参数列表错误（缺少，格式不匹配） |
| 401 | 未授权 |

| 错误码(code) | 描述(msg) |
| --- | --- |
| 403 | 访问受限，授权过期 |
| 404 | 资源，服务未找到 |
| 405 | 不允许的http方法 |
| 409 | 资源冲突，或者资源被锁 |
| 415 | 不支持的数据，媒体类型 |
| 500 | 系统内部错误 |
| 501 | 接口未实现 |

## 2.2 数据共享服务接口

### 2.2.1 数据资产数据共享

---

**接口定义**

- 接口名称：`数据资产数据共享`
- 接入地址：`/view/findXData`
- 请求方法：`POST`
- 功能描述：`获取数据资产信息`
- 授权验证：`是`

**输入参数**

| 参数名 | 类型 | 位置 | 必填 | 说明 |
| --- | --- | --- | --- | --- |
| pageNum | int | Path | 是 | 分页：当前页码 1 开始 |
| pageSize | int | Path | 是 | 分页：每页数据量 |
| tableName | String | Body | 是 | 表名 |
| dbType | String | Body | 是 | 数据库类型mysql、postgresql、mongodb |
| dbName | String | Body | 是 | 数据库名（geovis-dc-original , geovis-dc-bigdata , geovis-dc-smart） |
| matchingType | String | Body | 否 | 匹配模式like:模糊匹配 all：全量匹配 |
| columnName | String | Body | 否 | 字段名 |
| columnCondition | String | Body | 否 | 匹配值 |

**请求样例**

- 接入地址：http://192.168.131.21:30280/api/view/findXData/7b277461626c654e616d65273a2763697479272c27646254797065273a276d7973716c272c2764624e616d65273a2767656f7669732d64632d62696764617461277d?pageNum=1&pageSize=30
- 请求方法：`POST`
- 请求头:

```
Connection: keep-alive
AccessKeyId: 3o8Omxreadyagomr
sequenceId: 20161020153428000015
sign: da55be21096d188394c39dd307e7ce7aa3e4c5c38f9f171da39d3a151d0595bb
timestamp: 1533882163013
Content-type: application/json
version: 1.0
```

- 请求体:

```
{
    "conditionDtos": [{
        "columnCondition": "2566779834",
        "columnName": "code",
        "matchingType": "like"
    },
    {
        "columnCondition": "",
        "columnName": "city",
        "matchingType": "like"
    }]
}
```

- 返回结果:

```
{
    "total": 1,
    "data": [{
        "update_time": null,
        "code": "2566779834",
        "weather_station": "1",
        "province": "北京",
        "create_time": "2022-06-30 16:10:22",
        "city": "昌平",
        "station_url": "2",
        "lon": "116.235906",
        "id": 3371,
        "lat": "40.218085"
    }],
    "code": 200,
    "msg": "查询成功"
}
```

## 2.3 示例代码

### 2.3.1 java示例代码

```
import cn.hutool.http.HttpUtil;
import org.apache.commons.codec.binary.Hex;
import java.io.UnsupportedEncodingException;
import java.net.MalformedURLException;
import java.net.URL;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class TestSign {

    public static void main(String[] args) {
        String url = "http://192.168.131.21:30280/api/view/findXData?
pageNum=1&pageSize=10";
        //  String url = "http://127.0.0.1:8086/findXData?pageNum=1&pageSize=10";
        String AccessKeyId = "8wvwa2g2wxa3aOnk";
        String AccessKeySecret = "5cb9dOb07c05160d977d284d6274ee96";
        String timestamp = System.currentTimeMillis() + "";
        System.out.println(timestamp);
        String sign = generate(AccessKeyId, AccessKeySecret, timestamp);
        System.out.println(sign);
        String body = "{\"conditionDtos\":
[{\"columnCondition\":\"\",\"columnName\":\"code\",\"matchingType\":\"like\"},
{\"columnCondition\":\"\",\"columnName\":\"city\",\"matchingType\":\"like\"}],\"d
bName\":\"geovis-dc-
bigdata\",\"dbType\":\"mysql\",\"tableName\":\"city\",\"primarykey_orderby\":true
}";
        System.out.println(body);
        // 某些接口对Accept头有特殊要求，此处自定义头
        String result = HttpUtil
                .createPost(url)
                .header("accessKeyId", AccessKeyId)
                .header("timestamp", timestamp)
                .header("sequenceId", timestamp)
                .header("sign", sign)
                .header("version", "1.0")
                .header("Content-Type", "application/json;charset=UTF-8")
                .body(body)
                .execute()
                .body();
        System.out.println(result);
    }

    public static String generate(String AccessKeyId, String AccessKeySecret,
String timestamp) {
        String sign = "";
        try {
            AccessKeySecret = AccessKeySecret.trim();
            AccessKeySecret = AccessKeySecret.replaceAll("\"", "");
            StringBuffer sb = new StringBuffer();
            sb.append(AccessKeyId).append(AccessKeySecret).append(timestamp);
```

```
            sign = Hex.encodeHexString(MessageDigest.getInstance("SHA-
256").digest(sb.toString().getBytes("utf-8")));
        } catch (NoSuchAlgorithmException | UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return sign;
    }
}
```

### 2.3.2 javascript 示例代码

1.使用npm安装： npm install js-sha256

```
     //测试sha-256加密
      let url = "http://192.168.131.21:30280/api/view/findXData?
pageNum=1&pageSize=10"
      let AccessKeyId = "3o80mxreadyagomr"
      let AccessKeySecret = "9fbd6be7900b6c0d6bf7a2de132d63e5"
      let timestamp = new Date().getTime()
      console.log(timestamp)
      let sb  = AccessKeyId+AccessKeySecret+timestamp;
      var str=sha256.digest(sb)
      var hexCharCode = [];
      var chars =
["0","1","2","3","4","5","6","7","8","9","a","b","c","d","e","f"];
      for(var i = 0; i < str.length; i++) {
        var bit = (str[i] & 0x0f0) >> 4;
        hexCharCode.push(chars[bit]);
        var bit = str[i] & 0x0f;
        hexCharCode.push(chars[bit]);
      }
      let sign=hexCharCode.join("");
      console.info(sign)
      axios.post(url,
          {
            "conditionDtos": [{
              "columnCondition": "",
              "columnName": "code",
              "matchingType": "like"
            },
              {
                "columnCondition": "",
                "columnName": "city",
                "matchingType": "like"
              }],
            "dbName": "geovis-dc-bigdata",
            "dbType": "mysql",
            "tableName": "city"
          },
          {
            headers : {
              'accessKeyId': AccessKeyId,
              'timestamp': timestamp,
              'sequenceId': timestamp,
              'sign':sign,
```

```
            'version':'1.0',
            'Content-Type':'application/json;charset=UTF-8'
          }}
    ).then((res) => {
      console.log(res.data)
    })
```