更新白名单

接口描述

```
1、更新白名单
2、(channelId, timestamp, appId) 参与sign签名，并和sign一起通过url传递，请求体参数不参与签名，通过post请求体传递【请设置请求头contentType:application/json】
3、接口支持https协议
4、接口更新为局部更新，即不传的字段维持不变
```

接口URL

```
https://api.polyv.net/meet/v1/channel/auth-whitelist/update
```

请求方式

```
POST
```

接口约束

1、接口调用有频率限制，详细请查看

请求参数描述

参数名	必选	类型	说明
appId	true	String	账号appId【详见获取密钥】
timestamp	true	Long	当前13位毫秒级时间戳，3分钟内有效
sign	true	String	签名，为32位大写的MD5值【详见签名生成规则】
channelId	true	Integer	频道号

请求体参数描述

参数名	必选	类型	说明
oldCode	true	String	需要更新的原白名单记录，长度不超过64位
code	true	String	新的白名单记录，长度不超过64位
name	false	String	新的昵称，长度不超过64位（不传则不修改）
groupNo	false	Integer	新的分组序号，1~50之间（不传则不修改）
groupRole	false	String	新的组内身份（不传则不修改） leader：组长 member：组员

示例

```
https://api.polyv.net/meet/v1/channel/auth-whitelist/update?
appId=fyirmfdak4&timestamp=1657786379352&sign=CEA301164F9BA5076F156C1EB60F22F8&channelId=3220878
```

请求体参数

```
{
  "oldCode" : "oldCode",
  "code" : "code",
  "name" : "name",
  "groupNo" : 1,
  "groupRole" :"member"
}
```

响应参数描述

参数名	类型	说明
code	Integer	状态码，与 http 状态码相同，用于确定基本的响应状态

参数名	类型	说明
status	String	响应结果，由业务决定，成功返回success，失败返回error
success	Boolean	是否成功响应
requestId	String	请求ID，每次请求生成的唯一的UUID，仅可用于排查、调试，不应该和业务挂上钩
error	Object	状态码非200时的错误信息【详见Error字段描述】
data	String	成功时为true，错误时为空

Error参数描述

参数名	类型	说明
code	Integer	错误代码，用于确定具体的错误原因
desc	String	错误描述，与 error.code 对应

Java请求示例

快速接入基础代码请下载相关依赖源码， 点击下载源代码 ,下载后加入到自己的源码工程中即可。测试用例中的**HttpUtil.java 和 LiveSignUtil.java** 都包含在下载文件中。

> 强烈建议您使用直播Java SDK完成API的功能对接，直播Java SDK 对API调用逻辑、异常处理、数据签名、HTTP请求线程池进行了统一封装和优化。

```
private static final Logger log = LoggerFactory.getLogger(WhitelistUpdate.class);
/**
 * 更新白名单
 * @throws IOException
 */
@Test
public void testWhitelistUpdate() throws IOException, NoSuchAlgorithmException {
    //公共参数,填写自己的实际参数
    String appId=super.appId;
    String appSecret=super.appSecret;
    String userId = super.userId;
    String timestamp=String.valueOf(System.currentTimeMillis());
    //业务参数
    String url = "https://api.polyv.net/meet/v1/channel/auth-whitelist/update";
    String channelId = "2191532";
    String body = "{\n" +
        "  \"oldCode\" : \"oldCode\",\n" +
        "  \"code\" : \"code\",\n" +
        "  \"name\" : \"name\",\n" +
        "  \"groupNo\" : 1,\n" +
        "  \"groupRole\" :\"member\"\n" +
        "}";

    //http 调用逻辑
    Map<String,String> requestMap = new HashMap<>();
    requestMap.put("appId", appId);
    requestMap.put("timestamp",timestamp);
    requestMap.put("channelId",channelId);

    requestMap.put("sign",LiveSignUtil.getSign(requestMap, appSecret));
    url = HttpUtil.appendUrl(url, requestMap);
    String response = HttpUtil.postJsonBody(url, body, null);
    log.info("测试更新白名单接口返回值：{}",response);
    //do somethings

}
```

响应示例

系统全局错误说明详见全局错误说明

成功示例

```
{
    "code": 200,
    "status": "success",
    "message": "",
    "data": true
}
```

异常示例


```
{
  "code": 400,
  "status": "error",
  "error": {
    "code": 10003,
    "desc": "时间戳过期"
  },
  "success": false
}
```

```
{
  "code": 403,
  "status": "error",
  "error": {
    "code": 10002,
    "desc": "签名错误"
  },
  "success": false
}
```

```
{
  "code": 400,
  "status": "error",
  "requestId": "a7d7886f4a8c4938819b48c1b969ac04.66.16655445009590005",
  "error": {
    "code": 10001,
    "desc": "原始会员码不能为空"
  },
  "data": null,
  "success": false
}
```

```
{
  "code": 400,
  "status": "error",
  "requestId": "a7d7886f4a8c4938819b48c1b969ac04.63.16655444886950005",
  "error": {
    "code": 10001,
    "desc": "会员码不能为空"
  },
  "data": null,
  "success": false
}
```

```
{
  "code": 400,
  "status": "error",
  "requestId": "a7d7886f4a8c4938819b48c1b969ac04.57.16655448630610035",
  "error": {
    "code": 10001,
    "desc": "会员码长度需要介于1和64之间"
  },
  "data": null,
  "success": false
}
```

```
{
  "code": 400,
  "status": "error",
  "requestId": "a7d7886f4a8c4938819b48c1b969ac04.59.16655445255610009",
  "error": {
    "code": 10001,
    "desc": "昵称长度需要介于1和64之间"
  },
  "data": null,
  "success": false
}
```

```
{
  "code": 400,
  "status": "error",
  "requestId": "a7d7886f4a8c4938819b48c1b969ac04.58.16655455179100093",
  "error": {
    "code": 10001,
    "desc": "组内身份不正确"
  },
  "data": null,
  "success": false
}
```

```
{
  "code": 400,
  "status": "error",
  "requestId": "a7d7886f4a8c4938819b48c1b969ac04.61.16655444651140003",
  "error": {
    "code": 50202,
    "desc": "白名单不存在"
  },
  "data": null,
  "success": false
}
```

```
{
  "code": 400,
  "status": "error",
  "requestId": "a7d7886f4a8c4938819b48c1b969ac04.60.16655454803730087",
  "error": {
    "code": 50203,
    "desc": "会员码已存在"
  },
  "data": null,
  "success": false
}
```

```
{
  "code": 400,
  "status": "error",
  "requestId": "a7d7886f4a8c4938819b48c1b969ac04.63.16655455046100091",
  "error": {
    "code": 50204,
    "desc": "更新白名单失败，组内已有组长"
  },
  "data": null,
  "success": false
}
```