获取直播中视频片段的接口

接口URL

```
https://api.polyv.net/live/v2/channel/recordFile/{channelId}/streamRecordIndex
```

接口说明

```
1、获取当前直播频道直播中的视频片段
2、接口支持https协议
```

支持格式

```
JSON
```

请求方式

```
GET or POST
```

请求数限制

```
TRUE
```

请求参数

参数名	必选	类型	说明
appId	是	string	从API设置中获取，在直播系统登记的appId

参数名	必选	类型	说明
timestamp	是	string	当前13位毫秒级时间戳，3分钟内有效
startTime	是	string	视频片段的开始时间，格式为yyyy-MM-dd HH:mm:ss
endTime	是	string	视频片段的结束时间，格式为yyyy-MM-dd HH:mm:ss
fileName	否	string	转存点播的文件名，不保存默认视频名称为"精彩片段"
cataId	否	long	转存点播存放的目录ID
sign	是	String	签名，为32位大写的MD5值，生成签名的appSecret密钥作为通信数据安全的关键信息，严禁保存在客户端直接使用，所有API都必须通过客户自己服务器中转调用POLYV服务器获取响应数据 【详见签名生成规则】

响应成功JSON示例：

```
{
  "code": 200,
  "status": "success",
  "message": "",
  "data": "8205ac89d3f0c634988ef97f528aa745_8"
}
```

appId不正确

```
{
    "code": 400,
    "status": "error",
    "message": "application not found.",
    "data": ""
}
```

时间戳错误

```
{
    "code": 400,
    "status": "error",
    "message": "invalid timestamp.",
    "data": ""
}
```

签名错误

```
{
    "code": 403,
    "status": "error",
    "message": "invalid signature.",
    "data": ""
}
```

找不到该频道

```
{
    "code": 400,
    "status": "error",
    "message": "channel not found.",
    "data": ""
}
```

查询开始时间或查询结束时间为空

```
{
    "code": 400,
    "status": "error",
    "message": "startTime or endTime not found.",
    "data": ""
}
```

开始时间或查询结束时间非法

```
{
    "code": 400,
    "status": "error",
    "message": "startTime or endTime valid.",
    "data": ""
}
```

频道暂无此功能，请联系客服人员

```
{
    "code": 403,
    "status": "error",
    "message": "该频道暂不支持此api功能",
    "data": ""
}
```

转存失败，请重试，如果一直不成功，请及时查看PLYVO账号的空间以及套餐过期时间情况，并联系客服

```
{
    "code": 403,
    "status": "error",
    "message": "转存失败，请重试",
    "data": ""
}
```

字段说明

参数名	说明
code	请求状态响应码
status	请求状态
message	错误信息
data	成功将返回视频转存点播的vid

PHP请求示例

```
<?php

include 'config.php';
//接口需要的参数（非sign）赋值
$channelId = "108888";

$startTime = "2018-04-16 09:33:33";
$endTime = "2018-04-16 09:44:30";

$params = array(
 'appId'=> $appId,
 'timestamp'=> $timestamp,
 'startTime'=> $startTime,
 'endTime'=> $endTime,
  );
//生成sign
$sign = getSign($params); //详细查看config.php文件的getSign方法
//接口请求url
$url = "https://api.polyv.net/live/v2/channel/recordFile/$channelId/streamRecordIndex?
appId=$appId&timestamp=$timestamp&sign=$sign&startTime=$startTime&endTime=$endTime";

//输出接口请求结果
echo file_get_contents($url);

?>
```

JAVA请求示例

```
import com.live.util.EncryptionUtils;
import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;

import java.io.IOException;
import java.util.*;

public class TestStreamIndex {
    // 尽量把时间设置较长
    private RequestConfig requestConfig =
RequestConfig.custom().setSocketTimeout(60000).setConnectTimeout(15000)
            .setConnectionRequestTimeout(60000).build();

    private static final String APP_ID = "xxxx";
    private static final String APP_SECRET = "xxxxxxxxxxxx";
    private static final String STREAM_RECORD_INDEX_URL =
"http://api.polyv.net/live/v2/channel/recordFile/{param}/streamRecordIndex";

    public static void main(String[] args) {
        TestStreamIndex demo = new TestStreamIndex();
        int channelId = 183730;
        String start = "2018-04-17 10:54:50";
        String end = "2018-04-17 11:05:00";
        String fileName = "我的直播片段";
        long cataId = 1;
        demo.streamRecordIndexDemo(channelId, start, end, fileName, cataId);
    }

    private void streamRecordIndexDemo(int channelId, String start, String end, String fileName, long cataId) {
        Map<String, String> params = new HashMap<>();
        params.put("appId", APP_ID);
        params.put("startTime", start);
        params.put("endTime", end);
        params.put("timestamp", String.valueOf(System.currentTimeMillis()));
        params.put("fileName", fileName);
        params.put("cataId", String.valueOf(cataId));
        params.put("sign", this.generateSign(params, APP_SECRET));

        String result = sendHttpPost(getRealApiUrl(STREAM_RECORD_INDEX_URL, String.valueOf(channelId)), params);
        System.out.println("stream record index result=" + result);
    }

    private String getRealApiUrl(String url, String param){
        return url.replace("{param}", param);
    }
```

```
public String sendHttpPost(String httpUrl, Map<String, String> maps) {
    HttpPost httpPost = new HttpPost(httpUrl);
    List<NameValuePair> nameValuePairs = new ArrayList<>();
    for (String key : maps.keySet()) {
        nameValuePairs.add(new BasicNameValuePair(key, maps.get(key)));
    }
    try {
        httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs, "UTF-8"));
    } catch (Exception e) {
        e.printStackTrace();
    }
    return sendHttpPost(httpPost);
}

private String sendHttpPost(HttpPost httpPost) {
    CloseableHttpClient httpClient = null;
    CloseableHttpResponse response = null;
    HttpEntity entity = null;
    String responseContent = null;
    try {
        httpClient = HttpClients.createDefault();
        httpPost.setConfig(requestConfig);
        response = httpClient.execute(httpPost);
        entity = response.getEntity();
        responseContent = EntityUtils.toString(entity, "UTF-8");
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (response != null) {
                response.close();
            }
            if (httpClient != null) {
                httpClient.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return responseContent;
}


/**
 * 根据map里的参数构建加密串
 * @param map
 * @param secretKey
 * @return
 */
protected String generateSign(Map<String, String> map, String secretKey) {
    Map<String, String> params = this.paraFilter(map);
    // 处理参数，计算MD5哈希值
    String concatedStr = this.concatParams(params);
    String plain = secretKey + concatedStr + secretKey;
    String encrypted = EncryptionUtils.md5Hex(plain);
```

```
        // 32位大写MD5值
        return encrypted.toUpperCase();
    }

    /**
     * 对params根据key来排序并且以key1=value1&key2=value2的形式拼接起来
     * @param params
     * @return
     */
    private String concatParams(Map<String, String> params) {
        List<String> keys = new ArrayList<String>(params.keySet());
        Collections.sort(keys);
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < keys.size(); i++) {
            String key = keys.get(i);
            String value = params.get(key);
            sb.append(key).append(value);
        }
        return sb.toString();
    }

    /**
     * 除去数组中的空值和签名参数
     * @param sArray 签名参数组
     * @return 去掉空值与签名参数后的新签名参数组
     */
    private Map<String, String> paraFilter(Map<String, String> sArray) {
        Map<String, String> result = new HashMap<String, String>();
        if (sArray == null || sArray.size() <= 0) {
            return result;
        }
        for (String key : sArray.keySet()) {
            String value = sArray.get(key);
            if (value == null || value.equals("") || key.equalsIgnoreCase("sign")
                    || key.equalsIgnoreCase("sign_type")) {
                continue;
            }
            result.put(key, value);
        }
        return result;
    }

}
```