Tuya_Linux_设备

生产测试 SDK 编程

指南

V1.0

声　明

非常感谢与贵司的合作，如果您有什么疑问或需要随时联系我们。

- 我们已尽量保证手册内容的完整性和准确性，但也不免出现技术上不准确、与产品功能及操作不相符或印刷错误等情况，如有任何疑问或争议，请以我司最终解释为准。
- 手册内容会实时进行更新，恕不另行通知。
- 本手册中内容仅为开发者提供参考指导作用，请以 SDK 实际内容为准。

目 的

SDK 是软件开发者在开发涂鸦智能产品设备时监控互联网应用的开发套件。
本文档提供的示例代码仅为演示接口调用方法，不保证能直接拷贝编译。

读者对象

使用 SDK　　工程师、项目经理和产品经理

修订记录

编 号	版本号	修订内容	发布日期
1	V1.0	新版发布	2019.12

名词解释

名词	说明

目录

1.1 概述

本文档主要介绍生产测试SDK接口参考信息，包括主要功能，接口函数和回调函数。

1.2 实用性

待补充

2.1 简介

Linux 生产测试 SDK 是基于涂鸦私有网络通信协议、涂鸦私有 Linux 产测通信协议开发，用于 Linux 设备的授权、生产测试软件的二次开发。

本 SDK 文档共包括涂鸦网络通信库、Linux 设备固件通信库两部分组成。其中，涂鸦网络通信库用于与涂鸦云的通信。主要包括账号登录、获取授权码信息、模块授权、授权校验、授权日志上传等功能。Linux 设备固件通信库包括基于涂鸦 Linx 设备通信协议的模块产测功能，包括进入产测模式、授权信息读写、按键测试、信号强度测试等功能。

主要库名称：
网络通信库 TuyaCloudIfLib.dll
Linux 设备固件通信库 Tuya.Module.ProtocolServer.dll
主要依赖库：
BouncyCastle.Crypto.dll
Newtonsoft.Json.dll
Tuya.Core.dll
Tuya.Module.BottomCommunication.dll

2.1.2 涂鸦网络通信库接口总览

接口	说明
设置涂鸦云工作环境	SetUrlSftVer
账号登录	UserLogin
授权	ProdTokenAuth
获取 Token 信息	GetTokenInfo
授权有效性校验	ProdTokenAuthValidate
添加授权日志	AddAuthLog

2.1.3 涂鸦 Linux 设备固件通信库接口总览

接口	说明
设置 Linux 设备通信服务串口信息	CommunicationServerBuilder.UseSerialPort
创建 Linux 设备通信服务	CommunicationServerBuilder.Builder<LinuxComServer>
启动 Linux 设备通信服务	LinuxComServer.Start
停止 Linux 设备通信服务	LinuxComServer.Stop
进入测试模式	LinuxComServer.EnterTestMode
写入固件授权信息	LinuxComServer.WriteConfigInfo
读取固件授权信息	LinuxComServer.ReadConfigInfo
读取 MAC	LinuxComServer.ReadMasterMAC
写入 MAC	LinuxComServer.WriteMasterMAC
读取国家码	LinuxComServer.ReadCountryCode
写入国家码	LinuxComServer.WriteCountryCode

2.2 流程说明

部分内容为涂鸦产测授权 SDK 主要是要部分的相关设置、接口调用流程。

2.2.1 整机测试整体流程

整机测试主要包括账号登录、授权码信息获取、建立连接、进入测试模式、固件版本校验、授权等部分。测试开始前需确保涂鸦账号、授权码 Token 正确无误。

设置涂鸦云工作环境。用于设置涂鸦云的工作模式，设置软件版本信息等。

账号登录为使用涂鸦云授权的前提，账号密码需向涂鸦注册。登录成功后方可使用涂鸦云相关功能。

授权码信息部分作用为根据涂鸦提供授权码获取授权码配置的相关信息,如固件版本信息以及其他相关测试的配置等。

建立连接。调用程序并传入连接参数建立通信 Server，与设备建立连接。

进入产测模式。调用 Linux 设备固件通信库进入产测模式接口，根据通信协议返回正确数据后，固件进入产测成功，可以进行后续通信。如进入产测失败，则无法进行后续流程。

固件版本比较。此部分主要作用为确保固件中运行的固件版本与授权码中配置的固件信息一致，确保设备批次或烧录结果无误。方法为调用固件通信库接口将获取到的相关信息与授权码中相关信息进行校验。

授权。授权部分为设备向涂鸦云注册授权，只有授权通过后才可与涂鸦云进行正确连接。本部分涉及云端通信库授权、授权校验等接口，涉及到 Linux 设备固件通信库中写入设备授权信息、读取设备授权信息接口。此部分授权通过后授权测试结束。如授权失败，可能会造成整机产测无法进入。授权部分中除读写授权配置信息可使用自有通信协议外，云端通信库授权接口调用不可缺少、省略，否则都可能会造成后续流程失败。

2.2.1.1 授权流程

本部分介绍授权详细流程，其中涉及云端通信库、WIFI 设备固件通信库相关接口。

授权开始前需确认登录流程是否成功，以及 Token 相关信息获取无误。除此之外需自行创建序列号 SN 用于测试，SN 的作用为识别授权设备身份，以方便后续整机测试获取 Mac 正常。SN 的要求为独立不重复，且每个 SN 使用后不可重复授权(包括授权失败的情况)。

先进入产测模式，读取固件中的 MAC，用于授权接口的入参。

根据进入产测的返回值 aboutType 判断：aboutType&0x02 == 0x02 表示是加密芯片；aboutType&0x04 == 0x04 表示是需要 pskKey；如果是加密芯片需要加密的双向认证，双向认证的过程见下文。

授权接口调用。参数为 SN、授权码、mac、软件版本信息。调用成功后云端会返回授权会返回分配的 mac、uuid、accessKey 等信息，此信息用于写入固件设置固件相关参数。

注：授权错误、缺失或授权 SN 重复使用，授权接口会返回错误信息。

向固件写入授权信息。此处如使用涂鸦 WIFI 设备固件通信库，需确保 WIFI 通信服务开启成功、进入产测模式成功。分为 4 个步骤：

首先，判断授权返回的 mac 和从固件中获取的 mac 是否相同，不相同的话，重新写入 mac。

然后，根据授权码的类型确定写入信息的模型，通用 wifi 类型，写入 {prod_idx=productId,prod_test=productionTest,auzkey=accesskey,ap_ssid=wifihotspotname,ap_pwd=wifipassword}，如果进产测返回值需要 pskKey,则 json 中再增加一个 pskKey=pskkey。特殊情况，NB 模块或者 GPRS 模块，需要判断 SN 对应的 IMEI 是否为空。根调用向固件写入授权信息接口后，等待串口返回写入结果。

然后，如果授权码对应的国家码不为空，需要调用国家码写入的接口，等待串口返回写入结果。

然后，如果授权码支持写入配置文件，下载配置文件，如果配置文件的 url 为空，则报错，下载完成后，判断下载的文件是否是标准 json 文件，如果是标准 json 文件，需要压缩文件，如果不是，直接将文件写入后缀为.bin 的同名文件中。调用请求下载参数文件的接口，传入文件长度和校验值，等待返回，然后调用传输配置文件的接口，支持分包传输，每包都需要等待返回值，然后调用结束文件传输的接口。

上面操作完成之后，复位，如果是瑞昱的芯片，则调用软件复位的接口，其他则直接硬件复位。复位之后重新进产测。如果是加密芯片，则重新进行加密双向认证。

读取固件授权信息。读取固件授权信息即为读取上步骤中写入的授权信息，此方法用于

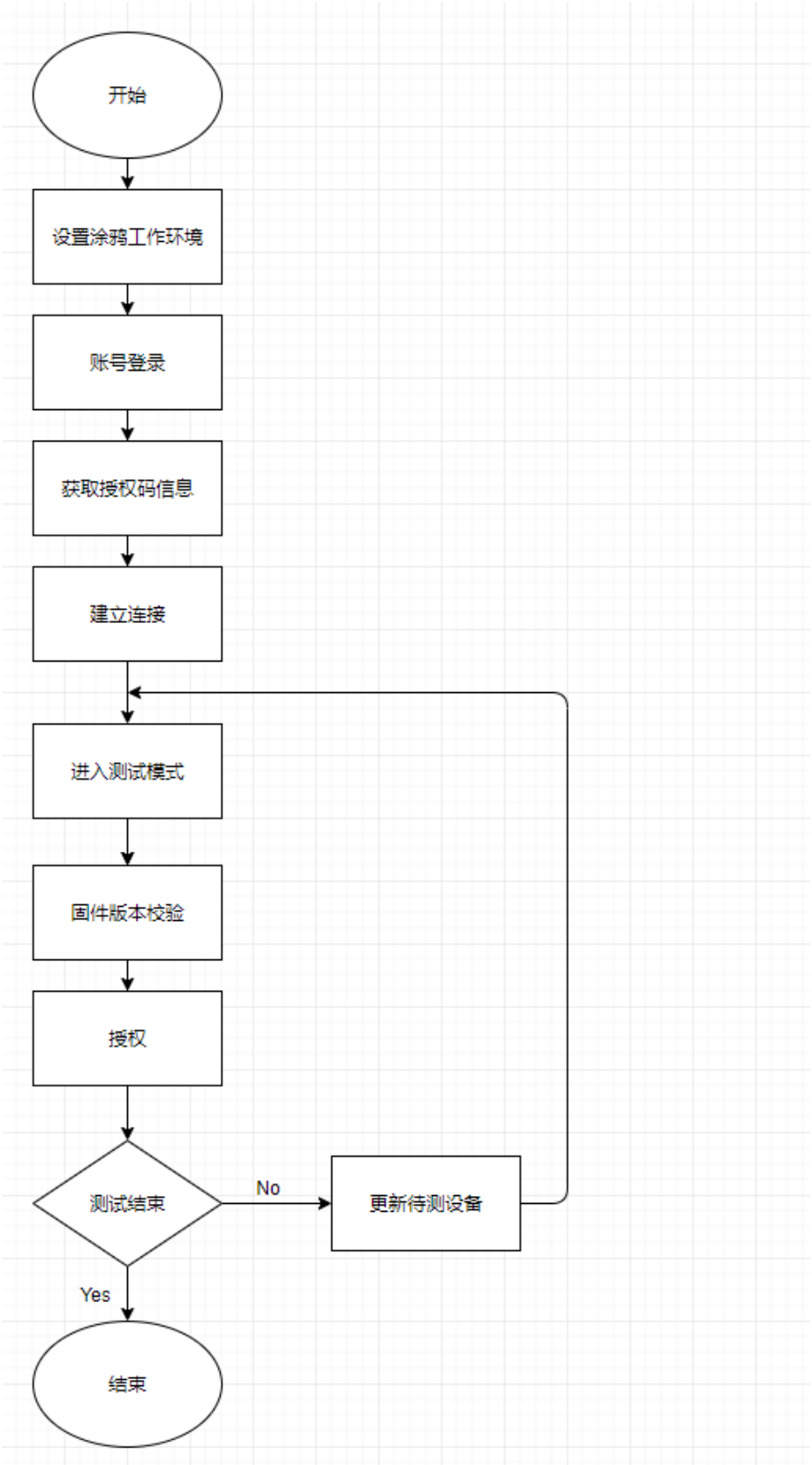
校验授权信息写入是否成功。返回信息包括 mac、uuid、accessKey 以及根据情况返回的 Pid。如果授权码对应的国家码不为空，则需要调用读取国家码接口，读取国家码并校验。

固件信息校验。用于比对写入、读出固件授权信息是否一致，确保授权无误。

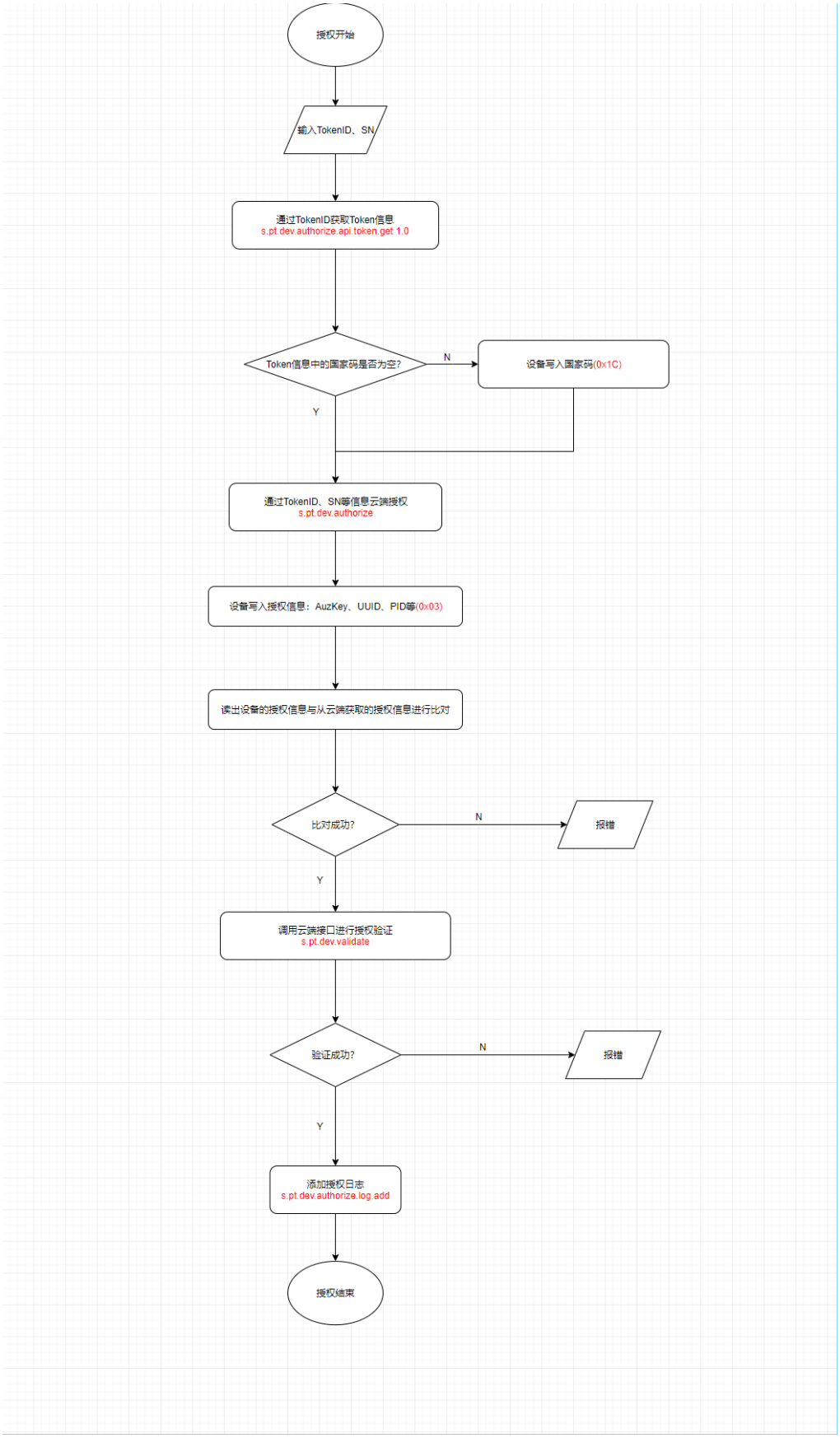
授权信息校验。调用授权接口并校验固件授权信息后，可确保写入固件授权信息与授权信息一致，此时需进行授权校验确保授权成功。参数包括 Token、SN、uuid、mac、accessKey、软件版本信息等。

添加授权日志。用于记录本次授权相关信息。需要上传的内容包括 Token、SN、uuid、时间戳、固件信息、授权状态等。添加授权日志接口调用返回正确后授权成功。

以上步骤除读写固件授权信息可使用自有协议外，其余流程(包括使用涂鸦 WIFI 设备固件通信协议的情况)必须确保返回成功方可进行下一项的调用，否则会导致授权失败。如 SN 已被授权接口使用过，无论后续流程是否成功，都不可再使用同一个 SN 重复授权。


开始
设置涂鸦工作环境
账号登录
获取授权码信息
建立连接
进入测试模式
固件版本校验
授权
测试结束
No
更新待测设备
Yes
结束

完整流程


授权开始
输入TokenID、SN
通过TokenID获取Token信息
s.pt.dev.authorize.api.token.get 1.0
Token信息中的国家码是否为空?
N
设备写入国家码(0x1C)
Y
通过TokenID、SN等信息云端授权
s.pt.dev.authorize
设备写入授权信息：AuzKey、UUID、PID等(0x03)
读出设备的授权信息与从云端获取的授权信息进行比对
比对成功?
N
报错
Y
调用云端接口进行授权验证
s.pt.dev.validate
验证成功?
N
报错
Y
添加授权日志
s.pt.dev.authorize.log.add
授权结束

授权流程

2.2.2 涂鸦网络通信函数说明

2.2.2.1 设置涂鸦云工作环境 SetUrlSftVer

函　数: void SetUrlSftVer(string sftVer, string sftIdentity, _LoginUrlType urlType = _LoginUrlType.Release)

参　数: [in] *sftVer* 调用 SDK 的软件的版本号

[in] *sftIdentity* 调用 SDK 的软件名称

[in] *urlType* 涂鸦云工作环境，默认为线上环境，

返回值: 无

说　明: 此接口用于选择涂鸦云的工作环境和设置调用 SDK 的应用程序的名称和版本信息。调用此接口后，每次调用涂鸦网络通信接口时，调用参数中自动附带应用程序的名称和版本信息。

注意：urlType 必须为_LoginUrlType.Release，否则可能导致产品授权无效。

2.2.2.2 账号登录 UserLogin

函　数: UserLoginRspParas UserLogin(UserLoginReqParas ReqParas, string apiVer = "1.0")

参　数: [in] *ReqParas* 账号登录请求参数，UserLoginReqParas 类型的对象

[in] *apiVer* API 版本，默认为 1.0

返回值: UserLoginRspParas 的对象，包含登录返回成功信息，以及错误信息等。

说　明: 此接口实现涂鸦云账号登录，实现用户身份验证。账号和密码可以联系项目经理申请。每个客户需要单独申请账号和密码，不同的账号拥有不同的权限范围和使用范围。绝大部分涂鸦网络通信接口都需要账号登录后才能访问。

2.2.2.3 授权 ProdTokenAuth

函　数: ProdTokenAuthRsp ProdTokenAuth(ProdTokenAuthReq ReqParas, string apiVer = "1.0")

参　数: [in] *ReqParas* 授权请求参数

[in] *apiVer* api 版本，默认为 1.0

返回值: ProdTokenAuthRsp 类型的对象，包含接口操作成功结果、错误信息、授权信息等。

说　明: 此接口上传产品的信息如 mac,SN,token 等信息到涂鸦云，校验通过后返回授权秘钥等信息，完成设备的授权。未经授权或授权失败的设备可能无法连接到涂鸦云。

2.2.1.4 获取 Token 信息 GetTokenInfo

函　数: GetTokenInfoRspParas GetTokenInfo(GetTokenInfoReqParas ReqParas, string apiVer = "1.0")
参　数: [in] GetTokenInfoReqParas 获取 Token 信息请求参数。
[in] apiVer api 版本，默认为 1.0。
返回值: Token 信息获取返回结果
说　明: 用于获取 Token 绑定信息。

2.2.1.5 授权有效性校验 ProdTokenAuthValidate

函　数: ProdTokenAuthValidateRsp ProdTokenAuthValidate(ProdTokenAuthValidateReq ReqParas, string apiVer = "1.0")
参　数: [in] ReqParas 授权有效性请求参数。
[in] apiVer api 版本，默认 1.0。
返回值: ProdTokenAuthValidateRsp 类型的对象，包含授权有效性校验结果。
说　明: 用于校验授权有效性，授权必须要通过授权有效性校验。

2.2.2.4 添加授权日志 AddAuthLog

函　数: AddAuthLogRspParas AddAuthLog(AddAuthLogReqParas ReqParas, string apiVer = "1.0")
参　数: [in]ReqParas 添加授权日志请求参数
[in]apiVer api 版本，默认 1.0。
返回值: AddAuthLogRspParas 的对象。
说　明: 授权完成后添加授权日志到涂鸦云。

2.2.3 涂鸦 Linux 设备固件通信函数说明

2.2.3.1 CommunicationServerBuilder.UseSerialPort 设置 Linux 通信服务串口信息

函　数: CommunicationServerBuilder UseSerialPort(ComPortParas paras)
参　数: [in]paras 串口参数
返回值: 返回 Linux 通信服务 Builder 对象
说　明: 用于设置 Linux 通信服务串口、波特率信息。

2.2.3.2 CommunicationServerBuilder.UseTCPClient 设置 Linux 通信服务 TCP 信息

函　数:　CommunicationServerBuilder.UseTCPClient (TCPClientParas paras)
参　数:　[in]paras　　TCP 参数
返回值:　返回 Linux 通信服务 Builder 对象
说　明:　用于设置 Linux 通信服务 TCP ip、port 信息。

2.2.3.4 CommunicationServerBuilde.Build<LinuxComServer> 创建 Linux 通信服务对象

函　数:　CommunicationServerBuilde.Build<LinuxComServer>()
参　数:　无
返回值:　返回 Linux 通信服务对象
说　明:　用于与设备的通信

2.2.3.5 启动 Linux 通信服务 LinuxComServer.Start

函　数:　bool LinuxComServer.Start()
参　数:　无
返回值:　true 成功　false:失败
说　明:　启动 Linux 设备串口通信服务

2.2.3.6 关闭 Linux 通信服务 LinuxComServer.Stop

函　数:　bool LinuxComServer.Stop()
参　数:　无
返回值:　true：成功　false:失败
说　明:　关闭 WIFI 设备串口通信服务

2.2.3.7 LinuxComServer.SwitchTestMode 进入测试模式

函 数:　bool SwitchMode(out EnterTestModeRsp rsp, int sendRetryTimes = 1, int receiveRetryTimes = 50, int retryInterval = 100)

参　数:	[out]rsp	进入测试模式返回结果
	[in]sendRestryTimes	串口命令最大重发次数，默认 1
	[in]receiveRetryTimes	串口命令接收最大重试次数，默认 50 次
	[in]retryInterval	串口命令接收重试间隔时间，单位毫秒,默认

100 毫秒

返回值： true：成功 false:失败

说　明： 向 Server 发送进入产测命令用以开启后续测试。

2.2.3.8 LinuxComServer.WriteConfigInfo 写入固件授权信息

函　数： bool WriteConfigInfo (WriteConfigInfoReq req, out BaseRsp rsp, int sendRetryTimes = 1, int receiveRetryTimes = 20, int receiveRetryInterval = 1000)

参　数：

[in]req	写入固件授权信息参数
[out]rsp	写入固件授权信息返回结果
[int]sendRestryTimes	串口命令最大发送次数，默认 1
[int]receiveRetryTimes	串口命令接收最大重试次数，默认 20 次
[in]retryInterval	串口命令接收重试间隔时间，单位毫秒,默认 1000 毫秒

返回值： true：成功 false:失败

说　明： 通过 Server 向固件向设备写入授权信息。

2.2.3.9 LinuxComServer.GetConfigInfo 读取固件授权信息

函　数： bool ReadConfigInfo (out ReadConfigInfoRsp rsp, int sendRetryTimes = 1, int receiveRetryTimes = 6, int receiveRetryInterval = 1000)

参　数：

[out]rsp	读取的固件授权信息
[int]sendRestryTimes	串口命令最大发送次数，默认 1
[int]receiveRetryTimes	串口命令接收最大重试次数，默认 6 次
[in]retryInterval	串口命令接收重试间隔时间，单位毫秒,默认 1000 毫秒

返回值： true：成功 false:失败

说　明： 通过 Server 读取固件授权信息。

2.2.3.10 LinuxComServer.WriteMasterMAC 写入 MAC

函　数： bool WriteMAC(WriteMasterMACReq req, out BaseRsp rsp, int sendRetryTimes = 1, int receiveRetryTimes = 6, int receiveRetryInterval = 1000)

参　数：

[out]req	写入 mac
[int]rsp	写入 mac 的返回
[int]sendRestryTimes	串口命令最大发送次数，默认 1
[int]receiveRetryTimes	串口命令接收最大重试次数，默认 6 次
[in]retryInterval	串口命令接收重试间隔时间，单位毫秒,默认 1000 毫秒

返回值： true：成功 false:失败

说　明： 通过 Server 写入 mac。

2.2.3.11 LinuxComServer.ReadMasterMAC 读取 MAC

函　数:	bool ReadMac(out ReadMasterMACRsp rsp, int sendRetryTimes = 1, int receiveRetryTimes = 6, int receiveRetryInterval = 1000)	
参　数:	[out]rsp	读取 MAC 的返回值
	[int]sendRestryTimes	串口命令最大发送次数，默认 1
	[int]receiveRetryTimes	串口命令接收最大重试次数，默认 6 次
	[in]retryInterval	串口命令接收重试间隔时间，单位毫秒,默认 1000 毫秒
返回值:	true：成功 false:失败 mac:mac 值	
说　明:	通过 Server 读取 MAC 信息。	

3.1 涂鸦网络通信库类说明

3.1.1 涂鸦云端运行环境枚举 _LoginUrlType

```
enum _LoginUrlType
{
    Release = 0,    //线上环境，默认
    Preview = 1,    //预发环境
    Daily = 2       //日常环境
}
```

3.1.2 账号登录请求参数 UserLoginReqParas

```
public class UserLoginReqParas
{
    public string username;   //用户名，必填
    public string password;   //密码,必填
    public string token;      //token,非必填
}
```

3.1.3 登录权限数据

```
public class PermissionsData
{
    public string permissionCode;   // 登录权限码
    public string permissionName;   // 登录权限名称
}
```

3.1.4 账号登录返回结果

```
public class UserLoginResult
{
    public string sessionId;                          //  登录 session id
    public List<PermissionsData> permissions; //  登录权限数据列表
}
```

3.1.5 账号登录返回参数 UserLoginRspParas

```
public class UserLoginRspParas{
    public bool success;            //接口执行结果，成功为 true，失败为 false
    public string errorMsg;         //  错误信息
    public string errorCode;        //错误码
    public string status;           //  操作状态，包含一些特定的错误信息或状态
    public long t;                  //接口调用时间
   public UserLoginResult result; //  账号登录返回结果
}
```

3.1.6 授权请求参数 ProdTokenAuthReq

```
public class ProdTokenAuthReq
{
        public string sn ;          //产品序列号
        public string tokenId ;   //激活码
        public string sftVersion ;//软件版本号
        public string muid ;       //乐鑫芯片的 muid，其他芯片传空字符串
        public string mac ;        //  产品的 mac 地址
        public string chipId ;     //  加密芯片的芯片 ID，无加密芯片传空串
}
```

3.1.7 授权结果 ProdTokenAuthResult

```
public class ProdTokenAuthResult
{
    public string uuid;           //模块 uuid
    public string mac;            //授权的 mac
    public string accessKey;    //授权秘钥
}
```

3.1.8 授权请求返回结果 ProdTokenAuthRsp

```
public class ProdTokenAuthRsp{
    public bool success; //接口执行结果，成功为 true，失败为 false
    public string errorMsg; //错误信息
    public string errorCode; //错误代码
    public string status;      //操作状态
    public long t; //接口调用时间戳
    public ProdTokenAuthResult result; //授权结果只有串口执行结果为 true 的时候有意义
}
```

3.1.9 获取 Token 信息请求参数 GetTokenInfoReqParas

```
public class GetTokenInfoReqParas
{
    public string tokenId; //激活码
    public string type;    //授权类型“wifi”
    public string sftVersion; //软件版本
    public string softwareName; //软件名称
}
```

3.1.10 Token 绑定信息 TokenInfo

```
public class TokenInfo
{
    public int amount; //授权总数
    public string firmwareConfigMd5; //配置文件 md5 校验码
    public string firmwareConfigUrl; //配置文件下载地址
    public string snPrefix; //sn 的前缀
    public string firmwareChip; //固件芯片平台
    public string firmwareProtocolType; //固件协议类型
    public string productionFirmwareMd5; //生产固件 md5 校验码
    public string productionFirmwareUrl; //生产固件下载地址
    public bool supportFirmwareConfig; //是否支持 oem 配置
    public string productionSubFirmwareUrl; //子固件下载地址
    public string moduleModelChip; //模组芯片平台
    public string productId; //产品 ID
    public string type; //激活码类型
    public string firmwareKey; //固件 key
    public string macRule; //是否云端分配 mac，“0”：不分配 mac，“1”：分配 mac
    public bool oem; //是否是 oem 固件
    public string countryCode; //国家码
    public string productionSubFirmwareMd5; //子固件 md5 校验码
```

```
    public string userareaFirmwareUrl; //用户区固件下载地址
    public string userareaFirmwareMd5; //用户区固件 md5 校验码
    public bool refreshUserarea; //是否刷新用户区固件
    public long effectiveTime; //激活码生效时间
    public int expires; //有效期
    public string id; //激活码
    public string productionId; //生产批号
    public bool test; //是否产测
    public int usedAmount; //已授权数量
    public string wifiHotspotName; //热点名称
    public string wifiPassword; //热点密码
    public string desc; //激活码描述信息
    public string manufacturer; //生产工厂
    public string productKey; //预留
    public string fingerprint; //固件名称
    public string firmwareVersion; //固件版本
    public int firmwareType; //预留
    public bool gpioTest; //预留
    public bool wifiTest; //预留
    public string baselineVersion; //预留
    public string flashRunSize; //flash 大小，单位 Mbits
}
```

3.1.11 授权请求返回结果

```
public class ProdTokenAuthRsp
{
    public bool success; //接口执行结果，成功为 true，失败为 false
    public string errorMsg; //错误信息
    public string errorCode; //错误码
    public string status; //操作状态，包含一些特定的错误信息或状态
    public long t; //接口调用时间戳
    public ProdTokenAuthResult result; //授权结果数据
}
```

3.1.12 授权信息校验请求参数 ProdTokenAuthValidateReq

```
public class ProdTokenAuthValidateReq      {
    public string tokenId ; //激活码
    public string sn; //产品序列号
    public string mac ; //mac 地址
    public string uuid ; //授权产品 uuid
```

```
    public string muid ; //产品芯片的 id
    public string accessKey ; //授权秘钥
    public string wifiHotspotName ; //热点名称
    public string wifiPassword ; //热点密码
    public string chipId ; //加密芯片 id
    public string sftVersion ; //软件版本
}
```

3.1.13 授权信息校验结果 ProdTokenAuthValidateResult

```
public class ProdTokenAuthValidateResult
{
    public bool result; //授权校验结果
    public string failureDepict; //失败原因描述信息
}
```

3.1.14 授权校验请求返回结果 ProdTokenAuthValidateRsp

```
public class ProdTokenAuthValidateRsp : BaseResponse
{
    public bool success; //授权校验执行结果
    public string errorMsg; //错误信息
    public string errorCode; //错误码
    public string status; //操作状态，包含一些特定的错误信息或状态
    public long t; //时间戳
    public ProdTokenAuthValidateResult result;   //   授权校验结果
}
```

3.1.15 授权日志信息 AuthLog

```
public class AuthLog
{
    public string hid; //wifi 类授权为 mac 地址
    public string uuid; //授权 uuid
    public string tokenId; //激活码
    public string sn; //产品 sn
    public string activeTime; //时间戳
    public string firmwareInfo; //固件指纹，包含固件名称和版本
    public string gpio; //gpio 测试结果“成功”或“失败”
    public string rssi; //wifi 信号强度测试结果
```

```
    public int authStatus; //授权状态，1：成功 ，0：失败
    public string authDepict; //授权状态描述，“成功”或“失败”
    public string countryCode; //国家码
}
```

3.1.16 添加授权日志返回结果 AddAuthLogReqParas

```
public class AddAuthLogReqParas
{
    public AuthLog log; //授权日志数据集
}
```

3.1.17 添加授权日志返回结果 AddAuthLogRspParas

```
public class AddAuthLogRspParas{
    public bool success; //接口执行结果
    public string errorMsg; //错误信息
    public string errorCode; //错误代码
    public string status; //操作状态，包含一些特定的错误信息或状态
    public long t; //时间戳
}
```

3.2 涂鸦 Linux 设备固件通信库类说明

3.2.1 串口信息设置参数 ComPortParas

```
public class ComPortParas
{
    public string PortName; //串口号如“COM3”
    public int BaudRate ; //波特率
    public int DataBits ; //数据为
    public Parity Parity ; //奇偶校验
    public StopBits StopBits ; //停止位数
    public int ReadBufferSize ; //读缓存大小
    public int WriteBufferSize ; //写缓存大小
    public int ReadTimeout ; //读超时时间，单位毫秒
}
```

3.2.2 TCP 信息设置参数 TCPClientParas

```
public class TCPClientParas
{
    public string HostName { get; set; }//目标 IP 地址
    public int Port { get; set; }//目标端口
    public int? ReceiveBufferSize { get; set; }//发送缓冲区大小
    public int? SendBufferSize { get; set; }//接受缓冲区大小
    public int? ReceiveTimeout { get; set; }//超时时间
}
```

3.2.3 写入设备固件信息 WriteConfigInfoReq

```
public class WriteConfigInfoReq
{
    public string Auzkey; //授权秘钥
    public string UUID; //授权 uuid
    public bool PID; //产品 PID
    public bool IsProductionTest; //是否为整机测试
    public string ApSSID; //授权的热点名称
    Public string ApPWD;//授权的热点密码
}
```

3.2.5 固件请求返回信息 BaseRsp

```
public class BaseRsp
{
    public bool Ret; //返回结果： true:成功 false:失败

}
```

3.2.6 固件授权信息读取结果

```
public class GetModuleInformationRsp
{
    public string AuthKey; //授权秘钥
    public string UUID; //授权 uuid
    public string PID; //产品 PID
    public bool IsProductionTest; //是否为整机测试
    public string ApSSID; //授权的热点名称
```

```
    Public string ApPWD;//授权的热点密码
}
```

4.1 获取授权信息示例代码（C#）

此代码进仅演示从涂鸦云端获取授权信息并完成授权校验的部分内容。（不包含写 mac 地址，写授权信息等其他功能）。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using TuyaCloudIfLib;
using Newtonsoft.Json;
//1. Login
 UserLoginReqParas loginReq= new UserLoginReqParas();
 loginReq.username = "******";
 loginReq.password = "######";
 UserLoginRspParas loginRsp= TuyaCloudIf.UserLogin(loginReq);
 if (loginRsp.success == false)
       MessageBox.Show(loginRsp.errorMsg);
 else
       MessageBox.Show("Login successfully !");

//2. Get token information
  GetTokenInfoReqParas GetTokenReq = new GetTokenInfoReqParas();
  GetTokenReq.tokenId= "XRVAygmzpvRZNxYN";
  GetTokenInfoRspParas GetTokenRsp = TuyaCloudIf.GetTokenInfo(GetTokenReq);
  if(GetTokenRsp.success == false)
      return false;

//3. Auth
  ProdTokenAuthReq AuthReq = new ProdTokenAuthReq();
  AuthReq.sn = "sn0000000002";
  AuthReq.mac = "ecfabc9a86f0";//mac read from module
  AuthReq.sftVersion = "test_app";
  AuthReq.muid = "009a86f0";// muid read from module
```

```
        AuthReq.chipId = "";
        AuthReq.tokenId = "XRVAygmzpvRZNxYN";
        ProdTokenAuthRsp AuthRsp= TuyaCloudIf.ProdTokenAuth(AuthReq);
        If(AuthRsp.success== false)
            Return false;
    //4. Auth validate
        ProdTokenAuthValidateReq ValidateReq = new ProdTokenAuthValidateReq();
        ValidateReq.accessKey = AuthRsp.result.accessKey;
        ValidateReq.muid = "009a86f0";//read
        ValidateReq.sn = "sn0000000002";
        ValidateReq.mac = AuthRsp.result.mac;
        ValidateReq.tokenId = "XRVAygmzpvRZNxYN";
        ValidateReq.uuid = AuthRsp.result.uuid;
        ValidateReq.chipId = "";
        ValidateReq.wifiHotspotName = "SmartLife";
        ValidateReq.wifiPassword = "";
        ValidateReq.sftVersion = "1.1";
        ProdTokenAuthValidateRsp ValidateRsp=
                    TuyaCloudIf.ProdTokenAuthValidate(ValidateReq);

        if (!ValidateRsp.success)
              MessageBox.Show("Validate fail ! ");
        else
              MessageBox.Show("Validate successfully! ");
```