TTLockSDK API

最近更新：2026-06-30

> 本文档描述 TTLockSDK（`@sciener/ttlock`）对外 API

本页目录

权限与工程配置

使用 TTLockSDK 前，**必须先完成权限声明与运行时授权**。不同业务模块所需权限如下。

权限对照表

权限	常量名	适用模块	是否必须	说明
蓝牙	`ohos.permission.ACCESS_BLUETOOTH`	TTLock、TTElectricMeter、TTWaterMeter、TTWiFiDoorMagnetic	**是**	扫描、连接、读写 BLE 设备
网络	`ohos.permission.INTERNET`	电表/水表添加、门锁/网关 OTA	按需	与开放平台 HTTP 交互时需要
NFC 标签	`ohos.permission.NFC_TAG`	TTNfcPassiveLock	**是**	读写 NFC 无源锁标签
蓝牙 MAC 持久化	`ohos.permission.PERSISTENT_BLUETOOTH_PEERS_MAC`	可选	否	部分场景可提升重连效率，按需开启

> **说明**：蓝牙扫描/连接在 HarmonyOS 上主要依赖 `ACCESS_BLUETOOTH`。若扫描不稳定，可参考 Demo 额外申请定位相关权限（`LOCATION` / `APPROXIMATELY_LOCATION` / `DISCOVER_BLUETOOTH` / `MANAGE_BLUETOOTH`），以适配不同系统版本策略。

SDK 导入与通用约定

导入方式

```
import {
  TTLock,
  TTElectricMeter,
  TTWaterMeter,
  TTNfcPassiveLock,
  TTWiFiDoorMagnetic,
  TTLog,
  SNError,
} from '@sciener/ttlock';
```

安装：

```
ohpm i @sciener/ttlock
```

回调约定（ITTLockResult）

除 `TTNfcPassiveLock` 外，蓝牙类 API 普遍使用 `ITTLockResult<T>` 回调：

```
export interface ITTLockResult<T> {
  success: (arg: T) => void;
  failure: (error: SNError) => void;
}
```

lockData 说明

- 门锁初始化成功后返回的加密字符串，后续绝大多数 `TTLock` 操作均依赖此参数。
- 请由应用侧安全存储；丢失后需重新初始化设备。

开启 SDK 日志

```
TTLock.printLog(true); // 建议在 Ability onCreate 中调用
```

TTLock

智能门锁、网关、配件等蓝牙设备的主入口类。

printLog

功能说明

控制 SDK 内部日志输出开关。

接口定义

```
static printLog(print: boolean): void
```

扫描

startScanLock

功能说明

扫描附近智能锁，持续回调发现的设备。需蓝牙权限且蓝牙已开启。

接口定义

```
static startScanLock(callBack: Callback<TTLockScanModel>): void
```

代码示例

```
TTLock.startScanLock((scanModel: TTLockScanModel) => {
  TTLog.info(`发现锁: ${scanModel.name} mac=${scanModel.mac} rssi=${scanModel.rssi}`);
});
// 页面离开时停止
TTLock.stopScan();
```

参数说明

参数	类型	说明
callBack	`Callback<TTLockScanModel>`	每发现一台锁回调一次

`TTLockScanModel` 主要字段：`mac`、`name`、`rssi`、`lockVersion`、`isInited`、`electricQuantity`、`lockSwitchState` 等。

---

startScanWirelessKeyboard

功能说明

扫描无线键盘。

接口定义

```
static startScanWirelessKeyboard(callBack: Callback<TTKeyboardScanModel>): void
```

---

startScanGateway / startScanDoorMagnetic / startScanWirelessKey

功能说明

分别扫描网关、蓝牙门磁、无线钥匙。

接口定义

```
static startScanGateway(progressBack: Callback<TTGatewayScanModel>): void
static startScanDoorMagnetic(progressBack: Callback<TTDoorMagneticScanModel>): void
static startScanWirelessKey(progressBack: Callback<TTKeyScanModel>): void
```

---

stopScan / stopScanGateway

功能说明

停止 BLE 扫描。`stopScan` 用于锁/键盘/门磁等；`stopScanGateway` 用于网关扫描。

接口定义

```
static stopScan(): void
static stopScanGateway(): void
```

锁初始化与重置

initLock

功能说明

初始化智能锁，成功后返回 `lockData`，需上传至开放平台完成绑定。

接口定义

```
static async initLock(config: IInitLockModel, callBack: ITTLockResult<string>): void
```

代码示例

```
TTLock.initLock(
  { mac: scanModel.mac, lockVersion: scanModel.lockVersion, clientPara: null },
  {
    success: (lockData: string) => {
      this.lockData = lockData;
    },
    failure: (error: SNError) => {
      TTLog.error(`初始化失败: ${error}`);
    },
  },
);
```

参数说明

参数	类型	说明
config.mac	`string`	扫描得到的锁 MAC
config.lockVersion	`string`	扫描得到的版本 JSON 字符串
config.clientPara	`string	null`	
callBack.success	`(lockData: string) => void`	成功返回 lockData

resetLock / resetLockByCode

功能说明

- `resetLock`：通过 lockData 恢复出厂。
- `resetLockByCode`：通过重置码 + MAC 恢复出厂。

接口定义

```
static resetLock(lockData: string, callBack: ITTLockResult<void>): void
static resetLockByCode(resetCode: string, mac: string, callBack: ITTLockResult<void>): void
```

开关锁与控制

controlLockWithControlAction

功能说明

蓝牙开锁/闭锁/查询等控制操作。

接口定义

```
static async controlLockWithControlAction(
  lockData: string,
  controlAction: TTControlAction,
  callBack: ITTLockResult<ITTControlModel>,
): void
```

代码示例

```
import { TTControlAction } from '@sciener/ttlock';

TTLock.controlLockWithControlAction(this.lockData, TTControlAction.unlock, {
  success: (info) => {
    TTLog.info(`开锁成功 uniqueId=${info.uniqueId} 电量=${info.electricQuantity}`);
  },
  failure: (error) => { TTLog.error(`${error}`); },
});
```

参数说明

参数	类型	说明
controlAction	`TTControlAction`	如 `unlock`、`lock` 等
成功回调	`ITTControlModel`	含 `lockTime`、`electricQuantity`、`uniqueId`

锁信息查询

方法	返回值	功能说明
`getLockTime(lockData, callBack)`	`number`	获取锁内时间戳（毫秒）
`timeCalibration(lockData, timestamp, callBack)`	`void`	校准锁时间
`getElectricQuantity(lockData, callBack)`	`number`	获取电量
`getLockVersion(mac, callBack)`	`Map<string, Object>`	获取锁版本
`getLockFeature(lockData, callBack)`	`string`	获取特征值
`getLockSystemInfo(lockData, callBack)`	`Map<string, Object>`	获取系统信息
`getLockSwitchState(lockData, callBack)`	`ITTSwitchState`	开关锁/门磁/反锁状态
`getOperationLog(lockData, type, callBack)`	`string`	获取操作日志

`ITTSwitchState` 字段说明：

- `lockSwitchState`：0 已关锁 / 1 已开锁 / 2 未知
- `doorSensorState`：0 无门磁 / 1 有门磁 / 2 未知

锁配置

方法	功能说明
`setAutomaticLockingPeriodicTime` / `getAutomaticLockingPeriodicTime`	自动闭锁周期

`setRemoteUnlockSwitch` / `getRemoteUnlockSwitch`	远程开锁开关
`setPassageMode` / `getPassageMode` / `deletePassageMode` / `clearPassageMode`	常开模式
`setLightTime` / `getLightTime`	照明时间（1~900 秒，0 关闭）
`setLockConfigWithType` / `getLockConfigWithType`	按类型配置锁参数
`setUnlockDirection` / `getUnlockDirection`	开锁方向
`setLockSound` / `getLockSound`	锁声音/音量/语言
`setLockFreezeState` / `getLockFreezeState`	冻结/解冻
`setScreenDisplaysPassword` / `getScreenDisplaysPassword`	带屏锁密码显示
`setSensitivity` / `getSensitivity`	灵敏度
`setLatchBolt` / `getLatchBolt` / `setMotorTorqueLevel`	斜舌/扭力
`setUnlockAngle` / `setLockAngle` / `setAutoAngle` / `getAngle`	贴锁角度校准
`supportFunction(lockData, func)`	判断是否支持某特征

密码管理

方法	功能说明
`modifyAdminPasscode`	修改管理员密码（4~9 位）
`getAdminPasscode`	获取管理员密码
`createCustomPasscode`	创建自定义密码
`modifyPasscode`	修改密码或有效期
`deletePasscode`	删除密码
`resetPasscodes`	重置所有密码
`getAllValidPasscodes`	获取全部有效密码
`getPasscodeVerificationParams`	获取密码校验参数
`recoverPasscode`	恢复密码
`resetEKey`	重置电子钥匙（待充分测试）

周期密码 `cyclicConfig` 格式示例：

```
[{ weekDay: 1, startTime: 10, endTime: 100 }] // 周一 00:10~01:40 有效
```

指纹 / IC 卡 / 人脸 / 掌静脉

分类	主要方法
指纹	`addFingerprint`、`modifyFingerprint`、`deleteFingerprint`、`clearAllFingerprint`、`getAllValidFingerprints`、`recoverFingerprint`、`writeFingerprint`
IC 卡	`addICCard`、`modifyICCard`、`deleteICCard`、`clearAllICCard`、`getAllValidICCard`、`recoverICCard`

人脸	`addFace`、`modifyFace`、`deleteFace`、`clearFace`、`getAllValidFaces`、`addFaceFeatureData`
掌静脉	`addPalmVein`、`modifyPalmVein`、`deletePalmVein`、`clearPalmVein`、`getAllValidPalmVein`

> `addICCard` 成功回调中 `status=2` 表示可以刷卡；指纹/人脸部分周期能力标注为待测，使用前请与硬件确认。

WiFi 锁

方法	功能说明
`lockScanNearbyWifi`	锁扫描周边 WiFi
`lockConfigWifi`	配置 WiFi
`cameraLockConfigWifi`	摄像头锁配网
`getLockWifiInfo`	获取 WiFi MAC/RSSI
`configLockServer`	配置服务器（默认 `wifilock.ttlock.com:4999`）
`configLockIP`	配置 IP（手动/自动）
`configWifiPowerSaving` / `clearWifiPowerSavingTime` / `getWifiPowerSavingTime`	WiFi 省电模式

无线键盘 / 无线钥匙 / 门磁配件

方法	功能说明
`initializeKeypad`	初始化无线键盘
`initializeMultifunctionalKeyPad`	初始化多功能键盘（锁与键盘需靠近）
`multifunctionalKeyPadGetAllStoredLocks`	查询已存锁列表
`multifunctionalKeyPadDeleteLockAtSpecifiedSlot`	删除指定槽位锁
`multifunctionalKeyPadAddFingerprint` / `multifunctionalKeyPadAddCard`	通过键盘添加指纹/卡片
`initializeDoorMagnetic`	初始化蓝牙门磁
`clearDoorMagnetic`	清除门磁
`setDoorMagneticAlertTime`	门磁报警时间（0~60 秒，0 关闭）
`getAccessoryElectricQuantity`	获取配件电量
`initializeWirelessKey` / `addWirelessKey` / `modifyWirelessKey` / `deleteWirelessKey` / `clearWirelessKey`	无线钥匙管理
`lockMangerWirelessKey`	锁与无线钥匙互绑
`getcWirelessKeyR2Feature` / `getWirelessKeySystemInfo` / `wirelessKeySupportFunction`	无线钥匙信息/特征

网关

方法	功能说明
`connectGateway` / `disConnectGateway`	连接/断开网关
`scanWiFiByGateway`	网关扫描周边 WiFi
`gatewayInit`	初始化网关（G2/G3/G4）
`upgradeGatewayMode`	进入网关升级模式
`gatewayConfigIp`	配置网关 IP
`gatewayConfigAPN`	配置 APN

`InitGatewayParameters` 要点：G2 需 `SSID` / `wifiPwd`； `gatewayVersion` 2=G2、3=G3、4=G4；名称不超过 48 字节。

酒店 / 梯控 / 取电 / 第三方设备

方法	功能说明
`setHotelCardSector`	设置酒店卡扇区（如 `"1,4,16"`，空串表示全部）
`setHotelDataWithHotelInfo`	设置酒店楼栋楼层
`activateHotelLiftFloors`	激活取电关联楼层
`setLiftControllableFloors` / `setLiftWorkMode`	梯控楼层与工作模式
`setPowerSaverWorkMode` / `setPowerSaverControllableLock`	取电开关模式与关联锁
`allowAddThirdPartyDevice` / `getThirdPartyDeviceResult` / `deleteThirdPartyDevice`	第三方设备管理

OTA 固件升级

方法	功能说明
`startLockDfu`	从服务器拉包升级门锁
`startLockDfuWithPackage`	使用本地固件包升级
`startGatewayDfu`	升级网关

TTElectricMeter（电表）

蓝牙智能电表业务入口。**需蓝牙权限**；添加/绑定设备还需配置开放平台 HTTP 参数。

setClientParam

功能说明

配置电表业务所需的开放平台 URL 与鉴权信息，**在 `addElectricMeter` 之前调用**。

接口定义

```
static setClientParam(url: string, clientId: string, accessToken: string): void
```

参数说明

参数	说明
url	开放平台 API 基地址
clientId	客户端 ID
accessToken	访问令牌

---

startScanElectricMeter

功能说明

扫描附近蓝牙电表。

接口定义

```
static startScanElectricMeter(callBack: Callback<TTElectricMeterScanModel>): void
```

代码示例

```
TTElectricMeter.startScanElectricMeter((model) => {
  TTLog.info(`电表 ${model.mac} 已初始化=${model.isInited} 剩余=${model.remainderKwh}`);
});
```

参数说明

`TTElectricMeterScanModel` 主要字段：`mac`、`name`、`rssi`、`isInited`、`isOn`、`payMode`、`totalKwh`、`remainderKwh`、`voltage`、`electricCurrent`。

---

stopScan

功能说明

停止电表扫描。

接口定义

```
static stopScan(): void
```

---

connect / cancelConnect

功能说明

- `connect`：连接指定 MAC 电表，成功后设备指示灯交替闪烁。
- `cancelConnect`：取消连接。

接口定义

```
static connect(mac: string, callBack: ITTLockResult<void>): void
static cancelConnect(mac: string, callBack: ITTLockResult<void>): void
```

---

addElectricMeter

功能说明

添加并绑定电表到账户，需先 `setClientParam`。

接口定义

```
static addElectricMeter(model: AddElectricMeterModel, callBack: ITTLockResult<TTElectricMeterAddResult>):
void
```

参数说明

字段	类型	说明
number	`string`	电表名称/编号
mac	`string`	电表 MAC
payMode	`number`	0 后付费 / 1 预付费
price	`string`	电价

成功返回 `electricMeterId`、`featureValue`。

---

deleteElectricMeter

功能说明

删除已绑定电表。

接口定义

```
static async deleteElectricMeter(mac: string, callBack: ITTLockResult<void>): void
```

用电控制与读数

方法	功能说明
`setPower(mac, on, callBack)`	合闸/拉闸
`setRemainingElectricity(mac, remainderKwh, callBack)`	设置剩余电量
`clearRemainingElectricity(mac, callBack)`	清空剩余电量
`readData(mac, callBack)`	读取电表状态 `ElectricMeterStatusModel`
`setPayMode(mac, payMode, price, callBack)`	设置付费模式与电价
`recharge(mac, rechargeAmount, rechargeKwh, callBack)`	充值
`setMaxPower(mac, maxPower, callBack)`	设置最大功率

`ElectricMeterStatusModel` 含：`payMode`、`onOff`、`voltage`、`electricCurrent`、`totalKwh`、`maxPower`、`remainderKwh`、`featureValue`。

设备信息与网络

方法	功能说明
`getFeatureValue(mac, callBack)`	获取特征值
`getElectricMeterDeviceInfo(mac, callBack)`	获取设备信息（含 Cat.1 信息）
`setElectricMeterApn(mac, apn, callBack)`	配置 APN
`setElectricMeterServer(mac, service, portNumber, callBack)`	配置服务器
`enterUpgradeMode(mac, callBack)`	进入升级模式（**固件升级流程尚未完善**）
`supportFunction(func, featureValue)`	判断特征是否支持

TTWaterMeter（水表）

蓝牙智能水表业务入口，API 结构与电表类似。**需蓝牙权限**；添加设备需 `setClientParam`。

setClientParam

功能说明

配置水表开放平台 URL 与鉴权，在 `addWaterMeter` 前调用。

接口定义

```
static setClientParam(url: string, clientId: string, accessToken: string): void
```

---

startScanWaterMeter

功能说明

扫描附近蓝牙水表，持续回调发现的设备。

接口定义

```
static startScanWaterMeter(callBack: Callback<TTWaterMeterScanModel>): void
```

代码示例

```
TTWaterMeter.startScanWaterMeter((model) => {
  TTLog.info(`水表 ${model.mac} 剩余=${model.remainderM3}m³`);
});
// 离开页面时
TTWaterMeter.stopScan();
```

参数说明

参数	类型	说明
callBack	`Callback<TTWaterMeterScanModel>`	每发现一台水表回调

`TTWaterMeterScanModel` 主要字段：`mac`、`name`、`rssi`、`isInited`、`isOn`、`payMode`、`totalM3`、`remainderM3`、`electricQuantity`。

---

stopScan

功能说明

停止水表扫描。

接口定义

```
static stopScan(): void
```

---

connect / cancelConnect

功能说明

连接/取消连接水表，连接成功后指示灯交替闪烁。

接口定义

```
static connect(mac: string, callBack: ITTLockResult<void>): void
static cancelConnect(mac: string, callBack: ITTLockResult<void>): void
```

---

addWaterMeter

功能说明

添加并绑定水表。

接口定义

```
static addWaterMeter(model: AddElectricMeterModel, callBack: ITTLockResult<TTWaterMeterAddResult>): void
```

参数说明

与电表 `AddElectricMeterModel` 结构相同：`number`、`mac`、`payMode`（0 后付费/1 预付费）、`price`。

---

deleteWaterMeter

功能说明

删除已绑定水表。

接口定义

```
static async deleteWaterMeter(mac: string, callBack: ITTLockResult<void>): void
```

用水控制与读数

方法	功能说明
`setWaterPower(mac, on, callBack)`	开阀/关阀（`on=true` 开，`false` 关）
`setRemainingWater(mac, remainderM3, callBack)`	设置剩余水量
`clearRemainingWater(mac, callBack)`	清空剩余水量
`readWaterData(mac, callBack)`	读取 `WaterMeterStatusModel`
`setWaterPayMode(mac, payMode, price, callBack)`	设置付费模式
`rechargeWater(mac, rechargeAmount, rechargeM3, callBack)`	充值
`setWaterTotalUsage(mac, totalM3, callBack)`	设置总用水量

设备信息与网络

方法	功能说明
`getWaterMeterFeatureValue(mac, callBack)`	获取特征值
`getDeviceInfo(mac, callBack)`	获取设备信息
`configAPN(mac, apn, callBack)`	配置 APN
`configServer(mac, service, portNumber, callBack)`	配置服务器
`waterMeterEnterUpgradeMode(mac, callBack)`	进入升级模式（**升级流程尚未完善**）
`supportFunction(func, featureValue)`	特征值能力判断

TTWiFiDoorMagnetic（WiFi 门磁）

WiFi 门磁设备蓝牙配网与管理。**需蓝牙权限**。

startScanWiFiDoorMagnetic

功能说明

扫描待配网的 WiFi 门磁。

接口定义

```
static startScanWiFiDoorMagnetic(callBack: Callback<TTDoorMagneticScanModel>): void
```

代码示例

```
TTWiFiDoorMagnetic.startScanWiFiDoorMagnetic((model) => {
  TTLog.info(`WiFi门磁 ${model.mac}`);
});
TTWiFiDoorMagnetic.stopScan();
```

stopScan

功能说明

停止 WiFi 门磁扫描。

接口定义

```
static stopScan(): void
```

---

initializeDoorMagnetic

功能说明

初始化 WiFi 门磁（配网 + 绑定），需传入 WiFi 与服务器参数。

接口定义

```
static initializeDoorMagnetic(
  mac: string,
  config: ConfigWiFiParameters,
  callBack: ITTLockResult<TTDeviceInfoModel>,
): void
```

参数说明

`ConfigWiFiParameters` 字段：

字段	说明
SSID	WiFi 名称
wifiPwd	WiFi 密码
serverAddress	服务器地址
portNumber	端口
aesKey	可选
contractNumber	可选

成功返回 `TTDeviceInfoModel`：`modelNum`、`hardwareRevision`、`firmwareRevision`、`electricQuantity`、`featureValue` 等。

信息与报警

方法	功能说明
`getFeatureValue(mac, callBack)`	获取特征值
`wiFiDoorMagneticWiFiMac(mac, callBack)`	获取 WiFi MAC
`wiFiDoorMagneticUUID(mac, callBack)`	获取 UUID
`wiFiDoorMagneticAuthCode(mac, callBack)`	获取授权码

`wiFiDoorMagneticVersion(mac, callBack)`	获取版本信息
`setAlarmTime(mac, openDoorTime, closeDoorTime, callBack)`	报警时间：未关门 0~~900 秒；长期未开门 0~~240 小时
`supportFunction(func, featureValue)`	特征能力判断

TTNfcPassiveLock（NFC 无源锁）

通过 NFC 与无源智能锁通信。**需** `ohos.permission.NFC_TAG`，且应用在前台、标签贴近期间完成操作。

接入准备

除在 权限与工程配置 中声明 `NFC_TAG` 外，还需完成以下配置。

NFC 能力声明

在 Ability 的 `skills` 中增加 NFC 意图，否则无法通过系统分发 NFC 标签：

```
{
  "abilities": [{
    "skills": [{
      "actions": [
        "action.system.home",
        "ohos.nfc.tag.action.TAG_FOUND"
      ]
    }]
  }]
}
```

NFC 生命周期注册

在 `UIAbility` 各生命周期中配合调用：

生命周期	调用方法
`onCreate`	`TTNfcPassiveLock.configureReaderFromWant(want)`
`onForeground`	`TTNfcPassiveLock.registerForegroundReader()`
`onBackground` / `onDestroy`	`TTNfcPassiveLock.unregisterForegroundReader()`

完整示例：

```
onCreate(want: Want): void {
  TTNfcPassiveLock.configureReaderFromWant(want);
}

onForeground(): void {
  TTNfcPassiveLock.registerForegroundReader();
}

onBackground(): void {
  TTNfcPassiveLock.unregisterForegroundReader();
}

onDestroy(): void {
  TTNfcPassiveLock.unregisterForegroundReader();
}
```

返回值约定（TTLockResult）

`TTNfcPassiveLock` 的业务方法返回 `Promise<TTLockResult<T>>`（与蓝牙类 `ITTLockResult` 不同）：

```
export interface TTLockResult<T> {
  isSuccess: boolean;
  error: SNError | undefined;
  data: T | undefined;
}
```

> `unlock` 成功后 `data` 为**增量 lockData**，需覆盖保存至本地。

---

configureReaderFromWant

功能说明

在 `UIAbility.onCreate` 中调用，保存 ElementName 供前台读卡注册使用。

接口定义

```
static configureReaderFromWant(want: Want): void
```

代码示例

```
onCreate(want: Want): void {
  TTNfcPassiveLock.configureReaderFromWant(want);
}
```

---

registerForegroundReader / unregisterForegroundReader

功能说明

- `registerForegroundReader`：在 `onForeground` 注册 NFC 前台读卡，标签信息写入 `TTNfcTagHolder`。
- `unregisterForegroundReader`：在 `onBackground` / `onDestroy` 注销监听。

接口定义

```
static registerForegroundReader(): void
static unregisterForegroundReader(): void
```

---

probeLaunchWantNfc

功能说明

检查设备 NFC 能力，并从启动 `Want` 探测标签（仅校验/日志，不写入 TagHolder）。

接口定义

```
static probeLaunchWantNfc(want: Want | null): boolean
```

---

clear

功能说明

清除当前缓存的 NFC 标签会话。

接口定义

```
static clear(): void
```

---

initLock

功能说明

初始化 NFC 无源锁。需手机贴近未初始化锁标签，且已注册前台读卡。

接口定义

```
static async initLock(): Promise<TTLockResult<ITTNfcInitLockResult>>
```

代码示例

```
let res = await TTNfcPassiveLock.initLock();
if (res.isSuccess) {
  this.lockData = res.data!.lockData;
  TTLog.info(`初始化成功 mac=${res.data!.lockMac}`);
} else {
  TTLog.error(`失败: ${res.error}`);
}
```

参数说明

成功时 `data` 含： `lockData` 、 `lockName` 、 `lockMac` 。

---

resetLock

功能说明

恢复出厂 / 重置无源锁。

接口定义

```
static async resetLock(lockData: string): Promise<TTLockResult<void>>
```

---

getLockState

功能说明

获取锁开关状态。

接口定义

```
static async getLockState(lockData: string): Promise<TTLockResult<number>>
```

参数说明

返回值	含义
0x00	闭锁
0x01	开锁
0x02	开锁中
0x03	闭锁中

---

unlock

功能说明

NFC 开锁。成功时 `data` 为更新后的 lockData，需覆盖保存。

接口定义

```
static async unlock(
  lockData: string,
  onProgress?: (electricQuantity: number) => void,
): Promise<TTLockResult<string>>
```

代码示例

```
let res = await TTNfcPassiveLock.unlock(this.lockData, (progress) => {
  TTLog.info(`马达充电进度 ${progress}%`);
});
if (res.isSuccess) {
  this.lockData = res.data!;
}
```

参数说明

参数	说明
onProgress	可选，马达充电进度 0~100

---

lock

功能说明

NFC 关锁。

接口定义

```
static async lock(
  lockData: string,
  onProgress?: (electricQuantity: number) => void,
): Promise<TTLockResult<void>>
```

NFC 常见错误码（TTNFCErrorCode）

错误码	含义
timeout (0x70)	超时，常见于未贴近标签或已切后台
nfcScan (0x65)	扫描/连接标签失败
nfcDisconnect (0x64)	标签连接断开
notInSettingMode (3)	锁未处于可初始化状态
permission (4)	权限不足

接入注意

- 操作期间保持 App **前台** + 标签**持续贴近**；切后台会触发 `unregisterForegroundReader`，导致 timeout。
- 建议在 UI 上展示「请贴近锁」引导，并监听 `onProgress` 充电进度。
- 调用 `initLock` / `unlock` / `lock` 前确认 `registerForegroundReader` 已成功。
- 失败时检查 `TTLockResult.isSuccess` 与 `error` 中的 `TTNFCErrorCode`。

注意事项与最佳实践

1. 权限优先

- 所有 BLE 业务：先 `verifyBluetoothPermission`，并确认 `getState()` 蓝牙已开。
- 电表/水表/OTA：额外需要 `INTERNET` 与有效的开放平台 `clientId` / `accessToken`。
- NFC 无源锁权限与接入流程见 TTNfcPassiveLock 章节。

2. 扫描与连接

- 同一时刻 SDK 共用扫描器，开始新扫描前调用对应 `stopScan()`。
- 页面 `aboutToDisappear` 或切后台时务必停止扫描，降低功耗与系统限频风险。
- 蓝牙操作请在 UI 线程发起，回调中更新 UI 时注意线程安全。

3. lockData 安全

- lockData 是后续所有锁操作的凭证，请加密存储，勿写入日志。

4. 电表 / 水表

- 流程建议：`setClientParam` → `startScan*` → `connect` → `add*` → 业务读写。
- `enterUpgradeMode` / `waterMeterEnterUpgradeMode` 仅进入升级模式，**完整 OTA 尚未完善**，生产环境慎用。
- 扫描广播 `version=2` 时部分字段可能为空，以 `readData` / `readWaterData` 为准。

5. 特征值能力判断

使用各类 `supportFunction` 前先获取 `featureValue`，避免对不支持能力的设备调用接口：

```
let supportCycle = await TTLock.supportFunction(lockData, TTLockFeatureValue.cyclePassword);
let supportCatOne = TTElectricMeter.supportFunction(TTElectricMeterFeature.catOne, featureValue);
```

6. 错误处理

- 蓝牙类失败回调为 `SNError`，可通过 `error.errorCode` / 文案定位问题。
- 常见蓝牙错误：未授权（`noBluetoothPermission`）、设备未进入添加模式、连接超时等。

7. 调试建议

```
TTLock.printLog(true); // 开发阶段开启
```