App与嵌入式通讯协议

从App接到蓝牙配对命令 🚀

🟧 设备正品认证流程

步骤	方向	📨 数据内容	✍️ 用途	⚙️ 说明
1	🔐→📱 Device→App	ID:<HEX>	设备发起加密认证	连接后立即推送，HEX字符串
2	📱→🔐 App→Device	ID:<解密HEX>	回传解密ID	utf-8→base64编码
3	🔐→📱 Device→App	VALID	正品认证通过	明文
4	📱→🔐 App→Device	"validation"	响应认证通过	utf-8→base64编码

🟦 PIN配对及钱包操作命令

步骤	方向	📨 数据内容	✍️ 用途	⚙️ 说明
5	📱→🔐 App→Device	"request"	请求PIN校验	utf-8→base64编码
6	🔐→📱 Device→App	PIN:<pin>,<flag>	下发PIN与处理标志	明文
7	📱→🔐 App→Device	pinCodeValue:<输入PIN>,receivedPin:<设备PIN>	用户PIN输入验证	utf-8→base64编码
8	📱→🔐 App→Device	"PIN_OK"	验证成功确认	utf-8→base64编码
9	📱→🔐 App→Device	address:<chainName>	获取各链区块链地址	utf-8→base64编码，分多次发送，间隔250ms；如发现缺失，可单独补发该命令请求缺失地址
10	🔐→📱 Device→App	<prefix><address>	返回链地址	明文，如 ETH:0x...
1 1	📱→🔐 App→Device	pubkey:<chain>,<hdpath>\n	获取部分链的公钥	utf-8→base64编码，结尾 \n
1 2	🔐→📱 Device→App	pubkeyData:<chain>,<pubkey>	返回链公钥	明文

📌 关键说明

- 📦 **所有 App → Device 的数据，均先 utf-8，再 base64 编码发送**
- ⏳ **连续命令建议间隔 ≥ 200ms，避免设备处理拥堵**
- 🔓 **数据明文传递，未做加密，仅用 base64 做数据包装**
- 💡 **每次写入请使用 writeCharacteristicWithResponseForService 保证响应**

从App接到确认签名对命令 📝🖋️

🟧 BLE 交易签名协议流程

步骤	方向	📨 数据内容	✍️ 用途	⚙️ 说明
1	📱→🔐 App→Device	destinationAddress:<付款地址>,<收款地址>,<手续费>,<链标识>	下发交易主要参数（第一步）	例如：destinationAddress:0x123abcDEF4567890 所有字段直接拼接，无空格；使用utf-8
2（功能等待开发）	🔐→📱 Device→App	PIN_SIGN_OKPIN_SIGN_FAILPIN_SIGN_CANCEL	用户密码验证通过	明文字符串，表示用户已在设备端输入P PIN_SIGN_CANCEL 表示用户主动取消。
3	🔐→📱 Device→App	Signed_OK	设备确认交易参数	明文字符串 Signed_OK ，表示设备已收到
4	📱→🌐 App→Server	POST请求： { "chain": "ethereum", "from": "<付款地址>", "to": "<收款地址>", "txAmount": "<交	获取nonce、gasPrice等预签	App向后端API（如accountAPI.getSigr 不同链定义。

步骤	方向	📨 数据内容	✍️ 用途	⚙️ 说明
		易金额>", ... }	名参数	
5️⃣	📱→🌐 App→Server	POST请求：根据上一步返回参数构造的数据结构，具体结构与链相关	获取presign数据（hex/json）	App向后端encode接口（如signAPI.en签名。
6️⃣	📱→🔐 App→Device	sign:<链标识>,<BIP44路径>,<presign数据>	下发预签名数据（进行实际签名）	例如：sign:ethereum,m/44'/60'/0'/0/0,0x据，整体utf-8编码再base64发送。
7️⃣	🔐→📱 Device→App	signResult:<签名后的交易数据> 或 signResult:ERROR	返回最终签名结果或错误	明文字符串，如 signResult:0xf86b8201...

从App接到收藏NFT命令 🖼️🔗

🟩 BLE 收藏NFT到冷钱包通讯协议

步骤	方向	📨 数据内容	✍️ 用途	⚙️ 说明
1️⃣	📱→🔐 App→Device	DATA_NFT_TEXT<n>SIZE	NFT名称传输头（标志+长度）	n为NFT名称utf-8字节数。举例：DATA_NFT_TEXT14SIZE。需**utf-8转base64**发送。
2️⃣	🔐→📱 Device→App	GET1，GET2，...	请求下一个NFT名称数据分包	设备每次请求一包（最多200字节），GET+序号，如GET1表示第一包。
3️⃣	📱→🔐 App→Device	NFT名称数据分包，base64	分包发送NFT名称正文	按200字节/包拆分，接收设备GET序号后逐包发送，utf-8→base64。
4️⃣	🔐→📱 Device→App	FINISH	NFT名称传输结束标志	明文，设备收到全部分包后发送，表示NFT名称部分收完，App继续下发图片部分。
5️⃣	📱→🔐 App→Device	DATA_NFT_IMG<m>SIZE	NFT图片传输头（标志+长度）	m为图片base64字节数。举例：DATA_NFT_IMG28321SIZE。需**utf-8转base64**发送。
6️⃣	🔐→📱 Device→App	GET1，GET2，...	请求下一个NFT图片分包	设备每次请求一包（最多200字节），GET+序号。
7️⃣	📱→🔐 App→Device	NFT图片数据分包，base64	分包发送NFT图片数据	按200字节/包拆分，收到设备GET后逐包发送，数据原本已是base64，无需再转码。
8️⃣	🔐→📱 Device→App	FINISH	NFT图片传输结束标志	明文，设备收到全部图片分包后，发送FINISH，表示全部收藏完成。

🆘 细节补充

- NFT名称、图片每部分**先发头**，设备收到后通过 GETn 分多包索取正文。
- **每包最多200字节**，App按序逐包应答，直到设备发 FINISH 。
- 头部内容均需utf-8→base64发送，正文内容按约定格式发送（名称utf-8→base64，图片原始base64）。
- 推荐每部分的头和数据之间**不要插入多余内容**，遵循命令即发。
- **如设备多次请求同一包（丢包重发）App应能容忍并正确返回**。

🗂️ 字段示例

- NFT名称头：DATA_NFT_TEXT12SIZE → base64: REFUQV9ORlRfVEVYVDEyU0laRQ==
- NFT图片头：DATA_NFT_IMG30580SIZE → base64: REFUQV9ORlRfSU1HMzA1ODBTWVpF
- NFT分包数据（如“CryptoCat”一包）：base64: Q3J5cHRvQ2F0

从App接到OTA固件升级命令 ⚙️⬆️

🟧 BLE OTA 固件升级协议流程

步骤	方向	📩 数据内容	✍️ 用途	⚙️ 说明
1️⃣	📱→🔐 App→Device	`DATA_OTA<文件字节数>SIZE`	固件头部（标志+总大小）	例如：`DATA_OTA163840SIZE`，即163840字节。utf-8转base64发送，通知设备准备接收数据。
2️⃣	📱→🔐 App→Device	固件内容分包，每包200字节，HEX字符串	固件分包数据	固件内容按200字节拆包，每包转为HEX字符串，再utf-8转base64发送。顺序连续下发，无需等待应答。
3️⃣	🔐→📱 Device→App	可扩展为 `OTA_OK` / `OTA_FAIL`（**建议支持**）	设备确认接收或异常反馈	固件全部接收完成后，设备可主动回复状态命令，`OTA_OK`表示升级数据已收全，`OTA_FAIL`表示异常。

🗂️ 字段示例与流程说明

- **固件头部示例**：`DATA_OTA12032SIZE`
  - 发送内容为 `DATA_OTA12032SIZE` → utf-8转base64
- **固件分包示例**：
  - 第1包HEX为 `aabbcc...`（200字节，HEX字符串长度=400）
  - 发送内容为 `aabbcc...` → utf-8转base64
- **分包机制**：
  - 固件全部数据，循环offset每200字节，分多包下发
  - App无需等设备GET或确认，可**顺序连续**写入

```
╔═══════════════════════════════════════╗
║      正在升级...                      ║
╠═══════════════════════════════════════╣
║ 已发送：3/48 包（7%）                 ║
║ [████▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒] ║
║ 请勿断开设备或关闭App                 ║
╚═══════════════════════════════════════╝
```

从App接到“确认地址”命令 🏦🔎

🟦 BLE 硬件钱包显示/确认地址协议

步骤	方向	📩 数据内容	✍️ 用途	⚙️ 说明
1️⃣	📱→🔐 App→Device	`verify:<chainName>`	显示地址请求命令	例如：`verify:bitcoin`、`verify:ethereum`。App按链类型查assetRouteDefs，utf-8→base64编码后写入。
2️⃣	🔐→📱 Device→App	`Address_OK`	地址显示完成反馈	明文。设备在屏幕弹窗展示目标链收款地址，用户校验，确认无误后设备回传`Address_OK`。
3️⃣	🔐→📱 Device→App	其他提示/错误码	（可选）异常/扩展命令	设备如遇异常可扩展`Address_FAIL`等命令（**建议App容错并友好提示**）。

🗂️ 命令与链类型映射举例

币种	显示命令
BTC 比特币	`verify:bitcoin`
ETH 以太坊	`verify:ethereum`
TRX 波场	`verify:tron`
SOL 索拉纳	`verify:solana`
COSMOS	`verify:cosmos`
...	其他见 assetRouteDefs

📄 说明与流程

- App端通过**链名/币种**查表获得命令（如`verify:bitcoin`），发送给设备（BLE写入，utf-8→base64）。

- 设备端**屏幕弹窗展示该链收款地址**，用户肉眼校验。
- 校验无误后，设备回BLE命令 `Address_OK`，App接收后显示“地址校验通过”提示。
- （如需异常反馈，设备可扩展`Address_FAIL`等命令，App需捕获并提示）

💡 交互建议

- **App需弹窗或Toast同步提示**：如“请在设备上核对收款地址”
- **设备屏幕建议有确认/取消按键**
- **如BLE断开/未回包应有超时重试/友好报错**

🔗 拓展

- 如多链支持，只需维护 assetRouteDefs，链越多表越全，命令结构不变。
- 若需显示多地址/子账号，命令结构可扩展（如带参数：`verify:bitcoin:sub1`）