Remoting接口文档

Remoting通信支持RPC和HTTP两种架构，参阅这里，使用完全相同的接口出入参格式，用法一致，不再分开表述。

参考：

简易远程消息协议

ApiHttpClient客户端

基础知识

针对常见基础功能，封装了客户端与服务端实现，例如登录、注销和心跳等。

主要部件

ClientBase是客户端基类，推荐应用客户端类继承该基类，仅需简单配置即可使用；

BaseDeviceController是服务端控制器基类，推荐应用服务端控制器类继承该基类，也可以直接拷贝该基类代码来使用；

功能设置

默认实现的功能比较多，都集中在ClientBase基类内部，需要通过Features来设置启用哪些功能。一般写在应用客户端构造函数内。例如ZeroIoT中的HttpDevice构造函数：

```
public HttpDevice(ClientSetting setting) : base(setting)
{
    // 设置动作，开启下行通知
    Features = Features.Login | Features.Logout | Features.Ping |
Features.Notify | Features.Upgrade;
    SetActions("Device/");
    Actions[Features.CommandReply] = "Thing/ServiceReply";
}
```

不仅通过Features设置了需要开启的功能，还通过SetActions指定了服务端接口的路径前缀（如Device/Login）。其中CommandReply接口比较特殊，独立设置其路径。

又如星尘节点客户端StarClient，星尘代理StarAgent和码神工具CrazyCoder都通过它连接星尘服务端节点管理平台，提供登录、注销、心跳、更新和指令下发等功能：

```
public StarClient()
{
    Features = Features.Login | Features.Logout | Features.Ping |
Features.Upgrade | Features.Notify | Features.CommandReply | Features.PostEvent;
    SetActions("Node/");

    Log = XTrace.Log;
}
```

再来看看星尘应用客户端AppClient，所有通过nuget引入Stardust的应用，都通过AppClient连接星尘服务端应用管理平台，提供登录、心跳、指令下发、监控数据上报、服务注册与消费等功能：

```
public AppClient()
{
    Features = Features.Login | Features.Ping | Features.Notify |
Features.CommandReply;
    SetActions("App/");

    Log = XTrace.Log;
}
```

客户端属性

ClientBase客户端主要属性。

名称	类型	说明
Server	String	服务端地址。支持http/tcp/udp，支持客户端负载均衡，多地址逗号分隔
Code	String	编码。设备编码DeviceCode，或应用标识AppId
Secret	String	密钥。设备密钥DeviceSecret，或应用密钥AppSecret
PasswordProvider	IPasswordProvider	密码提供者。用于保护密码传输，默认提供者为空，密码将明文传输。
OnLogined		登录完成后触发
Received		收到下行命令时触发
Status	LoginStatus	登录状态
Delay	Int32	请求到服务端并返回的延迟时间。单位ms
Span	TimeSpan	时间差。服务器时间减去客户端时间
MaxFails	Int32	最大失败数。心跳上报失败时进入失败队列，并稍候重试。重试超过该数时，新的数据将被抛弃，默认1440次，约24小时
Commands	IDictionary<String, Delegate>	命令集合。注册到客户端的命令与委托
Features	Features	客户端功能特性。默认登录注销心跳，可添加更新等
Actions	IDictionary<Features, String>	各功能的动作集合。记录每一种功能所对应的动作接口路径。
JsonHost	IJsonHost	Json主机。提供序列化能力
Setting	IClientSetting	客户端设置

异步登录

客户端刚启动时，网络可能尚未就绪。此时直接调用Login就会抛出异常。ClientBase提供了Open方法，内部通过定时器不断检测网络，并在网络恢复时调用Login，直到成功为止。

有可能客户端网络没有问题，而服务端网络有问题，或者中途网络无法连通，或者甚至服务端还没有启动。Open内部的定时登录机制同样有效，直到登录成功为止。

接口详解

各个接口按需选用，都不是必须实现。如果没有特殊说明，各接口都是客户端向服务端发起请求，并等待服务端响应。

各个应用接口前缀可能不同，下文使用变量表示，例如`/{Prefix}/Login`在ZeroIoT中就是`/Device/Login`。该前缀由前文功能设置中的SetActions指定，并保存在Actions属性中。少数场景可能会修改接口名，例如`/Device/Register`，具体以Actions中为准。

各接口请求中，非必要字段可以选填，应尽量提供相关信息，以支持服务端各项功能。

各接口默认使用POST请求方式，参数经Json序列化放在Body中。如果使用GET请求，则参数拼接在Url中。

登录获取令牌后，所有接口请求均需要在请求头中携带令牌，HTTP请求头：`Authorization: Bearer {token}`。

正常响应格式如下（code==0），下文响应表格仅列出data部分：

```
{
    "code": 0,
    "data": { /* 各接口响应表格字段 */ }
}
```

异常响应格式如下（code!=0），data部分为异常字符串：

```
{
    "code": 401,
    "data": "未登录"
}
```

登录Login

地址：`POST /{Prefix}/Login`

功能：向服务端提交编码和密钥，服务端验证通过后颁发令牌给客户端，后续所有接口调用带上令牌作为身份标识。

说明：登录请求一般还提供ClientId、版本信息、IP地址与MAC地址、唯一机器码、本地时间等信息，服务端接口能验证客户端身份，并颁发令牌。服务端找不到关联身份时，也可以根据相关信息注册一个身份，并通过登录响应返回编码和原始明文密钥。

请求（ILoginRequest）：

名字	类型	必要	说明
Code	String	√	编码
Secret	String	√	密钥
ClientId	String	√	实例。应用可能多实例部署，ip@proccessid
Version	String		版本
Compile	Int64		编译时间。UTC毫秒
IP	String		本地IP地址
Macs	String		MAC地址
UUID	String		唯一标识
Time	Int64		本地时间。UTC毫秒

响应（ILoginResponse）：

名字	类型	必要	说明
Code	String		编码。平台下发新编码，仅用于自动注册
Secret	String		密钥。平台下发新密钥，仅用于自动注册
Token	String	√	令牌。后续所有接口调用均需要在请求头携带该令牌。
Expire	Int32		令牌过期时间。单位秒。默认7200秒
Time	Int64		本地时间。客户端用于计算延迟，Unix毫秒（UTC）
ServerTime	Int64		服务器时间。客户端用于计算时间差，Unix毫秒（UTC）

注销Logout

地址：`GET /{Prefix}/Logout`

功能：向服务端申请，注销当前令牌。

请求：

名字	类型	必要	说明
reason	String		原因

响应（ILogoutResponse）：

名字	类型	必要	说明
Token	String		令牌。清空令牌后返回

心跳Ping

地址：`POST /{Prefix}/Ping`

功能：定时向服务端发送心跳，主要目的是链路保活，同时上报客户端的性能数据。

请求（IPingRequest）：

名字	类型	必要	说明
Time	Int64	√	本地UTC时间。Unix毫秒（UTC）
Memory	UInt64		内存大小
AvailableMemory	UInt64		可用内存大小
TotalSize	UInt64		磁盘大小。应用所在盘
AvailableFreeSpace	UInt64		磁盘可用空间。应用所在盘
CpuRate	Double		CPU占用率
Temperature	Double		温度
Battery	Double		电量
Signal	Int32		信号强度。WiFi/4G
UplinkSpeed	UInt64		上行速度。网络发送速度，字节每秒
DownlinkSpeed	UInt64		下行速度。网络接收速度，字节每秒
IP	String		本地IP地址。随着网卡变动，可能改变
Uptime	Int32		开机时间，单位s
Delay	Int32		延迟。请求到服务端并返回的延迟时间。单位ms

响应（IPingResponse）：

名字	类型	必要	说明
Time	Int64	√	本地时间。客户端用于计算延迟，Unix毫秒（UTC）
ServerTime	Int64	√	服务器时间。客户端用于计算时间差，Unix毫秒（UTC）
Period	Int32		心跳周期。单位秒
Token	String		令牌。现有令牌即将过期时，颁发新的令牌
Commands	CommandModel[]		下发命令
NewServer	String		新服务器地址。用于服务器迁移

CommandModel：

```
/// <summary>命令模型</summary>
public class CommandModel
{
    /// <summary>序号</summary>
    public Int64 Id { get; set; }

    /// <summary>命令</summary>
    public String Command { get; set; } = null!;

    /// <summary>参数</summary>
    public String? Argument { get; set; }

    /// <summary>开始执行时间。用于提前下发指令后延期执行，暂时不支持取消</summary>
    /// <remarks>
    /// 使用UTC时间传输，客户端转本地时间，避免时区差异。
    /// 有些序列化框架可能不支持带时区信息的序列化，因此约定使用UTC时间传输。
    /// </remarks>
    public DateTime StartTime { get; set; }

    /// <summary>过期时间。未指定时表示不限制</summary>
    /// <remarks>
    /// 使用UTC时间传输，客户端转本地时间，避免时区差异。
    /// 有些序列化框架可能不支持带时区信息的序列化，因此约定使用UTC时间传输。
    /// </remarks>
    public DateTime Expire { get; set; }

    /// <summary>跟踪标识。传输traceParent，用于建立全局调用链，便于查找问题</summary>
    public String? TraceId { get; set; }
}
```

升级更新Upgrade

地址：`GET /{Prefix}/Upgrade`

功能：定时请求服务端更新接口，获取满足当前客户端条件的升级更新信息，并根据信息执行自动更新流程。

请求：

名字	类型	必要	说明
channel	String		更新通道

响应（IUpgradeInfo/IUpgradeInfo2）：

名字	类型	必要	说明
Version	String	√	版本号
Source	String	√	更新包下载地址（相对路径将按服务地址补齐）
FileHash	String		文件哈希（用于完整性校验）
FileSize	Int64		文件大小
Preinstall	String		预安装脚本。解压后在解压目录执行（IUpgradeInfo2）
Executor	String		更新后要执行的命令（IUpgradeInfo2）
Force	Boolean		是否强制更新。覆盖后即刻重启应用（IUpgradeInfo2）
Description	String		描述

下行通知Notify

地址：`WS /{Prefix}/Notify`

功能：向服务端建立WebSocket连接，服务端通过下行链路（HTTP是WebSocket，RPC是TCP/UDP）向客户端发送命令，指示客户端即时执行特定流程。建立连接时同样会在请求头中携带`Authorization: Bearer {token}`。

方向：服务端->客户端

下发（CommandModel）：

名字	类型	必要	说明
Id	Int64		序号，用于去重
Command	String	√	命令
Argument	String		参数
StartTime	DateTime		开始执行时间（UTC）。支持提前下发后定时执行
Expire	DateTime		过期时间（UTC）。过期后不再执行
TraceId	String		跟踪标识（traceParent），用于分布式调用链

命令响应CommandReply

地址：`POST /{Prefix}/CommandReply`

功能：客户端执行完成服务端下发的命令后，通过CommandReply接口向上汇报执行状态与结果。

请求（CommandReplyModel）：

名字	类型	必要	说明
Id	Int64	√	命令序号
Status	CommandStatus	√	执行状态
Data	String		返回数据/备注

响应：标准响应（code==0）。通常无特定 data 字段，或返回受影响条数等简单结果。

上报事件PostEvents

地址：`POST /{Prefix}/PostEvents`

功能：客户端批量上报带有时间戳的事件日志。

请求（EventModel[]）：

名字	类型	必要	说明
Time	Int64	√	发生时间。Unix毫秒（UTC）
Type	String		事件类型。info/alert/error
Name	String	√	名称。事件名称，例如 LightOpen
Remark	String		内容。事件详情

响应：`Int32`，成功接收/写入的事件条数。

附注：

- 接口方法约定（HTTP 架构）
  - Login、Ping、CommandReply、PostEvents 使用 POST；
  - Logout、Upgrade 使用 GET；
  - 客户端会在必要时自动重试并处理 401 令牌过期，内部完成重新登录后重放请求。
- 接口前缀：默认通过 `SetActions("Device/")` 设置为 `Device/`，不同应用可自定义，如 `Node/`、`App/`。
- ClientId 语义：实例标识，形如 `ip@processId`。