# JSON RPC 2.0 协议接口文档

## 文档元数据

### 文档版本：1.5

### 生效日期：2025-05-16

### 修订记录：

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 主要变更说明 |
| 1.0 | 2025-05-08 | 首次发布，定义基础请求 / 响应结构 |
| 1.1 | 2025-05-08 | 移除成功响应中冗余的 data 字段，统一通过 result 承载结果，补充请求与响应的区分逻辑及双向调用说明 |
| 1.2 | 2025-05-08 | 增加对通知的定义及相关示例 |
| 1.3 | 2025-05-08 | 将文档元数据（版本、生效日期、修订记录）移至文档最前端，优化结构可读性 |
| 1.4 | 2025-05-15 | 增加了具体接口 |
| 1.5 | 2025-05-16 | 增加对时接口 |

## 1. 协议概述

本文档基于 **JSON RPC 2.0** 规范定义接口格式，用于设备配置参数查询场景（以获取串口参数为例）。JSON RPC 是一种轻量级的远程过程调用协议，通过 JSON 格式进行请求和响应的交互，具有无状态、跨平台的特点。该协议可通过字段结构差异明确区分客户端请求、服务器响应和通知，同时支持基于双向传输层（如 WebSocket）的双向调用。

## 2. 请求接口定义

### 2.1 通用请求结构

|  |
| --- |
| {  "jsonrpc": "2.0", // 协议版本号，固定为 "2.0"（必选）  "method": "string", // 调用的方法名（必选，如 "getUartPara"）  "id": "number|string|null", // 请求唯一标识（非通知型请求必选，通知型请求为 null）  "params": "object|array" // 方法参数（可选）  } |

### 2.2 字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必须 | 描述 |
| jsonrpc | string | 是 | 协议版本，固定为 "2.0" |
| method | string | 是 | 目标方法名，如 "getUartPara" |
| id | number/string/null | 非通知型请求必选 | 请求唯一标识：- 非通知型请求：需保证唯一性（如数字 88）- 通知型请求：设为 null（无需响应） |
| params | object/array | 否 | 方法入参，格式由方法定义 |

## 3. 通知定义

### 3.1 通知概述

通知是一种特殊的请求，客户端发送通知后不需要服务器返回响应。通知常用于客户端向服务器单向传递信息，如日志记录、状态更新等。

### 3.2 通知结构

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": "string",  "params": "object|array",  "id": null  } |

### 3.3 字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必须 | 描述 |
| jsonrpc | string | 是 | 协议版本，固定为 "2.0" |
| method | string | 是 | 要调用的方法名 |
| params | object/array | 否 | 方法的参数 |
| id | null | 是 | 通知的 id 必须为 null，表示不需要服务器响应 |

## 4. 响应接口定义

### 4.1 成功响应结构

|  |
| --- |
| {  "jsonrpc": "2.0", // 协议版本号（必选）  "id": "number|string", // 与请求对应的唯一标识（必选，与请求的 id 一致）  "result": "any" // 方法执行结果（必选，可为任意合法 JSON 类型）  } |

#### 字段说明

1. **result**：包含执行状态和具体数据（合并为单一对象），示例结果数据结构：

|  |
| --- |
| {  "status": "success", // 执行状态（建议包含）  "baudRate": 115200, // 波特率  "dataBits": 8, // 数据位  "stopBits": 1, // 停止位  "parity": "NONE" // 校验位  } |

### 4.2 错误响应结构

|  |
| --- |
| {  "jsonrpc": "2.0", // 协议版本号（必选）  "id": "number|string", // 与请求对应的唯一标识（必选，与请求的 id 一致）  "error": { // 错误信息（必选，替代 result 字段）  "code": "number", // 错误码（标准码或自定义码）  "message": "string",// 错误信息描述  "data": "any" // 可选，额外错误数据（如参数校验详情）  }  } |

## 5. 请求、响应和通知的区分方法

|  |  |  |  |
| --- | --- | --- | --- |
| 特征 | 客户端请求（非通知型） | 通知 | 服务器响应 |
| method 字段存在 | ✅ | ✅ | ❌ |
| id 字段 | 非 null | null | 与对应请求的 id 一致（非 null） |
| result 或 error 存在 | ❌ | ❌ | ✅（二者必有其一，互斥） |

**核心判别逻辑**：

1. 包含 method 且 id 非 null → 客户端请求（非通知型）；
2. 包含 method 且 id 为 null → 通知；
3. 不包含 method，但包含 result 或 error → 服务器响应。

## 6. 双向调用支持

JSON RPC 协议本身与传输层无关，若底层协议支持双向通信（如 WebSocket、长连接 TCP），可实现客户端与服务器互相调用：

1. **客户端 → 服务器**：可以是传统请求 - 响应模式（包含 method 字段且 id 非 null 的请求），也可以是通知（包含 method 字段且 id 为 null）；
2. **服务器 → 客户端**：服务器主动发送包含 method 的请求（需客户端保持连接并监听），同样可以分为需要客户端响应的请求和不需要响应的通知。

## 7. 注意事项

1. **协议版本强制要求**：所有消息必须包含 jsonrpc: "2.0"，否则视为无效。
2. **ID 唯一性和使用规则**：非通知型请求的 id 需唯一，用于匹配响应；通知的 id 必须为 null，且服务器无需回复。
3. **数据格式规范**：params（请求和通知）和 result（响应）需为合法 JSON 类型（对象、数组、基本类型等）。
4. **错误码规范**：
   1. 标准错误码：使用 -32700 到 -32000（如 -32600 无效请求，-32601 方法不存在）；
   2. 自定义错误码：建议从 -32000 开始（如 -32001 表示业务逻辑错误）。
5. **传输层选择**：单向调用可使用 HTTP，双向调用需选择全双工协议（如 WebSocket）。

## 8. 附录：JSON RPC 标准错误码

|  |  |
| --- | --- |
| 错误码 | 含义 |
| -32700 | JSON 解析错误 |
| -32600 | 无效请求（格式错误） |
| -32601 | 方法不存在 |
| -32602 | 无效参数 |
| -32603 | 内部错误 |
| -32000 ~ -32099 | 服务器自定义错误 |

## 9. 通知型接口说明

### 9.1 注册连接

此接口为客户端连上向服务器后发送的注册报文。

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": "register",  "id": "null",  "params": {  "time": "2025-05-12 12:11:10" , // 当前时间  "eqCode":” 4b313631 ff0e5042 ffff2137”, //设备编号唯一  "version":”czd20250419 1.0.2” //版本号  }  } |

### 9.2 心跳

此接口为客户端连上向服务器后，间隔10秒发送的心跳报文。

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": " heartbeat",  "id": "null",  "params": {  " time ": "2025-05-08 12:00:00" //当前时间  },  "id": null  } |

这是一个通知示例，客户端向服务器发送用户登录事件的日志记录，由于 id 为 null，服务器无需返回响应。

## 10.响应型接口

### 10.1 读取串口配置

#### 10.1.1请求

|  |
| --- |
| {  "jsonrpc": "2.0", // 协议版本号，固定为 "2.0"（必选）  "method": " getUartPara", // 调用的方法名（必选，如 "getUartPara"）  "id": "1", // 请求唯一标识（累加1）  "params": {  "name":"uart1" //可选uart1,uart2  }  } |

#### 10.1.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  "config": {  "baudrate": 115200,  "dataBits": 8,  "stopBits": 1,  "parity":0, //0=无校验，1=奇校验，2=偶校验  "protcal":5, //见协议列表  "mode":0 //0=rs232, 1= rs485  }  }  } |

### 10.2下发串口配置

#### 10.2.1请求

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": " setUartPara",  "id": 1, //唯一累加  " params ": {  "name":"uart1" //可选uart1,uart2  " config ": {  "baudrate": 115200,  "dataBits": 8,  "stopBits": 1,  "parity":0, //0=无校验，1=奇校验，2=偶校验  "protcal":5, //见协议列表  "mode":0 //0=rs232, 1= rs485  }  }  } |

#### 10.2.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  }  } |

### 10.3 读取can口配置

#### 10.3.1请求

|  |
| --- |
| {  "jsonrpc": "2.0", // 协议版本号，固定为 "2.0"（必选）  "method": " getCanPara", // 调用的方法名  "id": "1", // 请求唯一标识（累加1）  "params": {  "name":"can1" //可选can1  }  } |

#### 10.3.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  "config": {  "baudrate": 125000,  "protcal":4, //见协议列表  }  }  } |

### 10.4下发can配置

#### 10.4.1请求

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": " setCanPara",  "id": 1, //唯一累加  " params ": {  "name":"can1" //可选can1  " config ": {  "baudrate": 125000,  "protcal":4 //见协议列表  }  }  } |

#### 10.4.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  }  } |

### 10.5 读取can滤波字组

#### 10.5.1请求

|  |
| --- |
| {  "jsonrpc": "2.0", // 协议版本号，固定为 "2.0"（必选）  "method": " getCanLvBoPara", // 调用的方法名  "id": "1", // 请求唯一标识（累加1）  "params": {  "name":"can1" //可选can1  }  } |

#### 10.5.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  "config": {  "count": 0, //滤波字个数，最大256,=0接收所有帧  "mode": 0, //=0接收ID在滤波字组的所有帧,=1ID不在滤波字组的所有帧  "ids":[  11223344,84798291  ] //模式是16进制数，字节从高到低分别是0x11,0x22,0x33,0x44  }  }  } |

### 10.6下发can滤波字组

#### 10.6.1请求

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": " setCanLvBoPara ",  "id": 1, //唯一累加  " params ": {  "name":"can1" //可选can1  " config ": {  "count": 0, //滤波字个数，最大256,=0接收所有帧  "mode": 0, //=0接收ID在滤波字组的所有帧,=1ID不在滤波字组的所有帧  "ids":[  11223344,84798291  ] //模式是16进制数，字节从高到低分别是0x11,0x22,0x33,0x44  }  }  } |

#### 10.6.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一累加  "result": {  "status": "success",  }  } |

### 10.7 读取tcp配置

#### 10.7.1请求

|  |
| --- |
| {  "jsonrpc": "2.0", // 协议版本号，固定为 "2.0"（必选）  "method": " getTcpPara", // 调用的方法名  "id": "1", // 请求唯一标识（累加1）  "params": {  "name":"tcp1" //可选tcp1,tcp2  }  } |

#### 10.7.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  "config": {  "ip":”10.1.2.3”,  "port":1234,  "protcal":6, //见协议列表  "isRun":0 //0=不运行,1=运行    }  }  } |

### 10.8下发tcp配置

#### 10.8.1请求

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": " setTcpPara",  "id": 1, //唯一累加  " params ": {  "name":"tcp1" //可选tcp1,tcp2  " config ": {  "ip":”10.1.2.3”,  "port":1234,  "protcal":6, //见协议列表  "isRun":0 //0=不运行,1=运行  }  }  } |

#### 10.4.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  }  } |

### 10.9 重启

#### 10.9.1请求

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": " reboot",  "id": 1, //唯一累加  " params ": {  "key":"abcdefg" //协商好  }  } |

#### 10.9.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  }  } |

### 10.10对时

#### 10.10.1请求

|  |
| --- |
| {  "jsonrpc": "2.0",  "method": " timeSyn",  "id": 1, //唯一累加  " params ": {  "time":"2012-03-09 10:12:13" //时间格式  }  } |

#### 10.10.2正确响应

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 1, //唯一  "result": {  "status": "success",  }  } |

### 10.11 错误响应示例

|  |
| --- |
| {  "jsonrpc": "2.0",  "id": 88,  "error": {  "code": -32602, // 标准错误码：无效参数  "message": "Missing required parameter: name",  "data": { "missingParam": "name" }  }  } |

## 11 附录

### 11.1协议列表

|  |  |  |  |
| --- | --- | --- | --- |
| 协议编号 | 传输介质 | 协议名称 | 备注 |
| 1 | TCP | 透明上送协议 | 数据采用json格式 |
| 2 | Uart | 80G雷达协议 |  |
| 3 |  | 备用 |  |
| 4 | Can | 透明转发协议 |  |
| 5 | Uart | 透明转发协议 |  |
| 6 | TCP | json rpc 2.0协议 |  |