(译) JSON-RPC 2.0 规范(中文版)

- 起源时间: 2010-03-26(基于2009-05-24版本)
- 更新: 2013-01-04
- 作者: JSON-RPC工作组< json-rpc@googlegroups.com >
- 原文链接: http://www.jsonrpc.org/specification
- 翻译: leozvc < xxfs91@gmail.com >

Table of Contents

1.概述

JSON-RPC是一个无状态且轻量级的远程过程调用(RPC)协议。 本规范主要定义了一些数据结构及其相关的处理规则。它允许运行在基于socket,http等诸多不同消息传输环境的同一进程中。其使用JSON（RFC 4627）作为数据格式。

它为简单而生!

2.约定

文档中关键字"MUST"、"MUST NOT"、"REQUIRED"、"SHALL"、"SHALL NOT"、"SHOULD"、"SHOULD NOT"、"RECOMMENDED"、"MAY"和 "OPTIONAL" 将在RFC 2119 中得到详细的解释及描述。

由于JSON-RPC使用JSON，它具有与其相同的类型系统(见http://www.json.org或RFC 4627)。JSON可以表示四个基本类型(String、Numbers、Booleans和Null)和两个结构化类型(Objects和Arrays)。 规范中，术语“Primitive”标记那4种原始类型，“Structured”标记两种结构化类型。任何时候文档涉及JSON数据类型，第一个字母都必须大写：Object，Array，String，Number，Boolean，Null。包括True和False也要大写。

在客户端与任何被匹配到的服务端之间交换的所有成员名字应是区分大小写的。 函数、方法、过程都可以认为是可以互换的。

客户端被定义为请求对象的来源及响应对象的处理程序。

服务端被定义为响应对象的起源和请求对象的处理程序。

该规范的一种实现为可以轻而易举的填补这两个角色,即使是在同一时间,同一客户端或其他不相同的客户端。 该规范不涉及复杂层。

3.兼容性

JSON-RPC 2.0 的请求对象和响应对象可能无法在现用的JSON-RPC 1.0 客户端或服务端工作，然而我们可以很容易在两个版本间区分出2.0，总会包含一个成员命名为 “jsonrpc” 且值为“2.0”， 而1.0版本是不包含的。大部分的2.0实现应该考虑尝试处理1.0的对象，即使不是对等的也应给其相关提示。

4.请求对象

发送一个请求对象至服务端代表一个rpc调用， 一个请求对象包含下列成员：

jsonrpc

> 指定JSON-RPC协议版本的字符串，必须准确写为“2.0”

method

> 包含所要调用方法名称的字符串，以rpc开头的方法名，用英文句号（U+002E or ASCII 46）连接的为预留给rpc内部的方法名及扩展名，且不能在其他地方使用。

params

> 调用方法所需要的结构化参数值，该成员参数可以被省略。

id

> 已建立客户端的唯一标识id，值必须包含一个字符串、数值或NULL空值。如果不包含该成员则被认定为是一个通知。该值一般不为NULL[1]，若为数值则不应该包含小数[2]。

服务端必须回答相同的值如果包含在响应对象。 这个成员用来两个对象之间的关联上下文。

[1] 在请求对象中不建议使用NULL作为id值，因为该规范将使用空值认定为未知id的请求。另外，由于JSON-RPC 1.0 的通知使用了空值，这可能引起处理上的混淆。

[2] 使用小数是不确定性的，因为许多十进制小数不能精准的表达为二进制小数。

4.1通知

没有包含“id”成员的请求对象为通知， 作为通知的请求对象表明客户端对相应的响应对象并不感兴趣，本身也没有响应对象需要返回给客户端。服务端必须不回复一个通知，包含那些批量请求中的。

由于通知没有返回的响应对象，所以通知不确定是否被定义。同样，客户端不会意识到任何错误（例如参数缺省，内部错误）。

4.2参数结构

rpc调用如果存在参数则必须为基本类型或结构化类型的参数值，要么为索引数组，要么为关联数组对象。

- 索引：参数必须为数组，并包含与服务端预期顺序一致的参数值。
- 关联名称：参数必须为对象，并包含与服务端相匹配的参数成员名称。没有在预期中的成员名称可能会引起错误。名称必须完全匹配，包括方法的预期参数名以及大小写。

5.响应对象

当发起一个rpc调用时，除通知之外，服务端都必须回复响应。响应表示为一个JSON对象，使用以下成员：

jsonrpc

> 指定JSON-RPC协议版本的字符串，必须准确写为“2.0”

result

> 该成员在成功时必须包含。

> 当调用方法引起错误时必须不包含该成员。
>
> 服务端中的被调用方法决定了该成员的值。

error

> 该成员在失败是必须包含。
>
> 当没有引起错误的时必须不包含该成员。
>
> 该成员参数值必须为5.1中定义的对象。

id

> 该成员必须包含。
>
> 该成员值必须于请求对象中的id成员值一致。
>
> 若在检查请求对象id时错误（例如参数错误或无效请求），则该值必须为空值。

响应对象必须包含result或error成员，但两个成员必须不能同时包含。

5.1错误对象

当一个rpc调用遇到错误时，返回的响应对象必须包含错误成员参数，并且为带有下列成员参数的对象：

code

> 使用数值表示该异常的错误类型。 必须为整数。

message

> 对该错误的简单描述字符串。 该描述应尽量限定在简短的一句话。

data

> 包含关于错误附加信息的基本类型或结构化类型。该成员可忽略。 该成员值由服务端定义（例如详细的错误信息，嵌套的错误等）。

-32768至-32000为保留的预定义错误代码。在该范围内的错误代码不能被明确定义，保留下列以供将来使用。错误代码基本与XML-RPC建议的一样，url： http://xmlrpc-epi.sourceforge.net/specs/rfc.fault_codes.php

code	message	meaning
-32700	Parse error语法解析错误	服务端接收到无效的json。该错误发送于服务器尝试解析json文本
-32600	Invalid Request无效请求	发送的json不是一个有效的请求对象。
-32601	Method not found找不到方法	该方法不存在或无效
-32602	Invalid params无效的参数	无效的方法参数。
-32603	Internal error内部错误	JSON-RPC内部错误。

code	message	meaning
-32000 to -32099	Server error服务端错误	预留用于自定义的服务器错误。

除此之外剩余的错误类型代码可供应用程序作为自定义错误。

6.批量调用

当需要同时发送多个请求对象时，客户端可以发送一个包含所有请求对象的数组。

当批量调用的所有请求对象处理完成时，服务端则需要返回一个包含相对应的响应对象数组。每个响应对象都应对应每个请求对象，除非是通知的请求对象。服务端可以并发的，以任意顺序和任意宽度的并行性来处理这些批量调用。

这些相应的响应对象可以任意顺序的包含在返回的数组中，而客户端应该是基于各个响应对象中的id成员来匹配对应的请求对象。

若批量调用的rpc操作本身非一个有效json或一个至少包含一个值的数组，则服务端返回的将单单是一个响应对象而非数组。若批量调用没有需要返回的响应对象，则服务端不需要返回任何结果且必须不能返回一个空数组给客户端。

7.示例

Syntax:

```
--> data sent to Server
<-- data sent to Client
```

带索引数组参数的rpc调用:

```
--> {"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
<-- {"jsonrpc": "2.0", "result": 19, "id": 1}

--> {"jsonrpc": "2.0", "method": "subtract", "params": [23, 42], "id": 2}
<-- {"jsonrpc": "2.0", "result": -19, "id": 2}
```

带关联数组参数的rpc调用:

```
--> {"jsonrpc": "2.0", "method": "subtract", "params": {"subtrahend": 23, "minuend": 42},
"id": 3}
<-- {"jsonrpc": "2.0", "result": 19, "id": 3}

--> {"jsonrpc": "2.0", "method": "subtract", "params": {"minuend": 42, "subtrahend": 23},
"id": 4}
<-- {"jsonrpc": "2.0", "result": 19, "id": 4}
```

通知:

```
--> {"jsonrpc": "2.0", "method": "update", "params": [1,2,3,4,5]}
--> {"jsonrpc": "2.0", "method": "foobar"}
```

不包含调用方法的rpc调用:

```
--> {"jsonrpc": "2.0", "method": "foobar", "id": "1"}
<-- {"jsonrpc": "2.0", "error": {"code": -32601, "message": "Method not found"}, "id": "1"}
```

包含无效json的rpc调用:

```
--> {"jsonrpc": "2.0", "method": "foobar, "params": "bar", "baz]
<-- {"jsonrpc": "2.0", "error": {"code": -32700, "message": "Parse error"}, "id": null}
```

包含无效请求对象的rpc调用:

```
--> {"jsonrpc": "2.0", "method": 1, "params": "bar"}
<-- {"jsonrpc": "2.0", "error": {"code": -32600, "message": "Invalid Request"}, "id": null}
```

包含无效json的rpc批量调用:

```
--> [
        {"jsonrpc": "2.0", "method": "sum", "params": [1,2,4], "id": "1"},
        {"jsonrpc": "2.0", "method"
    ]
<-- {"jsonrpc": "2.0", "error": {"code": -32700, "message": "Parse error"}, "id": null}
```

包含空数组的rpc调用:

```
--> []
<-- {"jsonrpc": "2.0", "error": {"code": -32600, "message": "Invalid Request"}, "id": null}
```

非空且无效的rpc批量调用:

```
--> [1]
<-- [
    {"jsonrpc": "2.0", "error": {"code": -32600, "message": "Invalid Request"}, "id": null}
    ]
```

无效的rpc批量调用:

```
--> [1,2,3]
<-- [
    {"jsonrpc": "2.0", "error": {"code": -32600, "message": "Invalid Request"}, "id": null},
    {"jsonrpc": "2.0", "error": {"code": -32600, "message": "Invalid Request"}, "id": null},
    {"jsonrpc": "2.0", "error": {"code": -32600, "message": "Invalid Request"}, "id": null}
    ]
```

rpc批量调用:

```
--> [
    {"jsonrpc": "2.0", "method": "sum", "params": [1,2,4], "id": "1"},
    {"jsonrpc": "2.0", "method": "notify_hello", "params": [7]},
```

```
    {"jsonrpc": "2.0", "method": "subtract", "params": [42,23], "id": "2"},
    {"foo": "boo"},
    {"jsonrpc": "2.0", "method": "foo.get", "params": {"name": "myself"}, "id": "5"},
    {"jsonrpc": "2.0", "method": "get_data", "id": "9"}
    ]
<-- [
    {"jsonrpc": "2.0", "result": 7, "id": "1"},
    {"jsonrpc": "2.0", "result": 19, "id": "2"},
    {"jsonrpc": "2.0", "error": {"code": -32600, "message": "Invalid Request"}, "id": null},
    {"jsonrpc": "2.0", "error": {"code": -32601, "message": "Method not found"}, "id": "5"},
    {"jsonrpc": "2.0", "result": ["hello", 5], "id": "9"}
    ]
```

所有都为通知的rpc批量调用:

```
--> [
    {"jsonrpc": "2.0", "method": "notify_sum", "params": [1,2,4]},
    {"jsonrpc": "2.0", "method": "notify_hello", "params": [7]}
]

<-- //Nothing is returned for all notification batches
```

7.扩展

以rpc开头的方法名预留作为系统扩展，且必须不能用于其他地方。每个系统扩展都应该有相关规范文档，所有系统扩展都应是可选的。
