JSON-RPC

JSON-based light-weight RPC protocol

Based on: http://www.jsonrpc.org/specification

Overview

JSON-RPC is a stateless, light-weight remote procedure call (RPC) protocol.

It is transport agnostic in that the concepts can be used within the same process, over sockets, over http, or in many various message passing environments.

It is designed to be simple!

Overview

It uses JSON (RFC 4627) as data format.

The Client is defined as the origin of Request objects and the handler of Response objects.

The Server is defined as the origin of Response objects and the handler of Request objects.

Request object

- **jsonrpc** - a String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0";
- **method** - a String containing the name of the method to be invoked. Method names that begin with the word rpc followed by a period character (U+002E or ASCII 46) are reserved for rpc-internal methods and extensions and MUST NOT be used for anything else.
- **params** - a structured value that holds the parameter values to be used during the invocation of the method. This member MAY be omitted.
- **id** - an identifier established by the Client that MUST contain a String, Number, or NULL value if included. If it is not included it is assumed to be a notification. The value SHOULD normally not be Null and Numbers SHOULD NOT contain fractional parts

Response object

- **jsonrpc** - a String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0".
- **result** - this member is REQUIRED on success. This member MUST NOT exist if there was an error invoking the method. The value of this member is determined by the method invoked on the Server.
- **error** - this member is REQUIRED on error. This member MUST NOT exist if there was no error triggered during invocation. The value for this member MUST be an Object.
- **id** - this member is REQUIRED. It MUST be the same as the value of the id member in the Request Object. If there was an error in detecting the id in the Request object (e.g. Parse error/Invalid Request), it MUST be Null.

Examples

Request

```
{
      "jsonrpc": "2.0",
      "method": "subtract",
      "params": [42, 23],
      "id": 1
}
```

Response

```
{
      "jsonrpc": "2.0",
      "result": 19,
      "id": 1
}
```

Examples

Request

```
{
    "jsonrpc": "2.0",
    "method": "foobar",
    "id": 10
}
```

Response

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32601,
        "message": "Procedure not found."
    },
    "id": 10
}
```

Differences between 1.0 and 2.0

- client-server instead of peer-to-peer:
  - JSON-RPC 2.0 uses a client-server-architecture.
  - V1.0 used a peer-to-peer-architecture where every peer was both server and client.
- Transport independence:
  - JSON-RPC 2.0 doesn't define any transport-specific issues, since transport and RPC are independent.
  - V1.0 defined that exceptions must be raised if the connection is closed, and that invalid requests/responses must close the connection (and raise exceptions).
- Named parameters added
- Added optional extensions.

Differences between 1.0 and 2.0

- Reduced fields:
  - Request: params may be omitted
  - Notification: doesn't contain an id anymore
  - Response: contains only result OR error (but not both)
- "jsonrpc" field added:
  - added a version-field to the Request (and also to the Response) to resolve compatibility issues with JSON-RPC 1.0.
- Error-definitions added.

Batch

To send several Request objects at the same time, the Client MAY send an Array filled with Request objects.

The Server should respond with an Array containing the corresponding Response objects, after all of the batch Request objects have been processed.

Let’s do this code!

github.com/michal93cz/json-rpc/