JSONRPC

Welcome to JSONRPC

Introduction

he JSON RPC 2.0 aims to provide a binding for ODL Datastore, RPC and Notification operations and map them onto JSON RPC 2.0 calls over ZMQ and HTTP(S) transports. Other transports may be added as needed at a later date.

API agreement between ODL and the other JSON RPC endpoint is achieved by ensuring that they both use the same YANG model to describe all arguments in the RPC call. This includes core operations such as datastore access as well as any additional calls needed to implement service functions such as endpoint discovery, etc.

All data used in any of the JSON RPC 2.0 calls uses the same version of draft-json-netmod-yang as used in ODL (presently version 6). The project relies on existing Binding Independent APIs developed for netconf, does not require any changes in core ODL functionality and does not require any new APIs.

In addition to the ODL extension, the project also aims to provide a fully featured suite of libraries, bindings and tools for several languages to consume and produce data via JSON-RPC 2.0 in RFC 7951 as produced and consumed by ODL.

The specifics of using yang modeled data in JSON RPC 2.0 are described in detail in an IETF draft. The most up-to-date version is in draft-yang-json-rpc. This draft covers all applicable argument conventions. It does not cover various extensions necessary to achieve high performance when interfacing to opendaylight and other large scale systems. These are described in detail on the Asynchronous RPC page.

While RPC and data modeling in the context of extending ODL (and other yang based systems) are fairly straight-forward, there is a number of issues regarding the definitions of a transaction and integrity checking for off-ODL datastore operations. These are discussed in detail in: Transaction semantics. The rest of the datastore extension interface including data tree change notifications is covered in Extending the datastore

Last, but not least an essential part of any extension of functionality via remote procedure calling is the mapping of a remote procedure endpoint to a transport endpoint as well as how all of these mappings join together to form a consistent picture. This is described in detail in Configuration and governance.

Project Facts

Project Creation Date: April 21, 2016

Primary Contact: Anton Ivanov (aivanov) <anton.ivanov@kot-begemot.co.uk>

Project Lead: Anton Ivanov (aivanov) <anton.ivanov@kot-begemot.co.uk>

Committers:

- Anton Ivanov (aivanov) <anton.ivanov@kot-begemot.co.uk>
- David Spence (dspence) <david@roughsketch.co.uk>
- Shaleen Saxena (ssaxena) <shaleen.external@gmail.com>
- Richard Kosegi (rkosegi) <richard.kosegi@gmail.com>
- Tom Pantelis <tompantelis@gmail.com>

Mailing List: app-dev@lists.opendaylight.org

Meetings: See Community Meetings

Repository: git clone https://git.opendaylight.org/gerrit/jsonrpc

Jenkins: jenkins silo

Open Bugs: open bugs

Documentation

- https://docs.opendaylight.org/projects/jsonrpc/en/latest/
- Getting Started for Users
- Getting Started for Developers

Release Planning

- Oxygen: Release Plan

Release Notes

- Oxygen: Release Notes