spec

OpenRPC Specification

Version 1.3.2

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14 RFC2119 RFC8174 when, and only when, they appear in all capitals, as shown here.

In the following description, if a field is not explicitly **REQUIRED** or described with a MUST or SHALL, it can be considered OPTIONAL.

Introduction

The OpenRPC Specification defines a standard, programming language-agnostic interface description for JSON-RPC 2.0 APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic. When properly defined via OpenRPC, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Similar to what interface descriptions have done for lower-level programming, the OpenRPC Specification removes guesswork in calling a service.

Use cases for machine-readable JSON-RPC API definition documents include, but are not limited to:

- interactive documentation
- code generation for documentation
- clients
- servers
- automation of test cases.

OpenRPC documents describe a JSON-RPC APIs services and are represented in JSON format. These documents may either be produced and served statically or be generated dynamically from an application.

The OpenRPC Specification does not require rewriting existing JSON-RPC APIs. It does not require binding any software to a service — the service being described may not even be owned by the creator of its description. It does, however, require the capabilities of the service be described in the structure of the OpenRPC Specification. Not all services can be described by OpenRPC — this specification is not intended to cover REST APIs - It is exclusively for APIs which adhere to the JSON-RPC 2.0 spec. The OpenRPC Specification does not mandate a specific development process such as design-first or code-first. It does facilitate either technique by establishing clear interactions with a JSON-RPC API.

Definitions

OpenRPC Document

A document (or set of documents) that defines or describes an API. An OpenRPC document uses and conforms to the OpenRPC Specification.

Patterned Field

A field (key value pair) where the key name is supplied by the user, and the value is defined by the specification for the patterned field. The Field Pattern is a Regular expression.

Regular Expression

Regular expressions within the OpenRPC specification and tooling is RECOMMENDED to be a Perl Compatible Regular Expressions. That being said, tooling implementers SHOULD adhere to ECMA-262 6th Edition Regular Expressions.

Official OpenRPC Tooling

Tooling that is built, maintained and documented by the OpenRPC organization. It is meant to be used as a functional reference implementation of the Specification. Users of the OpenRPC Specification are encouraged to create versions of the tooling as their own organization/projects.

Versions

The OpenRPC Specification is versioned using Semantic Versioning 2.0.0.

The `major.minor` portion of the semver (for example `1.0.x`) SHALL designate the OpenRPC feature set. Typically, `.patch` versions address errors in this document, not the feature set. Tooling which supports OpenRPC 1.0.0 SHOULD be compatible with all OpenRPC `1.0.x` versions. The patch version SHOULD NOT be considered by tooling, making no distinction between `1.0.0` and `1.0.1` for example.

Subsequent minor version releases of the OpenRPC Specification (incrementing the `minor` version number) SHOULD NOT interfere with tooling developed to a lower minor version and same major version. Thus a hypothetical `1.1.0` specification SHOULD be usable with tooling designed for `1.0.0`.

An OpenRPC document compatible with OpenRPC 1.0.0 contains a required `openrpc` field which designates the semantic version of the OpenRPC that it uses.

Format

An OpenRPC document that conforms to the OpenRPC Specification is itself a JSON object, which must be represented in JSON format. Due to the nature of JSON-RPC APIs using JSON formats, strictly use JSON only as described here. If you wish to use any other format than JSON, it should be converted outside of any OpenRPC tooling.

It is RECOMMENDED that the OpenRPC document be named: `openrpc.json`. Tooling that requires an OpenRPC document as input MAY assume the default document location to be `./openrpc.json`, where the `./` represents the current working directory.

All field names in the specification are **case sensitive**. CamelCase SHOULD be used for all key names. This includes all fields that are used as keys in a map, except where explicitly noted that keys are **case insensitive**.

According to the JSON specification for objects, key names SHOULD be unique. However, To avoid ambiguity, all patterned fields in an OpenRPC document MUST have unique key names within the containing object.

Rich Text Formatting

Throughout the specification `description` fields are noted as supporting Github markdown formatting. Where OpenRPC tooling renders rich text it MUST support, at a minimum, markdown syntax as described by GitHub Flavored Markdown. Tooling MAY choose to ignore some GitHub Flavored Markdown features to address security concerns.

Service Discovery Method

JSON-RPC APIs can support the OpenRPC specification by implementing a service discovery method that will return the OpenRPC schema for the JSON-RPC API. The method MUST be named `rpc.discover`. The `rpc.` prefix is a reserved method prefix for JSON-RPC 2.0 specification system extensions. Below is the OpenRPC specification for the service discovery method:

```
{
  "methods": [
    {
      "name": "rpc.discover",
      "description": "Returns an OpenRPC schema as a description of this service",
      "params": [],
      "result": {
        "name": "OpenRPC Schema",
        "schema": {
          "$ref": "https://raw.githubusercontent.com/open-rpc/meta-schema/master/schema.json"
        }
      }
    }
  ]
}
```

Examples

Example OpenRPC documents can be found in the OpenRPC Examples Repository. There SHOULD be an example that uses every concept of the spec. These examples are to be used as the basis of testing for all the Official OpenRPC tooling.

Meta JSON Schema

Validating an OpenRPC document can be accomplished using the OpenRPC MetaSchema. The OpenRPC MetaSchema is based on the Draft 07 JSON Schema, and may be used as a JSON meta-schema for various tooling use. Each field in the Specification MUST be included in the OpenRPC MetaSchema, including all constraints that are possible to model with Draft 07 JSON Schema.

OpenRPC Object

This is the root object of the OpenRPC document. The contents of this object represent a whole OpenRPC document. How this object is constructed or stored is outside the scope of the OpenRPC Specification.

Field Name	Type	Description
openrpc	`string`	**REQUIRED**. This string MUST be the semantic version number of the OpenRPC Specification version that the OpenRPC document uses. The `openrpc` field SHOULD be used by tooling specifications and clients to interpret the OpenRPC document. This is *not* related to the API `info.version` string.
info	Info Object	**REQUIRED**. Provides metadata about the API. The metadata MAY be used by tooling as required.
servers	[Server Object]	An array of Server Objects, which provide connectivity information to a target server. If the `servers` property is not provided, or is an empty array, the default value would be a Server Object with a url value of `localhost`.
methods	[Method Object | Reference Object]	**REQUIRED**. The available methods for the API. While it is required, the array may be empty (to handle security filtering, for example).
components	Components Object	An element to hold various schemas for the specification.
externalDocs	External Documentation Object	Additional external documentation.

This object MAY be extended with Specification Extensions.

Info Object

The object provides metadata about the API. The metadata MAY be used by the clients if needed, and MAY be presented in editing or documentation generation tools for convenience.

Field Name	Type	Description
title	`string`	**REQUIRED**. The title of the application.
description	`string`	A verbose description of the application. GitHub Flavored Markdown syntax MAY be used for rich text representation.
termsOfService	`string`	A URL to the Terms of Service for the API. MUST be in the format of a URL.
contact	Contact Object	The contact information for the exposed API.
license	License Object	The license information for the exposed API.
version	`string`	**REQUIRED**. The version of the OpenRPC document (which is distinct from the OpenRPC Specification version or the API implementation version).

This object MAY be extended with Specification Extensions.

Contact Object

Contact information for the exposed API.

Field Name	Type	Description
name	`string`	The identifying name of the contact person/organization.
url	`string`	The URL pointing to the contact information. MUST be in the format of a URL.
email	`string`	The email address of the contact person/organization. MUST be in the format of an email address.

This object MAY be extended with Specification Extensions.

License Object

License information for the exposed API.

Field Name	Type	Description
name	`string`	**REQUIRED**. The license name used for the API.
url	`string`	A URL to the license used for the API. MUST be in the format of a URL.

This object MAY be extended with Specification Extensions.

Server Object

An object representing a Server.

Field Name	Type	Description
name	`string`	**REQUIRED**. A name to be used as the cannonical name for the server.
url	Runtime Expression	**REQUIRED**. A URL to the target host. This URL supports Server Variables and MAY be relative, to indicate that the host location is relative to the location where the OpenRPC document is being served. Server Variables are passed into the Runtime Expression to produce a server URL.
summary	`string`	A short summary of what the server is.

Field Name	Type	Description
description	`string`	An optional string describing the host designated by the URL. GitHub Flavored Markdown syntax MAY be used for rich text representation.
variables	Map[`string`, Server Variable Object]	A map between a variable name and its value. The value is passed into the Runtime Expression to produce a server URL.

This object MAY be extended with Specification Extensions.

Server Variable Object

An object representing a Server Variable for server URL template substitution.

Field Name	Type	Description
enum	`[string]`	An enumeration of string values to be used if the substitution options are from a limited set.
default	`string`	**REQUIRED**. The default value to use for substitution, which SHALL be sent if an alternate value is *not* supplied. Note this behavior is different than the Schema Object's treatment of default values, because in those cases parameter values are optional.
description	`string`	An optional description for the server variable. GitHub Flavored Markdown syntax MAY be used for rich text representation.

This object MAY be extended with Specification Extensions.

Method Object

Describes the interface for the given method name. The method name is used as the `method` field of the JSON-RPC body. It therefore MUST be unique.

Field Name	Type	Description
name	`string`	**REQUIRED**. The cannonical name for the method. The name MUST be unique within the methods array.
tags	[Tag Object | Reference Object]	A list of tags for API documentation control. Tags can be used for logical grouping of methods by resources or any other qualifier.
summary	`string`	A short summary of what the method does.
description	`string`	A verbose explanation of the method behavior. GitHub Flavored Markdown syntax MAY be used for rich text representation.
externalDocs	External Documentation Object	Additional external documentation for this method.
params	[Content Descriptor | Reference Object]	**REQUIRED**. A list of parameters that are applicable for this method. The list MUST NOT include duplicated parameters and therefore require name to be unique. The list can use the Reference Object to link to parameters that are defined by the Content Descriptor Object. All optional params (content descriptor objects with "required": false) MUST be positioned after all required params in the list.
result	Content Descriptor | Reference Object	The description of the result returned by the method. If defined, it MUST be a Content Descriptor or Reference Object. If undefined, the method MUST only be used as a notification.
deprecated	`boolean`	Declares this method to be deprecated. Consumers SHOULD refrain from usage of the declared method. Default value is `false`.
servers	[Server Object]	An alternative `servers` array to service this method. If an alternative `servers` array is specified at the Root level, it will be overridden by this value.
errors	[Error Object | Reference Object]	A list of custom application defined errors that MAY be returned. The Errors MUST have unique error codes.
links	[Link Object | Reference Object]	A list of possible links from this method call.
paramStructure	`"by-name"` | `"by-position"` | `"either"`	The expected format of the parameters. As per the JSON-RPC 2.0 specification, the params of a JSON-RPC request object may be an array, object, or either (represented as `by-position`, `by-name`, and `either` respectively). When a method has a `paramStructure` value of `by-name`, callers of the method MUST send a JSON-RPC request object whose `params` field is an object. Further, the key names of the `params` object MUST be the same as the `contentDescriptor.name`s for the given method. Defaults to `"either"`.
examples	[Example Pairing Object | Reference Object]	Array of Example Pairing Objects where each example includes a valid params-to-result Content Descriptor pairing.

This object MAY be extended with Specification Extensions.

Content Descriptor Object

Content Descriptors are objects that do just as they suggest - describe content. They are reusable ways of describing either parameters or result. They MUST have a schema.

Field Name	Type	Description
name	`string`	**REQUIRED**. Name of the content that is being described. If the content described is a method parameter assignable `by-name`, this field SHALL define the parameter's key (*ie* name).

Field Name	Type	Description
summary	`string`	A short summary of the content that is being described.
description	`string`	A verbose explanation of the content descriptor behavior. GitHub Flavored Markdown syntax MAY be used for rich text representation.
required	`boolean`	Determines if the content is a required field. Default value is `false`.
schema	Schema Object	**REQUIRED**. Schema that describes the content.
deprecated	`boolean`	Specifies that the content is deprecated and SHOULD be transitioned out of usage. Default value is `false`.

This object MAY be extended with Specification Extensions.

Schema Object

The Schema Object allows the definition of input and output data types. The Schema Objects MUST follow the specifications outline in the JSON Schema Specification 7 Alternatively, any time a Schema Object can be used, a Reference Object can be used in its place. This allows referencing definitions instead of defining them inline.

This object MAY be extended with Specification Extensions.

Example Pairing Object

The Example Pairing object consists of a set of example params and result. The result is what you can expect from the JSON-RPC service given the exact params.

Field Name	Type	Description
name	`string`	**REQUIRED** Name for the example pairing.
description	`string`	A verbose explanation of the example pairing.
summary	`string`	Short description for the example pairing.
params	[Example Object | Reference Object]	**REQUIRED** Example parameters.
result	Example Object | Reference Object	Example result. When undefined, the example pairing represents usage of the method as a notification.

This object MAY be extended with Specification Extensions.

Example Object

The Example object is an object that defines an example that is intended to match the `schema` of a given Content Descriptor.

Field Name	Type	Description
name	`string`	Cannonical name of the example.
summary	`string`	Short description for the example.
description	`string`	A verbose explanation of the example. GitHub Flavored Markdown syntax MAY be used for rich text representation.
value	Any	Embedded literal example. The `value` field and `externalValue` field are mutually exclusive. To represent examples of media types that cannot naturally represented in JSON, use a string value to contain the example, escaping where necessary.
externalValue	`string`	A URL that points to the literal example. This provides the capability to reference examples that cannot easily be included in JSON documents. The `value` field and `externalValue` field are mutually exclusive.

This object MAY be extended with Specification Extensions.

In all cases, the example value is expected to be compatible with the type schema of its associated value. Tooling implementations MAY choose to validate compatibility automatically, and reject the example value(s) if incompatible.

Link Object

The `Link object` represents a possible design-time link for a result. The presence of a link does not guarantee the caller's ability to successfully invoke it, rather it provides a known relationship and traversal mechanism between results and other methods.

Unlike *dynamic* links (i.e. links provided in the result payload), the OpenRPC linking mechanism does not require link information in the runtime result.

For computing links, and providing instructions to execute them, a runtime expression is used for accessing values in an method and using them as parameters while invoking the linked method.

Field Name	Type	Description
name	`string`	**REQUIRED**. Cannonical name of the link.
description	`string`	A description of the link. GitHub Flavored Markdown syntax MAY be used for rich text representation.
summary	`string`	Short description for the link.
method	`string`	The name of an *existing*, resolvable OpenRPC method, as defined with a unique `method`. This field MUST resolve to a unique Method Object. As opposed to Open Api, Relative `method` values ARE NOT permitted.

Field Name	Type	Description
params	Map[`string`, `Any` | Runtime Expression]	A map representing parameters to pass to a method as specified with `method`. The key is the parameter name to be used, whereas the value can be a constant or a runtime expression to be evaluated and passed to the linked method.
server	Server Object	A server object to be used by the target method.

A linked method must be identified directly, and must exist in the list of methods defined by the Methods Object.

When a runtime expression fails to evaluate, no parameter value is passed to the target method.

Values from the result can be used to drive a linked method.

Clients follow all links at their discretion. Neither permissions, nor the capability to make a successful call to that link, is guaranteed solely by the existence of a relationship.

This object MAY be extended with Specification Extensions.

Runtime Expression

Runtime expressions allow the user to define an expression which will evaluate to a string once the desired value(s) are known. They are used when the desired value of a link or server can only be constructed at run time. This mechanism is used by Link Objects and Server Variables.

The runtime expression makes use of JSON Template Language syntax.

The table below provides examples of runtime expressions and examples of their use in a value:

Runtime expressions preserve the type of the referenced value.

Error Object

Defines an application level error.

Field Name	Type	Description
code	Application Defined Error Code	**REQUIRED**. A Number that indicates the error type that occurred. This MUST be an integer. The error codes from and including -32768 to -32000 are reserved for pre-defined errors. These pre-defined errors SHOULD be assumed to be returned from any JSON-RPC api.
message	`string`	**REQUIRED**. A String providing a short description of the error. The message SHOULD be limited to a concise single sentence.
data	`any`	A Primitive or Structured value that contains additional information about the error. This may be omitted. The value of this member is defined by the Server (e.g. detailed error information, nested errors etc.).

Components Object

Holds a set of reusable objects for different aspects of the OpenRPC. All objects defined within the components object will have no effect on the API unless they are explicitly referenced from properties outside the components object.

Field Name	Type	Description
contentDescriptors	Map[`string`, Content Descriptor Object]	An object to hold reusable Content Descriptor Objects.
schemas	Map[`string`, Schema Object]	An object to hold reusable Schema Objects.
examples	Map[`string`, Example Object]	An object to hold reusable Example Objects.
links	Map[`string`, Link Object]	An object to hold reusable Link Objects.
errors	Map[`string`, Error Object]	An object to hold reusable Error Objects.
examplePairingObjects	Map[`string`, Example Pairing Object]	An object to hold reusable Example Pairing Objects.
tags	Map[`string`, Tag Object]	An object to hold reusable Tag Objects.

This object MAY be extended with Specification Extensions.

All the fixed fields declared above are objects that MUST use keys that match the regular expression: `^[a-zA-Z0-9\.\-_]+$`

Tag Object

Adds metadata to a single tag that is used by the Method Object. It is not mandatory to have a Tag Object per tag defined in the Method Object instances.

Field Name	Type	Description
name	`string`	**REQUIRED**. The name of the tag.
summary	`string`	A short summary of the tag.
description	`string`	A verbose explanation for the tag. GitHub Flavored Markdown syntax MAY be used for rich text representation.
externalDocs	External Documentation Object	Additional external documentation for this tag.

This object MAY be extended with Specification Extensions.

External Documentation Object

Allows referencing an external resource for extended documentation.

Field Name	Type	Description
description	`string`	A verbose explanation of the target documentation. GitHub Flavored Markdown syntax MAY be used for rich text representation.
url	`string`	**REQUIRED**. The URL for the target documentation. Value MUST be in the format of a URL.

This object MAY be extended with Specification Extensions.

Reference Object

A simple object to allow referencing other components in the specification, internally and externally.

The Reference Object is defined by JSON Schema and follows the same structure, behavior and rules.

Field Name	Type	Description
$ref	`string`	**REQUIRED**. The reference string.

This object cannot be extended with additional properties and any properties added SHALL be ignored.

Specification Extensions

While the OpenRPC Specification tries to accommodate most use cases, additional data can be added to extend the specification at certain points.

The extensions properties are implemented as patterned fields that are always prefixed by `"x-"`.

Field Pattern	Type	Description
^x-	Any	Allows extensions to the OpenRPC Schema. The field name MUST begin with `x-`, for example, `x-internal-id`. The value can be `null`, a primitive, an array or an object. Can have any valid JSON format value.

The extensions may or may not be supported by the available tooling, but those may be extended as well to add requested support (if tools are internal or open-sourced).