TODO: Specify other constrains such as length of strings…

# 1. Microrestjs Service Description Specification

## 1.1. Top Level

{

“microrestSpecification”: 1,

“info”: <Information Object>,

“config”: <Configuration Object>,

“operations”: <Operations Object>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| microrestSpecification | integer | **Required.** Specifies the version of Microrestjs Service Description Specification used to describe the service.  This value must be 1. |
| info | Information Object | **Required.** Provides meta-information about the service. |
| config | Configuration Object | **Required.** Provides some necessary configuration properties to run the service. |
| operations | Operations Object | **Required.** Defines the service operations.  The service must have at least one operation. |

## 1.2. Information Object

{

"name": <string: organization-service>,

"version": <string: MAJOR.MINOR.PATCH>,

"api": <integer>,

"description": <string>,

"keywords": [string],

"date": <string: YYYY-MM-DD>,

"authors": [Person Object],

"contributors": [Person Object],

"homepage": <string: URL>,

"repository": <Repository Object>,

"bugs": <Bugs Object>,

"termsOfService": <string: URL>,

"license": <string: SPDX Format>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| name | string | **Required.** Specifies the name of the service.  It must be unique in the world and should describe the service. Therefore, it is highly recommendable to follow the format: organization-service OR product-service. |
| version | string | **Required.** Specifies the version of the service.  The format of the version must be MAJOR.MINOR.PATCH and should follow the [Semantic Versioning 2.0.0](http://semver.org/) definition where an increment in   * MAJOR indicates that at least one service operation API has been modified and therefore the backwards-compatibility has been broken. * MINOR indicates that at least one new service operation has been added without changing the API of the previous implemented operations. The backwards-compatibility is assured. * PATCH indicates that an already implemented service operation has been fixed or re-implemented without modifying its API. The backwards-compatibility is assured. |
| api | integer | **Required.** Specifies the number of the service API.  It must be a positive integer. An increase of this number means that the service API has been changed. Therefore, the API number must be increased when the service has MAJOR or MINOR change. |
| description | string | Describes the service using natural language. |
| keywords | [string] | Specifies the list of keywords that characterize the service. |
| date | string | Specifies the release date of the current service version.  The date format must be YYYY-MM-DD as the definition of full-date in [RFC3339](http://xml2rfc.ietf.org/public/rfc/html/rfc3339.html#anchor14). |
| authors | [Person Object] | Provides information about the author/s of the service. |
| contributors | [Person Object] | Provide information about the contributor/s of the service. |
| homepage | string | Specifies the URL to the official website of the service.  The URL should follow the format: <http or https>://<hostname or IP>:<port>/<path> |
| repository | Repository Object | Specifies the public repository that manages the code of the service. |
| bugs | Bugs Object | Provides information to report service issues. |
| termsOfService | string | Specifies the URL to the Terms of Service webpage.  The URL should follow the format: <http or https>://<hostname or IP>:<port>/<path> |
| license | string | Specifies the license/s of the service.  Format:   * If the license is part of the SPDX licenses, the SPDX identifier should be included. E.g. “GPL-3.0”. * If the license has not been assigned a SPDX identifier or if the license is a custom one, the format should be “LicenseREF-LICENSE”. And a LICENSE file should be included. * If more than one license is used, SPDX expressions should be used. E.g. “(MIT OR MPL-1.0)”.   For more information, please check the official specification of [SPDX](http://spdx.org/). |

## 1.3. Person Object

{

"name": <string>,

"email": <string: EMAIL>,

"url": <string: URL>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| name | string | **Required** if included**.** Specifies the name of the person. |
| email | string | Specifies the email of the person.  The email should follow the format: <local-part>@<hostname or IP> |
| url | string | Specifies the URL to the webpage or social profile of the person.  The URL should follow the format: <http or https>://<hostname or IP>:<port>/<path> |

## 1.4. Repository Object

{

"type": <string>,

"url": <string: URL>,

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| type | string | **Required** if included. Specifies the type of the code repository. E.g. “git”, “svn”, … |
| url | string | **Required** if included. Specifies the URL to the code repository.  The URL should follow the format: <http or https>://<hostname or IP>:<port>/<path> |

## 1.5. Bugs Object

{

"url": <string: URL>,

"email": <string: EMAIL>,

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| url | string | **Required** if included1. Specifies the URL to the project’s issue tracker.  The URL should follow the format: <http or https>://<hostname or IP>:<port>/<path> |
| email | string | **Required** if included1. Specifies the email address to which issues should be reported.  The email should follow the format: <local-part>@<hostname or IP> |

1At least one of the fields is required if this object is included.

## 1.6. Configuration Object

{

"location": <string: “directory” or URL>,

"dependencies": <Dependencies Object>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| location | string | **Required.** Specifies the location of the service.  If the service is dynamic and does not have a fixed URL, this field must be filled with:   * “directory” to register the service in the default directory. * or “directory://<hostname or IP>:<port>/path” to register the service in an external directory.   Otherwise, if the service is static and will be always deployed on the same URL, this field can also be filled with that URL. In that case, the URL must follow the format: <http or https>://<hostname or IP>:<port>/<path> |
| dependencies | Dependencies Object | Specifies the external services that are necessary for the correct operation of this service. |

## 1.7. Dependencies Object

“<serviceName>”: {

"api": <integer>,

"url": <string: “directory” or URL>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| serviceName | string | **Required** if included. Specifies the name of the external service. |
| api | integer | **Required** if included. Specifies the API number of the external service. |
| url | string | **Required** if included. Specifies the location of the external service.  If the external service has been registered in a directory, this field must be filled with:   * “directory” if the external service has been registered in the directory defined by default. * or “directory://<hostname or IP>:<port>/path” if the external service has been registered in an external directory   Otherwise, if the external service has not been registered in a directory and is always deployed on the same URL, this field can also be filled with that URL. In that case, the URL must follow the format: <http or https>://<hostname or IP>:<port>/<path> |

## 1.8. Operations Object

"<operationName>": {

"request": <Request Object>,

"responses": <Responses Object>,

"errors": <Errors Object>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| operationName | string | **Required**. Specifies the name of the operation. This name must be equal to the name of the function that executes the operation.  This name must be unique and therefore no other operation of this service can have the same name. Moreover, it is highly recommendable to write the operation name using camelCase format. |
| request | Request Object | **Required**. Defines the properties of the HTTP request that executes this operation. |
| responses | Responses Object | **Required**. Defines all the possible HTTP responses that the operation can reply.  The operation must have at least one response. |
| errors | Errors Object | Defines all the possible errors and exceptions that the operation can reply. |

## 1.9. Request Object

{

"path": <string: URL Path>

"method": <string: HTTP Method>,

"parameters": [<Parameter Object>],

"body": <Body Object>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| path | string | **Required**. Specifies the URL path where the requests must be sent to execute the operation.  The path must follow the format:   * /<path> * or /:<pathParameter> * or any combination of the previous one   **Important:** the pair (path, method) must be unique and therefore no other operation of this service can have the same path and method. |
| method | string | **Required**. Specifies the method HTTP that is required to execute the operation.  This field must be filled with one of the following methods:   * “GET” * “HEAD” * “POST” * “PUT” * “DELETE” * “PATCH”   **Important:** the pair (path, method) must be unique and therefore no other operation of this service can have the same path and method. |
| parameters | [Parameter Object] | Defines the query or path parameters of the request. |
| body | Body Object | Defines the body objects that must be sent in the request to execute correctly the operation.  The definition of the body forces to send at least the defined objects correctly. In any case, other not-defined objects can be sent but their correctness will not be possible to check automatically. |

## 1.10. Responses Object

"<responseName>": {

"statusCode": <integer>,

"description": <string>,

"body": <Body Object>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| responseName | string | **Required**. Specifies the name of this response.  The response name must be unique and therefore no other response or error of this operation can have the same name. Furthermore, it is highly recommendable to use the name of the status code. E.g. “OK” if status code is 200, “ACCEPTED” if status code is 202, … |
| statusCode | integer | **Required**. Specifies the status code of this response.  The status code must follow the [HTTP Status Codes](http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml) specifications. |
| description | string | Specifies the meaning of this response and the reasons to reply this response. |
| body | Body Object | Defines the body objects that are included in the response.  The definition of the body forces to send at least the defined objects correctly. In any case, other not-defined objects can be sent but their correctness will not be possible to check automatically. |

## 1.11. Errors Object

"<errorName>": {

"statusCode": <integer>,

"description": <string>,

"body": <Body Object>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| errorName | string | **Required**. Specifies the name of this error response.  The error name must be unique and therefore no other response or error of this operation can have the same name. Furthermore, it is highly recommendable to use the name of the status code. E.g. “UNAUTHORIZED” if status code is 401, “NOT FOUND” if status code is 404, … |
| statusCode | integer | **Required**. Specifies the status code of this error response.  The status code must follow the [HTTP Status Codes](http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml) specifications. |
| description | string | Specifies the meaning of this error response and the reasons to reply this error response. |
| body | Body Object | Defines the body objects that are included in the error response.  The definition of the body forces to send at least the defined objects correctly. In any case, other not-defined objects can be sent but their correctness will not be possible to check automatically. |

## 1.12. Parameter Object

{

"name": <string>,

"in": <string: "query" or “path”>,

"description": <string>,

"required": <boolean>,

"type": <string>,

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| name | string | **Required** if included. Specifies the name of the parameter. |
| in | string | **Required** if included. Specifies if the parameter is a query parameter or a path parameter. Therefore, this field must be filled with “query” or “path”. |
| description | string | Specifies the meaning of the parameter and the accepted values. |
| required | boolean | **Required** if included. Specifies if the parameter is either required (true) or optional (false). |
| type | string | **Required** if included. Specifies the type of the parameter.  This field can be filled with one of the following types:   * “string” * “integer” * “number” * “boolean” |

## 1.13. Body Object

"<objectName>": {

"type": <string>,

"items": <string>,

"description": <string>,

"properties": <Body Object>,

"required": <boolean>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| objectName | string | **Required** if included. Specifies the name of the object. |
| type | string | **Required** if included. Specifies the type of the object.  This field can be filled with one of the following types:   * “object” * “string” * “integer” * “number” * “boolean” * “array” |
| items | string | **Required** if type is ”array”. Specifies the type of array items.  This field can be filled with one of the following types:   * “object” * “string” * “integer” * “number” * “boolean” |
| description | string | Specifies the meaning of the object and how to fill it. |
| properties | Body Object | **Required** if type is “object”. In this case, specifies the properties of the object. Each property is considered as a body object. This means that properties are like sub-objects and therefore can contain other sub-properties.  **Required** if type is “array” and items is “object”. In this case, specifies the kind of objects that the array can contain. Therefore, one and only one property can be specified in order to avoid the mixture of different object types. |
| required | boolean | **Required** if included. Specifies if the object is either required (true) or optional (false). |

## 1.14. Body Object Reference

{

"$ref": <JSON Reference>,

"required": <boolean>

}

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| $ref | JSON Reference | **Required** if included. Set a JSON reference to another object defined in the service description file. This is useful to reuse and avoid the duplication of objects. |
| required | boolean | Specifies if the object is either required (true) or optional (false).  When it is specified, the value *required* defined in the referenced object is overridden. |