



OpenAPI Specification

(fka Swagger RESTful API Documentation Specification)

Version 2.0

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#).

The Swagger specification is licensed under [The Apache License, Version 2.0](#).

Introductions

Swagger is a project used to describe and document RESTful APIs.

The Swagger specification defines a set of files required to describe such an API. These files can then be used by the Swagger-UI project to display the API and Swagger-Codegen to generate clients in various languages. Additional utilities can also take advantage of the resulting files, such as testing tools.

Revision History

Version	Date	Notes
2.0	2014-09-08	Release of Swagger 2.0
1.2	2014-03-14	Initial release of the formal document.
1.1	2012-08-22	Release of Swagger 1.1
1.0	2011-08-10	First release of the Swagger Specification

Definitions

Path Templating

Path templating refers to the usage of curly braces ({}) to mark a section of a URL path as replaceable using path parameters.

Mime Types

Mime type definitions are spread across several resources. The mime type definitions should be in compliance with [RFC 6838](#).

Some examples of possible mime type definitions:

```
1.  text/plain; charset=utf-8
2.  application/json
3.  application/vnd.github+json
4.  application/vnd.github.v3+json
5.  application/vnd.github.v3.raw+json
6.  application/vnd.github.v3.text+json
7.  application/vnd.github.v3.html+json
8.  application/vnd.github.v3.full+json
9.  application/vnd.github.v3.diff
10. application/vnd.github.v3.patch
```

HTTP Status Codes

Specification

Format

The files describing the RESTful API in accordance with the Swagger specification are represented as JSON objects and conform to the JSON standards. YAML, being a superset of JSON, can be used as well to represent a Swagger specification file.

For example, if a field is said to have an array value, the JSON array representation will be used:

```
1. {
2.   "field" : [...]
3. }
```

While the API is described using JSON it does not impose a JSON input/output to the API itself.

All field names in the specification are **case sensitive**.

The schema exposes two types of fields. Fixed fields, which have a declared name, and Patterned fields, which declare a regex pattern for the field name. Patterned fields can have multiple occurrences as long as each has a unique name.

File Structure

The Swagger representation of the API is made of a single file. However, parts of the definitions can be split into separate files, at the discretion of the user. This is applicable for **\$ref** fields in the specification as follows from the [JSON Schema](#) definitions.

By convention, the Swagger specification file is named **swagger.json**.

Data Types

Primitive data types in the Swagger Specification are based on the types supported by the [JSON-Schema Draft 4](#). Models are described using the [Schema Object](#) which is a subset of JSON Schema Draft 4.

An additional primitive data type **"file"** is used by the [Parameter Object](#) and the [Response Object](#) to set the parameter type or the response as being a file.

Primitives have an optional modifier property **format**. Swagger uses several known formats to more finely define the data type being used. However, the **format** property is an open **string**-valued property, and can have any value to support documentation needs. Formats such as **"email"**, **"uuid"**, etc., can be used even though they are not defined by this specification. Types that are not accompanied by a **format** property follow their definition from the JSON Schema (except for **file** type which is defined above). The formats defined by the Swagger Specification are:

Common Name	type	format	Comments
integer	integer	int32	signed 32 bits
long	integer	int64	signed 64 bits
float	number	float	
double	number	double	
string	string		

byte	string	byte	base64 encoded characters
binary	string	binary	any sequence of octets
boolean	boolean		
date	string	date	As defined by full-date - RFC3339
dateTime	string	date-time	As defined by date-time - RFC3339
password	string	password	Used to hint UIs the input needs to be obscured.

Schema

Swagger Object

This is the root document object for the API specification. It combines what previously was the Resource Listing and API Declaration (version 1.2 and earlier) together into one document.

Fixed Fields

Field Name	Type	Description
swagger	string	Required. Specifies the Swagger Specification version being used. It can be used by the Swagger UI and other clients to interpret the API listing. The value MUST be "2.0".
info	Info Object	Required. Provides metadata about the API. The metadata can be used by the clients if needed.
host	string	The host (name or ip) serving the API. This MUST be the host only and does not include the scheme nor sub-paths. It MAY include a port. If the host is not included, the host serving the documentation is to be used (including the port). The host does not support path templating .
basePath	string	The base path on which the API is served, which is relative to the host . If it is not included, the API is served directly under the host. The value MUST start with a leading slash (/). The basePath does not support path templating .
schemes	[string]	The transfer protocol of the API. Values MUST be from the list: "http", "https", "ws", "wss". If the schemes is not included, the default scheme to be used is the one used to access the Swagger definition itself.
consumes	[string]	A list of MIME types the APIs can consume. This is global to all APIs but can be overridden on specific API calls. Value MUST be as described under Mime Types .

produces	[String]	A list of MIME types the APIs can produce. This is global to all APIs but can be overridden on specific API calls. Value MUST be as described under Mime Types .
paths	Paths Object	Required. The available paths and operations for the API.
definitions	Definitions Object	An object to hold data types produced and consumed by operations.
parameters	Parameters Definitions Object	An object to hold parameters that can be used across operations. This property <i>does not</i> define global parameters for all operations.
responses	Responses Definitions Object	An object to hold responses that can be used across operations. This property <i>does not</i> define global responses for all operations.
securityDefinitions	Security Definitions Object	Security scheme definitions that can be used across the specification.
security	[Security Requirement Object]	A declaration of which security schemes are applied for the API as a whole. The list of values describes alternative security schemes that can be used (that is, there is a logical OR between the security requirements). Individual operations can override this definition.
tags	[Tag Object]	A list of tags used by the specification with additional metadata. The order of the tags can be used to reflect on their order by the parsing tools. Not all tags that are used by the Operation Object must be declared. The tags that are not declared may be organized randomly or based on the tools' logic. Each tag name in the list MUST be unique.
externalDocs	External Documentation Object	Additional external documentation.

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Info Object

The object provides metadata about the API. The metadata can be used by the clients if needed, and can be presented in the Swagger-UI for convenience.

Fixed Fields

Field Name	Type	Description
title	string	Required. The title of the application.



description	string	A short description of the application. HTML syntax can be used for rich text representation.
termsOfService	string	The Terms of Service for the API.
contact	Contact Object	The contact information for the exposed API.
license	License Object	The license information for the exposed API.
version	string	Required Provides the version of the application API (not to be confused with the specification version).

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Info Object Example:

```
1. {
2.   "title": "Swagger Sample App",
3.   "description": "This is a sample server Petstore server.",
4.   "termsOfService": "http://swagger.io/terms/",
5.   "contact": {
6.     "name": "API Support",
7.     "url": "http://www.swagger.io/support",
8.     "email": "support@swagger.io"
9.   },
10.  "license": {
11.    "name": "Apache 2.0",
12.    "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
13.  },
14.  "version": "1.0.1"
15. }
```

```
1. title: Swagger Sample App
2. description: This is a sample server Petstore server.
3. termsOfService: http://swagger.io/terms/
4. contact:
5.   name: API Support
6.   url: http://www.swagger.io/support
7.   email: support@swagger.io
8. license:
9.   name: Apache 2.0
10.  url: http://www.apache.org/licenses/LICENSE-2.0.html
11. version: 1.0.1
```

Contact Object

Contact information for the exposed API.

Fixed Fields

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.

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Contact Object Example:

```
1. {
2.   "name": "API Support",
3.   "url": "http://www.swagger.io/support",
4.   "email": "support@swagger.io"
5. }
```

```
1. name: API Support
2. url: http://www.swagger.io/support
3. email: support@swagger.io
```

License Object

License information for the exposed API.

Fixed Fields

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.

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

License Object Example:

```
1. {
2.   "name": "Apache 2.0",
3.   "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
4. }
```

```
1. name: Apache 2.0
2. url: http://www.apache.org/licenses/LICENSE-2.0.html
```

Paths Object

Patterned Fields



Field Pattern	Type	Description
/ {path}	Path Item Object	A relative path to an individual endpoint. The field name MUST begin with a slash. The path is appended to the basePath in order to construct the full URL. Path templating is allowed.
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Paths Object Example

```
1. {
2.   "/pets": {
3.     "get": {
4.       "description": "Returns all pets from the system that the user has access to",
5.       "produces": [
6.         "application/json"
7.       ],
8.       "responses": {
9.         "200": {
10.          "description": "A list of pets.",
11.          "schema": {
12.            "type": "array",
13.            "items": {
14.              "$ref": "#/definitions/pet"
15.            }
16.          }
17.        }
18.      }
19.    }
20.  }
21. }
```

```
1. /pets:
2.   get:
3.     description: Returns all pets from the system that the user has access to
4.     produces:
5.       - application/json
6.     responses:
7.       '200':
8.         description: A list of pets.
9.         schema:
10.          type: array
11.          items:
12.            $ref: '#/definitions/pet'
```

Path Item Object

Describes the operations available on a single path. A Path Item may be empty, due to [ACL constraints](#). The path itself is still exposed to the documentation viewer but they will not know which operations and parameters are available.

Fixed Fields

Field Name	Type	Description
------------	------	-------------

\$ref	string	Allows for an external definition of this path item. The referenced structure MUST be in the format of a Path Item Object . If there are conflicts between the referenced definition and this Path Item's definition, the behavior is <i>undefined</i> .
get	Operation Object	A definition of a GET operation on this path.
put	Operation Object	A definition of a PUT operation on this path.
post	Operation Object	A definition of a POST operation on this path.
delete	Operation Object	A definition of a DELETE operation on this path.
options	Operation Object	A definition of a OPTIONS operation on this path.
head	Operation Object	A definition of a HEAD operation on this path.
patch	Operation Object	A definition of a PATCH operation on this path.
parameters	[Parameter Object Reference Object]	A list of parameters that are applicable for all the operations described under this path. These parameters can be overridden at the operation level, but cannot be removed there. The list MUST NOT include duplicated parameters. A unique parameter is defined by a combination of a name and location . The list can use the Reference Object to link to parameters that are defined at the Swagger Object's parameters . There can be one "body" parameter at most.

Patterned Fields

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Path Item Object Example


```
4.   "summary": "Find pets by ID",
5.   "operationId": "getPetsById",
6.   "produces": [
7.     "application/json",
8.     "text/html"
9.   ],
10.  "responses": {
11.    "200": {
12.      "description": "pet response",
13.      "schema": {
14.        "type": "array",
15.        "items": {
16.          "$ref": "#/definitions/Pet"
17.        }
18.      }
19.    },
20.    "default": {
21.      "description": "error payload",
22.      "schema": {
23.        "$ref": "#/definitions/ErrorMessage"
24.      }
25.    }
26.  },
27.  "parameters": [
28.    {
29.      "name": "id",
30.      "in": "path",
31.      "description": "ID of pet to use",
32.      "required": true,
33.      "type": "array",
34.      "items": {
35.        "type": "string"
36.      },
37.      "collectionFormat": "csv"
38.    }
39.  ]
40. }
41. }
```


```
1. get:
2.   description: Returns pets based on ID
3.   summary: Find pets by ID
4.   operationId: getPetsById
5.   produces:
6.     - application/json
7.     - text/html
8.   responses:
9.     '200':
10.      description: pet response
11.      schema:
12.        type: array
13.        items:
14.          $ref: '#/definitions/Pet'
15.     default:
16.      description: error payload
17.      schema:
18.        $ref: '#/definitions/ErrorMessage'
19.   parameters:
20.     - name: id
21.       in: path
22.       description: ID of pet to use
23.       required: true
24.       type: array
25.       items:
26.         type: string
27.       collectionFormat: csv
```

Describes a single API operation on a path.



Fixed Fields

Field Name	Type	Description
tags	[string]	A list of tags for API documentation control. Tags can be used for logical grouping of operations by resources or any other qualifier.
summary	string	A short summary of what the operation does. For maximum readability in the swagger-ui, this field SHOULD be less than 120 characters.
description	string	A verbose explanation of the operation behavior. GFM syntax can be used for rich text representation.
externalDocs	External Documentation Object	Additional external documentation for this operation.
operationId	string	Unique string used to identify the operation. The id MUST be unique among all operations described in the API. Tools and libraries MAY use the operationId to uniquely identify an operation, therefore, it is recommended to follow common programming naming conventions.
consumes	[string]	A list of MIME types the operation can consume. This overrides the consumes definition at the Swagger Object. An empty value MAY be used to clear the global definition. Value MUST be as described under Mime Types .
produces	[string]	A list of MIME types the operation can produce. This overrides the produces definition at the Swagger Object. An empty value MAY be used to clear the global definition. Value MUST be as described under Mime Types .
parameters	[Parameter Object Reference Object]	A list of parameters that are applicable for this operation. If a parameter is already defined at the Path Item , the new definition will override it, but can never remove it. The list MUST NOT include duplicated parameters. A unique parameter is defined by a combination of a name and location . The list can use the Reference Object to link to parameters that are defined at the Swagger Object's parameters . There can be one "body" parameter at most.
responses	Responses Object	Required. The list of possible responses as they are returned from executing this operation.
schemes	[string]	The transfer protocol for the operation. Values MUST be from the list: "http", "https", "ws", "wss". The value overrides the Swagger Object schemes definition.
deprecated	boolean	Declares this operation to be deprecated. Usage of the declared operation should be refrained. Default value is false .

security	security Requirement Object]	A declaration of which security schemes are applied for this operation. The list of values describes alternative security schemes that can be used (that is, there is a logical OR between the security requirements). This definition overrides any declared top-level security . To remove a top-level security declaration, an empty array can be used.	
----------	---	--	---

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Operation Object Example

```
4.   ],
5.   "summary": "Updates a pet in the store with form data",
6.   "description": "",
7.   "operationId": "updatePetWithForm",
8.   "consumes": [
9.     "application/x-www-form-urlencoded"
10.  ],
11.  "produces": [
12.    "application/json",
13.    "application/xml"
14.  ],
15.  "parameters": [
16.    {
17.      "name": "petId",
18.      "in": "path",
19.      "description": "ID of pet that needs to be updated",
20.      "required": true,
21.      "type": "string"
22.    },
23.    {
24.      "name": "name",
25.      "in": "formData",
26.      "description": "Updated name of the pet",
27.      "required": false,
28.      "type": "string"
29.    },
30.    {
31.      "name": "status",
32.      "in": "formData",
33.      "description": "Updated status of the pet",
34.      "required": false,
35.      "type": "string"
36.    }
37.  ],
38.  "responses": {
39.    "200": {
40.      "description": "Pet updated."
41.    },
42.    "405": {
43.      "description": "Invalid input"
44.    }
45.  },
46.  "security": [
47.    {
48.      "petstore_auth": [
49.        "write:pets",
50.        "read:pets"
51.      ]
52.    }
53.  ]
54. }
```





```
4. description: ""
5. operationId: updatePetWithForm
6. consumes:
7. - application/x-www-form-urlencoded
8. produces:
9. - application/json
10. - application/xml
11. parameters:
12. - name: petId
13.   in: path
14.   description: ID of pet that needs to be updated
15.   required: true
16.   type: string
17. - name: name
18.   in: formData
19.   description: Updated name of the pet
20.   required: false
21.   type: string
22. - name: status
23.   in: formData
24.   description: Updated status of the pet
25.   required: false
26.   type: string
27. responses:
28.   '200':
29.     description: Pet updated.
30.   '405':
31.     description: Invalid input
32. security:
33. - petstore_auth:
34.   - write:pets
35.   - read:pets
```

External Documentation Object

Allows referencing an external resource for extended documentation.

Fixed Fields

Field Name	Type	Description
description	string	A short description of the target documentation. GFM syntax can be used for rich text representation.
url	string	Required. The URL for the target documentation. Value MUST be in the format of a URL.

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

External Documentation Object Example

```
1. {
2.   "description": "Find more info here",
3.   "url": "https://swagger.io"
4. }
```

Parameter Object

Describes a single operation parameter.

A unique parameter is defined by a combination of a [name](#) and [location](#).


There are five possible parameter types.

- Path - Used together with [Path Templating](#), where the parameter value is actually part of the operation's URL. This does not include the host or base path of the API. For example, in `/items/{itemId}`, the path parameter is `itemId`.
- Query - Parameters that are appended to the URL. For example, in `/items?id=###`, the query parameter is `id`.
- Header - Custom headers that are expected as part of the request.
- Body - The payload that's appended to the HTTP request. Since there can only be one payload, there can only be *one* body parameter. The name of the body parameter has no effect on the parameter itself and is used for documentation purposes only. Since Form parameters are also in the payload, body and form parameters cannot exist together for the same operation.
- Form - Used to describe the payload of an HTTP request when either `application/x-www-form-urlencoded`, `multipart/form-data` or both are used as the content type of the request (in Swagger's definition, the [consumes](#) property of an operation). This is the only parameter type that can be used to send files, thus supporting the `file` type. Since form parameters are sent in the payload, they cannot be declared together with a body parameter for the same operation. Form parameters have a different format based on the content-type used (for further details, consult <http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4>):
 - `application/x-www-form-urlencoded` - Similar to the format of Query parameters but as a payload. For example, `foo=1&bar=swagger` - both `foo` and `bar` are form parameters. This is normally used for simple parameters that are being transferred.
 - `multipart/form-data` - each parameter takes a section in the payload with an internal header. For example, for the header `Content-Disposition: form-data; name="submit-name"` the name of the parameter is `submit-name`. This type of form parameters is more commonly used for file transfers.

Fixed Fields

Field Name	Type	Description
name	string	Required. The name of the parameter. Parameter names are <i>case sensitive</i> . <ul style="list-style-type: none">• If in is "path", the <code>name</code> field MUST correspond to the associated path segment from the path field in the Paths Object. See Path Templating for further information.• For all other cases, the <code>name</code> corresponds to the parameter name used based on the in property.
in	string	Required. The location of the parameter. Possible values are "query", "header", "path", "formData" or "body".
description	string	A brief description of the parameter. This could contain examples of use. GFM syntax can be used for rich text representation.
required	boolean	Determines whether this parameter is mandatory. If the parameter is in "path", this property is required and its value MUST be <code>true</code> . Otherwise, the property MAY be included and its default value is <code>false</code> .

If [in](#) is "body":

schema	Schema Object	Required. The schema defining the type used for the body parameter.	
--------	-------------------------------	--	---

If [in](#) is any value other than "body":

Field Name	Type	Description
type	string	Required. The type of the parameter. Since the parameter is not located at the request body, it is limited to simple types (that is, not an object). The value MUST be one of "string", "number", "integer", "boolean", "array" or "file". If type is "file", the consumes MUST be either "multipart/form-data", " application/x-www-form-urlencoded" or both and the parameter MUST be in "formData".
format	string	The extending format for the previously mentioned type . See Data Type Formats for further details.
allowEmptyValue	boolean	Sets the ability to pass empty-valued parameters. This is valid only for either query or formData parameters and allows you to send a parameter with a name only or an empty value. Default value is false .
items	Items Object	Required if type is "array" . Describes the type of items in the array.
collectionFormat	string	Determines the format of the array if type array is used. Possible values are: <ul style="list-style-type: none">csv - comma separated values foo,bar.ssv - space separated values foo bar.tsv - tab separated values foo\tbar.pipes - pipe separated values foo bar.multi - corresponds to multiple parameter instances instead of multiple values for a single instance foo=bar&foo=baz. This is valid only for parameters in "query" or "formData". Default value is csv .
default	*	Declares the value of the parameter that the server will use if none is provided, for example a "count" to control the number of results per page might default to 100 if not supplied by the client in the request. (Note: "default" has no meaning for required parameters.) See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-6.2 . Unlike JSON Schema this value MUST conform to the defined type for this parameter.
maximum	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.2 .
exclusiveMaximum	boolean	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.2 .
minimum	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.3 .
exclusiveMinimum	boolean	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.3 .

maxLength	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.1 .
minLength	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.2 .
pattern	string	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.3 .
maxItems	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.2 .
minItems	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.3 .
uniqueItems	boolean	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.4 .
enum	[*]	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.5.1 .
multipleOf	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.1 .

Patterned Fields

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Parameter Object Examples

Body Parameters

A body parameter with a referenced schema definition (normally for a model definition):

```
1. {
2.   "name": "user",
3.   "in": "body",
4.   "description": "user to add to the system",
5.   "required": true,
6.   "schema": {
7.     "$ref": "#/definitions/User"
8.   }
9. }
```

```
1. name: user
2. in: body
3. description: user to add to the system
4. required: true
5. schema:
6.   $ref: '#/definitions/User'
```

A body parameter that is an array of string values:

```
4.   "description": "user to add to the system",
5.   "required": true,
6.   "schema": {
7.     "type": "array",
8.     "items": {
9.       "type": "string"
10.    }
11.  }
12. }
```

```
1. name: user
2. in: body
3. description: user to add to the system
4. required: true
5. schema:
6.   type: array
7.   items:
8.     type: string
```

Other Parameters

A header parameter with an array of 64 bit integer numbers:

```
1. {
2.   "name": "token",
3.   "in": "header",
4.   "description": "token to be passed as a header",
5.   "required": true,
6.   "type": "array",
7.   "items": {
8.     "type": "integer",
9.     "format": "int64"
10.  },
11.   "collectionFormat": "csv"
12. }
```

```
1. name: token
2. in: header
3. description: token to be passed as a header
4. required: true
5. type: array
6. items:
7.   type: integer
8.   format: int64
9. collectionFormat: csv
```

A path parameter of a string value:

```
1. {
2.   "name": "username",
3.   "in": "path",
4.   "description": "username to fetch",
5.   "required": true,
6.   "type": "string"
7. }
```

```
1. name: username
2. in: path
3. description: username to fetch
4. required: true
5. type: string
```

An optional query parameter of a string value, allowing multiple values by repeating the query parameter:

```
4.   "description": "ID of the object to fetch",
5.   "required": false,
6.   "type": "array",
7.   "items": {
8.     "type": "string"
9.   },
10.  "collectionFormat": "multi"
11. }
```

```
1. name: id
2. in: query
3. description: ID of the object to fetch
4. required: false
5. type: array
6. items:
7.   type: string
8. collectionFormat: multi
```

A form data with file type for a file upload:

```
1. {
2.   "name": "avatar",
3.   "in": "formData",
4.   "description": "The avatar of the user",
5.   "required": true,
6.   "type": "file"
7. }
```

```
1. name: avatar
2. in: formData
3. description: The avatar of the user
4. required: true
5. type: file
```

Items Object

A limited subset of JSON-Schema's items object. It is used by parameter definitions that are not located `in "body"`.

Fixed Fields

Field Name	Type	Description
type	string	Required. The internal type of the array. The value MUST be one of "string", "number", "integer", "boolean", or "array". Files and models are not allowed.
format	string	The extending format for the previously mentioned <code>type</code> . See Data Type Formats for further details.
items	Items Object	Required if <code>type</code> is "array" . Describes the type of items in the array.
collectionFormat	string	Determines the format of the array if type array is used. Possible values are: <ul style="list-style-type: none"><code>csv</code> - comma separated values <code>foo,bar</code>.<code>ssv</code> - space separated values <code>foo bar</code>.<code>tsv</code> - tab separated values <code>foo\tbar</code>.<code>pipes</code> - pipe separated values <code>foo bar</code>. Default value is <code>csv</code> .

default		Declares the value of the item that the server will use if none is provided. (Note: "default" has no meaning for required items.) See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-6.2 . Unlike JSON Schema this value MUST conform to the defined <code>type</code> for the data type.
maximum	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.2 .
exclusiveMaximum	boolean	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.2 .
minimum	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.3 .
exclusiveMinimum	boolean	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.3 .
maxLength	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.1 .
minLength	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.2 .
pattern	string	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.3 .
maxItems	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.2 .
minItems	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.3 .
uniqueItems	boolean	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.4 .
enum	[*]	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.5.1 .
multipleOf	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.1 .

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger Schema. The field name MUST begin with <code>x-</code> , for example, <code>x-internal-id</code> . The value can be <code>null</code> , a primitive, an array or an object. See Vendor Extensions for further details.

Items Object Examples

Items must be of type string and have the minimum length of 2 characters:

```
1. {
2.   "type": "string",
3.   "minLength": 2
4. }
```

```
1. type: string
2. minLength: 2
```

```
1. {
2.   "type": "array",
3.   "items": {
4.     "type": "integer",
5.     "minimum": 0,
6.     "maximum": 63
7.   }
8. }
```

```
1. type: array
2. items:
3.   type: integer
4.   minimum: 0
5.   maximum: 63
```

Responses Object

A container for the expected responses of an operation. The container maps a HTTP response code to the expected response. It is not expected from the documentation to necessarily cover all possible HTTP response codes, since they may not be known in advance. However, it is expected from the documentation to cover a successful operation response and any known errors.

The `default` can be used as the default response object for all HTTP codes that are not covered individually by the specification.

The **Responses Object** MUST contain at least one response code, and it SHOULD be the response for a successful operation call.

Fixed Fields

Field Name	Type	Description
default	Response Object Reference Object	The documentation of responses other than the ones declared for specific HTTP response codes. It can be used to cover undeclared responses. Reference Object can be used to link to a response that is defined at the Swagger Object's responses section.

Patterned Fields

Field Pattern	Type	Description
{ HTTP Status Code }	Response Object Reference Object	Any HTTP status code can be used as the property name (one property per HTTP status code). Describes the expected response for that HTTP status code. Reference Object can be used to link to a response that is defined at the Swagger Object's responses section.
<code>^x-</code>	Any	Allows extensions to the Swagger Schema. The field name MUST begin with <code>x-</code> , for example, <code>x-internal-id</code> . The value can be <code>null</code> , a primitive, an array or an object. See Vendor Extensions for further details.

Responses Object Example

A 200 response for successful operation and a default response for others (implying an error):

```
4.   "schema": {
5.     "$ref": "#/definitions/Pet"
6.   },
7. },
8. "default": {
9.   "description": "Unexpected error",
10.  "schema": {
11.    "$ref": "#/definitions/ErrorMessage"
12.  }
13. }
14. }
```

```
1. '200':
2.   description: a pet to be returned
3.   schema:
4.     $ref: '#/definitions/Pet'
5. default:
6.   description: Unexpected error
7.   schema:
8.     $ref: '#/definitions/ErrorMessage'
```

Response Object

Describes a single response from an API Operation.

Fixed Fields

Field Name	Type	Description
description	string	Required. A short description of the response. GFM syntax can be used for rich text representation.
schema	Schema Object	A definition of the response structure. It can be a primitive, an array or an object. If this field does not exist, it means no content is returned as part of the response. As an extension to the Schema Object , its root type value may also be "file" . This SHOULD be accompanied by a relevant produces mime-type.
headers	Headers Object	A list of headers that are sent with the response.
examples	Example Object	An example of the response message.

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Response Object Examples

Response of an array of a complex type:

```

4.     "type": "array",
5.     "items": {
6.       "$ref": "#/definitions/VeryComplexType"
7.     }
8.   }
9. }

```

```

1. description: A complex object array response
2. schema:
3.   type: array
4.   items:
5.     $ref: '#/definitions/VeryComplexType'

```

Response with a string type:

```

1. {
2.   "description": "A simple string response",
3.   "schema": {
4.     "type": "string"
5.   }
6. }

```

```

1. description: A simple string response
2. schema:
3.   type: string

```

Response with headers:

```

1. {
2.   "description": "A simple string response",
3.   "schema": {
4.     "type": "string"
5.   },
6.   "headers": {
7.     "X-Rate-Limit-Limit": {
8.       "description": "The number of allowed requests in the current period",
9.       "type": "integer"
10.    },
11.    "X-Rate-Limit-Remaining": {
12.      "description": "The number of remaining requests in the current period",
13.      "type": "integer"
14.    },
15.    "X-Rate-Limit-Reset": {
16.      "description": "The number of seconds left in the current period",
17.      "type": "integer"
18.    }
19.  }
20. }

```

```

1. description: A simple string response
2. schema:
3.   type: string
4. headers:
5.   X-Rate-Limit-Limit:
6.     description: The number of allowed requests in the current period
7.     type: integer
8.   X-Rate-Limit-Remaining:
9.     description: The number of remaining requests in the current period
10.    type: integer
11.   X-Rate-Limit-Reset:
12.     description: The number of seconds left in the current period
13.     type: integer

```

Response with no return value:



1. description: **object** created

Headers Object

Lists the headers that can be sent as part of a response.

Patterned Fields

Field Pattern	Type	Description
{name}	Header Object	The name of the property corresponds to the name of the header. The value describes the type of the header.

Headers Object Example

Rate-limit headers:

```
1. {
2.   "X-Rate-Limit-Limit": {
3.     "description": "The number of allowed requests in the current period",
4.     "type": "integer"
5.   },
6.   "X-Rate-Limit-Remaining": {
7.     "description": "The number of remaining requests in the current period",
8.     "type": "integer"
9.   },
10.  "X-Rate-Limit-Reset": {
11.    "description": "The number of seconds left in the current period",
12.    "type": "integer"
13.  }
14. }
```

```
1. X-Rate-Limit-Limit:
2.   description: The number of allowed requests in the current period
3.   type: integer
4. X-Rate-Limit-Remaining:
5.   description: The number of remaining requests in the current period
6.   type: integer
7. X-Rate-Limit-Reset:
8.   description: The number of seconds left in the current period
9.   type: integer
```

Example Object

Allows sharing examples for operation responses.

Patterned Fields

Field Pattern	Type	Description
{ mime type }	Any	The name of the property MUST be one of the Operation produces values (either implicit or inherited). The value SHOULD be an example of what such a response would look like.

Example Object Example

Example response for application/json mimetype of a Pet data type:

```
4.   "type": "Dog",
5.   "color": "Black",
6.   "gender": "Female",
7.   "breed": "Mixed"
8. }
9. }
```

```
1. application/json:
2.   name: Puma
3.   type: Dog
4.   color: Black
5.   gender: Female
6.   breed: Mixed
```

Header Object

Field Name	Type	Description
description	string	A short description of the header.
type	string	Required. The type of the object. The value MUST be one of "string", "number", "integer", "boolean", or "array".
format	string	The extending format for the previously mentioned type . See Data Type Formats for further details.
items	Items Object	Required if type is "array" . Describes the type of items in the array.
collectionFormat	string	Determines the format of the array if type array is used. Possible values are: <ul style="list-style-type: none">csv - comma separated values <code>foo,bar</code>.ssv - space separated values <code>foo bar</code>.tsv - tab separated values <code>foo\tbar</code>.pipes - pipe separated values <code>foo bar</code>. Default value is <code>csv</code> .
default	*	Declares the value of the header that the server will use if none is provided. (Note: "default" has no meaning for required headers.) See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-6.2 . Unlike JSON Schema this value MUST conform to the defined type for the header.
maximum	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.2 .
exclusiveMaximum	boolean	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.2 .
minimum	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.3 .
exclusiveMinimum	boolean	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.3 .
maxLength	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.1 .

maxLength	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.2 .
pattern	string	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.2.3 .
maxItems	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.2 .
minItems	integer	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.3 .
uniqueItems	boolean	https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.3.4 .
enum	[*]	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.5.1 .
multipleOf	number	See https://tools.ietf.org/html/draft-fge-json-schema-validation-00#section-5.1.1 .

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Header Object Example

A simple header with of an integer type:

```
1. {
2.   "description": "The number of allowed requests in the current period",
3.   "type": "integer"
4. }
```

```
1. description: The number of allowed requests in the current period
2. type: integer
```

Tag Object


Allows adding meta data to a single tag that is used by the [Operation Object](#). It is not mandatory to have a Tag Object per tag used there.

Fixed Fields

Field Name	Type	Description
name	string	Required. The name of the tag.
description	string	A short description for the tag. GFM syntax can be used for rich text representation.
externalDocs	External Documentation Object	Additional external documentation for this tag.

Patterned Fields

Field Pattern	Type	Description
---------------	------	-------------

^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.	
-----	-----	---	---

Tag Object Example

```
1. {
2.     "name": "pet",
3.     "description": "Pets operations"
4. }
```

```
1. name: pet
2. description: Pets operations
```

Reference Object

A simple object to allow referencing other definitions in the specification. It can be used to reference parameters and responses that are defined at the top level for reuse.

The Reference Object is a [JSON Reference](#) that uses a [JSON Pointer](#) as its value. For this specification, only [canonical dereferencing](#) is supported.

Fixed Fields

Field Name	Type	Description
\$ref	string	Required. The reference string.

Reference Object Example

```
1. {
2.     "$ref": "#/definitions/Pet"
3. }
```

```
1. $ref: '#/definitions/Pet'
```

Relative Schema File Example

```
1. {
2.     "$ref": "Pet.json"
3. }
```

```
1. $ref: 'Pet.yaml'
```

Relative Files With Embedded Schema Example

```
1. {
2.     "$ref": "definitions.json#/Pet"
3. }
```

```
1. $ref: 'definitions.yaml#/Pet'
```

Schema Object

The Schema Object allows the definition of input and output data types. These types can be objects, but also primitives and arrays. This object is based on the [JSON Schema Specification Draft 4](#) and uses a predefined subset of it. On top of this subset, there are extensions provided by this specification to allow for more complete documentation.

Further information about the properties can be found in [JSON Schema Core](#) and [JSON Schema Validation](#). Unless stated otherwise, the property definitions follow the JSON Schema specification as referenced here.

- \$ref - As a [JSON Reference](#)
- format (See [Data Type Formats](#) for further details)
- title
- description ([GFM syntax](#) can be used for rich text representation)
- default (Unlike JSON Schema, the value MUST conform to the defined type for the Schema Object)
- multipleOf
- maximum
- exclusiveMaximum
- minimum
- exclusiveMinimum
- maxLength
- minLength
- pattern
- maxItems
- minItems
- uniqueItems
- maxProperties
- minProperties
- required
- enum
- type




The following properties are taken from the JSON Schema definition but their definitions were adjusted to the Swagger Specification. Their definition is the same as the one from JSON Schema, only where the original definition references the JSON Schema definition, the [Schema Object](#) definition is used instead.

- items
- allOf
- properties
- additionalProperties

Other than the JSON Schema subset fields, the following fields may be used for further schema documentation.

Fixed Fields

Field Name	Type	Description
discriminator	string	Adds support for polymorphism. The discriminator is the schema property name that is used to differentiate between other schema that inherit this schema. The property name used MUST be defined at this schema and it MUST be in the required property list. When used, the value MUST be the name of this schema or any schema that inherits it.
readOnly	boolean	Relevant only for Schema " properties " definitions. Declares the property as "read only". This means that it MAY be sent as part of a response but MUST NOT be sent as part of the request. Properties marked as readOnly being true SHOULD NOT be in the required list of the defined schema. Default value is false .
xml	XML Object	This MAY be used only on properties schemas. It has no effect on root schemas. Adds Additional metadata to describe the XML representation format of this property.

EXTERNAL DOCS	External Documentation Object	Additional external documentation for this schema.	
example	Any	A free-form property to include an example of an instance for this schema.	

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Composition and Inheritance (Polymorphism)

Swagger allows combining and extending model definitions using the **allOf** property of JSON Schema, in effect offering model composition. **allOf** takes in an array of object definitions that are validated *independently* but together compose a single object.

While composition offers model extensibility, it does not imply a hierarchy between the models. To support polymorphism, Swagger adds the support of the **discriminator** field. When used, the **discriminator** will be the name of the property used to decide which schema definition is used to validate the structure of the model. As such, the **discriminator** field MUST be a required field. The value of the chosen property has to be the friendly name given to the model under the **definitions** property. As such, inline schema definitions, which do not have a given id, *cannot* be used in polymorphism.

XML Modeling

The [xml](#) property allows extra definitions when translating the JSON definition to XML. The [XML Object](#) contains additional information about the available options.

Schema Object Examples

Primitive Sample

Unlike previous versions of Swagger, Schema definitions can be used to describe primitive and arrays as well.

```
1. {
2.   "type": "string",
3.   "format": "email"
4. }
```

```
1. type: string
2. format: email
```

Simple Model

```
4.   "name"
5. },
6.   "properties": {
7.     "name": {
8.       "type": "string"
9.     },
10.    "address": {
11.      "$ref": "#/definitions/Address"
12.    },
13.    "age": {
14.      "type": "integer",
15.      "format": "int32",
16.      "minimum": 0
17.    }
18.  }
19. }
```

```
1. type: object
2. required:
3. - name
4. properties:
5.   name:
6.     type: string
7.   address:
8.     $ref: '#/definitions/Address'
9.   age:
10.    type: integer
11.    format: int32
12.    minimum: 0
```

Model with Map/Dictionary Properties

For a simple string to string mapping:

```
1. {
2.   "type": "object",
3.   "additionalProperties": {
4.     "type": "string"
5.   }
6. }
```

```
1. type: object
2. additionalProperties:
3.   type: string
```

For a string to model mapping:

```
1. {
2.   "type": "object",
3.   "additionalProperties": {
4.     "$ref": "#/definitions/ComplexModel"
5.   }
6. }
```

```
1. type: object
2. additionalProperties:
3.   $ref: '#/definitions/ComplexModel'
```

Model with Example



```
4.     "id": {
5.       "type": "integer",
6.       "format": "int64"
7.     },
8.     "name": {
9.       "type": "string"
10.    }
11.  },
12.  "required": [
13.    "name"
14.  ],
15.  "example": {
16.    "name": "Puma",
17.    "id": 1
18.  }
19. }
```

```
1. type: object
2. properties:
3.   id:
4.     type: integer
5.     format: int64
6.   name:
7.     type: string
8. required:
9. - name
10. example:
11.   name: Puma
12.   id: 1
```

Models with Composition



```
4.     "type": "object",
5.     "required": [
6.         "message",
7.         "code"
8.     ],
9.     "properties": {
10.        "message": {
11.            "type": "string"
12.        },
13.        "code": {
14.            "type": "integer",
15.            "minimum": 100,
16.            "maximum": 600
17.        }
18.    }
19. },
20. "ExtendedErrorModel": {
21.     "allOf": [
22.         {
23.             "$ref": "#/definitions/ErrorModel"
24.         },
25.         {
26.             "type": "object",
27.             "required": [
28.                 "rootCause"
29.             ],
30.             "properties": {
31.                 "rootCause": {
32.                     "type": "string"
33.                 }
34.             }
35.         }
36.     ]
37. }
38. }
39. }
```

```
1. definitions:
2.   ErrorModel:
3.     type: object
4.     required:
5.       - message
6.       - code
7.     properties:
8.       message:
9.         type: string
10.      code:
11.        type: integer
12.        minimum: 100
13.        maximum: 600
14.   ExtendedErrorModel:
15.     allOf:
16.       - $ref: '#/definitions/ErrorModel'
17.       - type: object
18.       required:
19.         - rootCause
20.     properties:
21.       rootCause:
22.         type: string
```

Models with Polymorphism Support

```
4.     "type": "object",
5.     "discriminator": "petType",
6.     "properties": {
7.       "name": {
8.         "type": "string"
9.       },
10.    "petType": {
11.      "type": "string"
12.    }
13.  },
14.  "required": [
15.    "name",
16.    "petType"
17.  ]
18. },
19. "Cat": {
20.   "description": "A representation of a cat",
21.   "allOf": [
22.     {
23.       "$ref": "#/definitions/Pet"
24.     },
25.     {
26.       "type": "object",
27.       "properties": {
28.         "huntingSkill": {
29.           "type": "string",
30.           "description": "The measured skill for hunting",
31.           "default": "lazy",
32.           "enum": [
33.             "clueless",
34.             "lazy",
35.             "adventurous",
36.             "aggressive"
37.           ]
38.         }
39.       },
40.       "required": [
41.         "huntingSkill"
42.       ]
43.     }
44.   ]
45. },
46. "Dog": {
47.   "description": "A representation of a dog",
48.   "allOf": [
49.     {
50.       "$ref": "#/definitions/Pet"
51.     },
52.     {
53.       "type": "object",
54.       "properties": {
55.         "packSize": {
56.           "type": "integer",
57.           "format": "int32",
58.           "description": "the size of the pack the dog is from",
59.           "default": 0,
60.           "minimum": 0
61.         }
62.       },
63.       "required": [
64.         "packSize"
65.       ]
66.     }
67.   ]
68. }
69. }
70. }
```

```
4.   discriminator: petType
5.   properties:
6.     name:
7.       type: string
8.     petType:
9.       type: string
10.  required:
11.    - name
12.    - petType
13.  Cat:
14.    description: A representation of a cat
15.    allOf:
16.      - $ref: '#/definitions/Pet'
17.      - type: object
18.        properties:
19.          huntingSkill:
20.            type: string
21.            description: The measured skill for hunting
22.            default: lazy
23.            enum:
24.              - clueless
25.              - lazy
26.              - adventurous
27.              - aggressive
28.          required:
29.            - huntingSkill
30.  Dog:
31.    description: A representation of a dog
32.    allOf:
33.      - $ref: '#/definitions/Pet'
34.      - type: object
35.        properties:
36.          packSize:
37.            type: integer
38.            format: int32
39.            description: the size of the pack the dog is from
40.            default: 0
41.            minimum: 0
42.          required:
43.            - packSize
```

XML Object

A metadata object that allows for more fine-tuned XML model definitions.

When using arrays, XML element names are *not* inferred (for singular/plural forms) and the **name** property should be used to add that information. See examples for expected behavior.

Fixed Fields

Field Name	Type	Description
name	string	Replaces the name of the element/attribute used for the described schema property. When defined within the Items Object (items), it will affect the name of the individual XML elements within the list. When defined alongside type being array (outside the items), it will affect the wrapping element and only if wrapped is true . If wrapped is false , it will be ignored.
namespace	string	The URL of the namespace definition. Value SHOULD be in the form of a URL.
prefix	string	The prefix to be used for the <u>name</u> .

attribute	boolean	Declares whether the property definition translates to an attribute instead of an element. Default value is false .
wrapped	boolean	MAY be used only for an array definition. Signifies whether the array is wrapped (for example, <books><book/><book/></books>) or unwrapped (<book/><book/>). Default value is false . The definition takes effect only when defined alongside type being array (outside the items).

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

XML Object Examples

The examples of the XML object definitions are included inside a property definition of a [Schema Object](#) with a sample of the XML representation of it.

No XML Element

Basic string property:

```
1. {
2.   "animals": {
3.     "type": "string"
4.   }
5. }
```

```
1. animals:
2.   type: string
```

```
1. <animals>...</animals>
```

Basic string array property (**wrapped** is **false** by default):

```
1. {
2.   "animals": {
3.     "type": "array",
4.     "items": {
5.       "type": "string"
6.     }
7.   }
8. }
```

```
1. animals:
2.   type: array
3.   items:
4.     type: string
```

```
1. <animals>...</animals>
2. <animals>...</animals>
3. <animals>...</animals>
```

XML Name Replacement

```

4.     "xml": {
5.       "name": "animal"
6.     }
7.   }
8. }

```

```

1. animals:
2.   type: string
3.   xml:
4.     name: animal

```

```

1. <animal>...</animal>

```

XML Attribute, Prefix and Namespace

In this example, a full model definition is shown.

```

1. {
2.   "Person": {
3.     "type": "object",
4.     "properties": {
5.       "id": {
6.         "type": "integer",
7.         "format": "int32",
8.         "xml": {
9.           "attribute": true
10.        }
11.      },
12.      "name": {
13.        "type": "string",
14.        "xml": {
15.          "namespace": "http://swagger.io/schema/sample",
16.          "prefix": "sample"
17.        }
18.      }
19.    }
20.  }
21. }

```

```

1. Person:
2.   type: object
3.   properties:
4.     id:
5.       type: integer
6.       format: int32
7.       xml:
8.         attribute: true
9.     name:
10.      type: string
11.      xml:
12.        namespace: http://swagger.io/schema/sample
13.        prefix: sample

```

```

1. <Person id="123">
2.   <sample:name
3.     xmlns:sample="http://swagger.io/schema/sample">example</sample:name>
4. </Person>

```

XML Arrays

Changing the element names:

```

4.     "items": {
5.         "type": "string",
6.         "xml": {
7.             "name": "animal"
8.         }
9.     }
10. }
11. }

```

```

1. animals:
2.   type: array
3.   items:
4.     type: string
5.     xml:
6.       name: animal

```

```

1. <animal>value</animal>
2. <animal>value</animal>

```

The external **name** property has no effect on the XML:

```

1. {
2.   "animals": {
3.     "type": "array",
4.     "items": {
5.       "type": "string",
6.       "xml": {
7.         "name": "animal"
8.       }
9.     },
10.   "xml": {
11.     "name": "aliens"
12.   }
13. }
14. }

```

```

1. animals:
2.   type: array
3.   items:
4.     type: string
5.     xml:
6.       name: animal
7.   xml:
8.     name: aliens

```

```

1. <animal>value</animal>
2. <animal>value</animal>

```

Even when the array is wrapped, if no name is explicitly defined, the same name will be used both internally and externally:

```

1. {
2.   "animals": {
3.     "type": "array",
4.     "items": {
5.       "type": "string"
6.     },
7.     "xml": {
8.       "wrapped": true
9.     }
10. }
11. }

```



```

4.     type: string
5.   xml:
6.     wrapped: true

```

```

1. <animals>
2.   <animals>value</animals>
3.   <animals>value</animals>
4. </animals>

```

To overcome the above example, the following definition can be used:

```

1. {
2.   "animals": {
3.     "type": "array",
4.     "items": {
5.       "type": "string",
6.       "xml": {
7.         "name": "animal"
8.       }
9.     },
10.    "xml": {
11.      "wrapped": true
12.    }
13.  }
14. }

```

```

1. animals:
2.   type: array
3.   items:
4.     type: string
5.     xml:
6.       name: animal
7.   xml:
8.     wrapped: true

```

```

1. <animals>
2.   <animal>value</animal>
3.   <animal>value</animal>
4. </animals>

```

Affecting both internal and external names:

```

1. {
2.   "animals": {
3.     "type": "array",
4.     "items": {
5.       "type": "string",
6.       "xml": {
7.         "name": "animal"
8.       }
9.     },
10.    "xml": {
11.      "name": "aliens",
12.      "wrapped": true
13.    }
14.  }
15. }

```

```

1. animals:
2.   type: array
3.   items:
4.     type: string
5.     xml:
6.       name: animal
7.   xml:
8.     name: aliens
9.     wrapped: true

```

4. `</aliens>`



If we change the external element but not the internal ones:

```
1. {
2.   "animals": {
3.     "type": "array",
4.     "items": {
5.       "type": "string"
6.     },
7.     "xml": {
8.       "name": "aliens",
9.       "wrapped": true
10.    }
11.  }
12. }
```

```
1. animals:
2.   type: array
3.   items:
4.     type: string
5.   xml:
6.     name: aliens
7.     wrapped: true
```

```
1. <aliens>
2.   <aliens>value</aliens>
3.   <aliens>value</aliens>
4. </aliens>
```

Definitions Object

An object to hold data types that can be consumed and produced by operations. These data types can be primitives, arrays or models.

Patterned Fields

Field Pattern	Type	Description
{name}	Schema Object	A single definition, mapping a "name" to the schema it defines.

Definitions Object Example

```
4.   "properties": {
5.     "id": {
6.       "type": "integer",
7.       "format": "int64"
8.     },
9.     "name": {
10.      "type": "string"
11.    }
12.  },
13.  "Tag": {
14.    "type": "object",
15.    "properties": {
16.      "id": {
17.        "type": "integer",
18.        "format": "int64"
19.      },
20.      "name": {
21.        "type": "string"
22.      }
23.    }
24.  }
25. }
26. }
```

```
1. Category:
2.   type: object
3.   properties:
4.     id:
5.       type: integer
6.       format: int64
7.     name:
8.       type: string
9. Tag:
10.  type: object
11.  properties:
12.    id:
13.      type: integer
14.      format: int64
15.    name:
16.      type: string
```

Parameters Definitions Object

An object to hold parameters to be reused across operations. Parameter definitions can be referenced to the ones defined here.

This does *not* define global operation parameters.

Patterned Fields

Field Pattern	Type	Description
{name}	Parameter Object	A single parameter definition, mapping a "name" to the parameter it defines.

Parameters Definition Object Example

```
4.   "in": "query",
5.   "description": "number of items to skip",
6.   "required": true,
7.   "type": "integer",
8.   "format": "int32"
9. },
10.  "limitParam": {
11.    "name": "limit",
12.    "in": "query",
13.    "description": "max records to return",
14.    "required": true,
15.    "type": "integer",
16.    "format": "int32"
17.  }
18. }
```

```
1. skipParam:
2.   name: skip
3.   in: query
4.   description: number of items to skip
5.   required: true
6.   type: integer
7.   format: int32
8. limitParam:
9.   name: limit
10.  in: query
11.  description: max records to return
12.  required: true
13.  type: integer
14.  format: int32
```

Responses Definitions Object

An object to hold responses to be reused across operations. Response definitions can be referenced to the ones defined here.

This does *not* define global operation responses.

Patterned Fields

Field Pattern	Type	Description
{name}	Response Object	A single response definition, mapping a "name" to the response it defines.

Responses Definitions Object Example

```
1. {
2.   "NotFound": {
3.     "description": "Entity not found."
4.   },
5.   "IllegalInput": {
6.     "description": "Illegal input for operation."
7.   },
8.   "GeneralError": {
9.     "description": "General Error",
10.    "schema": {
11.      "$ref": "#/definitions/GeneralError"
12.    }
13.  }
14. }
```

4. description: **Illegal** input **for** operation.
5. **GeneralError**:
6. description: **General Error**
7. schema:
8. \$ref: '#/definitions/GeneralError'



Security Definitions Object

A declaration of the security schemes available to be used in the specification. This does not enforce the security schemes on the operations and only serves to provide the relevant details for each scheme.

Patterned Fields

Field Pattern	Type	Description
{name}	Security Scheme Object	A single security scheme definition, mapping a "name" to the scheme it defines.

Security Definitions Object Example

```
1. {
2.   "api_key": {
3.     "type": "apiKey",
4.     "name": "api_key",
5.     "in": "header"
6.   },
7.   "petstore_auth": {
8.     "type": "oauth2",
9.     "authorizationUrl": "http://swagger.io/api/oauth/dialog",
10.    "flow": "implicit",
11.    "scopes": {
12.      "write:pets": "modify pets in your account",
13.      "read:pets": "read your pets"
14.    }
15.  }
16. }
```

```
1. api_key:
2.   type: apiKey
3.   name: api_key
4.   in: header
5. petstore_auth:
6.   type: oauth2
7.   authorizationUrl: http://swagger.io/api/oauth/dialog
8.   flow: implicit
9.   scopes:
10.     write:pets: modify pets in your account
11.     read:pets: read your pets
```

Security Scheme Object

Allows the definition of a security scheme that can be used by the operations. Supported schemes are basic authentication, an API key (either as a header or as a query parameter) and OAuth2's common flows (implicit, password, application and access code).

Fixed Fields

Field Name	Type	Validity	Description
type	string	Any	Required. The type of the security scheme. Valid values are " basic ", " apiKey " or " oauth2 ".

description	string	Any	A short description for security scheme.
name	string	apiKey	Required. The name of the header or query parameter to be used.
in	string	apiKey	Required The location of the API key. Valid values are "query" or "header".
flow	string	oauth2	Required. The flow used by the OAuth2 security scheme. Valid values are "implicit", "password", "application" or "accessCode".
authorizationUrl	string	oauth2 ("implicit", "accessCode")	Required. The authorization URL to be used for this flow. This SHOULD be in the form of a URL.
tokenUrl	string	oauth2 ("password", "application", "accessCode")	Required. The token URL to be used for this flow. This SHOULD be in the form of a URL.
scopes	Scopes Object	oauth2	Required. The available scopes for the OAuth2 security scheme.

Patterned Fields

Field Name	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Security Scheme Object Example

Basic Authentication Sample

```
1. {
2.   "type": "basic"
3. }
```

```
1. type: basic
```

API Key Sample

```
1. {
2.   "type": "apiKey",
3.   "name": "api_key",
4.   "in": "header"
5. }
```

```
1. type: apiKey
2. name: api_key
3. in: header
```

Implicit OAuth2 Sample

```
4.   "flow": "implicit",
5.   "scopes": {
6.     "write:pets": "modify pets in your account",
7.     "read:pets": "read your pets"
8.   }
9. }
```

1. type: oauth2
2. authorizationUrl: <http://swagger.io/api/oauth/dialog>
3. flow: **implicit**
4. scopes:
5. write:pets: modify pets **in** your account
6. read:pets: read your pets

Scopes Object

Lists the available scopes for an OAuth2 security scheme.

Patterned Fields

Field Pattern	Type	Description
{name}	string	Maps between a name of a scope to a short description of it (as the value of the property).

Patterned Objects

Field Pattern	Type	Description
^x-	Any	Allows extensions to the Swagger 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. See Vendor Extensions for further details.

Scopes Object Example

```
1. {
2.   "write:pets": "modify pets in your account",
3.   "read:pets": "read your pets"
4. }
```

1. write:pets: modify pets **in** your account
2. read:pets: read your pets


Security Requirement Object

Lists the required security schemes to execute this operation. The object can have multiple security schemes declared in it which are all required (that is, there is a logical AND between the schemes).

The name used for each property MUST correspond to a security scheme declared in the [Security Definitions](#).

Patterned Fields

Field Pattern	Type	Description
---------------	------	-------------

{name}	[string]	Each name must correspond to a security scheme which is declared in the Security Definitions . If the security scheme is of type "oauth2", then the value is a list of scope names required for the execution. For other security scheme types, the array MUST be empty.	
--------	----------	--	---

Security Requirement Object Examples

Non-OAuth2 Security Requirement

```
1. {
2.   "api_key": []
3. }
```

```
1. api_key: []
```

OAuth2 Security Requirement

```
1. {
2.   "petstore_auth": [
3.     "write:pets",
4.     "read:pets"
5.   ]
6. }
```

```
1. petstore_auth:
2.   - write:pets
3.   - read:pets
```

Specification Extensions

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

The extensions properties are always prefixed by "x-" and 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).

Security Filtering

Some objects in the Swagger specification may be declared and remain empty, or completely be removed, even though they are inherently the core of the API documentation.

The reasoning behind it is to allow an additional layer of access control over the documentation itself. While not part of the specification itself, certain libraries may choose to allow access to parts of the documentation based on some form of authentication/authorization.

Two examples for this:

1. The [Paths Object](#) may be empty. It may be counterintuitive, but this may tell the viewer that they got to the right place, but can't access any documentation. They'd still have access to the [Info Object](#) which may contain additional information regarding authentication.
2. The [Path Item Object](#) may be empty. In this case, the viewer will be aware that the path exists, but will not be able to see any of its operations or parameters. This is different than hiding the path itself from the [Paths Object](#) so the user will not be aware of its existence. This allows the documentation provider a finer control over what the viewer can see.



[Sign In](#)

[Try Free](#)



- [Swagger IRC](#)
 - [Swagger Community](#)
 - [Swagger Projects](#)
- [Support](#)
 - [News](#)
 - [Contact Us](#)
- [SwaggerHub Enterprise](#)
 - [SwaggerHub vs OSS](#)
 - [SwaggerHub Integrations](#)
- [Open Source Docs](#)
 - [Swagger Inspector Docs](#)
 - [SwaggerHub Docs](#)



Products

Over 6 million developers, testers and operations teams use SmartBear products to deliver the world's best applications.

[Explore SmartBear Products](#) ➤

Company

- [About SmartBear](#)
- [Careers](#)
- [Newsroom](#)
- [Upcoming Events](#)

Global Headquarters

450 Artisan Way
Somerville, MA 02145

+1 (617) 684-2600

Info@SmartBear.com

Copyright © 2019 SmartBear Software

[Privacy](#) [Terms of Use](#) [Site Map](#)

