Mashery I/O Docs

Configuration Guide

August 2013

Revised: 9/4/2013 10:00 PM

www.mashery.com

Copyright Notice

Trademarks

Mashery, Inc.
717 Market Street, Suite 300
San Francisco, CA 94103

Part Number: IO-Docs-cfg-04-2012

Contents

Chapter 1.
About this Guide

Introduction

This guide describes how to configure Mashery I/O Docs using the Administration Dashboard. Mashery I/O Docs is an interactive API exploration and testing tool that runs on your Developer Portal. It enables your developer partners to perform API calls from within the documentation with their own API keys with described form-based parameter inputs fields and easy-to-read color-coded and formatted payload outputs.

For some APIs, I/O Docs may serve as a full-on replacement for traditional long-form documentation. However, it can also serve as a useful compliment to regular documentation, especially in cases of APIs that have more advanced authentication and request methods.

Assumptions

This guide assumes that you have Administration Dashboard access and proper role privileges to configure content on the Developer Portal.

Chapter Overview

The Mashery I/O Docs Configuration Guide is divided into the following chapters:

- **Chapter 2: Prerequisites and Enablement Overview** - Provides you with a quick overview of the requirements and the process to activate I/O Docs on your Developer Portal via the Administration Dashboard.
- **Chapter 3: Top-Level Configuration Overview** - A high-level view of the global Developer Portal configuration settings and I/O Docs Definitions
- **Chapter 4: I/O Docs Definition JSON Schema** - A deep dive into the description schema that describes your resources, methods and parameters. This schema definition is used to render the interactive docs.

Chapter 2. Prerequisites and Enablement Overview

Prerequisites

I/O Docs is available on all default Mashery deployments, meaning that options to enable and configure I/O Docs are available in the Administrative Dashboard. If for any reason the options to enable and configure I/O Docs are not visible, please contact your Client Service Manager for more information.

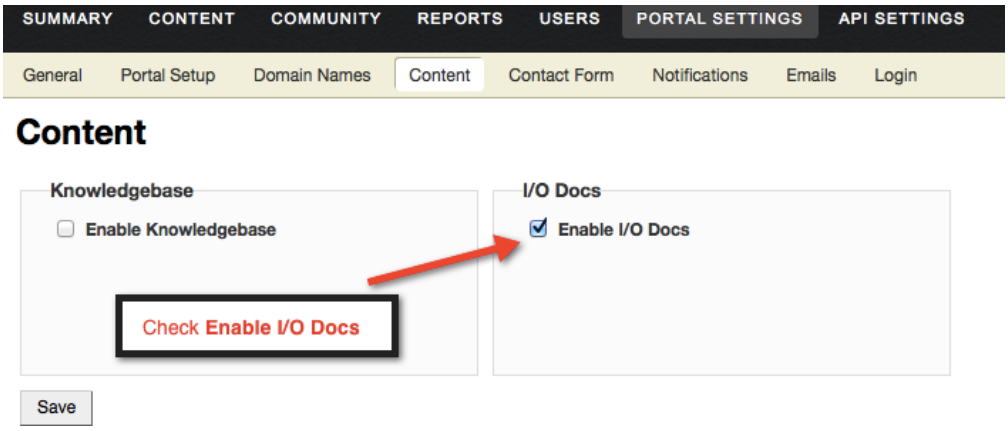
RESTful API Support

Currently, I/O Docs only supports RESTful APIs, using the GET, POST, DELETE and PUT methods. SOAP APIs are not supported because the resource, methods and parameters do not construct a SOAP request.

Enablement Overview

To enable I/O Docs:

1. Access to the Mashery Admin Dashboard.
2. Enable the I/O Docs Content Module by navigating to **Portal Settings > Content** and checking the box adjacent to **Enable I/O Docs:**



3. Click **Save**.

   After completing these steps, I/O Docs is enabled. The URL for I/O Docs on your portal is: **http://*devhost.devdomain.com*/io-docs**, for example, if your portal URL is **http://developer.acme.com**, then the URL for I/O Docs would be **http://developer.acme.com/io-docs**. To configure I/O Docs properly for your APIs, please proceed to Chapter 3.

Chapter 3. Top-Level Configuration Overview

I/O Docs Definitions Overview

Definitions for I/O Docs are used to generate the rendered I/O Docs interface, including the groups of resources, methods and parameters. Definitions are associated with APIs configured in the API Settings section of the Administration Dashboard. An API is allowed only one I/O Docs definition.

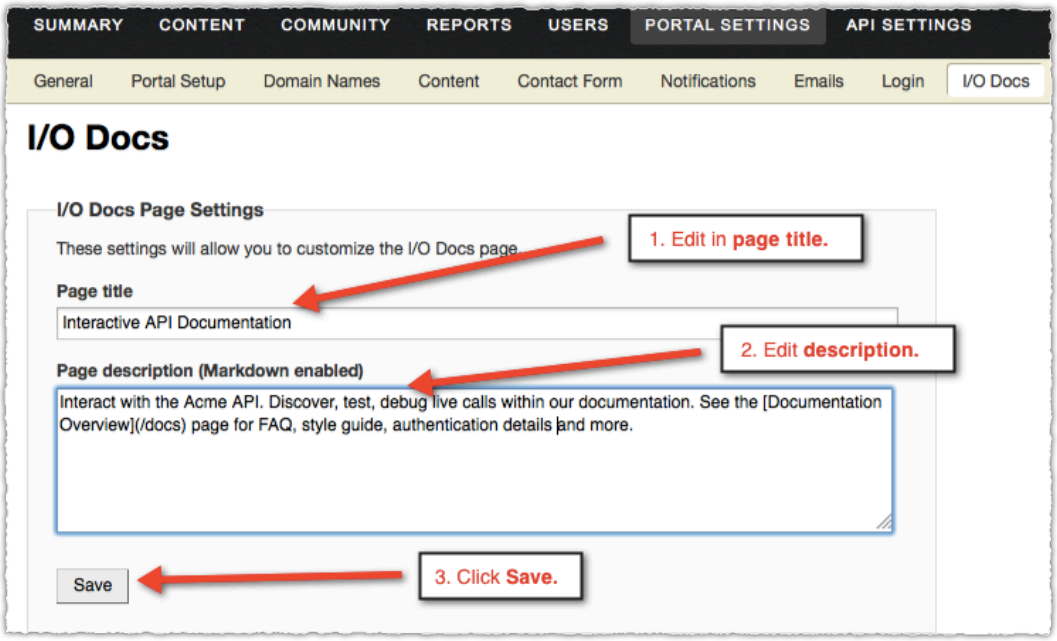
Note that although an I/O Docs definition is associated with an API, it is functionally separate. This means that you can freely configure your I/O Docs definition to highlight only certain resources, methods or parameters of your choice.

The definitions are simple JSON objects modeled after the Google Discovery Document format (GDDF). The schema format has been extended to address authentication and signature methods. More detailed information about the schema can be found in Chapter 4.

Global I/O Docs Page Settings

You can provide a page title header and description that appears at the top of every rendered I/O Docs page. These attributes can be set by following these steps:

1. Access the Mashery Admin Dashboard
2. Add page title and description by navigating to the **Portal Settings > I/O Docs** section, and editing these fields in the I/O Docs Page Settings. Click **Save**.



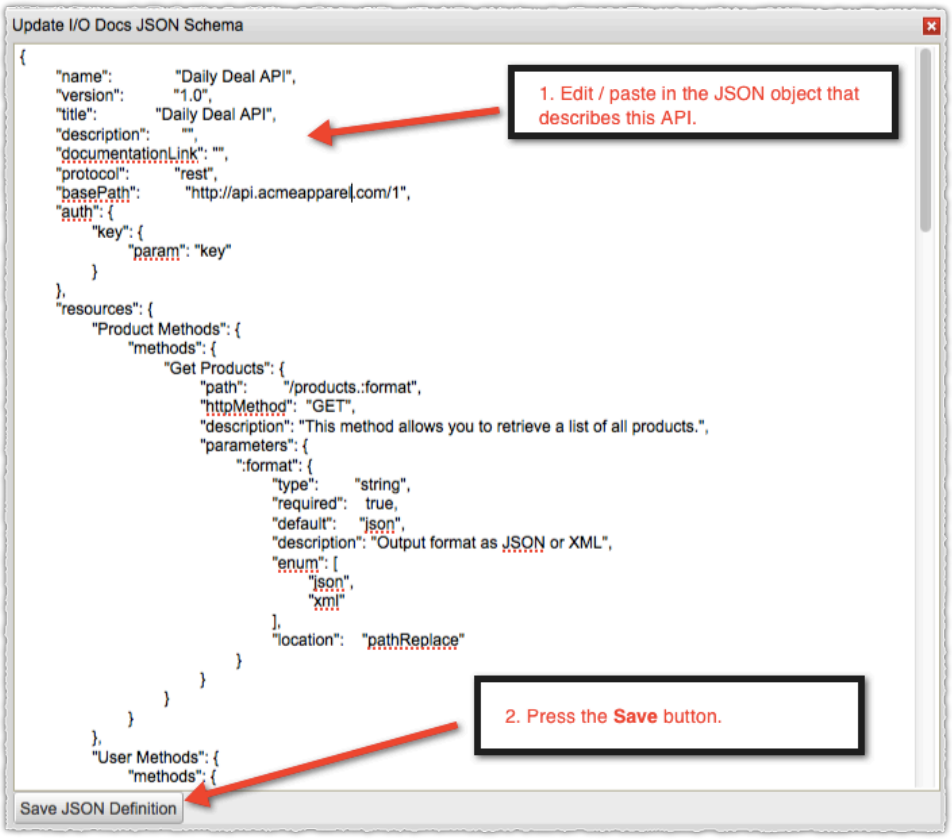
Adding I/O Docs Definitions

If you have enabled I/O Docs by following directions in Chapter 2, the next step is to add a definition to one or more APIs.

To add I/O Docs definitions:

1. Access the Mashery Admin Dashboard
2. Add a definition to an API by navigating to the **Portal Settings > I/O Docs** and clicking **Add Definition** adjacent to the API Name that you wish to configure:

3. Provide the JSON schema that describes the entire API's set of resources, methods, and parameters that you wish to expose via I/O Docs. The JSON format details can be found in Chapter 4. Click **Save JSON Definition**:



Markdown support in I/O Docs

I/O docs (PHP version) supports Markdown syntax to be used within the description properties at the API, method, and parameter level. There are however some limitations with using markdown in I/O Docs JSON based definitions.

I/O Docs Markdown Known Limitations and Issues

Due to the nature of the I/O docs view implementation, the API description supports a subset of the Markdown syntax. Specifically, links and tags that utilize quotes are not supported at the API description level and only supported at the resource level, within the method and parameter descriptions level. For example:

```
"description": "This is [an example](http://example.com) inline link."
```

Other limitations

- Links, Images, and Tables are not supported at the API description level (at the top of the page).

- Heading level 2 (H2) renders as a drop-down list when used at the API description level (see image below)
- Bulleted lists render as indented items without bullets.
- Automatic links are not supported; this syntax will not render a link in I/O docs: <http://example.com/>

Markdown uses blank lines to indicate that a "new line" is desired in the resulting HTML. A blank line is any line that looks like a blank line — a line containing nothing but spaces or tabs is considered blank. To represent this Markdown syntax correctly within the I/O docs JSON definition requires adding a "\n" before the markdown in certain cases. For example, the following code will show something as Heading level 1 (H1):

`"\n# Level 3 Header"` will show as Level 3 Header in I/O Docs

Chapter 4. I/O Docs Definition Schema Details

JSON Schema Overview

The schema is a JSON object containing top-level API properties along with resources, methods and parameters that describe how API requests are formed and transmitted. The Mashery I/O Docs Definition Schema is based in part on the Google Discovery Document format. Some properties within the GDDF are disregarded by the I/O Docs schema processor and will not be referenced below or in the examples. The GDDF has been extended to address authentication and signature methods.

JSON Object Property Details

```
{
  "name": "value",
  "version": "value",
  "title": "value",
  "description": "value",
  "protocol": "rest",
  "basePath": "value",
  "auth": {
    "key": {
      "param": "value",
      "location": "value",
      "secret": {
        "param": "value",
        "type": "value"
      }
    },

    "basicAuth": "value",

    "oauth": {
      "version": "value",
      "base_uri": "value",
      "authorize_uri": "value",
      "access_token_uri": "value",
      "auth_flows": [
         "value"
      ],
      "options": { }
    }
```

```
  },
  "resources": {
    "value": {
      "methods": {
        "value": {
          "path": "value",
          "httpMethod": "value",
          "description": "value",
          "parameters": {
            "value": {
              "description": "value",
              "default": "value",
              "required": "value",
              "enum": [
                "value"
              ],
              "enumDescriptions": [
                "value"
              ],
              "location": "value"
            }
          }
        }
      }
    }
  }
}
```

Property Name	Value	Description
name	string	The name of the API. This is not shown/displayed anywhere in the rendered page.
version	string	The version of the API, e.g. "v1.0".
title	string	The title of the API that is displayed in the API selection drop-down menu, e.g. "Daily Deals API".
description	string	The description of the API. This is displayed directly under the API selection drop-down menu when the corresponding API is selected, e.g. "The Daily Deals API provides a real-time look into the current daily deals based on ZIP code and radius".
protocol	string	The protocol described by this document, e.g. "rest".

Property Name	Value	Description
basePath	string	The base URL path for REST requests, including scheme name. Port number is optional. e.g. "https://api.acme.mashery.com:443"
auth	object	Links to key and/or OAuth objects.
auth.key	object	Links to param and secret objects.
auth.key.location	string	Location of API key. Default location is the query string for GET requests, and encoded parameter request body for POST requests. Also determines location of signature key/value pair. To override default location, valid values are "query", "pathReplace", "body" and "header". Additionally, you can supply one or more of the above values separated by a comma. (e.g. "query,header" would place the API key parameter in both the query string and as a request header value pair).
auth.key.param	string	Name of the API key parameter, e.g. "api_key" or "key"
auth.key.secret	object	Links to the param and type objects.
auth.key.secret.param	string	Name of the API key secret parameter, e.g. "secret"
auth.key.secret.type	string	Type of secret signature method, e.g. "signed_md5" or "signed_sha256"
auth.basicAuth	boolean	Whether the basic authentication is required, either "true" or "false".
auth.oauth	object	OAuth version and flow definition.
auth.oauth.version	string	OAuth version (e.g. “2.0”).
auth.oauth.base_uri	string	The base URI for the OAuth service endpoints. This value prepends the *authorize_uri* and *access_token_uri* values below, e.g. "http://yourdomain.com" (optional)
auth.oauth.authorize_uri	string	The endpoint where the end-user is sent to authorize their account and grant permissions, e.g. "/oauth/authorize". If *base_uri* is not defined, a fully qualified

Property Name	Value	Description
		URI is required.
auth.oauth.access_token_uri	string	The endpoint where the access token is granted, e.g. "/oauth/access_token". If *base_uri* is not defined, a fully qualified URI is required.
auth.oauth.access_token_location	string	The location of the access token. Eg setting this value to “header” will pass the OAuth token in the request header. Default, if not specified, is query. This value can also be set to “body”.
auth.oauth.auth_flows	array	Array of strings, each string containing one of the four supported flows, e.g. "auth_code", "client_cred", "password_cred", or "implicit"
auth.oauth.options	object	Links to OAuth options objects, e.g. "authorize" (often used to hold *authorize* and *scope* details – see examples #3 and #4 below). (optional)
resources	object	The resources in the API.
resources.*value*	string	An individual resources description or name. Contains methods related to this resource.
resources.(value).methods	object	Methods on this resource.
resources.(value).methods.*value*	string	Description for any methods on this resource. The value at this level (string) contains name of this method.
resources.(value).methods.(value).path	string	The URI path of this REST method. Should be used in conjunction with the basePath property at the API-level.
resources.(value).methods.(value).httpMethod	string	HTTP method used by this method, e.g. "GET", “POST”, “PUT”, “DELETE”.
resources.(value).methods.(value).description	string	Description of this method.
resources.(value).methods.(value).parameters	object	Details for all parameters in this method.
resources.(value).methods.(value).parameters.*value*	string	Details for a single parameter in this method (object). The value assigned at this

Property Name	Value	Description
		level (string) would be the name of the parameter.
resources.(value).methods.(value).parameters.(value).description	string	A description of this object.
resources.(value).methods.(value).parameters.(value).default	string	The default value of this property (if one exists).
resources.(value).methods.(value).parameters.(value).required	boolean	Whether the parameter is required, either "true" or "false".
resources.(value).methods.(value).parameters.(value).enum	list	Values this parameter may take.
resources.(value).methods.(value).parameters.(value).enumDescriptions	list	The descriptions for the enums. Each position maps to the corresponding value in the "enum" array.
resources.(value).methods.(value).parameters.(value).location	string	Whether this parameter goes in the query, path (for REST requests) or header. Valid values are "query", "pathReplace", "body" and "header".

Example #1 – Key Only Auth

This is an example of an I/O Docs Definition for an API that uses key-based authorization, where the key is passed in a query string parameter named "api_key".

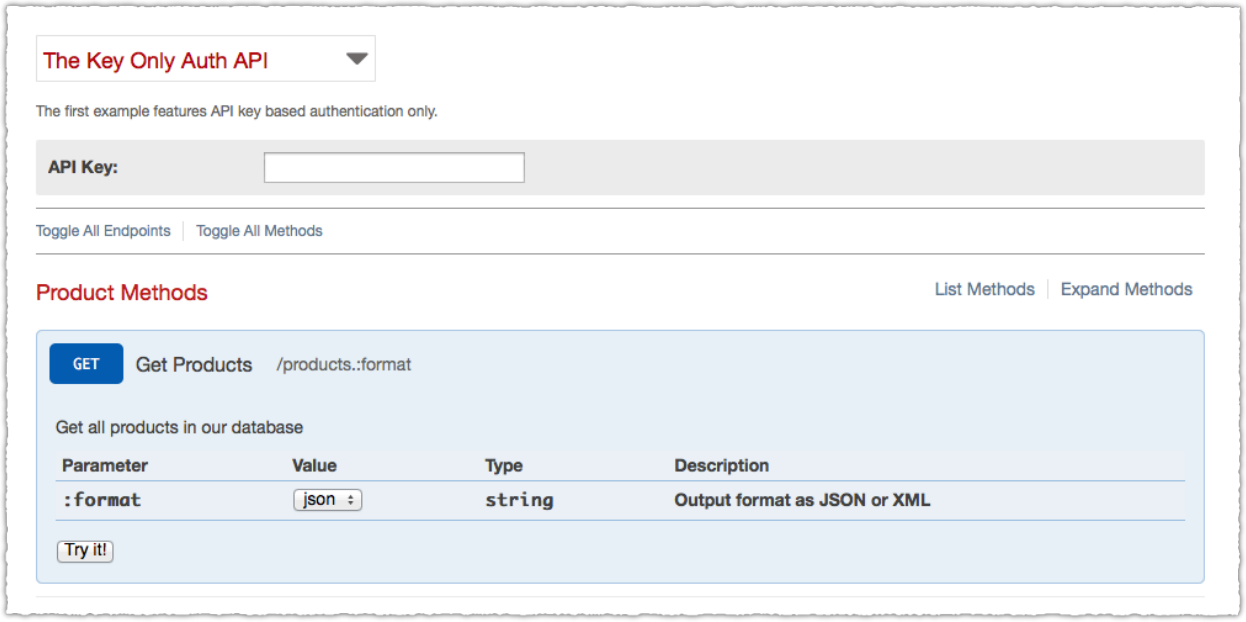
Example #1 - I/O Docs Definition

```
{
    "name": "Example #1 API",
    "version": "1.0",
    "title": "The Key Only Auth API",
    "description": "The first example features API key based authentication only.",
    "protocol": "rest",
    "basePath": " http://api.example1.com/v1",
    "auth": {
        "key": {
            "param": "api_key",
            "location": "query"

        }
    },
    "resources": {
        "Product Methods": {
```

```
            "methods": {
                "Get Products": {
                    "path": "/products.:format",
                    "httpMethod": "GET",
                    "description": "Get all products in our database",
                    "parameters": {
                        ":format": {
                            "type": "string",
                            "required": true,
                            "default": "json",
                            "description": "Output format as JSON or XML",
                            "enum": [
                                "json",
                                "xml"
                            ],
                            "location": "pathReplace"
                        }
                    }
                }
            }
        }
    }
}
```

Example #1 - I/O Docs Rendered



Example #2 – Key, Secret and Signature Auth

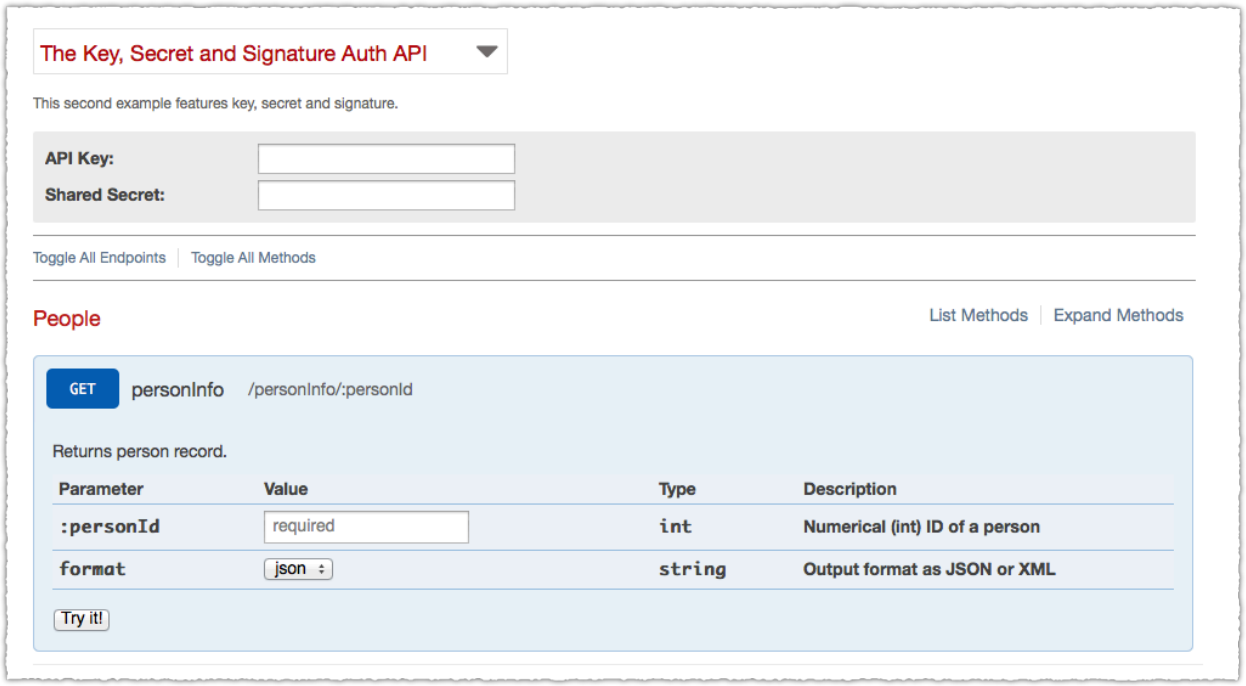
This is an example of an I/O Docs definition for an API that uses key-based authorization with an MD5 hash signature. The string that is hashed is a concatenation of the following values: API key, secret, epoch (POSIX time).

Example #2 - I/O Docs Definition

```
{
    "name": "Example #2 API",
    "version": "1.0",
    "title": "The Key, Secret and Signature Auth API",
    "description": "This second example features key, secret and signature.",
    "protocol": "rest",
    "basePath": "http://api.example2.com/v2",
    "auth": {
        "key": {
            "param": "api_key",
            "location": "query"
            "secret": {
               "param": "sig",
               "type": "signed_md5"
            }
      }
},
    "resources": {
        "People": {
            "methods": {
                "personInfo": {
                    "path": "/personInfo/:personId",
                    "httpMethod": "GET",
                    "description": "Returns person record. ",
                    "parameters": {
                        ":personId": {
                            "type": "int",
                            "required": true,
                            "default": "",
                            "description": "Numerical (int) ID of a person",
                            "location": "pathReplace"
                        },
                        "format": {
                            "type": "string",
                            "required": true,
                            "default": "json",
                            "description": "Output format as JSON or XML",
                            "enum": [
                                "json",
                                "xml"
                            ],
                            "location": "query"
                        }
                    }
                }
            }
        }
```

```
    }
}
```

Example #2 - I/O Docs Rendered



Example #3 – OAuth 2.0 API

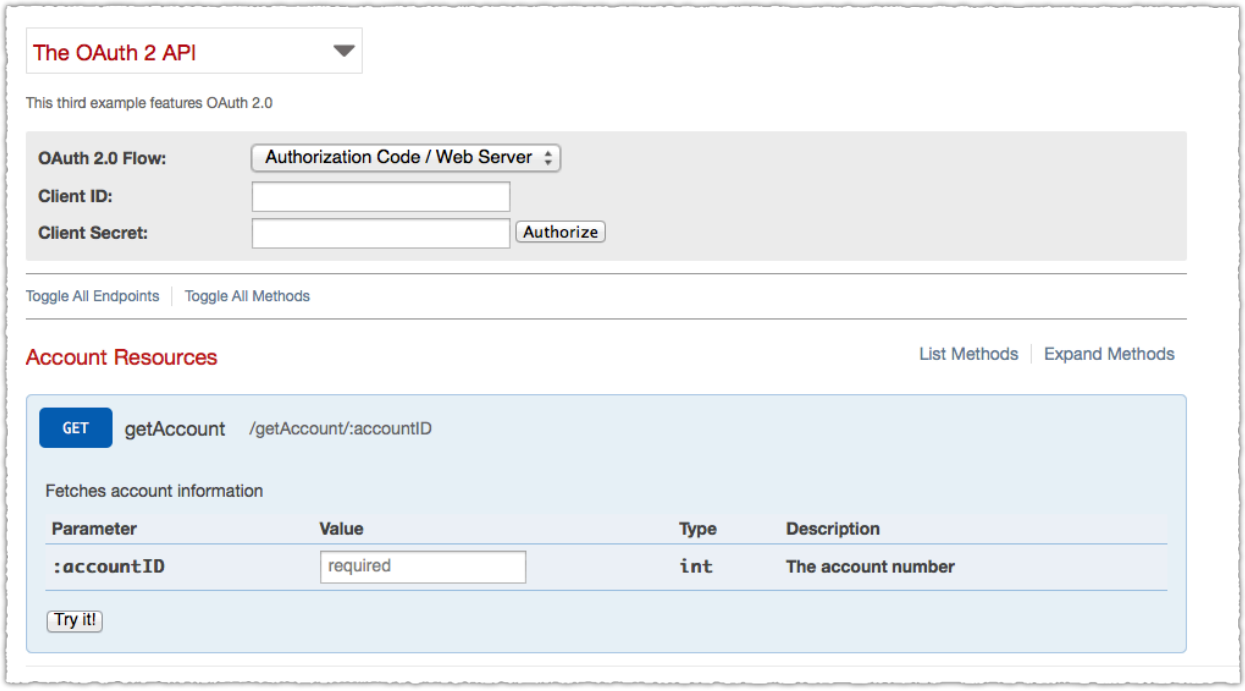
This is an example of an I/O Docs Definition for an API that uses OAuth 2.0 and the authentication code flow. Additional options for scope are also included in this configuration. OAuth 1.0 and 1.0a are not supported by I/O Docs.

Example #3 - I/O Docs Definition

```
{
    "name": "Example #3 API",
    "version": "1.0",
    "title": "The OAuth 2 API",
    "description": "This third example features OAuth 2.0",
    "protocol": "rest",
    "basePath": " http://api.example3.com/v3",
    "auth": {
        "oauth": {
            "version": "2.0",
            "auth_flows": ["auth_code"],
            "base_uri": "http://api.example3.com",
            "authorize_uri": "/oauth/authorize",
```

```
            "access_token_uri": "/oauth/token",
            "access_token_location": "header",
            "options": {
                "authorize": {
                    "scope": [
                       "read",
                       "write",
                       "execute"
                    ]
                }
            }
        }
    },
    "resources": {
        "Account Resources": {
            "methods": {
                "getAccount": {
                    "path": "/getAccount/:accountID",
                    "httpMethod": "GET",
                    "description": "Fetches account information",
                    "parameters": {
                        ":accountID": {
                            "type": "int",
                            "required": true,
                            "default": "",
                            "description": "The account number",
                            "location": "pathReplace"
                        }
                    }
                }
            }
        }
    }
}
```

Example #3 - I/O Docs Rendered



Example #4 – Google OAuth 2.0 API

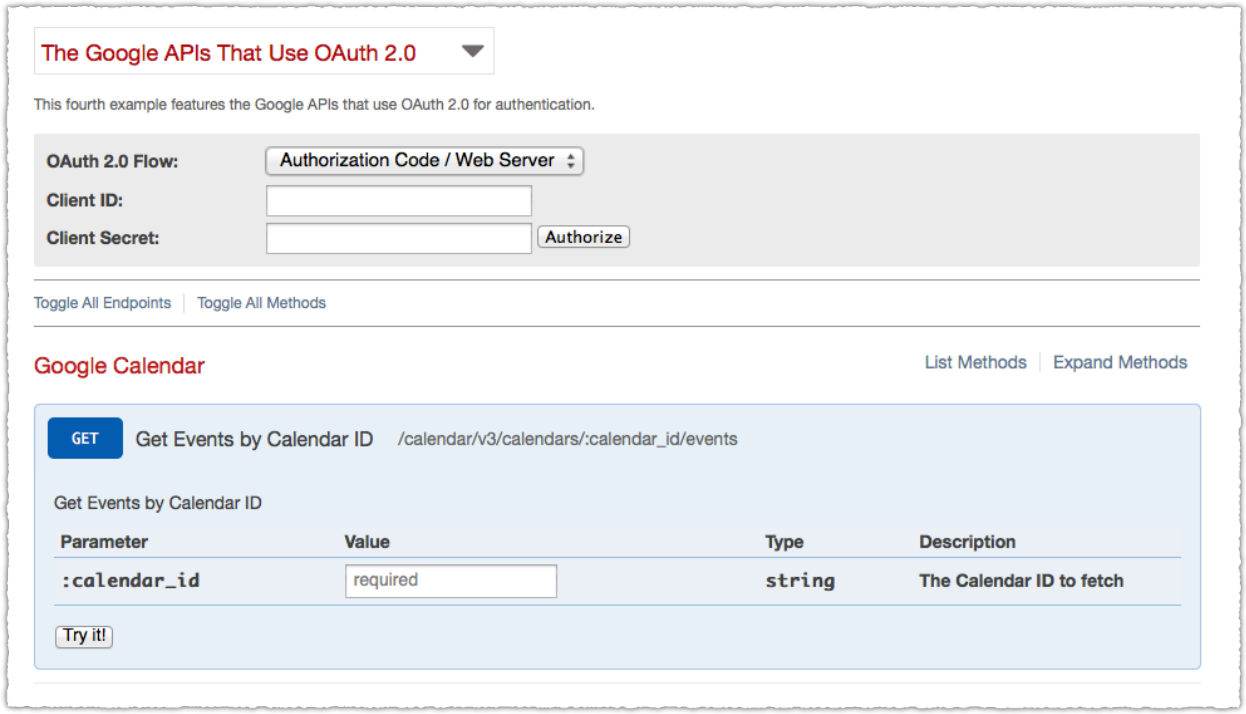
This is an example of an I/O Docs definition for the Google APIs that use OAuth 2.0.

Example #4 - I/O Docs Definition

```
{
    "name": "Google APIs",
    "version": "1.0",
    "title": "The Google APIs That Use OAuth 2.0",
    "description": "This fourth example features the Google APIs that use OAuth 2.0
for authentication.",
    "protocol": "rest",
    "basePath": "https://www.googleapis.com",
    "auth": {
        "oauth": {
            "version": "2.0",
            "base_uri": "https://accounts.google.com",
            "auth_flows": [
                "auth_code",
                "implicit"
            ],
            "authorize_uri": "/o/oauth2/auth",
            "access_token_uri": "/o/oauth2/token",
```

```
            "options": {
                "authorize": {
                    "scope": ["https://www.googleapis.com/auth/calendar"],
                    "access_type": "online",
                    "approval_prompt": "force"
                },
                "auto_exchange_auth_code": true
            }
        }
    },
    "resources": {
        "Google Calendar": {
            "methods": {
                "Get Events by Calendar ID": {
                    "description": "Get Events by Calendar ID",
                    "httpMethod": "GET",
                    "path": "/calendar/v3/calendars/:calendar_id/events",
                    "parameters": {
                        ":calendar_id": {
                            "required": "true",
                            "default": "",
                            "type": "string",
                            "description": "The Calendar ID to fetch",
                            "location": "pathReplace"
                        }
                    }
                }
            }
        }
    }
}
```

Example #4 - I/O Docs Rendered



Example #5 – Post Body

This is an example of an I/O Docs Definition for an API that uses key-based authorization, where the key is passed in a query string parameter named "api_key".

Example #5 - I/O Docs Definition

```
{
   "name":"acme daily deal api definition",
   "version":"1.0",
   "title":"Acme Daily Deal API",
   "description":"This Daily Deal API is used to find the most current daily
          deals.",
   "protocol":"rest",
   "basePath":"http://api.acmeapparelstore.com/",
   "auth":{
      "key":{
         "param":"apikey",
         "location": "query"
      }
   },
   "resources":{
           "Review Daily Deal":{
            "methods": {
                          "PostDailyDealReview": {
                        "path":"deal",
                        "httpMethod":"POST",
                        "description":"Submit a review on the the most current
```

```
          Daily Deal.",
                         "parameters":{
                                        "reviewBody": {
                                   "description": "Some well-formed JSON",
                                   "default": "{ \"some\": \"well-formed\", \"json\":
          \"string\" }",
                                   "type": "textarea",
                                   "required": true,
                                   "location": "body"
                              },
                              "Content-Type":{
                                   "type":"string",
                                   "required":"true",
                                   "description":"Content type of the payload",
                                   "default":"application/json",
                                   "location": "header"
                              }
                                     }
                        }
                          }
             }
        }
}
```

Example #5 - I/O Docs Rendered