Table of Contents

api发布者手册

工作流程

你只需负责api的核心业务部分, 由本系统替你完成:

- 确认哪些应用可以调用你的api
- 限制应用对api的调用频率
- 获取当前是哪位用户通过哪个应用访问你的api信息
- 获取api的调用统计数据
- 为你的API调用者提供一套一致友好的测试工具和文档, 让你免去反复讲诉之苦

你需要做的是:

1. 编写api描述文件
2. 部署api描述文件

此外, 为了更好的用户体验. 你还可以:

- 让服务可被自动探测
- 让后端的签名密钥可被自动配置.

服务的自发现: /api.txt

放置一个api.txt 在你的应用根目录, 例如:

`http://www.example.com/api.txt`

内容:

```
#apilist
./api01.json   //相对api.txt的地址
api02.json
api03.json
http://www.example.com/path/to/api02.json  //绝对url
/paht/to/api04.json //绝对地址
http://www.example.com/api.txt   //另一个api.txt
...
```

密钥的传递流程

附录: 保留参数

后端Api不能使用下列参数, 前端的API调用以下参数会被过滤掉.

- apiver
- client_id
- client_secret
- sign
- sign_method
- sign_time

api-json文件

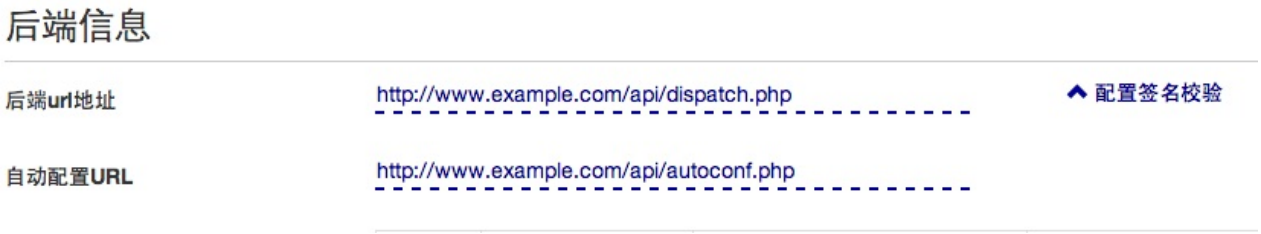
放到一个可供http访问的位置即可. 比如: http://www.example.com/apidb/orders.json

分段讲解

基本信息


```
{
    "summary": "分销系统",
    "prefix": "b2b",
    "mode": "path", //path: rest风格, param: 根据某个参数进行调度
    "param_name": "method", //mode=param时,  调度的参数名
    ...
}
```

后端信息




```
{
    ...
    "url": "http://www.example.com/api/dispatch.php",
    "auto_config_url": "http://www.example.com/api/autoconf.php",
    ...
}
```

后端配置

```
    {
        ...
        "config_values": [
            { "name": "token", "description": "token", "defaultvalue": "", "issecret": fa
lse },
            ...
        ],
        ...
    }
```

后端全局参数


```
    {
        ...
        "global_params": [
            { "name": "api_version", "param_type": "request", "value_type": "expr", "form
at": "3.2" },
            { "name": "return_data", "param_type": "request", "value_type": "expr", "form
at": "json" },
            { "name": "ac", "param_type": "request", "value_type": "sign", "format": "sho
pex" }
        ]
        ...
    }
```

api定义

api基本信息

```
{
    ...
    "apis": {
        "search_order_lists": {
            "path": "",
            "method": [ "post" ],
            "summary": "获取订单列表",
            "notes": "",
            "limit_seconds": 0,
            "limtt_count": 0,
            ...
            }
       }
    ...
}
```

api参数

必填	名称	值类型	说明	-
☐	start_time	date	2013-06-01	⊗
☐	end_time	date	2013-08-20	⊗
☐	status	json	{"status":"active","ship	⊗

添加参数

```
{
    ...
    "apis": {
        "search_order_lists": {
            ...
            "params": [
                { "name": "start_time", "desc": "2013-06-01",
                    "required": true, "type": "date", "param_type": "param" },
                { "name": "end_time", "desc": "2013-06-30",
                    "required": false, "type": "date", "param_type": "param" },
                { "name": "status", "desc": "订单状态",
                    "required": false, "type": "date", "param_type": "param" }
            ],
            ...
            }
       }
    ...
}
```

api返回结果

异常返回

http状态码	错误码	说明	-
200	0		

添加异常返回

```
{
    ...
    "apis": {
        "response": "response data format"
    ...
}
```

完整例子:

```
{
    "url": "http://www.example.com/api/dispatch.php",
    "prefix": "b2b",
    "summary": "分销系统",
    "apis": {
        "search_order_lists": {
            "path": "",
            "method": [ "POST" ],
            "summary": "获取订单列表",
            "notes": "",
            "limit_seconds": 0,
            "limtt_count": 0,
            "params": [
                { "name": "start_time", "desc": "2013-06-01",
                    "required": true, "type": "date", "param_type": "param" },
                { "name": "end_time", "desc": "2013-06-30",
                    "required": false, "type": "date", "param_type": "param" },
                { "name": "status", "desc": "订单状态",
                    "required": false, "type": "date", "param_type": "param" }
            ],
            "response": "",
            "exception": [
                {
                    "httpcode": 200,
                    "code": 0,
                    "message": ""
                }
            ]
        }
    },
    "mode": "path",
    "param_name": "act",
    "models": {},
    "auto_config_url": "http://www.example.com/api/autoconf.php",
    "config_values": [
        { "Name": "token", "Description": "token", "DefaultValue": "", "IsSecret": false
}
    ],
    "global_params": [
        { "name": "api_version", "paramtype": "REQUEST", "valuetype": "expr", "format": "
3.2" },
        { "name": "return_data", "paramtype": "REQUEST", "valuetype": "expr", "format": "
json" },
        { "name": "ac", "paramtype": "REQUEST", "valuetype": "sign", "format": "shopex" }
    ]
}
```

使用**php sdk**编写服务端

使用php的sdk可以大大简化api json的编写, 代码如下:

输出**api json**

```
//定义api
$provider = new prism_provider("/real/api/router");

$provider->add("get_id_by_domain",
        prism_api("id",
            prism_params("arg1", "useinput"),
            prism_params("arg2", "useinput"),
            prism_params("arg3", "useinput"),
            prism_params("arg4", "useinput")
        )
    );

输出api的json格式
$provider->output_json();
```

如此即可输出api的配置json. 适用于已有许多接口, 仅希望用JSON输出结构.

```
{
   "url":"\/prism_sdk\/examples\/server.php",
   "interface":"",
   "sandbox_url":"",
   "resource_content_types":"text\/json",
   "prefix":"",
   "summary":"",
   "apis":{
      "get_id_by_domain":{
         "path":"",
         "method":[
            "POST"
         ],
         "summary":"",
         "notes":"",
         "require_oauth":"",
         "backend_timeout_second":"",
         "params":[
            {
               "name":"arg1",
               "desc":"useinput",
               "required":false,
               "type":"string",
               "param_type":"request"
            },
            {
               "name":"arg2",
               "desc":"useinput",
```

```
                "required":false,
                "type":"string",
                "param_type":"request"
            },
            {
                "name":"arg3",
                "desc":"useinput",
                "required":false,
                "type":"string",
                "param_type":"request"
            },
            {
                "name":"arg4",
                "desc":"useinput",
                "required":false,
                "type":"string",
                "param_type":"request"
            }
        ],
        "response":"",
        "exception":[

        ]
    }
},
"mode":"params",
"param_name":"method",
"models":[

],
"auto_config_url":"",
"config_values":[

],
"global_params":[

]
}
```

处理**api**调用

如果在定义api的时候定义相应的handle方法, 则可以进行api的调度以及签名, 错误处理等操作.

```
class my_api_handler{

    public function get_id_by_domain($params){
        return $_SERVER;
    }

}

$provider->handler(new my_api_handler);

if(array_key_exists('show_api_json', $_GET)){
    $provider->output_json();
}elseif(array_key_exists("method", $_REQUEST)){
    $provider->dispatch($_REQUEST["method"]);
}
```

设置签名验证方式

```
$provider->set_validation("prism_sign_validation", "get_secret_by_key");
```

`set_validation(prism_validation, construct args...)`

其中, 用于验证的类需要符合接口 `prism_validation`

```
interface prism_validation{
    function get_config();
    function get_global_params();
    function validate();
}
```

`prism_sign_validation` 是sdk自带. 采用key与secret的组合进行签名, 需要设置一个回调函数用来验证key与secret的正确性.