ALIBABA CLOUD

阿里云

控制台

后端服务开发文档

文档版本：20201111

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.工作台托管-技术接入指南

对接流程

架构图

调用链路

说明

ISV无法拿到用户的ak，Aliyun会针对用户的请求颁发accessToken，ISV要调用OPEN API，得携带accessToken调用Aliyun-Api-Service。

前端改造

基础框架

目前没有对使用何种 JS 框架做约束，支持 React / VUE / Angular 三大框架。

但是需要对入口文件初始化代码逻辑稍加改造，从而能接入阿里云的微前端容器。具体如下：

Angular

```
// 引入一个 portal 包
import { boostrap } from '@alicloud/console-os-ng-portal';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { NgZone } from '@angular/core';
import { Router } from '@angular/router';
import { AppModule } from './app/app.module';

// 修改一下入口写法并 export
export default boostrap({
 bootstrapFunction: props => {
  return platformBrowserDynamic().bootstrapModule(AppModule);
 },
 template: '<app-root />',
 Router,
 NgZone: NgZone,
});
```

React

```
// 引入一个 portal 包
import { mount }  from '@alicloud/console-os-react-portal';
import React from 'react';
import App from './app';

// 修改一下入口写法并 export
export default mount(
 (props) => <App {...props} />,
 document.getElementById('app')
);
```

VUE

```
// 引入一个 portal 包
import { mount } from '@alicloud/console-os-vue-portal';
import App from './App.vue';

// 修改一下入口写法并 export
export default mount({
 el: '#app',
 i18n,
 router,
 store,
 render: h => h(App),
 created() {}
});
```

组件库

截止目前，没有要求强制使用阿里云提供的组件库 SDK。

但是在应用的 UI 设计上，需要阿里云的设计师评估。风格差异太大的话则需要修改。

请求调用

由于后端服务会部署在阿里云这边，所以请求的地址和消息体需要修改：

```
General
- Request URL: https://work.console.aliyun.com/data/plugin.json
- Request Method: POST

Form Data
- product: ${isvCode}-${appCode}
- action: ${actionName}
- params: ${bizParamsMap}
- sec_token: ${window.ALIYUN_CONSOLE_CONFIG.SEC_TOKEN}
```

返回的数据是：

```
{
 "requestId": String,
 "code": String,
 "successResponse": Boolean,
 "data": Object
}
```

工程化

根目录下增加一个 `abc.json` 文件。此文件用来给阿里云内部构建使用，文件名和路径不可自定：

```
{
 "assets": {
  "type": "builder",
  "builder": {
   "name": "@ali/builder-breezr"
  }
 }
}
```

并且增加一个 `build-utils.js` 文件。此文件提供一个实用函数，用来给构建逻辑调用，文件名和路径可以自定：

```
const path = require('path');

// ISV 自己托管的资源根目录地址，比如 https://somewhere.oss-cn-hangzhou.aliyuncs.com/dest
const ISV_CDN_PATH = '/';
const CONSOLE_OS_DEV_CDN_PATH = 'http://localhost:4200';

const isCloudBuild = () => {
```

```
 return process.env.BUILD_ENV === 'cloud';
};

// 云构建时涉及到的一些逻辑
const getPublicPathFromArg = () => {
 const argvString = process.env.BUILD_ARGV;
 const outputPublicPath = argvString.split('=')[1];
 let publicPath = outputPublicPath.substr(0, outputPublicPath.length - 3);
 if (!publicPath.endsWith('/')) {
  publicPath += '/';
 }
 return publicPath;
};

// 得到最终的资源线上根目录地址
const getPublicPath = () => {
 return process.env.CONSOLE_OS_DEV_LOCAL === 'true' ?
  CONSOLE_OS_DEV_CDN_PATH : (
   isCloudBuild() ?
    getPublicPathFromArg() :
    ISV_CDN_PATH
  );
};

// 构建的目标目录
const getOutputDir = () => {
 return isCloudBuild() ?
  path.resolve(process.cwd(), process.env.BUILD_DEST_DIR || process.env.BUILD_DEST) :
  path.resolve(__dirname, 'dist');
};

module.exports = {
 getPublicPath,
 getOutputDir
};
```

最后，改造构建逻辑。根据不同框架下的项目修改对应的构建配置文件。

Angular 工程

修改 `webpack.config.js` ，作如下修改：

```
const osAngularWebpack = require('@alicloud/console-os-ng-builder');
const { getPublicPath, getOutputDir } = require('./build-utils');

module.exports = osAngularWebpack({
 output: {
  // 构建输出目录
  path: getOutputDir(),
  // 部署到的根目录地址，使用 build-utils.js 提供的实用函数
  publicPath: getPublicPath()
 }
}, {
 // 新增应用 ID 作为钩子
 id: `${isvCode}-${appCode}`
});
```

直接使用 Webpack 构建的工程（React / Non-vue-cli 等）

修改 `webpack.config.js`，作如下修改：

```
const Chain = require('webpack-chain');
const merge = require('webpack-merge');
const { chainOsWebpack } = require('@alicloud/console-toolkit-plugin-os');
const { getPublicPath, getOutputDir } = require('./build-utils');

const chain = new Chain();
chainOsWebpack({
 // 新增应用 ID 作为钩子
 id: `${isvCode}-${appCode}`
})(chain);

module.exports = merge({
 output: {
  // 构建输出目录
  path: getOutputDir(),
  // 部署到的根目录地址，使用 build-utils.js 提供的实用函数
  publicPath: getPublicPath()
 }
}, chain.toConfig());
```

使用了 vue-cli 的工程

修改 `vue.config.js` ，增加几个 option 如下：

```
const { getPublicPath, getOutputDir } = require('./build-utils');
const path = require('path');

module.exports = {
 // 构建输出目录
 outputDir: getOutputDir(),
 // 部署到的根目录地址，使用 build-utils.js 提供的实用函数
 publicPath: getPublicPath(),
 // 插件配置
 pluginOptions: {
  consoleOs: {
   // 新增应用 ID 作为钩子
   id: `${isvCode}-${appCode}`
  }
 }
};
```

并且安装 devDeps：

```
npm install vue-cli-plugin-console-os -D
```

部署

ISV 将前端代码仓库托管到 云效，并且增加 console-fe 这个用户作为仓库成员。阿里云这边会执行取代码、运行构建脚本、部署到 CDN 整套逻辑。

最终上线的应用地址是： `https://work.console.aliyun.com/${isvCode}-${appCode}`

前端资源文件会部署到 `https://cb.alibabausercontent.com` 这个 CDN 下，而不是应用地址下。

后端服务

后端服务部署在ISV账号申请的vpc里面，此账号为阿里云持有管控。不提供登录，会提供日志查询权限。

ISV的服务要以约定的形式发布，一个action代表一个服务。

- url :http://${isv_ip}/api/${action}?userInfo=${userInfo}&accessToken=${accessToken}&params=${params}
- method: get & post
- 入参：
  - action: ISV自定义的action
  - params: 前端传递过来的参数,toJsonString,URLEncoder.encode(params)
  - userInfo: AliYun OneConsole透传,toJsonString,URLEncoder.encode(userInfo)

    - aliyunPK: String 阿里云账号id
    - parentPk: String 阿里云主账号
    - accountStructure: Integer, 账号类型 2-主账号 3-子账号 4-角色账号
    - roleId: 角色id(角色账号登陆的情况才存在)
    - roleName: 角色name(角色账号登陆的情况才存在)
  - accessToken: Aliyun OneConsole生成，用户权限认证
  - traceId，请将参数传给aliyun-api-service，保证请求可被追踪回溯，排查问题的依据。

    [规范]
  - locale，包括zh_CN、en_US和ja，分别表示中文、英文和日文
- ISV返回数据的结构

```
// 正常返回和错误返回都要按这个格式
{
  requestId: String,
 code: String,
 message: String,
 data: <T>
}
```

ISV 调用 Aliyun-Api-Service

api-service是一个proxy，isv可以通过accessToken和appToken直接调用OPEN API。这里，Aliyun针对每个ISV做了调用白名单，只有在白名单里的产品才能被调用。

- url: http://${api-service-url}/data/openapi
- method: get & post
- 说明：使用post方式，请参考vpc控制台下的/data/api.json发送formdata请求。
- 入参：
  - product: request.getSysProduct(), sdk里的产品ID
  - version: request.getSysVersion()，sdk里的api version
  - params: OPEN API参数,toJsonString,URLEncoder.encode(params)，如果是post，不需要encode。
  - accessToken: OneConsole生成，用户权限认证，为了向后兼容，该字段依然存在
  - consoleKey: 调用api-service的凭证，即原来的 accessToken
  - consoleSecret: 调用api-service的凭证，即原来的 accessToken
  - action：对应OPEN API action
  - regionId: 调用的OPEN API region，我们会依据这个拿云产品的endpoint
  - traceId，请将参数传给aliyun-api-service，保证请求可被追踪回溯，排查问题的依据。

    [规范]
- 出参

正常返回OPEN API的请求结果

错误格式也满足错误OPEN API格式，具体参照「api-service错误码」

离线任务接口

针对离线的任务，也提供给ISV里面accessToken，但需要用户sts授权。

- 用途：获取accessToken接口
- url: http://${api-service-url}/data/queryAccessToken
- method: get
- 入参：
  - uid: 主账号uid
  - appToken：颁发给ISV的唯一身份token，不是isvCode和appCode，请不要泄露。
- 出参

```
// data里是accessToken
{
  requestId: String,
 code: String,
 message: String,
 data: String
}
```

测试阶段，用自身账号做角色扮演测试，测试也请根据权限最小原则来添加策略内容，避免上线前再做回归测试。示例：

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeRegions",
        "ecs:DescribeAvailableResource",
        "ecs:DescribeImages",
        "ecs:CreateImage"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {}
    },
    {
      "Effect": "Allow",
      "Action": [
        "vpc:DescribeVpcs",
        "vpc:DescribeVSwitches"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {}
    }
  ]
}
```

api-service 错误码

错误码分为三种：

- OPEN API的返回结果直接透传，错误码和http状态码参照各个云产品，比如 Ecs
- api-service返回的业务错误码，http状态码统一返回400，参照ErrorCode表
- api-service目前无法感知的异常，错误码用SystemError，http状态码统一返回500，表示内部错误，请联系我们排查问题！

```
public enum ErrorCode {

  ApiNotExist("ApiNotExist", "The specified api is not exist"),
  ApiVersionNotExist("ApiVersionNotExist", "The specified api version is not exist"),
  ApiActionNotExist("ApiActionNotExist", "The specified api action is not exist"),
  ApiDefineNotExist("ApiDefineNotExist", "The specified restful api define is not exist"),
  ApiDefineWrong("ApiDefineWrong", "The specified restful api define is wrong"),
  UnSupportedApiType("UnSupportedApiType", "The specified api type is not supported"),
  ApiInvokeException("ApiInvokeException", "The api call has some accidents, please try again"),
  ApiFormatException("ApiFormatException", "The specified api information parsing failed"),
  ApiParamsEncodingException("ApiParamEncodingException", "Some parameters of this specified api enc
oding error"),
  ApiEndpointFormatWrong("ApiEndpointFormatWrong", "The specified api endpoint format error"),
  ApiEndpointNotExist("ApiEndpointNotExist", "The specified api endpoint is not exist"),
  AccountInfoDecodeError("AccountInfoDecodeError", "Account information parsing failed"),
  SystemError("SystemError", "Some accidents have appeared in the system, please try again"),
  AccountInfoNotExist("AccountInfoNotExist", "Account information does not exist"),
  ApiVersionForbidden("ApiVersionForbidden", "The version param is not allowed here"),
  ForbiddenAction("ForbiddenAction", "The specified api action is forbidden"),
  MultiApiContentEmpty("MultiApiContentEmpty", "Content of this request is empty"),
  MultiApiActionMismatch("MultiApiActionMismatch", "The specified api action is not allowed"),
  ApiNoResponse("ApiNoResponse", "This api failed to respond, please try again"),
  ApiReadTimeOut("ApiReadTimeOut", "This api read timed out, please try again"),
  ApiConnectTimeout("ApiConnectTimeout", "This api connect timed out, please try again"),
  ApiConnectionRefused("ApiConnectionRefused", "This api connection refused, please try again"),
  ApiConnectionReset("ApiConnectionReset", "This api connection reset, please try again"),
  ApiUnknownEndpoint("ApiUnknownEndpoint", "This api endpoint is not reachable"),
  ApiEndpontNoRouteToHost("ApiEndpontNoRouteToHost", "This api endpoint is not reachable"),
  MissingApiParam("MissingApiParam", "Missing the necessary request parameters, please add and try agai
n"),
  StsTokenError("StsTokenError", "get sts token failed"),
  InvalidParams("InvalidParams", "The params cannot convert to Map"),
  AppTokenNotExist("AppTokenNotExist", "isv app token not exist"),
  AccessTokenInvalid("AccessTokenInvalid", "access token invalid"),
  ActionNotInWhiteList("ActionNotInWhiteList", "this action is not in white list");

  private String code;
  private String msg;
}
```

用户购买流程回调

用户通过阿里云市场购买插件服务，购买完成我们会回调isv的接口告知用户的购买状态，包括购买，过期，续费等。

ISV提供的回调服务格式同**「 ISV后端服务 」**

具体的action含义参考 文档

OSS使用

需要使用oss的场景，为了防止用户数据泄露，我们不允许ISV使用自己账号的OSS。通过sts的方式，isv能获取到指定bucket的操作ak。

- url: http://${api-service-url}/data/

  queryOssCredentials
- method: get & post
- 入参：
  - appToken: String.同离线调用，需要提前找我们做白名单处理。
- 出参
-

```
// data里是sts ak信息
{
  requestId: String,
 code: String,
 message: String,
 data: {
  "SecurityToken":"xxx",
  "AccessKeyId":"xxxx",
  "AccessKeySecret":"xxx",
  "Expiration":"2020-06-17T08:11:31Z"
 }
}
```

测试环境

出于安全考虑，我们需要对各个环节都做白名单处理，测试环境架构图：

isv可以本地调试前端代码，并把后端代码放在自己的环境(有公网IP)下进行调试。

需要isv提供的配置如下：

```
{
  "[isvCode]-[appCode]":{ // isvCode标识isv名，appCode表示插件名，由isv提供。
    "local_debug": {
      "is_debug": true, // 是否开始本地调试【isv填写】
      "url":"http://218.17.169.171:31574" // 后端服务的公网ip&port【isv填写】
    },
    "online":{
      "ips":[
        "xxxxxxxx", // 访问前端的来源ip白名单【isv提供】
      ],
      "account":{ // 测试账号的uid、ak 【isv提供】
        "uid": xxxxxx,
        "accessKeyId":"xxxxx",
        "accessKeySecret":"xxxxxx"
      }
    },
    "offline":{
      "ips":[ // isv后端服务公网ip 【isv填写】
        "xxxxxx"
      ],
      "appToken":"xxxxx", //由阿里颁发
      "uids":[ // 允许sts调用的uid 【isv填写】
        xxxxxx
      ]
    }
  }
}
```

说明

- 本机的公网IP(
  http://www.cip.cc/
)发给我们，做白名单处理，否则无权限访问。
- isv需按照上述json结构提供数据，如无sts调用需求，则offline部分无需提供。
- isvCode标识isv名，appCode表示插件名，由isv提供。
- 支持sts调用，isv需要在允许sts调用的uid所在账号创建一个ram role，命名参考图中，并给console.aliyuncs.com授权，阿里这边需要在api-service上配置role name。如图：
- 需要绑定host：114.55.202.134 work.console.aliyun.test
- 测试环境，api-service的访问通过114.55.202.134 访问即可

- 前端页面访问路径

  http://work.console.aliyun.test/[isvCode]-[appCode]

Mock接口

背景

我们假定拥有以上元数据

1. isvCode: testIsv
2. appCode: testApp
3. 一个RAM用户（子用户）登录阿里云打开使用插件 子用户的UID 为 12345678 对应的主账号的UID为88888888
4. 调用一个ISV封装的 testAction的接口
5. 阿里云给ISV提供的服务地址为11.22.33.44

> *因为阿里云要求阿里云必须在/data 这个路由下提供接口， 即真实的调用路径为*
> *http://11.22.33.44/data/testAction*

5. ISV通过这个接口需要的参数为pageSize 和 pageNumber

前端调用

General
- Request URL: https://work.console.aliyun.com/data/plugin.json
- Request Method: POST

Form Data
- product: testIsv-TestApp
- action: testAction
- params: {"pageSize": 1, pageNumber: "20"} //这些是ISV自己定义的需要透传的参数
- sec_token: XjJHIHy0HDf1I2bLotfLGE //从 window.ALIYUN_CONSOLE_CONFIG.SEC_TOKEN获取

经过阿里云预处理后传递给ISV的参数

```
http://11.22.33.44/api/testAction?userInfo=%7B%22aliyunPk%22%3A%2212345678%22%2C%22parentPk%22%3A%228888888%22%2C%22accountStructure%22%3A3%2C%22roleId%22%3A%22%22%2C%22roleName%22%3A%22%22%7D}&accessToken=${accessToken&params=%7B%22pageSize%22%3A1%2C%22pageNumber%22%3A%2220%22%7D  //参数经过encode
```

ISV返回的结果

正常返回

```
{
  "code":"200",
  "data":["1", "2", "3"],
  "requestId":"ac10012716012633996472243 6",
  "message":""
}
```

异常返回

```
{
  "code":"Permission Forbidden",
  "data":null,
  "requestId":"ac10012716012633996472243 6",
  "message":"该用户没有使用权限."
}
```

阿里云返回给前端 （透传）

ISV 正常返回

```
{
  "code":"200",
  "data":["1", "2", "3"],
  "requestId":"ac10012716012633996472243 6",
  "message":""
}
```

ISV 异常返回

```
{
  "code":"Permission Forbidden",
  "data":null,
  "requestId":"ac10012716012633996472243 6",
  "message":"该用户没有使用权限."
}
```

2.聚石塔自管-技术接入指南

聚石台接入模式概览

Oauth授权

step1: RAM 配置Oauth应用管理

获取如下信息:

```
aliyun.oauth.clientId=xxxxx
aliyun.oauth.clientSecret=xxxxxx
aliyun.oauth.redirectPath=xxxxx
```

step2: 获取身份信息

按照阿里云 RAM Oauth 的方式实现三方授权，解决身份登录问题，得到关键的 `id_token` 、 `access_token` :

```
{
 "access_token": "eyJraWQiOiJrMTIzNCIsImVu****",
 "token_type": "Bearer",
 "expires_in": 3600,
 "refresh_token": "Ccx63VVeTn2dxV7ovXXfLtAqLLERA****",
 "id_token": "eyJhbGciOiJIUzI1****"
}
```

step3: 内容解析

通过 JWT 来进行解析，得到如下结构:

```
{
 "exp": 1517539523,
 "sub": "123456789012****",
 "aud": "4567890123456****",
 "iss": "https:\/\/oauth.aliyun.com",//中国站
 "iat": 1517535923,
 "name": "alice", //登录用户的显示名称
 "login_name":"alice@example.com", // 阿里云账号的登录名
 "aid": "123456789012****", //对应的阿里云账号ID
 "uid": "123456789012****", //登录用户的阿里云账号ID
}
```

因为阿里云的 Oauth 暂时没办法通过 `access_token` 进行 OpenAPI 调用，所以需要借助于阿里云的服务角色来实现 OpenAPI 调用。

OpenAPI调用

关于阿里云OpenAPI调用，需要经过三步：

step1 权限申请

按照RAM的授权策略，定义插件需要的权限点，例如：

```
{
  "Version":"1",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ecs:DescribeInstances",
        "ecs:DescribeInstanceTypes"
      ],
      "Resource":[
        "*"
      ],
      "Condition":{

      }
    },
    {
      "Effect":"Allow",
      "Action":[
        "rds:DescribeDBInstances",
        "rds:DescribeAvailableResource"
      ],
      "Resource":[
        "*"
      ],
      "Condition":{

      }
    }
  ]
}
```

提交给阿里云进行审核，通过后，会创建服务角色。同时阿里云会给 ISV 颁发一个 `appToken` ，作为 OpenAPI 调用的凭证。

step2 用户 RAM 授权

用户购买插件以后，跳转到 RAM 授权页面，在用户账号下创建服务角色，授予 ISV OpenAPI 调用权限。

授权以后，得到类似的服务角色：

该角色，会具有 step1 的授权策略。

当然，开发阶段也可以手动创建服务角色，然后增加权限点。

step3 执行OpenAPI调用

- 当前的 `Endpoint` 是 `http://100.100.0.242`
- 按照 ISV应用接入指南 进行调用
  - 首先根据 `appToken` 、 `idToken` 获得 `accessToken`
  - 根据 `accessToken` 完成 `OpenAPI` 调用

例如：

```
// 获取accessToken，有效期是15分钟
curl http://100.100.0.242/api/acs/queryAccessToken?idToken=xxx&appToken=xxx
// 根据accessToken调用OpenAPI
curl http://100.100.0.242/api/acs/openapi?accessToken=xxx&product=ecs&action=DescribeRegions
```

聚石塔部署环境介绍

概述

ISV接入企业工作台的第二个步骤就是需要将已经打包好的应用部署在阿里云上。那么首先就需要购买相应的云产品，例如ECS、RDS、OSS、弹性IP等，然后根据应用的架构进行影响的部署，最后进行线上的调试和测试。那么上述这几个步骤我们将借助阿里云“聚石塔”的基础能力来实现。

聚石塔介绍

聚石塔是一套为ISV部署和运维应用的PaaS平台，为ISV可以高效的开发、调试、运维，以及保障数据安全而推出的一项基础服务。通过该服务，ISV的云主机可依据系统需要对带宽、和机器性能进行动态扩容，升降配，实现资源的弹性扩展，按需付费。

聚石塔的安全规则

1. ISV部署在聚石塔内的应用不得通过自定义接口与塔外系统进行数据交互，确有塔内系统与塔外系统数据交互的业务场景，须提交业务场景及字段说明，平台审核通过后，开发者应通过开放平台奇门标准接口体系对塔外系统交互.
2. 为了您的数据安全云数据库不允许直接从数据库中dump数据，因此类似into outfile，load_file，mysqldump，Backup table等数据dump的命令会被过滤掉。若有数据dump需求，后续走专门的申请流程（例如提供公司公章等材料），采用dump工具进行数据dump。

聚石塔应用架构规则

规则名称	规则要求
系统部署	购买ECS和RDS,应用部署在ECS上，数据库迁移至RDS上。
RDS选型	RDS类型分为MYSQL和SQLSERVER。
数据库安装	ECS允许安装数据库客户端软件，来管理RDS数据库(DB)，但是数据库(DB)不允许安装在ECS服务器上
系统架构	应用系统入塔要求使用三层架构，业务逻辑在服务端ECS上运行，RDS数据只允许ECS上服务层访问，客户端只作为应用可视化展现.
API调用	所有API调用要求在塔内发起调用；
业务逻辑	云管理平台系统： 运维工具类系统 其他类型的系统，其主要功能模块涉及到用户的资源数据，业务逻辑必须在塔内运行；
数据存储	涉及订单和用户数据的业务履行处理的数据必须存储在RDS中；

资源订购与应用部署

概览：

一般情况下需要创建两个应用，一个应用作为web应用，即前台工程，用于和前端页面交互以及后端核心业务逻辑交互的应用程序；另一个应用作为后台应用，即后台工程，作为核心业务逻辑层，主要后端业务逻辑编写以及数据持久化和数据库访问。两个应用之间的通信，采用dubbo的服务化框架进行通信。如果是第一次使用聚石塔，可以参考此应用创建实践。

应用部署主要基于两种部署方式：

1.虚拟机方式

2.容器方式部署（推荐）

一、前置准备工作

1.注册淘宝账号（已有淘宝账号，可以忽略）

2.登录聚石塔，进入控制台

3.进入新控制台

二、容器部署方式详细步骤（推荐方式）：

步骤说明

三、虚拟机方式详细步骤：

1.点击“资源视图”进入云产品导航页面

2.选择并点击进入相应的云产品控制台（以云服务ECS控制台为例）

3.选择相应的region，并点击“创建实例”

4.根据实际情况，订购相应的ECS。其他云产品订购流程与ECS基本相同。

4.1.创建应用

4.2.填写应用表单

注意：业务类型请选择”其他云应用“；发布方式请选择”虚拟机发布“

4.3.应用列表

点击查看进入应用详情页

4.4.关联资源

点击右上角”关联资源“，后选择需要关联的资源类型，以ECS为例

5. 应用发布

5.1 点击应用发布

5.2 选择应用的系统架构

5.3 创建发布单

5.4 发布单列表

更多内容请参考文档

域名统一接入

SLB处理

> *因为SLB支持『转发策略』功能，那么可以借助该功能实现path的白名单机制*

日志审计

最终策略

因为主账号在阿里云侧，因此，可以基于主账号修改SLB

- 上线前，在SLB层面接入SLS日志审计
- 如果有白名单机制：
  - 默认情况下，流量转发的服务器组，都是不可用
  - 设置服务器组，把允许的请求转发的固定服务器组
- 基于EIP，进行DNS绑定，同时在SLB上上传https证书

建议应用域名

<appName>.<isvdomain>.com

例如公司 monitormaster

monitorapp.monitormaster.com