# Q/NCB IT

宁波通商银行股份有限公司技术标准

Q/NCB IT 001—2024



# 宁波通商银行开放平台 API 接口标准及调用规范

企业标准信息公共服务平台
公开 2024年12月20日 10点22分

2024-12-20 发布

宁波通商银行股份有限公司 发布

# 目　录

**前　言** ..... **4**
**宁波通商银行开放平台 API 接口标准及调用规范** ..... **5**
**1 范围** ..... **5**
**2 规范性引用文件** ..... **5**
**3 术语和定义** ..... **5**
**4 缩略语** ..... **6**
**5 概述** ..... **6**
**6 标准内容** ..... **7**
6.1 协议规范 ..... 7
6.2 Schema 规范 ..... 7
6.3 URL 设计规范 ..... 8
6.4 HTTP 方法规范 ..... 8
6.5 状态码规范 ..... 9
6.6 错误处理规范 ..... 9
6.7 认证和授权规范 ..... 9
6.8 限流规范 ..... 10
**7 实施要求** ..... **10**
7.1 宣传 ..... 10
7.2 培训 ..... 10
7.3 监督检查 ..... 10

# 前　言

本标准编制说明：

本规范根据中国人民银行发布的《商业银行应用程序接口安全管理规范》（JR/T 0185—2020）进行合理转化编制，规定了我行开放平台API接口标准及调用规范。

本标准由宁波通商银行股份有限公司提出并归口

本标准编制单位：宁波通商银行股份有限公司

本标准主要编制人：总行信息科技部 吴斌

# 宁波通商银行开放平台 API 接口标准及调用规范

## 1 范围

本标准规定了我行开放平台API接口标准及调用规范。

## 2 规范性引用文件

中国人民银行发布的《商业银行应用程序接口安全管理规范》（JR/T 0185—2020）

## 3 术语和定义

### 3.1 应用程序接口 application programming interface

一组预先定义好的功能，开发者可通过该功能（或功能的组合）便捷地访问相关服务，而无需关注服务的设计与实现。

### 3.2 应用方 application agency

调用商业银行应用程序接口的机构。

### 3.3 应用程序接口唯一标识 application programming interface unique ID

由商业银行自行定义，用于区分商业银行应用程序接口功能的唯一标识。

### 3.4 应用程序接口统一识别码 uniform application programming interface ID

商业银行依据行业主管部门发布的编码规则，生成的商业银行应用程序接口统一识别码。

注：用于标识商业银行机构代码、接口类型、服务类别、接口顺序号等内容。

### 3.5 应用软件开发工具包 software development kit

基于特定软件包、软件框架、硬件平台、操作系统等建立应用程序时所使用的软件开发工具集合。

### 3.6 应用唯一标识 application unique ID

在应用方身份核验通过后，根据其调用的金融产品与服务类型，由商业银行为其授予的唯一标识。

注：包括服务器端应用标识与移动终端应用软件标识两种。

### 3.7 应用鉴别密文 application secret

应用合法性鉴别凭证，与应用唯一标识配套使用，以验证通过 API 方式接入的应用合法性，接入验证通过后，即可完成系统对接，调用应用程序接口或使用应用程序接口提供的功能和数据。

## 4 缩略语

下列缩略语适用于本文件。

API：应用程序接口（Application Programming Interface）

API_ID：接口唯一标识（Application Programming Interface unique ID）

App_ID：应用唯一标识（Application unique ID）

App_Secret：应用鉴别密文（Application Secret）

DDoS：分布式拒绝服务攻击（Distributed Denial of Service）

U_API_ID：应用程序接口统一识别码（Uniform Application Programming Interface ID）

SDK：应用软件开发工具包（Software Development Kit）

SSL：安全套接层协议（Secure Sockets Layer）

TLS：安全传输层协议（Transport Layer Security）

MAC：消息鉴别码（Message Authentication Code）

## 5 概述

商业银行应用程序接口服务是一种依托 API 技术实现内部与外部互联的金融服务模式。商业银行通过为合作伙伴提供用以互联的应用程序接口，输出自身金融服务能力与信息技术能力，为增加金融生态黏性提供有益补充。外部机构能够通过互联网渠道，调用商业银行应用程序接口（外部 API），获取商业银行提供的各类服务，其逻辑结构见图 1。

商业银行应用程序接口服务的参与方主要包括用户、应用方以及商业银行，商业银行通过 API 直接连接或 SDK 间接连接方式向应用方和用户提供应用程序接口服务，实现商业银行服务的对外输出。

用户发起商业银行应用程序接口应用请求，并接收由应用方或商业银行返回的处理结果。 应用方负责接收并处理用户请求，通过应用程序接口向商业银行提交相关请求、接收返回结果，依 照流程进行服务请求处理或反馈用户。

商业银行构建商业银行应用程序接口、应用程序接口服务层和银行业务系统以提供商业银行应用程序接口服务。商业银行应用程序接口服务层将应用方请求转发至银行业务系统处理，并将处理结果反馈应用方或用户，包含认证鉴权、流量控制、监控分析、报文交换、服务组合等功能，不涉及具体业务逻辑处理，实现对商业银行应用程序接口和应用方的管理。

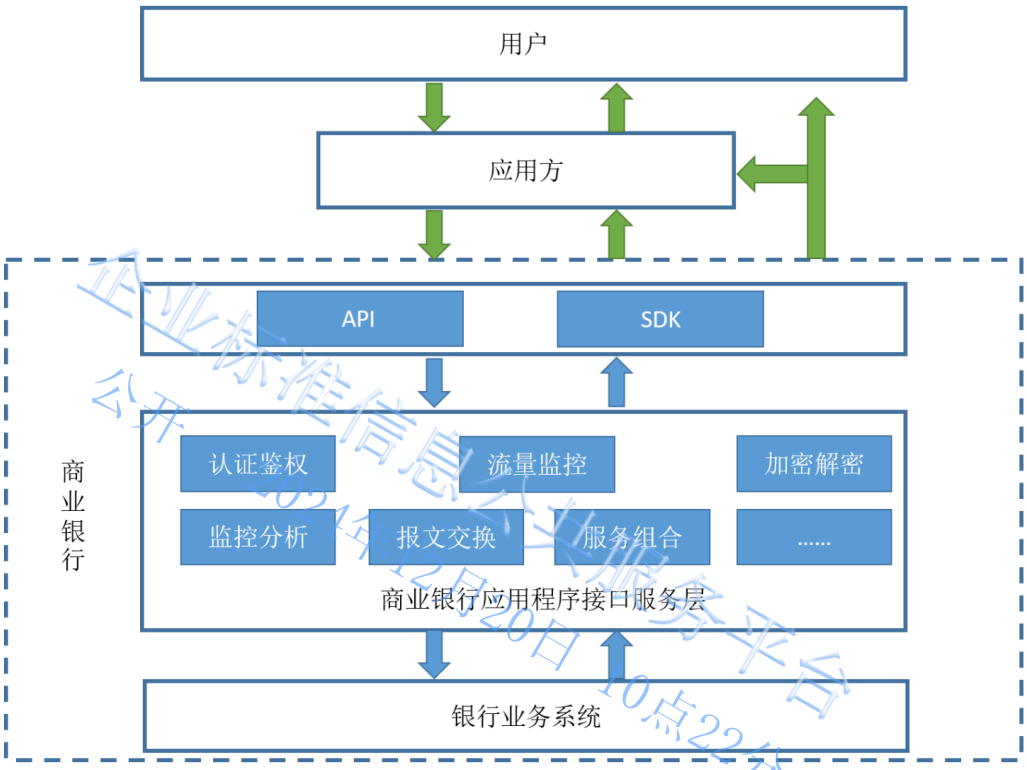


图 1 商业银行应用程序接口逻辑图



## 6 标准内容

### 6.1 协议规范

【规则 1-1】API 与用户的通信协议，使用 HTTPS 协议。

### 6.2 Schema 规范

**【规则 2-1】**URI 的格式定义如下：**URI = scheme "://" authority "/" path \[ "?" query \] \[ "#" fragment \]**

URL 是 URI 的一个子集(一种具体实现)，对于 REST API 来说一个资源一般对应一个唯一的 URI（URL）。在 URL 的设计中，我们会遵循一些规则，使接口看起透明易读，方便使用者调用。

**【规则 2-2】**"/"分隔符一般用来对资源层级的划分。为了避免混淆，"/"不应该出现在 URL 的末尾。

**【规则 2-3】**URL 中尽量使用连字符"-"代替下划线"_"的使用。如：http://api.example.restapi.org/blogs/mark-masse/entries/this-is-my-first-post。

**【规则 2-4】**URL 应该统一使用小写字母。

【规则 2-5】URL 中不要包含文件(脚本)的扩展名。例如 .json 之内的就不要出现了。

【规则 2-6】JSON 中的所有字段都应该用小写.

### 6.3 URL 设计规范

【规则 3-1】资源是 Restful API 的核心元素，所有的操作都是针对特定资源进行的。而资源就是 URL（Uniform Resource Locator）表示的，所以简洁、清晰、结构化的 URL 设计是至关重要的。

正例

/users/:username/repos
/users/:org/repos
/repos/:owner/:repo
/repos/:owner/:repo/tags
/repos/:owner/:repo/branches/:branch

【规则 3-2】URL 不要带有动词

反例

URL 是 /users/show/1

正例

URL 是 /users/1

### 6.4 HTTP 方法规范

【规则 4-1】所有针对资源的操作都是使用 HTTP 方法指定的比较常用的 HTTP/1.1 动词有下面 5 个：

- **GET**：从服务器取出资源（一项或多项）。
- **POST**：在服务器新建一个资源。
- **PUT**：在服务器更新资源（客户端提供改变后的完整资源）。
- **PATCH**：在服务器更新资源（更新资源的部分属性）。
- **DELETE**：从服务器删除资源。

【规则 4-2】HTTP 协议涉及到的一种重要性质：幂等性(Idempotence)，HTTP 方法的幂等性是指一次和多次请求某一个资源应该具有同样的副作用。

| HTTP Method | 幂等 | 安全 |
| --- | --- | --- |
| OPTIONS | yes | yes |
| GET | yes | yes |
| HEAD | yes | yes |
| PUT | yes | no |
| POST | no | no |
| DELETE | yes | no |
| PATCH | no | no |

### 6.5 状态码规范

**【规则 6-1】**HTTP 应答中，需要带一个很重要的字段：status code。状态码都是三位的整数，大概分成了几个区间：

2XX：请求正常处理并返回

3XX：重定向，请求的资源位置发生变化

4XX：客户端发送的请求有错误

5XX：服务器端错误

### 6.6 错误处理规范

**【规则 6-1】**应该在 response body 中通过 message 给出明确的错误信息（一般来说，返回的信息中将 message 作为键名，出错详情作为键值即可）。给出错误 message，能更好地让客户端知道具体哪里有问题，进行快速修改。

```
{
 "message":"错误详情"
}
```

### 6.7 认证和授权规范

**【规则 7-1】**验证和授权是两件事情：

1. 验证（Authentication）是为了确定用户是其申明的身份，比如提供账户的密码。不然的话，任何人伪造成其他身份（比如其他用户或者管理员）是非常危险的

2. 授权（Authorization）是为了保证用户有对请求资源特定操作的权限。比如用户的私人信息只能自己能访问，其他人无法看到；有些特殊的操作只能管理员可以操作，其他用户有只读的权限等等

如果没有通过验证（提供的用户名和密码不匹配，token 不正确等），需要返回 401 Unauthorized 状态码，并在 body 中说明具体的错误信息；而没有被授权访问的资源操作，需要返回 403 Forbidden 状态码，还有详细的错误信息。

### 6.8 限流规范

**【规则 8-1】**如果对访问的次数不加控制，很可能会造成 API 被滥用，甚至被 DDOS 攻击。

对用户的请求限流之后，要有方法告诉用户它的请求使用情况，推荐使用的三个相关的头部：

**X-RateLimit-Limit**: 用户每个小时允许发送请求的最大值

**X-RateLimit-Remaining**：当前时间窗口剩下的可用请求数目

**X-RateLimit-Rest**: 时间窗口重置的时候，到这个时间点可用的请求数量就会变成 X-RateLimit-Limit 的值

## 7 实施要求

### 7.1 宣传

面向行内下发本规范，成为技术标准。

### 7.2 培训

行内培训及学习。

### 7.3 监督检查

本规范内容作为技术监督检查内容不定期进行抽查。