中国物品编码中心

接口设计规范

V1.0

**2024年8月**

目录

[文档版本历史 3](#_Toc2547)

[1 概述 4](#_Toc1746)

[2 基本要求 5](#_Toc26807)

*[01. 信息通讯安全](#_Toc29486)* [5](#_Toc29486)

[02. 支持高开发 5](#_Toc22081)

[03. 可监控 6](#_Toc11474)

[04. 系统资源的动态扩展 6](#_Toc21559)

[05. 异常处理机制 6](#_Toc7535)

[06. 业务扩展 6](#_Toc386)

[3 接口通讯方式 7](#_Toc23622)

[01. 同步请求/应答方式 7](#_Toc9333)

[02. 异步请求/应答方式 7](#_Toc14888)

[03. 会话方式 7](#_Toc1849)

[04. 广播通知方式 7](#_Toc10932)

[05. 事件订阅方式 7](#_Toc16555)

[06. 文件传输 7](#_Toc23616)

[07. 可靠消息传输 7](#_Toc5382)

[4 传输控制要求 9](#_Toc7246)

[01. 负载均衡 9](#_Toc13592)

[02. 伸缩性与动态配置管理 9](#_Toc21638)

[03. 网络调度 9](#_Toc6091)

[5 接口设计原则 10](#_Toc3200)

[01. 充分理由 10](#_Toc18760)

[02. 单一职责 10](#_Toc32488)

[03. 高内聚低耦合 10](#_Toc9229)

[04. 状态及信息 10](#_Toc1221)

[05. 控制数据量 10](#_Toc19351)

[06. 禁止随意扩展参数 11](#_Toc28645)

[6 接口技术 12](#_Toc15773)

[7 接口规范 13](#_Toc30142)

[01. 域名规范 13](#_Toc13809)

[02. API路径规范 13](#_Toc5701)

[03. 版本控制规范 14](#_Toc17015)

[04. API命名规范 14](#_Toc1669)

[05. 方法操作 14](#_Toc4059)

[06. 请求参数规范 15](#_Toc9479)

[07. 返回数据规范 15](#_Toc31978)

[8 接口文档规范 16](#_Toc28223)

[9 接口管理 17](#_Toc12727)

[01. 接口分类、编码排序 17](#_Toc22939)

[02. 在线文档 17](#_Toc8441)

文档版本历史

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **变更内容** | **变更时间** | **主要参与人员** |
| V1.0 | 初稿 | 2024.08 | 栗许静 |
|  |  |  |  |

# 概述

本文旨在规范接口标准，确保系统之间的数据交互安全有效和可靠。接口标准规范适用于所有系统之间的数据交互。为服务使用方和服务提供方提供开发参考。

基本要求包括信息通讯安全、数据格式、数据传输、接口协议和错误处理等方面。其中，信息通讯安全是最基本的要求，必须得到严格遵守。

信息通讯安全包括身份认证、数据加密、防篡改和防重放等方面。必须确保数据的完整性、保密性和可用性。同时，必须保证系统之间的身份认证和授权，防止非法访问。

数据格式必须符合规范，确保数据传输的正确性和可靠性。数据传输必须采用可靠的传输协议，确保数据的完整性和可靠性。接口协议必须符合规范，确保系统之间的数据交互的正确性和可靠性。

错误处理必须及时、准确地处理错误信息，确保系统之间的数据交互的稳定性和可靠性。必须提供完善的错误处理机制包括错误码、错误信息和错误处理流程等方面。

总之，接口标准规范是确保系统之间的数据交互安全、有效和可靠的基础，必须得到严格遵守。

# 基本要求

## *信息通讯安全*

1. 安全评估：保证接口的自身安全，通过接口实现技术上的安全控制，做到对安全事件的“可知、可控、可预测"，是实现系统安全的一个重要基础。
2. 访问控制：如果客户端很频繁的请求服务器，会给给服务器造成很大的压力，需要对客户端对 API的请求做一些限制。比如单位时间内同一IP只能请求多少次。
3. 防恶意代码：我们的系统内置了强大的恶意代码防护机制，保证用户的设备不会受到病毒、木马等恶意代码的攻击。
4. 加密
5. 接口调用方和接口提供方约定好统一的参数加密算法(加密方法参照接口规范的请求参数规范)。
6. 接口调用方在调用时把加密后的\_sign放在参数中去请求接口。
7. 接口提供方接到响应后，判断时间戳是不是在有效时间内(这个时间间隔根据你的安全范围可以是10分钟，5分钟，20秒等，过期失效，前提是需要保证接口提供方和调用方的服务器时间为准确的网络同步时间)。
8. 把参数中除了\_sign以外的参数进行加密，然后把加密结果和传过来的\_sign比较，相同则执行调用请求。
9. 如果服务器和客户端的时间没有同步，可以返回错误的同时候在返回一个服务器的当前时间，客户端接收到该错误后再请求上一个接口，时间则传服务器刚刚返回的时间。
10. 涉及到比较重要的信息，可以用AES对value进行加密，防止抓包拉取到上传的数据。

## 支持高开发

我们的系统在设计时考虑到多线程的情况，采用了分布式架构和负载均衡技术，能够支持大规模的并发访问。

## 可监控

我们的系统具备完善的监控功能，可以实时监控系统运行状态、用户访问情况等，及时发现并解决问题，保证系统的稳定性和可靠性。

日志全覆盖：系统在进行日志记录时，覆盖之前的日志信息。这种方式可以避免日志文件过大，减少存储空间的占用但是，如果系统出现问题，之前的日志信息将无法被恢复，因此需要谨慎使用。

1. 记录正常运行信息
2. 记录异常捕获信息
3. 统一规范日志打印格式，以满足运维需求和格式统一规范

## 系统资源的动态扩展

保证在充分利用系统资源的前提下，实现系统平滑的移植和扩展，同时在系统并发增加时提供系统资源的动态扩展，以保证系统的稳定性。

## 异常处理机制

表单验证、唯一性检查、或其他可预期的错误。我们需要编写特定代码来捕获这类错误，并抛出一个包含提示信息的全局异常，捕获并返回给客户端。

## 业务扩展

系统在运行过程中，根据需要扩展业务功能例如，当用户需求发生变化时，可以增加新的业务功能以满足用户需求。这种方式可以提高系统的灵活性和可扩展性，但是需要考虑业务变化的频率和影响范围。

# 接口通讯方式

## 同步请求/应答方式

客户端向服务器端发送服务请求，客户端阻塞等待服务器端返回处理结果。这种方式可以确保请求和响应的顺序和一致性，但是需要考虑响应时间和系统性能等问题。

## 异步请求/应答方式

客户端向服务器端发送服务请求，与同步方式不同的是，在此方式下，服务器端处理请求时，客户端继续运行；当服务器端处理结束时返回处理结果。这种方式可以提高系统的并发性和响应速度，但是需要考虑请求和响应的匹配和处理方式。

## 会话方式

客户端与服务器端建立连接后，可以多次发送或接收数据，同时存储信息的上下文关系。会话方式可以通过不同的协议来实现，例如 TCP 和 UDP。

## 广播通知方式

由服务器端主动向客户端以单个或批量方式发出未经客户端请求的广播或通知消息，客户端可在适当的时候检查是否收到消息并定义收到消息后所采取的动作。

## 事件订阅方式

客户端可事先向服务器端订阅自定义的事件，当这些事件发生时，服务器端通知客户端事件发生，客户端可采取相应处理。事件订阅方式使客户端拥有了个性化的事件触发功能，极大方便了客户端及时响应所订阅的事件。

## 文件传输

客户端和服务器端通过文件的方式来传输消息，并采取相应处理。

## 可靠消息传输

在接口通讯中，基于消息的传输处理方式，除了可采用以上几种通讯方式外，还可采用可靠消息传输方式，即通过存储队列方式，客户端和服务器端来传输消息，采取相应处理。这种方式的优点是可以确保消息的完整性和正确性，从而避免数据传输中的错误和丢失。可靠消息传输可以通过TCP 协议来实现。

# 传输控制要求

传输控制利用高速数据通道技术实现把前端的大数据量并发请求分发到后端从而保证应用系统在大量客户端同时请求服务时，能够保持快速、稳定的工作状态。而传输控制要求是指在网络通信中对传输过程进行控制的要求。这些要求包括传输速度、传输可靠性、传输安全性等。为了满足这些要求，需要选择合适的协议和技术来进行网络通信同时还需要对网络进行优化和管理，以提高网络的性能和稳定性。

## 负载均衡

有必要时为了确保接口服务吞吐量最大，接口应自动地在系统中完成动态负载。通过负载均衡，可以避免单个服务器过载，提高系统的吞吐量和响应速度。常见的负载均衡算法包括轮询、加权轮询、最少连接数等。

## 伸缩性与动态配置管理

由系统自动伸缩管理方式或动态配置管理方式实现队列管理、存取资源管理以及接口应用的恢复处理等。

## 网络调度

当对接口有较高通讯保障要求时可能会在在双方接口之间设置多个网络通道需要实现接口的多数据通道和容错性，保证当有一网络通道通讯失败时，进行自动的切换，实现接口连接的自动恢复。

# 接口设计原则

## 充分理由

不是随便一个功能、需求就要加个接口。每新建一个接口，就要有充分的理由和考虑，即这个接口的存在是十分有意义额价值的，无意义的接口不仅增加了维护的难度，更重要是对于程序的可控性的大大降低，接口也会十分臃肿。

## 单一职责

一个接口只负责一个业务功能。

## 高内聚低耦合

一个接口要包含完整的业务功能，而不同接口之间的业务关联要尽可能的小。还是查询会员的例子，有时查询会员的同时，可能该会员的相关信息要随之发生变化(如状态)，如果这时一条完整的业务流水线，那么就应该在一个接口里完成，而不应再单独设立接口去操作完成。就是说一个接口不应该随着另一个变化而变化或以某几个接口为前提而存在。

## 状态及信息

提供必要的接口调用状态信息。调用是否成功如果失败，那么失败的原因是什么。这些必要的信息必须要告诉给客户端。提供必要的接口调用状态信息。调用是否成功如果失败，那么失败的原因是什么。这些必要的信息必须要告诉给客户端。

## 控制数据量

一个接口返回不应该包含过多的数据量，过多的数据量不仅处理复杂，对数据传输的压力也非常大，会导致客户端反应缓慢。过多的数据量很多时候都是接口划分不明确。

## 禁止随意扩展参数

拓展接口可能是难以避免的，但是不要随意就加参数，加参数一定是必要且有意义的，需求改变前首先应考虑现有接口内部维护是否能满足需求，而不要通过加个参数来方便自己实现需求的难度，因为参数的更变会直接导致客户端调用的变化，容易产生版本兼容性问题。

# 接口技术

接口技术是 API设计中的一个重要方面。API应该使用标准的协议和格式，以便客户端可以轻松地与 API进行通信。常见的 API技术包括 REST、SOAP 和 GraphQL 等。

# 接口规范

接口规范是API设计中的另一个重要方面。API应该定义明确的接口规范，包括请求和响应的格式、参数和状态码等。这可以确保 API的一致性和可靠性，并且可以减少客户端的错误。

## 域名规范

在API设计中，域名规范也是非常重要的。API应该使用明确的域名，并且应该将API的版本号包含在域名中。每个项目要有且仅有一个自己唯一的域名(或ip地址)+端口。如果一个域名(或ip 地址)满足不了要求，那么就需要再添加一个。

1. HTTP接口

HTTP 接口是一种基于 HTTP 协议的接口，通常用于 Web应用程序之间的通信。在设计HTTP接口时，需要考虑以下几个方面:请求方法、请求参数、请求头、请求体、响应状态码响应头和响应体。

1. WebService接口

WebService 是一种基于 Web 的应用程序接口，它使用XML和 HTTP 协议进行通信。在设计WebService 接口时，需要考虑以下几个方面:SOAP 协议、WSDL文档、请求参数、响应参数、请求头、响应头和错误处理。必须在路径中添加api目录，且须以字母开头，以“/”结尾。

## API路径规范

API路径规范是指在设计API时，需要遵循一定的路径规则，以便于统一管理和维护。在设计API路径时，需要考虑以下几个方面:路径格式、路径参数、路径变量和路径层级。

1. HTTP接口

在设计HTTP接口路径时，需要遵循RESTul风格的路径规范，即使用 HTTP 请求方法来表示对资源的操作，并使用路径变量来表示资源的唯一标识。例如，GET/users/fid}表示获取用户信息，POST/users表示创建用户，

1. WebService接口

在设计 WebService 接口路径时，需要遵循 SOAP 协议的路径规范，即使用 WSDL文档来定义接口的路径和参数。

## 版本控制规范

版本控制规范是指在设计 API时，需要考虑API的版本管理和升级策略。在设计 API版本控制时，需要考虑个方面:版本号格式、版本号命名规则、版本号升级策略和版本号兼容性。例如，使用v1、v2等格式来表示API的版本号使用语义化版本号规则来命名版本号，使用向后兼容的升级策略来保证 API的稳定性和可扩展性。

## API命名规范

API命名规范是指在设计和定义API时应该遵循的规范和标准。在 API命名时，我们需要注意以下几点:

1. API的名称应该清晰明了，能够准确描述API的功能和作用；
2. API的参数应该清晰明了，不应该存在歧义或不必要的参数；
3. API的返回值应该符合预期，不应该存在错误或异常情况；
4. API的命名应该符合行业规范和标准，不应该存在歧义或不规范的命名方式。

通过遵循 API命名规范，我们可以提高API的可读性和可维护性，从而提高 API的质量和稳定性。

## 方法操作

1. 新增方法：该方法用于新增数据，具体实现方式为向服务器发送一个POST请求，并携带新增数据的参数。成功新增数据后，服务器会返回一个成功提示信息。
2. 删除方法：该方法用于删除数据，具体实现方式为向服务器发送一个DELETE请求，并携带要删除的数据的ID参数。成功删除数据后，服务器会返回一个成功提示信息。
3. 修改方法：该方法用于修改数据，具体实现方式为向服务器发送一个PUT请求，并携带要修改的数据的ID参数和修改后的参数。成功修改数据后，服务器会返回一个成功提示信息。
4. 获取方法：该方法用于获取单个数据，具体实现方式为向服务器发送一个GET请求，并携带要获取的数据的ID参数。成功获取数据后，服务器会返回一个包含数据的 JSON格式对象。
5. 获取列表方法：该方法用于获取数据列表，具体实现方式为向服务器发送一个GET请求，并携带获取数据列表的参数。成功获取数据列表后，服务器会返回一个包含数据列表的 JSON格式数组。

## 请求参数规范

本部分旨在说明在接口请求中，列表请求的特殊规范。在列表请求中，需要指定每页的数据量和当前页码。具体规范如下:

1. 请求参数中需要包含pageSize和pageNo 两个参数，分别表示每页数据量和当前页码。
2. pageSize 参数的取值范围为 1~100，pageNo参数的取值范围为 1~XXX，超出范围将返回错误信息。
3. 返回结果中需要包含 totalPages和XXX两个参数，分别表示总页数和总数据量。

## 返回数据规范

本部分旨在说明接口返回数据的规范。接口返回数据需要符合以下规范:

1. 返回数据必须为 JSON 格式。
2. 返回数据中需要包含code和 message 两个参数，分别表示接口返回码和错误信息。
3. 返回数据中需要包含 data 参数，表示接口返回的数据内。
4. 如果接口返回的数据为空，data 参数的取值为 null。

# 接口文档规范

本章旨在说明接口文档的规范。接口文档需要包含以下内：

1. 接口名称和接口地址。
2. 接口请求参数和请求参数说明。
3. 接口返回数据和返回数据说明。
4. 接口使用示例。
5. 接口更新记录。

# 接口管理

本章旨在说明接口管理的规范。接口管理需要遵循以下规范：

## 接口分类、编码排序

在接口管理中，需要对接口进行分类和编码排序，以方便查找和管理。可以按照功能模块、业务类型等方面进行分类，同时给每个接口分配一个唯一的编码，方便快速定位。

## 在线文档

为了方便开发人员查阅接口文档，可以将接口文档制作成在线文档，方便查找和管理。在线文档可以包括接口详情、参数说明、返回值等信息，方便开发人员快速了解接口的使用方法和返回值。