接口测试用例设计

1.接口测试过程中问题反馈：

1. 传入参数处理不当，导致程序crash；
2. 类型溢出，导致数据读出和写入不一致；
3. 因对象权限未进行校验，可以访问其他用户敏感信息
4. 状态处理不当，导致逻辑错乱
5. 逻辑校验不完善，可利用漏洞获取非正当利益
6. 接口测试用例设计

### 1.针对输入设计

对于接口来说，输入就是入参。常见参数类型有:

（1）数值型(int,long,float,double等）

（2）字符串类型

（3）数组或链表

（4）结构体

1.1数值类型

等价类：1.取值范围内2.取值范围外

边界法：1.取值范围边界

特殊值：1.0，负数等等

遍历法：1.取值范围所有数值遍历

例如检查权限的接口：TaskChecker.checkTask(int taskID) taskID的取值范围是1-35，那么设计时考虑：

1-35范围内和范围外的值；

1-35的边界：0，1,35,36；

类型的特殊值：-1,0

数据类型的边界值：int的最小值最大值；

因为1-35代码的权限ID不同，可能需要遍历1-35的每个值。

常见问题和风险：

1. 特殊值处理不当导致程序异常退出；
2. 类型边界溢出
3. 取值范围外值未正确返回错误信息

1.2 字符串型

字符串长度

1. 等价类：取值范围内，取值范围外
2. 边界值：String类型的最大长度
3. 特殊值：0，即空字符串

字符串内容

1. 特定类型：英文，中文，大小
2. 特殊字符：,.><?/”’%#%^@\*()\_)等
3. 敏感字符：如法轮功等

例如接口转换设置闹钟的接口DateUtil.getDayOfDDHH(String ddhh)，用例可以考虑：

长度为4位，比4位少，比4位多；

边界值：String的最大长度；

特殊值：空字符；

字符串内容可考虑类型：数字，非数字；

特殊字符。

如果是输入用户输入且其他用户可见的内容，则还需要考虑敏感字是否被正常过滤。

可能出现的问题和风险：

传入非特定类型程序异常退出

超长字符未进行处理，导致存储、显示等异常

其他用户可见设置的敏感字

1.3 数组或链表类型

1. 数值或链表
2. 成员个数：等价类：等价范围内，边界法：规定边界，类型边界，特殊值： 0等
3. 成员内容：重复数据，非法数据

例如批量提交任务的接口submitTask(int[] taskID)，参数用例设计考虑：

正常取值：1-5个权限，范围外：6个权限；

边界值：1-35的边界值，请求允许最大最小值；

特殊值：0个；

合法ID和不合法的；

重复的ID等。

可能存在的问题和风险：

0个item时程序异常退出；

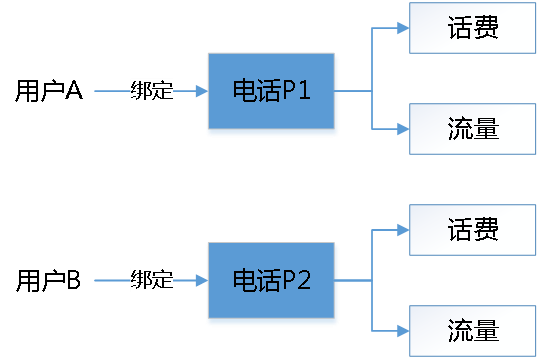
重复的item处理时未去重导致结果异常等。

### 2.按照逻辑进行设计



1. 时间约束：22：00之前
2. 数值约束：积分200；限量5个
3. 状态约束：登录手机管家
4. 权限约束：管理员

2.1操作对象分析



对象分析主要是针对合法和不合法对象进行操作。例如下述用例子:

用户A查询电话P1话费：

用户A查询电话P1流量；

用户A查询电话P2话费；

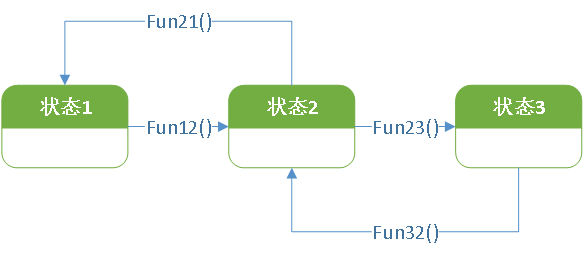
用户A查询电话P2流量。

常见的问题和风险：

* 用户可访问非权限内的其他用户信息、敏感信息，从而利用这些信息谋取利益。

2.2 状态转换分析

被测逻辑可以抽象成状态机，各个状态之间根据功能逻辑从一个状态切换到另一个状态。如果我们打乱了这个次序，从一个状态切换到另一个不在它下一状态集中的状态，那么逻辑将会打乱，就会出现逻辑问题。



那么可以这样设计：

（1）正常的状态切换：未领取状态，领取任务后变为已领取状态；已领取满足任务条件提交后，变成已完成状态；完成后可以再次领取任务。

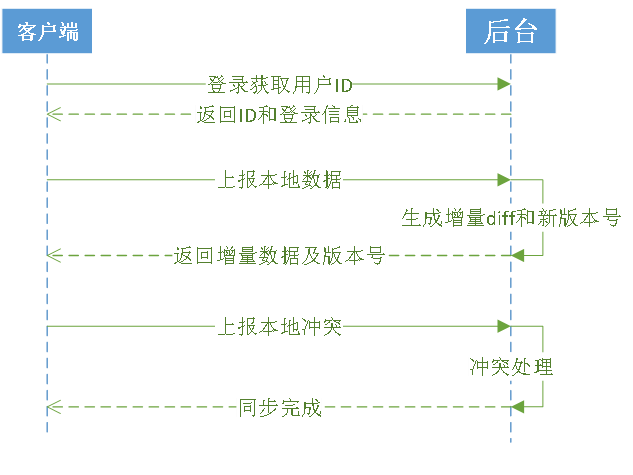
（2） 非正常的状态切换：未领取任务满足任务条件直接提交任务；已领取时再次领取任务等等。

2.3时序分析

在一些复杂的活动中，一个活动是由一系列动作按照指定顺序进行的，这些动 作形成一个动作流，只有按照这个顺序依次执行，才能得到预期结果。

在正常的流程里，这些动作是根据程序调用依次进行的，并不会打乱，在接口测试时，需要考虑如果不安装时序执行，是否会出现问题。

例如，客户端数据同步是由客户端触发进行的，期间的同步用户无法干预。功能测试的时候可见的就是是否能正常进行同步，而进一步分析，同步流程实际涉及了一组动作。



常见的问题和风险：

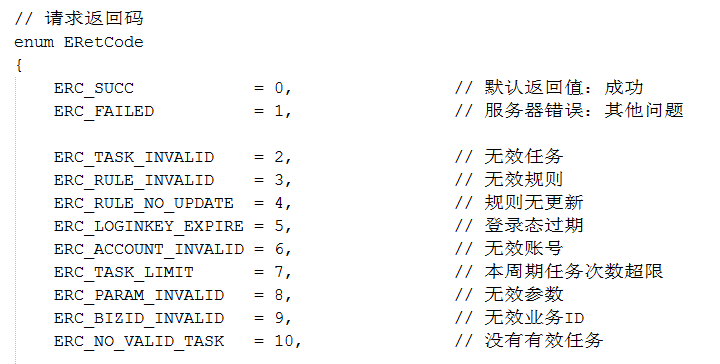
非顺序执行后，数据出现异常，可能还会出现程序其他异常

通过打乱顺序获取利益

### 2.3针对输出设计

接口处理正确的结果可能只有一个，但是错误异常返回结果有很多情况很多值。如果知道返回结果有很多种，就可以针对不同结果设计用例。例如提交积分任务的时候我们通常能想到的是返回正确和错误，错误可能想到：无效任务，无效登录态，但是不一定能否完全覆盖所有错误码，而接口返回定义的返回码可以设计更多用例：





覆盖返回码也是用例设计的一种思路。

常见问题和风险：

（1）错误前端处理不足，导致前端异常；

（2）错误提示处理不当，导致用户看到晦涩的错误码；

（3）错误提示不当，导致用户不知道哪里出了问题，如何解决。

****2.3.2 接口超时****

接口正常情况下是有返回的，那么如果接口不返回呢？也就是说接口超时后的处理也是测试需要考虑的部分。如果超时处理不当，可能会引起以下问题：

（1）未进行超时处理，导致整个流程阻塞

（2）超时后又收到接口返回，导致逻辑出现错乱

## **2.4 其他测试设计**

****2.4.1 已废弃接口测试****

已废弃协议，是指之前有定义，但是因为需求变更或其他原因，目前版本不用。这些接口虽然不再使用，但有可能代码并没有及时删除。如果利用技术手段调用这些接口，可能获取额外利益。

例如，任务之前有个清理任务，在一个版本需求里将清理任务替换为下载任务。在新版本客户端已不再调用完成清理任务的接口；但是如果该接口未关闭，用户就可以继续请求submitTask(int taskID)接口完成清理任务获得积分。

因此新版本在考虑兼容旧版本的同时，还应做好相关废弃接口的检查，避免用户获得额外利益。

****2.4.2 接口设计合理性分析****

接口定义是否合理可以从以下几个方面分析：

（1）接口字段是否冗余；

（2）接口是否冗余；

（3）接口是否返回了调用方期望得到的信息；

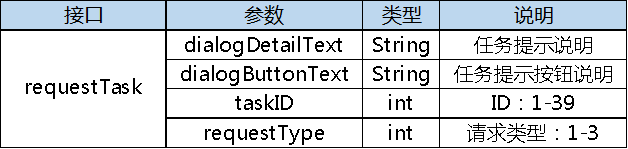
（4）接口定义是否可满足所有调用需求；

（5）接口定义调用是否方便。

具体例子：

下面举一个完整例子，通过上述方法来分析如何对接口进行用例设计。

某模块提供了一个接口给其他模块，用户请求任务，接口定义如下：

[](https://i.imgur.com/TVhT73y.png)

****2.5.1 针对输入设计****

dialogDetailText（dialogButtonText类似）

长度

1）正常：请安装提示进行操作；

2）边界：一个字：请；长度非常长：无悬浮窗权限，可能影响XX功能无法使用，请开始悬浮窗权限，以便获得更好的用户体验；甚至更长；

3）特殊：空字符串。

内容

1）特定类型：中文，英文，数字等；

2）特殊字符：/n/r/t ,.><?\*$&^%~"ஜღ℡♬€✎等；

3）敏感字符：非用户设置，不涉及。

taskID（requestType类似）

等价类

取值范围内：1,5,10等；

取值范围外：0，99。

边界法

取值范围边界：0,1,38,39,40

数据类型边界：-2147483648 ,2147483648

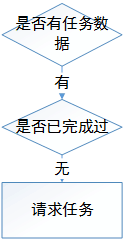
特殊值：0,-1等。

遍历法：1,2,3,4,5…38,39对应每种不同ID。

****2.5.2 针对逻辑设计****

（1）约束条件分析

去引导某功能需要：未完成过任务，任务有任务数据。

[](https://i.imgur.com/uHramoR.png)

那么用例可以是：以下情况下调requestTask：

1）未使用过有任务数据时；

2）未使用无任务数据时；

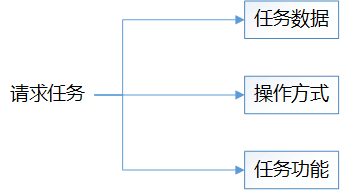
3）使用过有任务数据时；

4）使用过无任务数据时。

如果有其他约束条件类似设计。

（2）操作对象分析

调用请求接口后，会显根据任务数据，引导对应的任务。任务数据，任务操作方式，任务功能都可以是对象。

[](https://i.imgur.com/UfOJIkc.png)

1）任务数据

数据类型：本地，云端 等

数据有效性：正确数据，错误数据

2）操作方式

方式：安装，下载，打开等等 。

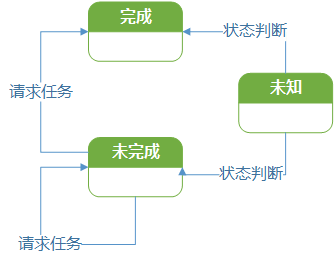
3）任务功能

功能：用户操作了该功能，未正常操作该功能；什么都不操作；完成一个任务功能；完成多个任务功能；任务功能使用顺序等等。

对象：还需要关注，会不会操作到不合法的对象，例如任务数据和功能不对应等问题。

（3）状态转换分析

功能是有4个状态的：完成，未完成，未知。状态图如下：这里是产品里涉及的状态转换：

[](https://i.imgur.com/fouRlIu.png)

针对该状态：

1）正常状态转换：未完成状态请求并完成任务后是否可变成完成状态；未完成状态请求但不完成，还是未完成状态。

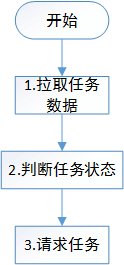
2）走不到的状态路径：未知和完成状态请求任务，不能进行进行该任务。

（4）时序分析

从时序的角度分析，调用请求接口前需要以下2步动作：

1）拉取任务数据；

2）判断任务状态。

[](https://i.imgur.com/0TxF0ZY.png)

从时序得到的用例有：

正常时序：按照正常时序请求1 2 3；

缺失的时序

缺少动作1调2 3；缺少动作2调1 3；缺少动作1和2直接调。

打乱的时序

打乱的时序：2 1 3，还可以有1 3 2，2 3 1，3 1 2，3 2 1。

针对处理逻辑的设计中，可能使用某一种或某几种方式就可以将用例覆盖前，故实际使用中，可能不会全部使用，只要找到最合适的方式覆盖用例即可。

****2.5.3 针对输出分析****

请求任务接口返回的数据是任务完成结果，即返回完成，未完成两种状态（未知都作为完成返回）。

从结果可以考虑遍历：

1）未完成

2）完成

3）完成-未知

从接口处理时间分析，考虑：请求后快速返回，很长时间才返回，甚至不返回结果的情况。

上下游接口的数据依赖无非就是准备测试数据。

假如一个事务需要顺序调用3个接口，A B C， C依赖于AB， 而AB有数据依赖，这时候就需要准备好A和B的数据。

数据一般有两种方式生成

动态方式：假如B依赖A创造的数据，那么**每次执行B之前必须执行A去做数据创建**

静态方式：独立统一的**测试数据库**, ABC需要的数据都可以从库里拿到