多元产品的基于Python的测试自动化方案



目录

目录 2

1. 概述 3

1.1 测试自动化架构 3

1.2 接口测试的范围界定 4

1.3 性能测试 6

1.4 测试用例生成工具 6

1.5 加解密服务器 6

1.6 其它 7

2. 服务测试方案 7

2.1 范围界定 7

2.2 测试方案 7

2.3 接入服务（server\_busproxy）的测试方案 8

2.4 二级服务的测试方案 9

2.5 行情子系统的测试 10

2.6 如何获取服务的功能规格 10

2.7 服务的启动与监控 10

3. 接口系统测试 11

# 概述

## 范围界定

广义的接口测试包括系统测试的接口测试、子系统测试和服务测试。

系统测试是以整个系统为测试对象，只是用测试用例取代真实的管理客户端和交易客户端。TAS 2.0的系统测试是手工测试先行，后续再考虑回归测试的自动化。所以接口测试目前不考虑系统测试。

子系统测试如接入子系统测试、交易子系统测试和银行子系统测试等。交易子系统测试是我们确定的接口测试重点，参见《TAS 2.0测试工作安排.xls》。测试方法用测试用例模拟前置服务，对交易子系统（委托服务+资金服务+撮合服务）进行测试。

服务测试则是以单个服务为测试对象，与该服务存在消息交互的周边服务都用测试用例模拟。目前没有计划开展服务测试。但是，如果在子系统测试中发现bug很多，因问题定位而导致测试进度很慢，则有必要考虑将测试级别再下降，开展针对单个服务的测试，以提高测试效率。

服务测试又可以分为两类：

* 总线类服务的测试：该类服务只有总线接口和数据库接口。总线上与该服务存在消息交互的其它服务都用测试用例模拟，而数据库则使用真实的。
* 存在其它接口的服务的测试：比如接入服务、前置服务、存在外部行情源接口的行情服务，存在银行接口的银行服务等。

接口测试也包括性能/压力测试。

## 测试自动化架构

TAS 2.0的接口测试，采用基于Python语言(Python 2.7)的测试自动化方案， 选用开源测试框架nose作为测试框架。测试用例可以在Windows或者Linux上执行。具体内容可以参考测试部的nose培训文档或者nose的官方文档。

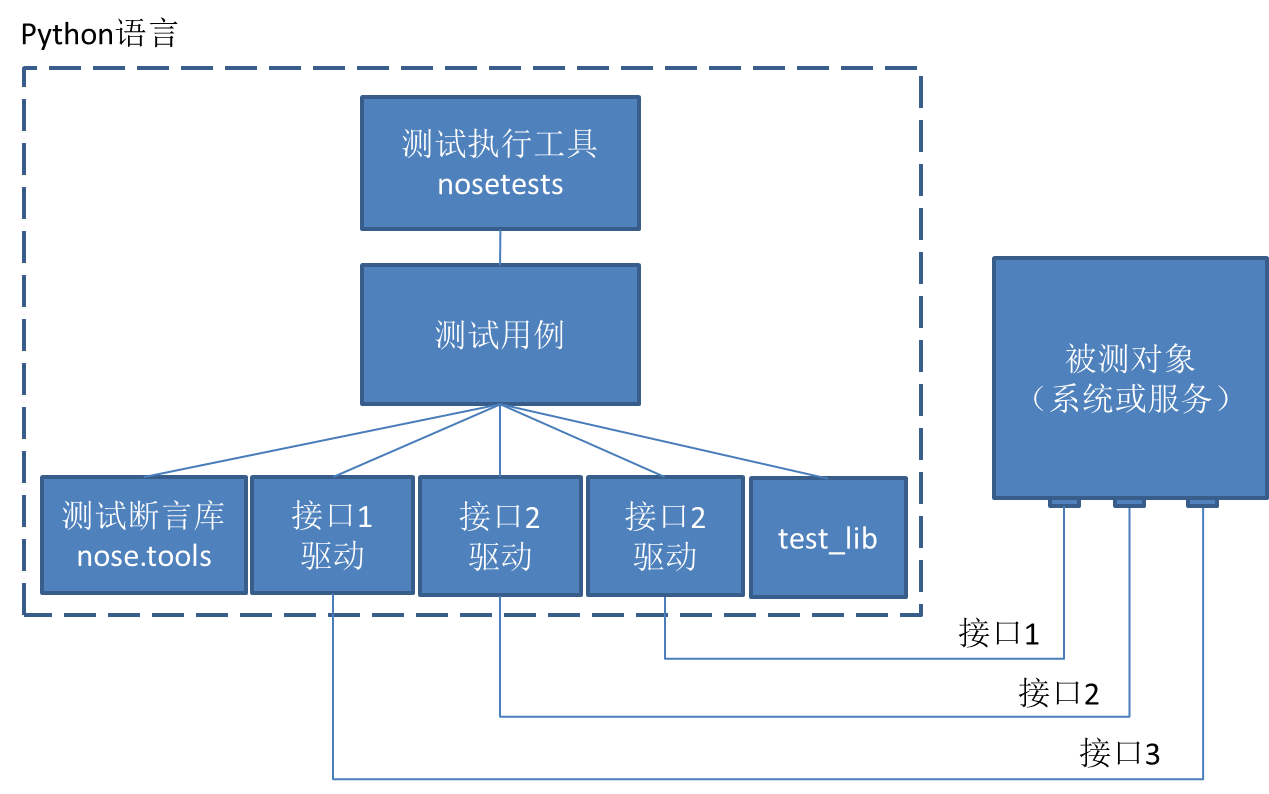


图 1.1 测试自动化工具的架构

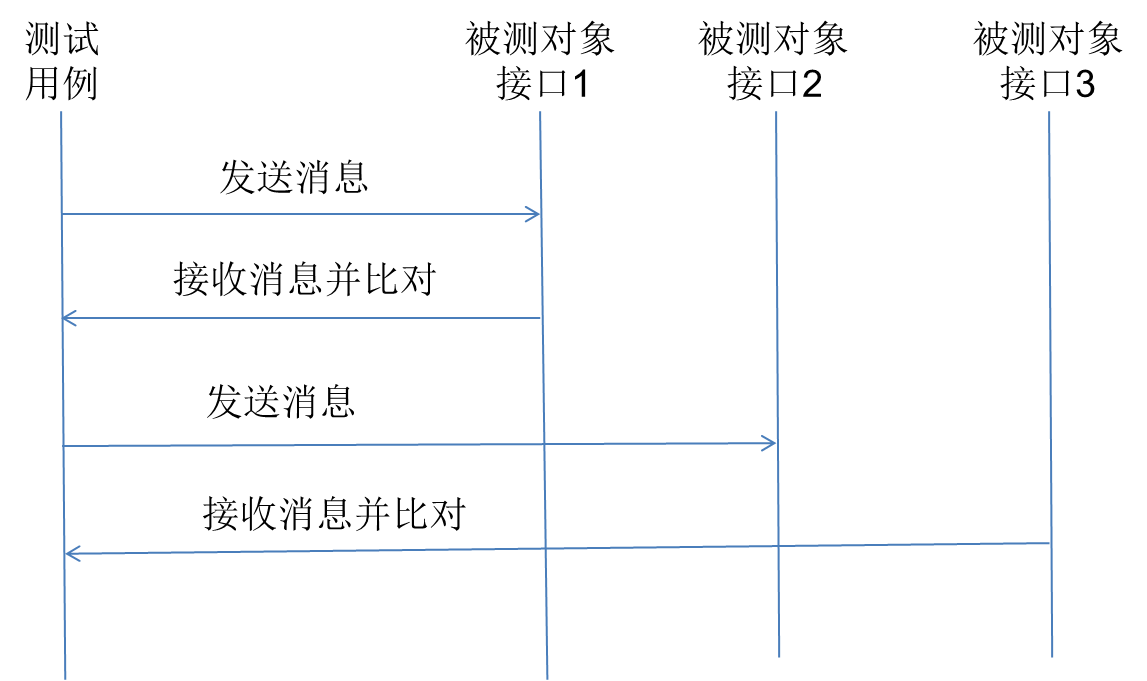


图 1.2 测试用例以顺序方式进行消息发送和接收

在上述框架之下，选择或自行开发下列测试工具：

* 在Haigha（Python语言的AMQP client，RabbitMQ官方推荐）基础上封装RabbitMQ的测试驱动。
* 选用Google ProtoBuffer官方工具（支持Python）作为protobuf消息的编解码器工具。
* 在公司MISM模块的基础上封装TCP接口的消息加解密服务器。
* 对于TAS 2.0系统的其它私有接口，今后开展相应测试前，再自行开发其测试驱动。

## 性能与压力测试

在TAS 1.0版本上已经开发出Python语言的服务性能测试工具（消息总线接口），稍加修改即可用于TAS 2.0的服务测试。

## 防御性编程与契约式设计

接口测试一般包括异常输入测试，这就涉及各服务/组件之间的接口保护应该做到什么程度的问题，到时候开发和测试可能会为此出现争议。TAS 2.0的业务目标（例如交易所联盟）对软件质量提出了更高的要求，所以应该引入契约式设计，或者至少将防御性编程作为接口设计的一个设计原则明确下来。

# 交易子系统的测试方案

## 测试方案概述

测试对象包括委托服务、资金服务和撮合服务。前置服务用测试用例模拟。如下图所示：

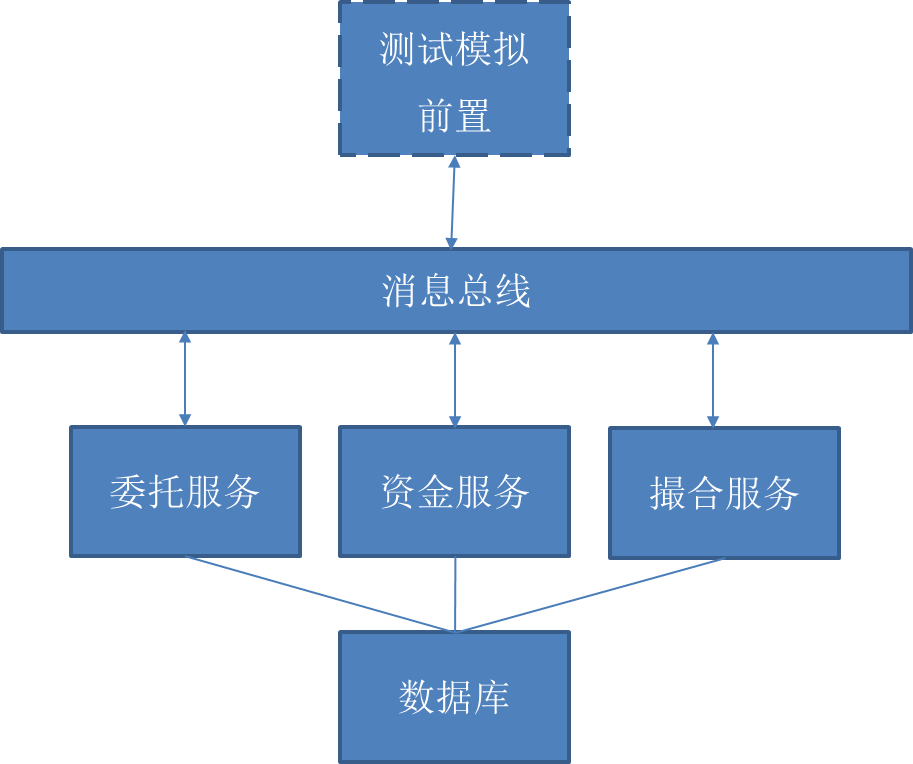
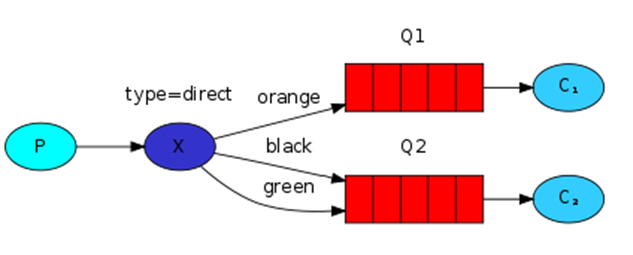


图 2.1 交易子系统的测试

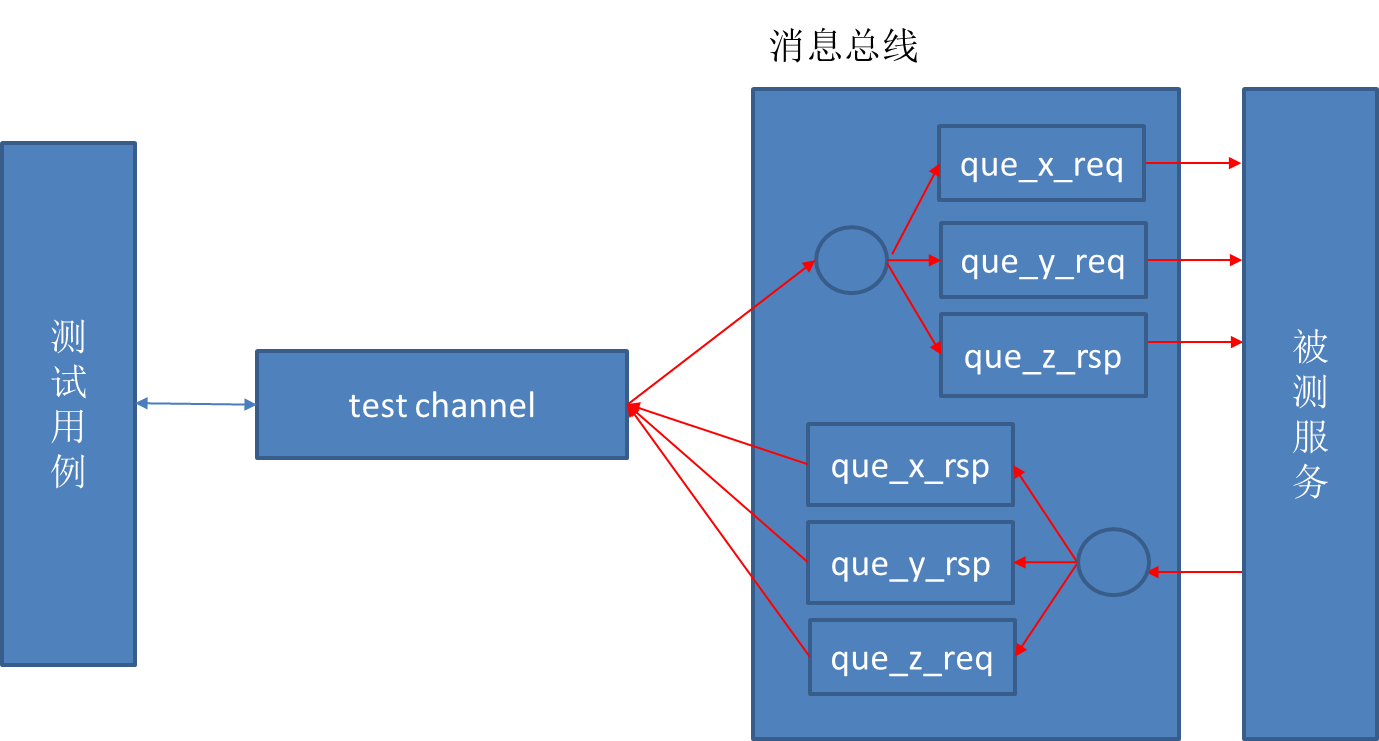
由于没有配套的管理子系统，数据库的数据通过PLSQL直接构造。

## 测试方案

RabbitMQ有多种使用方式，TAS产品使用的是其中的direct方式。在Direct方式下，服务通过消息总线发送消息，只需要使用特定的路由键（Routing key）作为路由标识。同样，服务通过消息总线接收消息，也只需要将特定的路由键绑定一个已存在的消息队列(Queue)或者自己新建的消息队列。消息的发送者和接收者之间已经被消息总线隔离，被测服务只需要知道路由键和消息队列，而并不知道消息的接收者和发送者是谁。



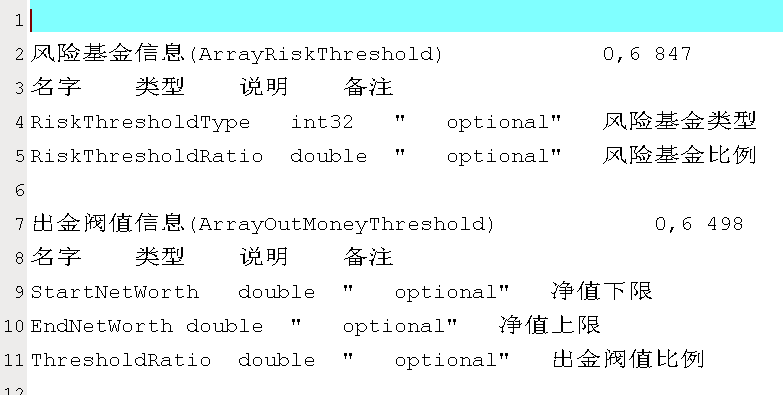
所以，在做服务测试的时候，测试用例并不需要真的去模拟各个周边服务（比如该服务有几个连接，每个连接上有几个channel，每个channel发送或接收那些消息等）。在使用Haigha client的情况下，测试用例只需要一个channel，既可以用这个channel向其它服务发送消息，也可以在这个channel上用routing\_key绑定接收消息的消息队列。



## 测试前的准备工作

获取proto文件并编译为pb2文件。

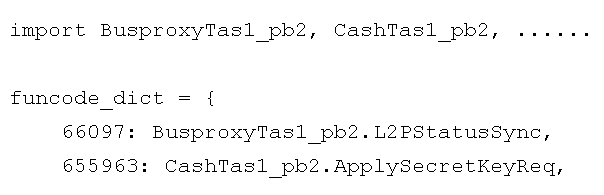
获取待测版本的《TAS一级系统接口文档.xls》，将其中“全部接口”页的内容全部拷贝至一个txt文件，删除标题行，变成如下格式：



再用一个工具(gen\_funcode\_dict.py)将上述txt文件转换为funcode.py：

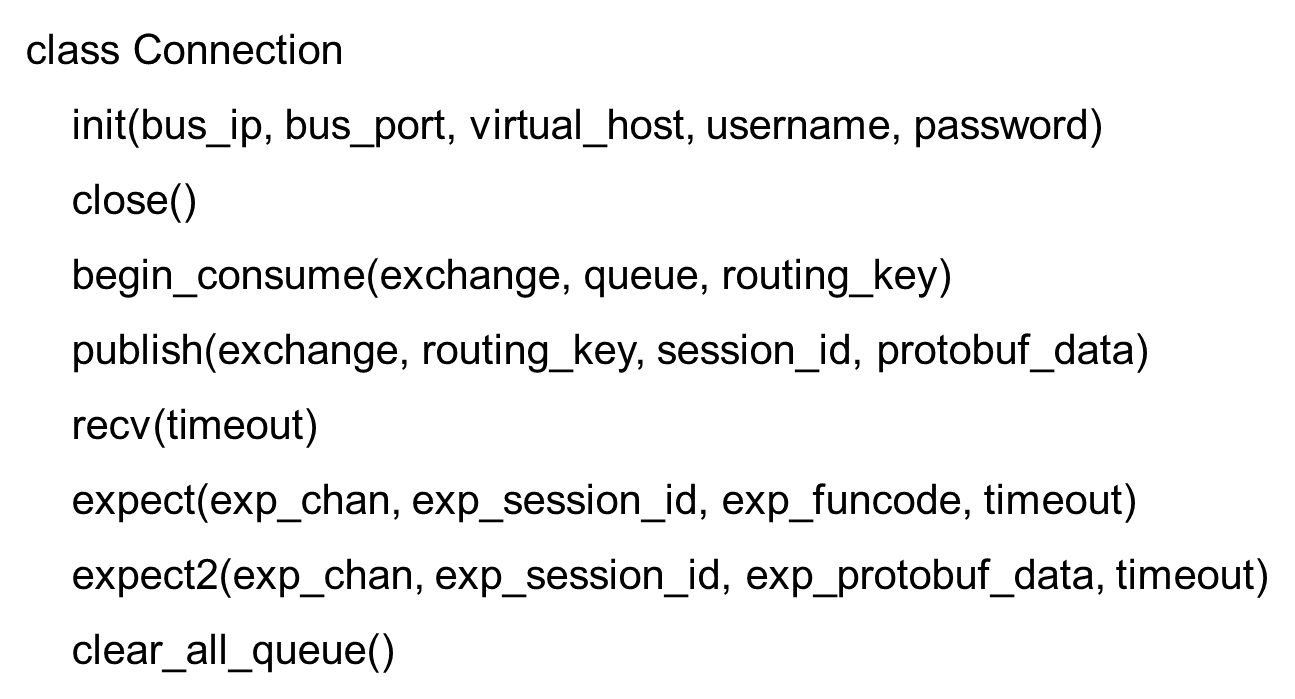
D:\tas2\tools>python gen\_funcode\_dict.py level\_1\_service\_interface.def>temp\_funcode.py

生成的funcode.py如下：



## 消息总线驱动的接口设计

该驱动是在pika（AMQP的Python语言client）之上的封装，为测试用例提供了方便易用的接口。



其中，expect()还会检查收到的响应消息，看其RetCode是否等于0。

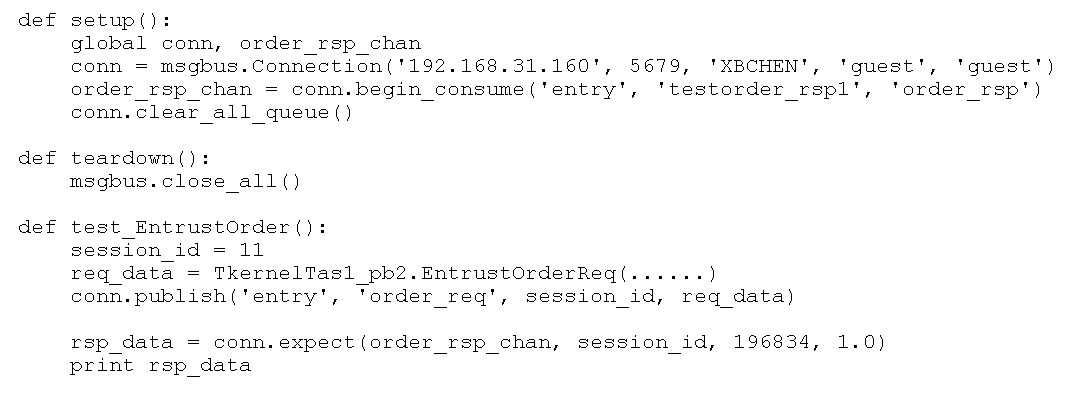
该驱动在TAS 1.0上已经通过实际测试。

## 测试数据的赋值

编写测试用例时，工作量最大的部分应该是测试数据的构造。而测试用例代码里行数最多的也应该是测试数据的赋值语句。测试数据的赋值使用google.protobuf提供的数据接口，范例如下图所示：



## 测试用例范例



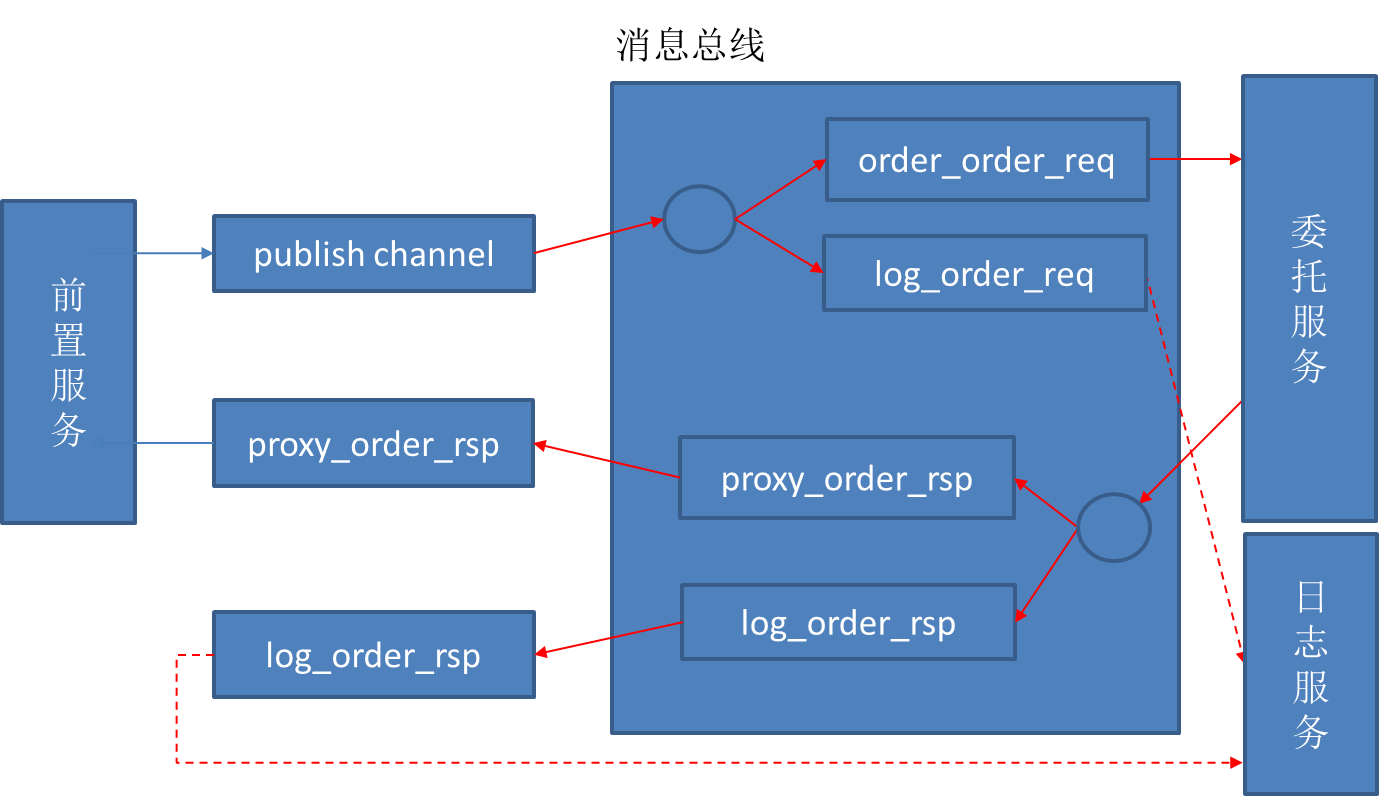
其中的数据赋值语句的具体内容，参看上一节。

## 服务的启动与监控工具

如果TAS 2.0的服务日志能够提供足够多的信息，那么后续就可以编写服务的启动与监控工具。就是在测试用例之外，编写单独的Python脚本来启动被测服务，并启动成功后对其进程与日志进行持续的监控。

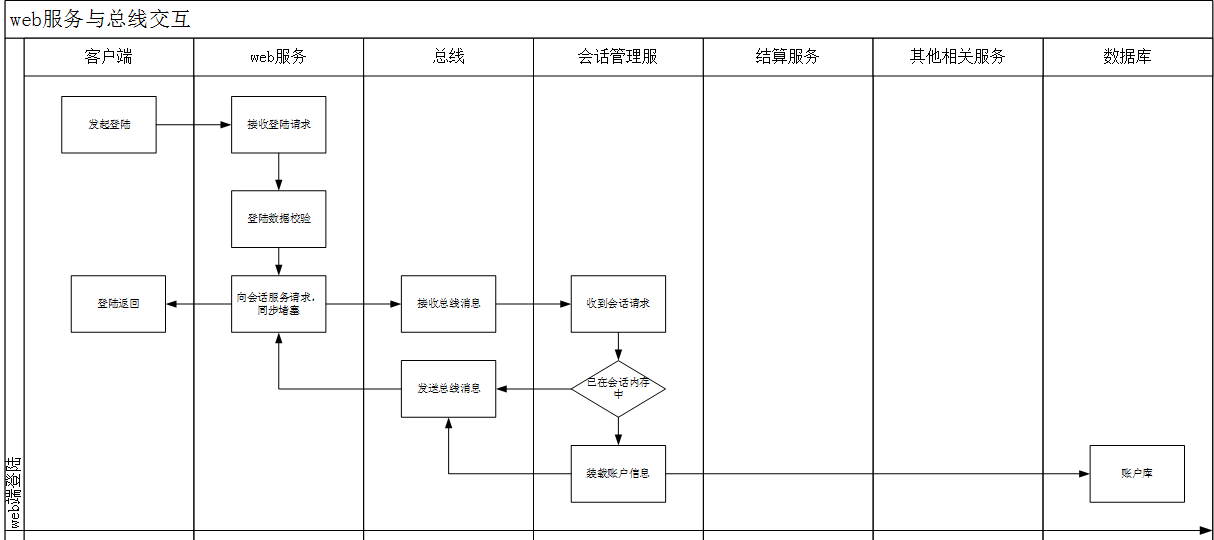
## 消息总线的清理

由于存在多路分发的情况（对于一个路由键，有多个接收消息队列，如下图所示），而被测服务或者测试用例只会读取其中一个消息队列。另外，各个服务会发出心跳消息。如果长时间不做清理，这些没有接收者的消息队列会越来越长，大量占用甚至耗尽RabbitMQ所在服务器的内存。所以，每次测试执行之后，teardown里应该自动进行消息总线清理。



## 服务测试需要的输入

1. 各个业务的消息交互图，如下图（Tas2.0总线交互-1.0-黄颖天）所示：



1. protobuf消息定义文件
2. protobuf消息与消息队列的对应关系
3. funcode与protobuf消息的映射关系

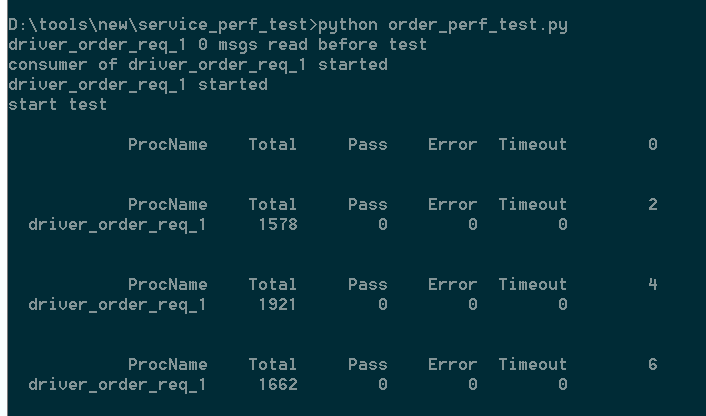
# 总线服务的性能测试

## 概述

总线服务的性能测试工具，用Python开发。考虑到工作量的关系，界面采用了简单的命令行窗口。

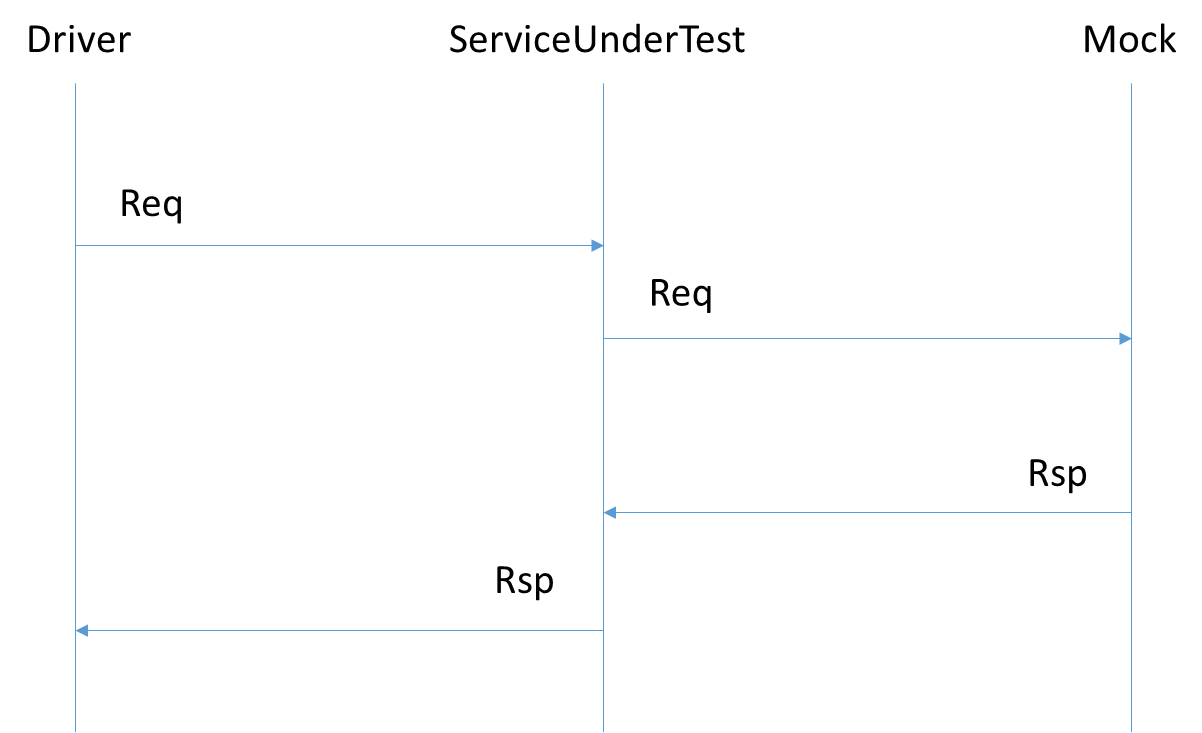
为了规避Python GIL（全局解释器锁）对性能的影响，采用了多进程的设计，并且采用multiprocessing库以简化代码。

编写服务的性能测试用例，只需要很少的Python知识和基本的RabbitMQ知识。



## 工具设计

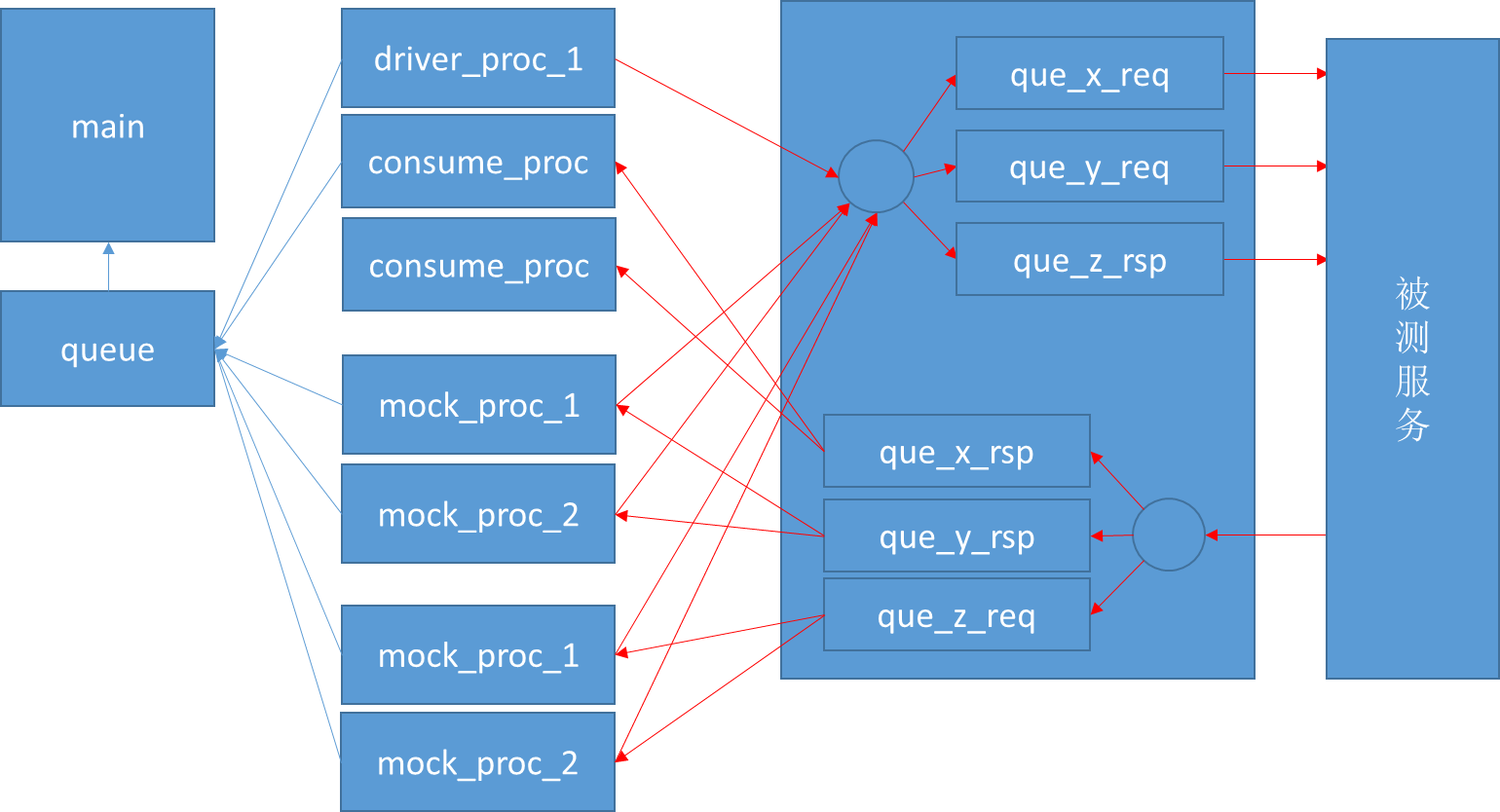
主进程负责启动mock进程、driver\_consumer进程和driver主进程，并在性能测试过程中持续接收、处理和输出测试报告。



driver主进程负责向被测服务发送请求消息，这些请求消息是预先编码过的二进制消息。

driver\_consumer进程负责接收来自被测服务的响应消息，并检查其RetCode是否为0。

mock进程则负责接收来自被测服务的请求消息并作出响应。



Driver主进程、Driver\_consumer进程和Mock进程都独立向主进程发送测试数据，频度固定为每秒一次。

为了保证性能，可以启动多个mock进程来处理单个请求消息队列的输出，也可以启动多个driver\_consumer进程来处理driver的单个响应消息队列。

## 性能测试用例范例

## 存在问题

实测发现pika用在性能测试上的时候，表现不佳，推测是因为同步接口在消息发送和接收之间存在相互锁定。

后面进行服务的性能测试时，打算试用RabbitMQ官方推荐的另一个python client：Haigha。Haigha的接口与pika基本相同（所以性能测试工具改动不多），只是工作方式是异步的，预计性能更好，而且这对于写测试用例来说更加方便。

# 其它服务的测试方案（略）

## 二级服务



## 前置服务

## 银行服务

## 行情服务

# 系统测试之接口测试（略）

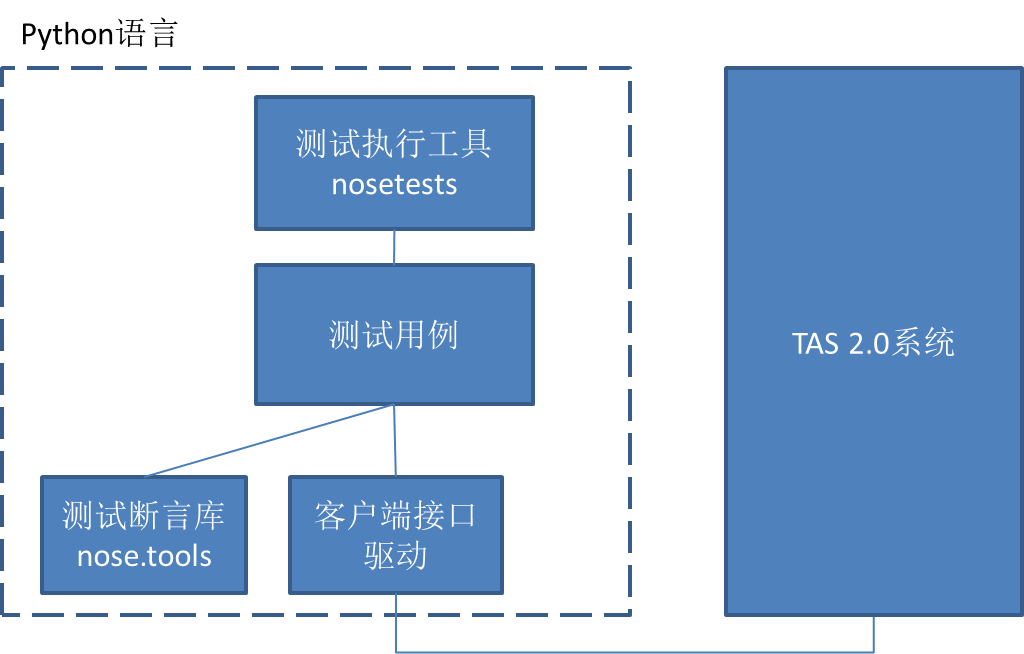
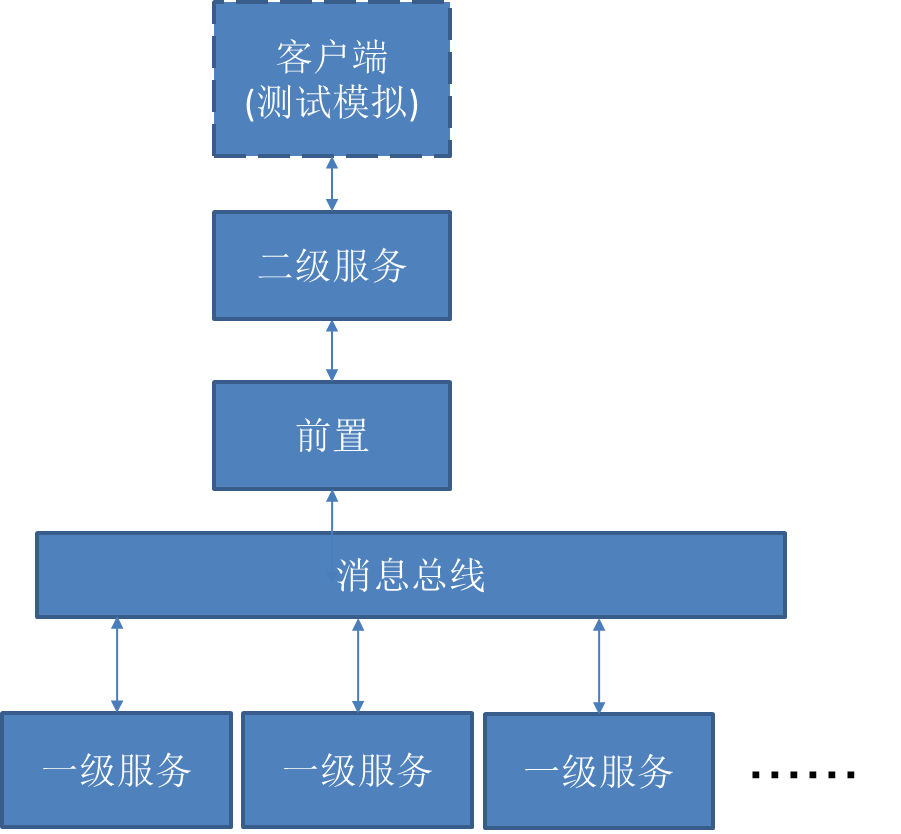
## 目的

以更为稳定的消息接口代替易变的GUI，减少测试用例脚本的维护成本，提高测试执行速度。

主要测试接口：

* 交易客户端消息接口(交易)
* 交易客户端消息接口(行情)
* 管理客户端消息接口。
* 外部行情源接口
* 银行接口

## 测试工具架构



## 客户端接口驱动

## 测试用例生成工具

在TAS 1.0版本上已经开发出管理客户端的测试用例生成工具。预计做少量修改后可以用于TAS 2.0。这样，系统测试人员就可以用录制-编辑-执行的方式来实现接口测试自动化，显著降低转向Python自动化测试的学习成本。

## 加解密服务器

已经将TAS 1.0的MISM模块封装为独立的加解密服务器，如下图所示。由于TAS 2.0的MISM模块与TAS 1.0接口兼容，所以今后模拟客户端做测试的时候，简单用TAS 2.0的MISM代码替换TAS 1.0的，再重新编译即可。

