My Document, My Way

Chen Jin Fen

2024年9月20日

文档信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 撰写人 |  | 日期 |  | 页码 | 共 X 页 | 版本号 |  |
| 审核 | | | | | | | |
| 用户代表 |  | 签字： | | | 年 月 日 | | |
| 部门内部审核 | 详见《单元测试总结报告》 | | | | | | |
| 审批 | | | | | | | |
| 项目经理/部门经理 |  | 签字： | | | 年 月 日 | | |

目录

[1 前言 3](#_Toc177895749)

[1.1 目的 3](#_Toc177895750)

[1.2 范围 3](#_Toc177895751)

[1.3 定义、缩写词 3](#_Toc177895752)

[2 测试原则 4](#_Toc177895753)

[3 测试方法 5](#_Toc177895754)

[4 测试过程 7](#_Toc177895755)

[4.1 星盟发送DU报文到深航流程 7](#_Toc177895756)

[4.1.1 parseSubMessage节点 7](#_Toc177895757)

[4.2 深航发送DU报文到星盟流程 8](#_Toc177895758)

[4.2.1 parseASM节点 8](#_Toc177895759)

[4.2.2 parseMVT节点 8](#_Toc177895760)

[4.2.3 parseDIV节点 9](#_Toc177895761)

[5 测试总结 10](#_Toc177895762)

# 前言

## 目的

本文档记录说明深航FLIFO系统的单元测试的测试原则、测试方法、测试过程和测试总结，以作为后续进行系统 开发测试的依据，同时也为相关人员评审、验收项目提供依据。

## 范围

本文档主要说明深航FLIFO平台中DU模块的单元测试总结报告，覆盖DU模块的各个消息流中的业务处理相关节点。

## 定义、缩写词

本文档涉及到的名词缩写说明如下：

FLIFO

Flight Information

EDI

Eletric Data Interchange

EDIFACT

电子数据交换的标准协议

Q/R

Query and Response

DU

Dynamic Update

IFOC

航班运行信息管理系统

PADIS

PASSENGER AND AIRPORT DATA INTERCHANGE STANDARD

ZH

深航缩写代码

TravelSky

航信

SkyAlliance

星空联盟

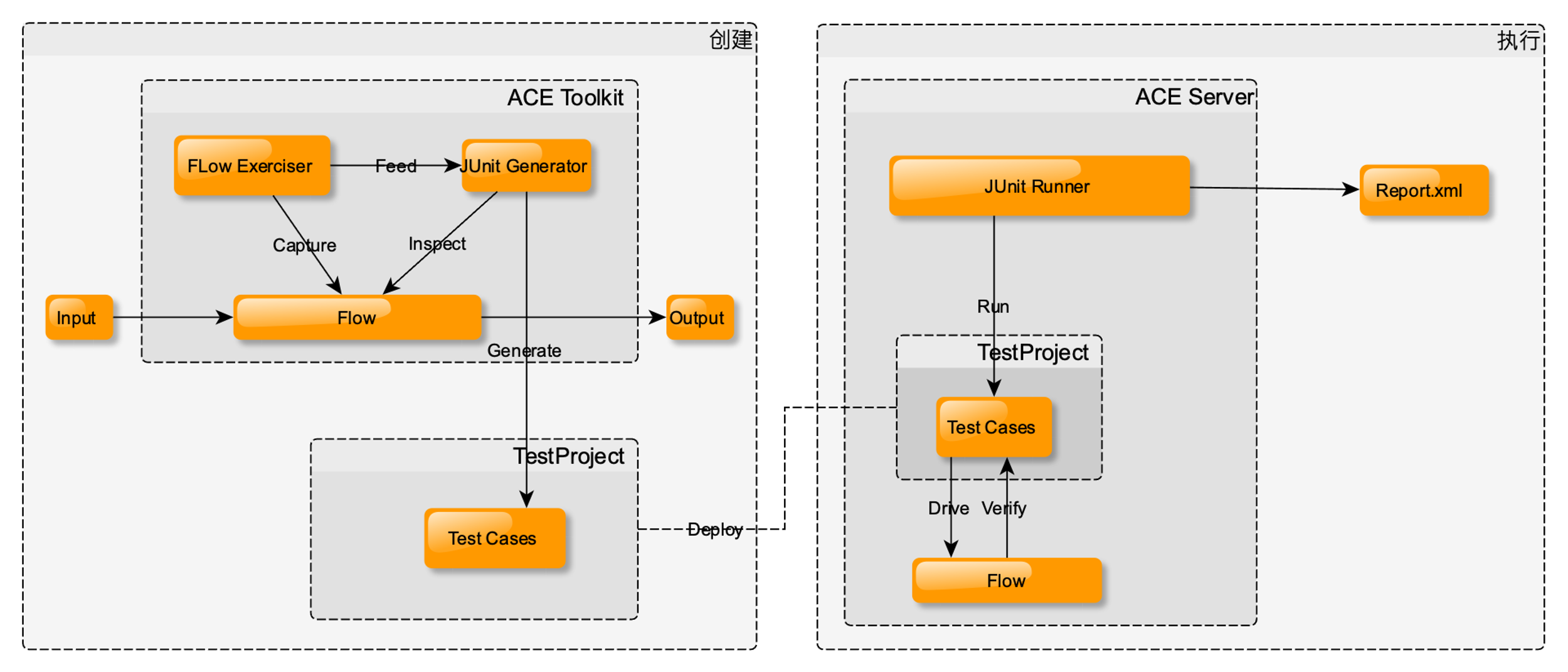
# 测试原则

在开发FLIFO系统过程中的单元测试会遵循以下原则：

* 采用业界主流的测试框架，例如junit
* 遵循业界主流的最佳实践
* 针对ACE产品的特殊性，对于ACE提供的已封装好的组件不进行单元测试， 只对消息流中具有业务处理逻辑的节点进行单元测试

# 测试方法

FLIFO系统的单元测试采用基于junit框架的方法，具体方法过程如下图所示：



1. DU星盟发送报文到深航

上图各个步骤说明如下：

1. 用ACE的开发工具toolkit打开需测试的消息流，点击消息流流程图左上角的 Flow Exerciser的最左边按钮，启动Flow Exerciser
2. 触发消息流
3. 点击Flow Exerciser右边第二个show path按钮，获取流经消息流各个节点的 数据
4. 检查需测试节点的输入数据和输出数据，确保其符合功能需求
5. 右键点击需测试节点，在弹出菜单中选择 generate test cases
6. 在弹出的对话框中选择捕获的数据作为test case的输入和输出
7. 重复以上步骤对消息流中的需测试节点生成test case并保存在一个Test Project 中
8. 把生成的Test Project部署到一个独立的ACE server的工作目录下，运行以下命令 执行test project中的test cases并生成报告：

* IntegrationServer -w <path-to-testing-ACE-server-working-dir> \  
   --mq-queue-manager-name dummy \  
   --start-msgflows false \  
   --no-nodejs \  
   --admin-rest-api -1 \  
   --test-project <name of the Test Project> \  
   --test-junit-options "--reports-dir=<path-to-the-junit-report-output>"

# 测试过程

按照上一章节说明的测试方法，对DU模块中的包含业务处理逻辑的节点进行测试， 过程如下。

## 星盟发送DU报文到深航流程

星盟发送DU报文到深航流程如下图所示，整个流程中只有parseSubMessage节点具有 业务处理逻辑的代码，其余是主要ACE提供的封装组件或简单的变量值传递节点，所以 该消息流对 parseSubMessage节点进行单元测试。

### parseSubMessage节点

#### case 1 - 正常ASM消息

1. 正常ASM消息测试案例

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 1 | 发送正常ASM消息 | 正常的ASM报文 | 文件输出目录为IB\_ASM，文件名为ASM\_yyyymmddhhMMss\_xxxx.txt | 与期望符合 | 通过 |

以下是junit执行结果：

#### case 2 - 正常MVT消息

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 2 | 发送正常MVT消息 | 正常的MVT报文 | 文件输出目录为IB\_MVT，文件名为MVT\_yyyymmddhhMMss\_xxxx.txt | 与期望符合 | 通过 |

以下是junit执行结果：

#### case 3 - 正常DIV消息

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 3 | 发送正常消息 | 正常的DIV报文 | 文件输出目录为IB\_DIV，文件名为DIV\_yyyymmddhhMMss\_xxxx.txt | 与期望符合 | 通过 |

以下是junit执行结果：

#### case 4 - 无效消息

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 4 | 发送无效ASM消息 | 无效的ASM报文 | 文件输出目录为IB\_ERROR，文件名为ERROR\_yyyymmddhhMMss\_xxxx.txt | 与期望符合 | 通过 |

以下是junit执行结果：

## 深航发送DU报文到星盟流程

深航发送DU报文到星盟流程如下图所示，整个流程中只有parseASM/parseMVT/parseDIV节点 具有业务处理逻辑的代码，其余是主要ACE提供的封装组件或简单的变量值传递节点，所以 该消息流对 parseASM/parseMVT/parseDIV节点进行单元测试。

### parseASM节点

#### case 1 - 正常ASM报文

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 1 | 发送正常ASM消息 | 正常的ASM报文 | GlobalEnvironment中的MSG\_FORMAT为ASM，无异常 | 与期望符合 | 通过 |

以下是junit执行结果：

#### case 2 - 无效报文

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 2 | 发送无效ASM消息 | 无效的ASM报文 | GlobalEnvironment中的MSG\_FORMAT为空，抛出异常 | 与期望符合 | 通过 |

以下是junit执行结果：

### parseMVT节点

#### case 1 - 正常MVT消息

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 1 | 发送正常MVT消息 | 正常的MVT报文 | GlobalEnvironment中的MSG\_FORMAT为MVT，无异常 | 与期望符合 | 通过 |

以下是junit执行结果：

#### case 2 - 无效报文

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 2 | 发送无效MVT消息 | 无效的MVT报文 | GlobalEnvironment中的MSG\_FORMAT为空，抛出异常 | 与期望符合 | 通过 |

以下是junit执行结果：

### parseDIV节点

#### case 1 - 正常DIV消息

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 1 | 发送正常DIV消息 | 正常的DIV报文 | GlobalEnvironment中的MSG\_FORMAT为DIV，无异常 | 与期望符合 | 通过 |

以下是junit执行结果：

#### case 2 - 无效消息

| case # | case说明 | 输入 | 期望输出 | 实际输出 | 测试结果 |
| --- | --- | --- | --- | --- | --- |
| 2 | 发送无效DIV消息 | 无效的DIV报文 | GlobalEnvironment中的MSG\_FORMAT为空，抛出异常 | 与期望符合 | 通过 |

以下是junit执行结果：

# 测试总结

测试项目共创建并执行了10个单元测试案例，模拟了各个消息流的大部分执行路径，验证了系统的功能和 需求说明相符合，为后继的系统集成测试打下良好基础。