## 测试人员Jira使用规范

一 创建功能测试用例

在jira中每个创建的test type issue是一个test case，每个test case应包含的字段内容： Summary, Priority, Linked Issues, Epic Link, Assignee, Sprint, Labels, Test Details, Attachments。

1. **Summary**

简洁清晰描述测试用例的测试功能点，推荐用测试类型和模块名称作为前缀，具体要测试的功能点，测试结果是positive normal或negative abnormal，建议不要太长，详细内容可在描述字段填写。

Template: Type\_Component\_测试点

Example: APP\_我的爱车\_车辆信息\_P用户未设置spin码新增授权

其他类似以API\_，PERF\_作为前缀。

1. **Priority**

分为Highest, High, Medium, Low, Lowest 5个优先级等级。

Highest：对核心或重要或基础功能的可用性测试，其他功能的正常运作要依赖于这些功能的测试用例集合；

High：最常执行以保证功能性是稳定的，目标的行为和能力可以正常的工作，重要的错误和边界测试的测试用例集合；

Medium：对功能的更详细全面的测试，检查功能的多方面包括边界，错误、中断、断网、容错、UI和配置的测试用例集合；

Low和Lowest：比较少被用到的功能流程，最少被执行的测试用例，性能、压力、稳定性、安全等；

1. **Description**

被测试功能点的详细描述或说明，相关功能的需求、流程、UI/UE、测试所需数据路径等。

1. **Linked Issues**

与该测试用例相关的User Story，task，bug都可以进行关联，并选择好各自之间关联的关系。

1. **Epic Link**

将该测试用例与相关的Epic ticket进行关联。

1. **Assignee**

分配给需要完成测试步骤的人员，一般是测试人员。

1. **Sprint**

必填项，表示这个测试用例是用来测试哪个Sprint中开发的功能。

1. **Labels**

必填项，申明该测试用例的测试类别，UI，API，性能，兼容性，integration，smoke，regression，not applicable等，可以同时拥有多个标签。

1. **Test Details**

Test step：Detailed atomic executable operation，包括前期数据，环境准备步骤和后期环境清理步骤；

Test data：每一个测试操作步骤需要用到的测试数据；

Expected result：每一个测试步骤执行后期望得到的结果；

Attachment：与该测试步骤相关的各种文件，测试脚本，配置文件，图片等。

1. **Automated**

该测试用例是否已经实现的自动化**。**

1. **Depricated**

Test Case 是否废弃

二 创建接口测试用例

与功能测试用例类似，其中内容稍有不同：

1. **Description**

API，URL，输入参数、类型及范围，response类型、数据和返回码等接口描述信息

1. **Test Details**

Test Steps: 测试的pre-condition，准备环境和数据的脚本文件，测试脚本中对应的api case名字及编号，还原环境的步骤和脚本。

1. **Attachments**

脚本代码。

三 创建缺陷

提交Bug的描述需包含以下内容：

1. **Summary**

简洁清晰描述出现的问题，推荐用模块名称作为前缀，bug产生场景和结果，建议不要太长，详细内容可在描述字段填写。

Example: [Charging] 充电历史接口未展示前三个月的充电历史信息

1. **Severity**

分为Block，Critical，Major和Minor四个严重性等级。

Block为系统无法执行、崩溃或严重资源不足、应用模块无法启动或异常退出、无法测试、造成系统不稳定等；

Critical即影响系统功能或操作，主要功能存在严重缺陷，但不会影响到系统稳定性；

Major即不严重的功能缺陷，界面，性能或兼容性缺陷；

Minor即易用性及建议性问题；

1. **Priority**

分为Highest, High, Medium, Low, Lowest 5个优先级等级。

Highest：即表示问题必须马上解决，否则系统根本无法达到预定的需求；

High：表示问题的修复很紧要很急迫，关系到系统的主要功能模块能否正常运行；

Medium：表示有时间就要马上解决，否则系统偏离需求较大或预定功能不能正常实现；

Low：表示问题不影响需求的实现，但影响其他使用方面；

Lowest：即问题在系统发布以前确认解决或确认可以不予解决；

1. **Description**

Pre-Condition: 非必填，如有必要设置需写明在什么情况或场景下发生

Steps: 重现bug的详细步骤

Expected Result: 期望得到的结果

Actual Result: 实际得到的结果

1. **发现阶段(stage)**

必填项，bug发现阶段，迭代中选择sprint，专项测试时选择对应选项

1. **Phone OS and 测试手机型号**

APP功能测试时使用的手机类型、具体型号和系统版本号

1. **Fixing Projects**

必填项，按照bug产生和修复方来填写，提交bug时默认选择：APP，确认为后端bug可选择：EXP，一旦确定bug root cause属于哪一层（包括cp/sp）就固定不变，不随bug的流转而改写。

1. **Bug Type**

Bug的种类，表示哪个方面出现的问题。

Functional, UI, Performance, Compatibility, User Experience。

1. **Reason**

表明在哪个阶段引入的bug。

Requirement definition, UI/UE design, system design, back end, front end, middle ware and test comprehension。

1. **Linked Issues**

与该测试用例相关的User Story，task，test case都可以进行关联，并选择好各自之间关联的关系。

1. **Components**

非必填项，填写与该缺陷产生的功能点相关的模块，便于开发快速定位和bug统计分析。

1. **Assignee**

分配给处理bug的人员，可根据bug生命周期不同阶段分配给相应人员。

1. **Labels**

非必填项，申明该缺陷的其他属性或标签，如是一个regression ，bug bash，integration bug，可以同时拥有多个标签。

1. **Fix Version**

该缺陷需要在哪个版本被修复。

1. **Affects Version**

该缺陷影响的版本，即在哪个版本中被发现。

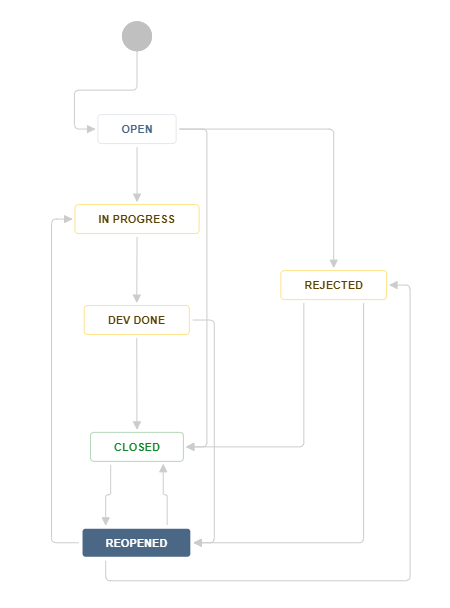
1. **Epic Link**

将该测试用例与相关的feature ticket进行关联。

1. **Sprint**

必填项，表示这个缺陷是在哪个Sprint中引入的问题。

四 Bug生命周期管理



**Open**：测试人员发现并创建bug，把bug assign给对应功能或模块的开发人员或设计人员或在不确定的情况下assign给接口人进行转发，此时bug处于open状态；

**Rejected**：开发人员或设计人员确认为非缺陷，在comment中进行说明并reject这个bug，assign给测试人员；

**In Progress**：开发人员或设计人员确认是引入的缺陷并开始解决bug，并可设置预估时间和log work time；

**DEV DONE**：开发人员或设计人员找到问题原因并解决，且经过对修复的bug本身，可能对其他模块产生影响和修复bug可能引起新的bug做基本的测试通过后，在comment中描述root cause和solution并设置bug状态为fixed，重新assign给测试人员进行验证和测试；

**Closed**：测试人员对修复的bug本身，可能对其他模块产生影响和修复bug可能引起新的bug做详细的测试通过后，在comment中描述测试过程和结果并可关闭bug；或者bug是被rejected，确认为非缺陷，可直接关闭bug；

**Reopen**：首先bug已处于dev done/rejected/closed状态，如在后续开发测试中，又产生了regression，则重新reopen该bug ticket，并assign给相关开发或设计人员进行确认，如是regression bug则进入**In Progress**阶段，如确认为非缺陷，则把bug reassign给测试人员，由测试人员关闭bug；