[1. 框架 2](#_Toc396918568)

[1.1 结构 2](#_Toc396918569)

[1.2 流程 2](#_Toc396918570)

[2. 测试工具与RF间接口 3](#_Toc396918571)

[2.1 测试数据获取 3](#_Toc396918572)

[2.2 测试报告 4](#_Toc396918573)

[3. 测试工具 4](#_Toc396918574)

[3.1 数据查询 4](#_Toc396918575)

[3.2 上报测试报告 5](#_Toc396918576)

[3.3 异常处理 5](#_Toc396918577)

[3.4 数据查询方法 5](#_Toc396918578)

[4. RF 6](#_Toc396918579)

[4.1 编写数据查询 6](#_Toc396918580)

[4.2 编写结果检查 6](#_Toc396918581)

[4.3 转换为测试脚本 6](#_Toc396918582)

[4.4 变量记录 6](#_Toc396918583)

[4.5 检查测试结果 7](#_Toc396918584)

[4.6 异常处理 7](#_Toc396918585)

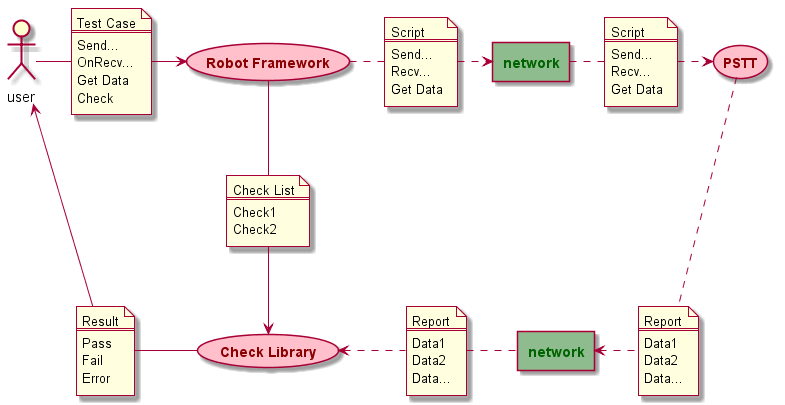
# 框架

## 结构

用户编写测试用例，由RF翻译为测试脚本，并生成测试检查清单。测试脚本通过网络传递给PSTT，PSTT执行脚本，并生成测试报告。测试报告回传给RF，RF检查测试报告中的内容，输出测试结果。

测试结果包含三种：

* Pass：测试通过；
* Fail：测试不通过；
* Error：测试执行过程中有错误。

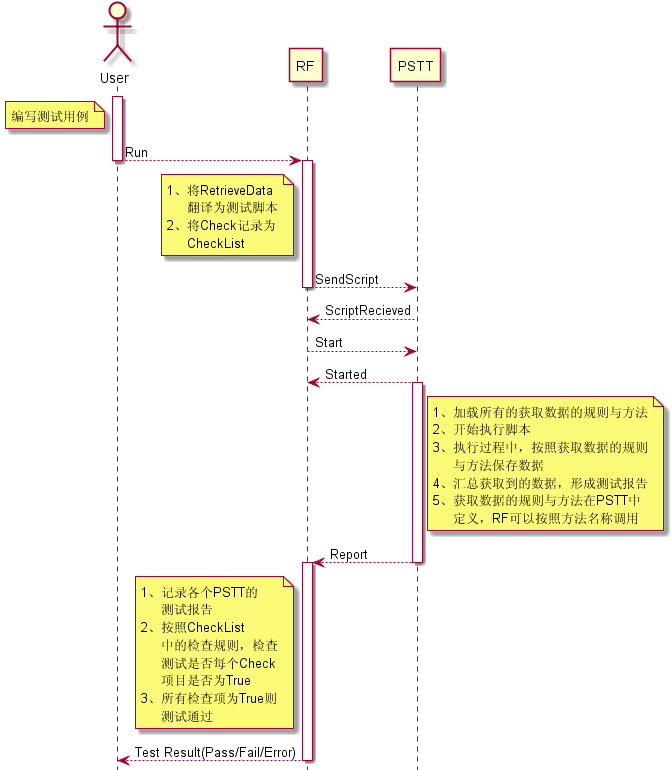


## 流程

数据的检查分两个步骤，一是数据的获取，另一是对获取到的数据的检查。这样做的原因是，数据的获取方法相对固定，如读取某个参数值、记录某类消息的个数等，但是检查方法灵活多变，因此要将变化的部分与不变的部分分割开，以便能够灵活扩展。

数据获取部分，在PSTT实现。PSTT提供各种固定的数据获取方法，在RF的测试用例中描述。测试用例执行过程中，PSTT按照指定的方法获取数据，并记录在测试报告中，PSTT中的测试用例执行完成后，PSTT将所有获取到的数据汇总，通过测试报告发送给RF。

数据的检查部分在RF实现。在测试用例中，描述要对数据做什么样的检查，包括数值大小的判断、字符串的判断、与或非逻辑等。检查方法在测试用例中编写，检查过程由RF的库来执行。最终的测试结果依照各个检查的结果而定，只有所有检查通过，才算测试通过。



# 测试工具与RF间接口

## 测试数据获取

**[RetrieveData]**

**Operation**: <operation name>

**Param**: <para>|<para>=<value>;…

**Return**: <variable>

**StartMessage**: name=<msg>|alias=<alias>

**StartParameter**: <msg>.<attr>…=<value>|<variable>; …

**StartIndex**: <index>

**EndMessage**: name=<msg>|alias=<alias>

**EndParameter**: <msg>.<attr>…=<value>|<variable>; …

**EndIndex**: <index>

其中：

Operation：要执行的获取数据的操作；

Param：获取数据需要的参数，如指定的消息名称、参数等；

Return：返回值的变量名称。每个变量名称对应一个返回值，可以在测试用例中后续引用该名称；

Start…|End…：获取数据的范围，可以不指定，如果指定的话，可以指定起始/结束消息的名称、参数、第几个等

## 测试报告

测试报告与其他RF—PSTT消息的格式应该保持一致，建议使用xml或者json等结构化的数据描述语言。如果按照json格式，则测试报告格式如下：

{

“message type”:”report”,

“<variable\_name>” : <value>,

“<variable\_name>” : <value>,

“<variable\_name>” : <value>,

“<variable\_name>”: {“<parameter}”: <value>,

“<parameter}”: <value>,}

…

}

其中，嵌套深度最大为两层。

结构化的数据描述语言中，类型、数组、结构体等都能够描述，因此这种格式具有较强的传递数据的能力。

# 测试工具

## 数据查询

测试工具在收到测试用例后，记录用例中的检查项，然后开始执行测试用例。测试过程中，或者测试完成后，按照检查项中的要求获取测试过程数据。

测试检查项目，是在测试工具中的预定义过程，在测试用例中可以通过检查项的名称来调用这些检查。测试工具根据测试用例中描述的检查项目的名称，找到合适的检查方法，并将测试用例中携带的检查参数带入检查过程，完成一次检查。

测试检查项目中，可以指定检查的范围，用于描述检查的起始消息、终止消息以及其参数特征、个数索引等。这些描述都是可选的，可以任意组合。消息的名称与参数，描述方法与Receive相同，表示的是开始/结束的消息的特征。当指定了消息名称或参数时，Index描述的是在匹配的消息中从第几个开始/结束；如果没有指定消息名称和参数，Index描述的就是在所有消息中从第几个开始/结束。

测试用例中的每个检查，都会有一个“返回值”，用于标识一次检查。当测试工具获取到检查项指定的数据时，将数据与返回值关联存储，在测试报告中返回。

## 上报测试报告

当测试用例执行结束，并且测试工具完成所有的检查后，测试工具向RF上报测试报告。测试报告的格式参见2.2节，其中每一组值对应一个检查项目，测试报告仅上报检查项目的返回值名称和值。

测试报告在测试用例结束后发送。测试用例结束后，工具先向RF发送End，然后发送Report。

为简化处理，测试报告中的数据类型目前只支持字符串和数值以及由这两种类型构成的结构体。

## 异常处理

当测试发生异常时，不需要向RF上报测试报告。由于发生异常代表着测试无法执行下去，因此其中的检查项目已经没有意义了。这时RF将根据Exception中的信息标识测试用例Error，不再检查测试报告中的各项内容。

## 数据查询方法

需要支持的数据查询方法包括：

**MessageCount(msg, condition)：**

**符合指定条件的消息个数。**

msg：消息名称。

condition：用于指定消息中携带的参数必须符合的条件。可选。

返回值：整型。json格式：{“<variable\_name>”:<value>}。

**MessageParameterCount(msg, index, condition, para\_name)：**

**消息中携带的某参数的个数。**

msg：消息名称。

condition：用于指定消息中携带的参数必须符合的条件。可选，不携带时只需要匹配消息名称。

index：如果有多个符合条件的消息，index指示选择其中第几个，从1开始，可选，默认值为1。

para\_name：参数名称。使用“.”分隔的参数名称，与PSTT上的消息树一致。只能携带一个参数。

返回值：整型。json格式：{“<variable\_name>”:<value>}。

**MessageParameterValue(msg, index, condition, para\_names)：**

**符合条件的消息参数取值。**

msg：消息名称。

condition：用于指定消息中携带的参数必须符合的条件。可选，不携带时只需要匹配消息名称。

index：如果有多个符合条件的消息，index指示选择其中第几个，从1开始，可选，默认值为1。

para\_name：参数名称。使用“.”分隔的参数名称，与PSTT上的消息树一致。可以携带多个参数，通过“,”分隔。

返回值：结构。json格式：

{“<variable\_name>”: {“<parameter}”: <value>,

“<parameter}”: <value>}}

**MessageOrder(msgs)：**

**消息的顺序是否符合预期。**

msgs：消息名称，通过“,”分隔。

返回值：整数，0-否，1-是。json格式：{“<variable\_name>”:0|1}

其余操作后续根据需求逐渐增加。

# RF

## 编写数据查询

在RF中，数据查询采用一个新的命令RetrieveData来编写，供工具识别。命令格式参见2.1节。

## 编写结果检查

在RF中数据的检查时与数据查询分开的，这部分工作全部由RF来完成。数据检查使用一个新的关键字Check来编写，这个关键字在RF处理，因此不需要转换为PSTT的脚本。关键字格式如下：

Check <Expression>

其中Expression是一个逻辑表达式，其中可以对测试报告中返回的各种返回值，以及测试用例中使用到的各种变量做检查。检查方法包括：数学运算、逻辑运算、函数。字符串的各种计算，由函数提供。具体提供哪些函数，在RF中可以定制，在测试用例中使用。

目前计划实现的内容包括：

* 数学运算：+、-、×、/、()
* 逻辑运算：&&、||、!
* 函数：字符串长度、字符串正则表达式匹配、字符串正则表达式提取子串。

## 转换为测试脚本

在RF上某测试用例开始执行测试后，测试用例首先被转换为测试工具可识别的测试脚本。在测试检查方面，会将数据提取关键字转换为脚本RetrieveData，格式如2.1。RetrieveData是测试脚本中的一个普通操作，与SendMessage、RecvMessage等处理方式一致。在RF与测试工具间，不需要为RetrieveData新增消息，在SendScript时，所有测试执行时的操作，包括RetrieveData将一起分发给测试工具。

对于Check关键字，RF不需要转换为测试脚本，在RF内部，将要检查的逻辑表达式记录在Checklist中。

## 变量记录

测试用例中可以使用的变量，在各个测试工具间将相互广播，这一过程通过RF操作，因此RF能够得知测试用例中的变量值。RF在广播变量的同时，自己也维护一份变量列表，后续检查时可以对变量直接做检查。

## 检查测试结果

测试结束后，RF收到测试工具的测试报告。RF遍历记录的Checklist，对其中的每个逻辑表达式做演算，检查测试报告中与变量的值是否符合预期的逻辑关系。

每个检查项的结果有三种：Pass、Fail、Error。只有一个测试用例的所有检查项的结果都为Pass的时候，这个用例才算测试通过。当至少一个检查项的结果为Fail，其余检查结果为Pass的时候，测试用例不通过。当至少一个检查项的结果为Error，其余检查结果为Pass或Fail时，测试用例执行异常。

## 异常处理

执行用例时，当RF收到测试工具上报的Exception时，RF立即停止所有工具的执行，不检查测试报告，直接标记测试用例异常。测试用例异常的原因从Exception中读取。