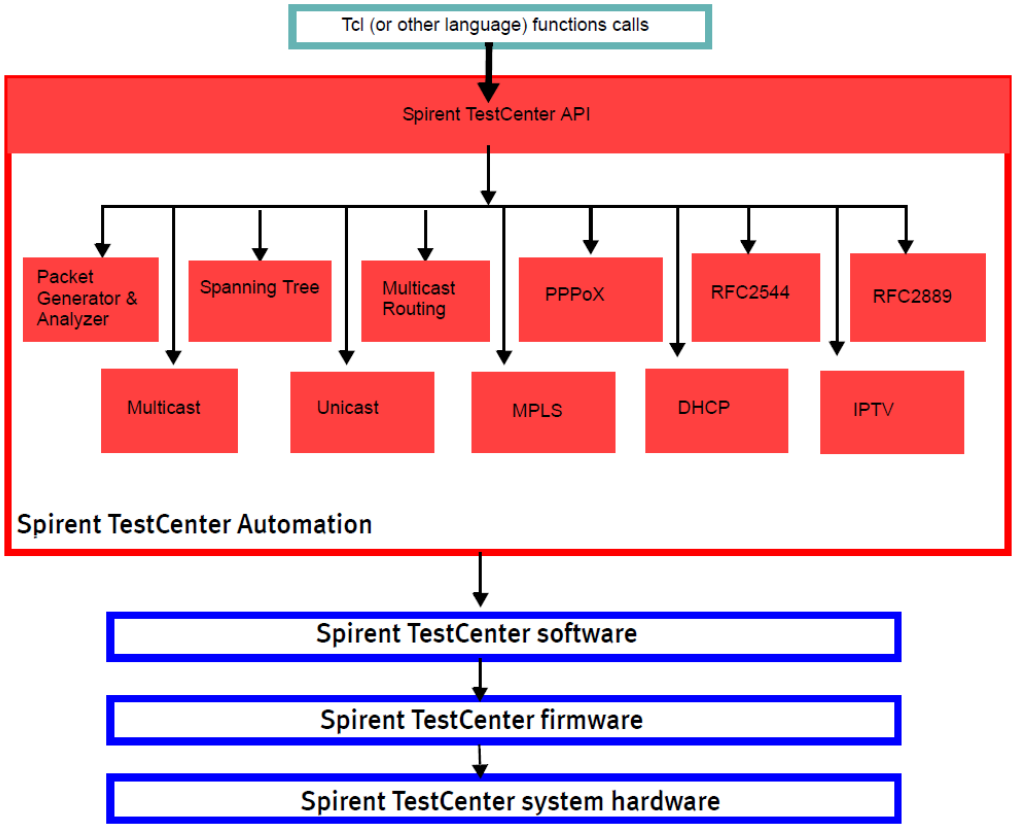
Spirent TestCenter WiFi API 实现自动化测试开发实现

一、TestCenter 自动化测试框架

在使用 Spirent TestCenter 自动化时，需要通过调用 TestCenter API 函数来创建测试配置进行测试。在进行自动化测试时，在执行测试的环境中需要安装 Spirent TestCenter 客户端，以便于 API 可以驱动 Spirent TestCenter 硬件进行测试。下图是 Spirent TestCenter 自动化和 Spirent TestCenter 核心系统（Spirent TestCenter 软件、固件和系统硬件）之间的关系。

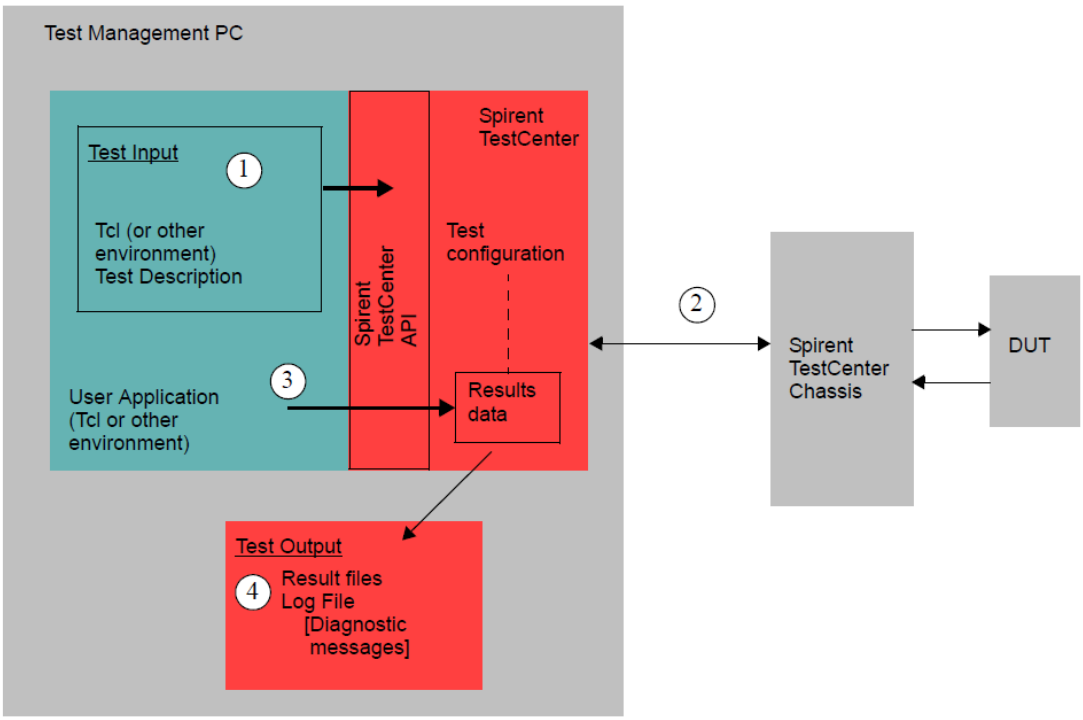


Spirent TestCenter API

Spirent TestCenter 自动化提供了强大的 API。通过这些 API 可以创建和运行测试，并且自动获取到测试结果。API 定义了一组函数，可以从 TCL、Ruby、Python 等脚本环境中调用这些函数。

这些 API 由一组用于 Spirent TestCenter 所有自动化库的公共函数组成。这些公共函数可以反映出 Spirent TestCenter 库的体系结构。因为这些库共享公共的接口，学会了如何编写一种功能的测试程序，就很容易编写其他测试程序。

这些 API 支持 Spirent TestCenter 提供的所有特性。当然也包括 WIFI 功能。



上图是使用 Spirent TestCenter 进行自动化测试的逻辑过程。

二、测试环境搭建

Spirent TestCenter 支持多种脚本语言使用 API 进行自动化测试。这里以 Ruby 语言为例介绍搭建 Spirent TestCenter 自动化开发环境。

第一步：打开 spirenttestcenter.rb 并且把 STC_INSTALL_DIR 字符串替换为 Spirent TestCenter Application 的安装路径。

路径当中要使用前向斜杠（/）. 在安装路径的开始和结束位置添加上（'）。例如：

在替换前：

```
ENV['STC_PRIVATE_INSTALL_DIR'] = STC_INSTALL_DIR
```

在替换后：

```
ENV['STC_PRIVATE_INSTALL_DIR'] = 'C:/Program Files/Spirent Communications/Spirent TestCenter 5.00/Spirent TestCenter Application/'
```

第二步：确认 Ruby gem 在当前的路径中，然后通过命令生成 spirenttestcenter gem。命令如下：

```
gem build spirenttestcenter.gemspec
```

第三步：安装生成的 gem 文件，命令如下：

gem install spirenttestcenter”X.X.X.gem

这里的 x 要替换成 Spirent TestCenter 的实际版本号。

第四步：检查确认 Ruby 配置正确。可以使用如下的命令：

irb

require 'rubygems'

require 'spirenttestcenter'

Stc.get('system1', 'version')

如果 Ruby 返回 Spirent TestCenter 的版本号，那么表示 Ruby 测试环境已经配置完成。

三、使用 API 开发 WiFi 功能自动化

使用 Spirent TestCenter 开发 WiFi 功能，可以包含如下几个过程：

1. 实例化端口
2. 配置无线端口参数
3. 在无线端口上创建模拟终端
4. 扫描端口信号
5 设置用户模板
6. 启动终端

如下是各功能单元 API 的实现参考。

（1） 实例化端口

```
# Physical topology
szChassisIp1 = "10.29.0.49";
szChassisIp2 = "10.29.0.45";
txPortLoc = "//#{szChassisIp1}/1/1";
rxPortLoc = "//#{szChassisIp2}/1/1";

# Create the root project object
hProject = Stc.create("project", "system1")

hPortTx = Stc.create("port", hProject, "location"=>txPortLoc, "useDefaultHost"=>false)
hPortRx = Stc.create("port", hProject, "location"=>rxPortLoc, "useDefaultHost"=>false)
```

```
# Attach ports.
# Connects to chassis, reserves ports and sets up port mappings all in one step.
# By default, connects to all previously created ports.
Stc.perform("AttachPorts")
```

（2） 配置无线端口参数

```
# Create the Ieee80211 Physical module object
hIeee80211Phy1 = Stc.create("Ieee80211Phy ", hPortTx,
                        " EnableRadioPower "=>” TRUE”,
                        " MimoType "=>” MIMO_4X4”,
                        " Sma1 "=>”TRUE”,
                        " Sma2 "=>”TRUE”,
                        " Sma3 "=>”TRUE”,
                        " Sma4 "=>”TRUE”,
                        " Sma5 "=>”TRUE”,
                        " Sma6 "=>”TRUE”,
                        " Sma7 "=>”TRUE”,
                        " Sma8 "=>”TRUE”,
                        " ApplyAllAttenuators "=>” TRUE”,
                        " ChannelBandwidth20Mhz "=>” TRUE”,
                        " ChannelBandwidth80Mhz "=>” TRUE”,
                         "ChannelBandwidth160Mhz Host"=>” TRUE”)

Stc.config( hPortTx,”ActivePhy-targets”=> " hIeee80211Phy1")
```

（3） 在无线端口上创建模拟终端

```
#create Host to emulate Terminal
hEmulatedDevice = Stc.create("EmulatedDevice", hProject,
                         “DeviceCount”=> "1" )
hEthIIIf = Stc.create("EthIIIf" , hEmulatedDevice ,
                        “SourceMac”=> "00:10:94:00:00:01")

hIpv4If=Stc.create("Ipv4If" ,hEmulatedDevice ,
                        “Address”=> "192.85.1.3" ,
                        “PrefixLength”=> "24" ,
                        “Gateway”=> "192.85.1.1" )

Stc.config( hEmulatedDevice, “TopLevelIf-targets”=> hIpv4If )
Stc.config( hEmulatedDevice, “PrimaryIf-targets”=> hIpv4If )
Stc.config( hIpv4If, “StackedOnEndpoint-targets”=> hEthIIIf )
Stc.config(hPortTx, “AffiliationPort-sources”=> hEmulatedDevice )

hIeee80211ClientConfig = Stc.create( "Ieee80211ClientConfig" , hEmulatedDevice ,
```

“ClientsMacAddressesList”=> "00:10:00:00:00:01")

（4） 扫描端口信号

Stc.perform(“Ieee80211Scanning”, “HandleList” => hPortTx)

（5） 设置用户模板

```
hIeee80211ClientProfile = Stc.create( "Ieee80211ClientProfile" ,hProject ,
                “BSSID "00:00:00:00:00:00" ,
                “EncryptionType "ENCRYPT_TYPE_OPEN" ,
                “PassPhrase {password} ,
                “ActiveEncryption "TRUE" )

Stc.config( hIeee80211ClientConfig, “AffiliatedUEClientProfile-targets " =>
hIeee80211ClientProfile )
```

（6） 启动终端

Stc.perform(“Ieee80211Associate’, “ClientList” => hIeee80211ClientConfig

以上部分就是 WiFi 部分的功能自动化实现。如果需要扩展发流等功能，需要结合其他功能 API 来实现。