TesterLibrary

发布 *1.0.0.07212216*

Xinertel

2022 年 07 月 21 日

Contents:

1 TesterLibrary package **1**
- 1.1 Subpackages 1
- 1.2 Submodules 499
- 1.3 TesterLibrary.base module 499
- 1.4 TesterLibrary.data module 984
- 1.5 Module contents 984

2 Indices and tables **985**

Python 模块索引 **987**

索引 **989**

1 TesterLibrary package

1.1 Subpackages

1.1.1 TesterLibrary.Overall package

Submodules

TesterLibrary.Overall.common module

TesterLibrary.Overall.common.**connect_chassis**(*Chassis*)

连接测试仪表机箱后台.

参数 **Chassis** (*str*) -- 机箱主机 IP 地址列表

返回 Chassis 对象列表

返回类型 list

实际案例

robotframework:

```
| ${Hosts}= | Create List | 192.168.0.10 | 192.168.0.10 |
| ${Chassis} | Connect Chassis | Chassis=${Hosts} |
```

TesterLibrary.Overall.common.**create_bgp_ipv4_flowspec_performance**(*Session*, *MaxRouteCount*, *SourcePrefix*, *DestPrefix*)

创建 bgpflowspec 性能条目

参数

- **Session** -- bgp session
- **MaxRouteCount** -- 支持最大 bgpls 数量
- **SourcePrefix** -- 接口列表
- **DestPrefix** -- BGP ipv4 路由列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

Examples:

TesterLibrary.Overall.common.**create_peclsp_for_srte**(*Excel*, *Session*, *TunnelCount=16000*, *PcelspCount=4000*, *SymbolicNameIdentification='Tunnel'*)

从 Excel 表格创建 SRTE 性能测试 PCE LSP

参数

- **Excel** -- Excel 文件完整路径
- **Session** -- PceLspConfig 对象
- **TunnelCount** -- 隧道数量
- **PcelspCount** -- pcelsp 数量

返回 布尔值 Bool (范围：True / False)

返回类型 bool

Examples:

TesterLibrary.Overall.common.**del_objects**(*Objects*)

删除测试仪表相关对象

:param : param Objects: 测试仪表相关对象:param : type Objects: 类型为：list，测试仪表相关对象 object 列表

Returns: 布尔值 Bool (范围：True / False)

实际案例

robotframework:

```
| ${Port} | Get Ports |
| Del Objects | Port=${Port} |
```

TesterLibrary.Overall.common.**edit_configs**(*Configs*, ***kwargs*)

TesterLibrary.Overall.common.**edit_overall_setting**(***kwargs*)

编辑测试仪表统全局参数

关键字参数

- 流全局配置，参数支持： -- PortSendMode: 端口发送模式

  SYNCHRONOUS ASYNCHRONOUS

  MeshCreationMode: 拓扑创建模式

  PortBased EndpointBased

- 二层学习，参数支持： -- Rate: 速率（帧/秒），类型：number，值范围：1-4294967295, 默认值：100

  RepeatCount: 重复次数, 类型：number，值范围：0-4294967295, 默认值：3

  DelayTime: 学习前延迟时间, 类型：number，值范围：0-4294967295, 默认值：1

  RxLearningEncapsulation: 封装类型

NO_ENCAPSULATION TX_ENCAPSULATION

- **ARP/ND** 选项，参数支持： -- EnableAutoArp: 使能自动 ARP/ND, 类型: bool, 默认值：True

  StopOnArpFail: ARP/ND 失败自动停止测试, 类型: bool, 默认值：False

  AutoArpWaitTime: 自动 ARP/ND 等待时间（秒）, 类型：number，值范围：0-4294967295, 默认值：30

- **LM** 全局配置，参数支持： (*Y.1731*) -- TestModeType: 测试模式, 类型: string, 默认值：TYPE_NORMAL

  TYPE_NORMAL TYPE_CC_SCALE_MODE TYPE_CC_SCALE_MODE_WITHOUT_RX

  LmrRxFCfStart: LMR 帧的 RxFCF 初始值, 类型：number，值范围：0-4294967295, 默认值：1

  LmrRxFCfStep: LMR 帧的 RxFCF 更新步长, 类型：number，值范围：0-65535, 默认值：1

  LmrTxFCbStart: LMR 帧的 TxFCF 初始值, 类型：number，值范围：0-4294967295, 默认值：1

  LmrTxFCbStep: LMR 帧的 TxFCF 更新步长, 类型：number，值范围：1-65535, 默认值：9

  LmmTxFCfOffset: LMM 帧的 TxFCF 的偏移值, 类型：number，值范围：0-32767, 默认值：0

  LmrRxFCfOffset: LMR 帧的 RxFCF 的偏移值, 类型：number，值范围：0-32767, 默认值：0

  LmrTxFCbOffset: LMR 帧的 TxFCF 的偏移值, 类型：number，值范围：0-32767, 默认值：0

  DmTimeUnit: DM 时间统计单位, 类型：string, 默认值：TYPE_NORMAL

  TIME_MS TIME_NS

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| edit_overall_setting | PortSendMode=ASYNCHRONOUS |␣
↪RxLearningEncapsulation=TX_ENCAPSULATION | EnableAutoArp=False |␣
↪TestModeType=TYPE_CC_SCALE_MODE |
```

TesterLibrary.Overall.common.**get_config_children**(*Configs*, *Children*)

TesterLibrary.Overall.common.**get_configs**(*Configs=None*, *KeyType='handle'*, *Upper=None*)

获取测试仪表指定对象

参数

- **Configs** -- 测试仪表端口对象类型列表, 类型为：list
- **KeyType** -- 返回字典使用指定类型作为字典的 key，支持：handle、name
- **Upper** -- 指定上层节点获取对象

返回 object} 或者 {'name': object}

返回类型 字典 {'handle'

实际案例

robotframework:

```
| ${Result} | Get Configs | KeyType=name |
| ${Result} | Get Configs | Configs=StreamTemplate | KeyType=handle |
| ${Result} | Get Configs | Configs=BgpProtocolConfig | KeyType=name |
| ${Result} | Get Configs | Configs=StreamTemplate | KeyType=handle | Upper=$
↪{Port_1} |
| ${Result} | Get Configs | Configs=BgpProtocolConfig | KeyType=name | Upper=$
↪{Port_1} |
```

TesterLibrary.Overall.common.**init_tester**(*Product='BIGTAO'*, *Mode='performance'*, *Log=True*)

初始化测试仪表

参数

- **Product** (*str*) -- 测试产品类型, 支持 BIGTAO 和 DARYU
- **Mode** (*str*) -- 统计模式, 支持 Performance 和 DB
- **Log** (*bool*) -- 是否使能机箱日志

返回 sys_entry 测试仪表根节点对象

返回类型 (sys_entry)

实际案例

robotframework:

```
| ${result} | init tester | Product=DARYU | Mode=Performance |
```

TesterLibrary.Overall.common.**load_case**(*Path*)

测试仪表加载配置文件

参数 **Path** (*str*) -- 配置文件路径, 类型 string (例如："C:/test.xcfg")

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${result} | Load Case | Path=='C:/test.xcfg' |
```

TesterLibrary.Overall.common.**reset_tester**()

清空测试仪表所有配置

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Reset Tester |
```

TesterLibrary.Overall.common.**save_case**(*Path*)

测试仪表保存配置文件

参数 **Path** (*str*) -- 配置文件路径, 例如：”C:/test.xcfg”

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${result} | Save Case | Path=='C:/test.xcfg' |
```

Module contents

1.1.2 TesterLibrary.Port package

Submodules

TesterLibrary.Port.capture module

TesterLibrary.Port.capture.**create_capture_byte_pattern**(*Port*, ***kwargs*)

在指定端口上创建 Byte Pattern

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **CustomCapturePatternOperator** (*str*) -- 表达式位运算符：, 类型为：string, 默认值：AND, 支持参数

  AND OR XOR

- **CustomCapturePatternNot** (*bool*) -- 表达式取反：, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UseFrameLength** (*bool*) -- 使用 Frame 长度：, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Data** (*str*) -- 最小值：, 类型为：string, 取值范围：十六进制字符串, 默认值：0x0,
- **MaxData** (*str*) -- 最大值：, 类型为：string, 取值范围：十六进制字符串, 默认值：0xff,
- **Mask** (*str*) -- 掩码：, 类型为：string, 取值范围：十六进制字符串, 默认值：0xff,
- **Offset** (*int*) -- 偏移位：, 类型为：number, 取值范围：0-16378, 默认值：0
- **MinFrameLength** (*int*) -- 最小长度, 当 UseFrameLength 为 True 有效：, 类型为：number, 取值范围：64-16383, 默认值：64

- **MaxFrameLength** (*int*) -- 最大长度, 当 UseFrameLength 为 True 有效：, 类型为：number, 取值范围：64-16383, 默认值：16383

返回 Byte Pattern 唯一索引字符串 string，例如：CapturePattern_1

返回类型 str

实际案例

robotframework:

```
| Create Capture Byte Pattern | Port=${Port} | Data=0x0 0x01 | Mask=0xff 0xff␣
↪| Offset=0 | CustomCapturePatternOperator=OR | CustomCapturePatternNot=True␣
↪|
```

TesterLibrary.Port.capture.**create_capture_pdu_pattern**(*Port*, *HeaderTypes*, *FieldName*, ***kwargs*)

在指定端口上创建 Pdu Pattern

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **HeaderTypes** (*list*) -- 指定报文结构 EthernetII Vlan IPv4 IPv6 Igmpv1 Igmpv1Query Igmpv2 Igmpv2Query Igmpv3Report Igmpv3Query Icmpv4EchoRequest Icmpv4EchoReply
- **FieldName** (*str*) -- 过滤字段名
- **CustomCapturePatternOperator** (*str*) -- 表达式位运算符：, 类型为：string, 默认值：AND, 支持参数

  AND OR XOR
- **CustomCapturePatternNot** (*bool*) -- 表达式取反：, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Value** (*str*) -- 最小值：, 类型为：string
- **MaxValue** (*str*) -- 最大值：, 类型为：string
- **Mask** (*str*) -- 掩码：, 类型为：string

返回 Pdu Pattern 唯一索引字符串 string，例如：CapturePduPattern_1

返回类型 str

实际案例

robotframework:

```
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoReply |
| Create Capture Pdu Pattern | Port=${Port} | HeaderTypes=${HeaderTypes} |␣
↪FieldName=Icmpv4EchoReply_1.code | Value=4 | MaxValue=5 |
```

TesterLibrary.Port.capture.**download_packages**(*Port*, *FileDir*, *FileName*, *MaxCount=0*, *TimeOut=30*)

下载指定端口捕获到的数据包

参数

- **Port** (Port) -- 测试仪表端口对象 object
- **FileDir** (*str*) -- 报文保存的路径, ("D:/test")

- **FileName** (*str*) -- 报文保存的文件的名称
- **MaxCount** (*int*) -- 下载报文最大数量，默认值 0, 表示下载端口上捕获到的所有报文
- **TimeOut** (*int*) -- 下载报文的超时时间单位秒，超时时间内为下载完成则下载失败, 默认值 30

返回 下载数据包文件的绝对路径 (例如：”D:test10.0.5.10_1_1dowload.pcap”)

返回类型 (str)

实际案例

robotframework:

```
| ${Port} | Get Ports |
| &{File} | Download Packages | Port=${Port} | FileDir=D:/test |␣
↪FileName=download |
```

TesterLibrary.Port.capture.**edit_capture**(*Ports*, ***kwargs*)

编辑端口数据捕获参数

参数 **Ports** -- 测试仪表端口对象列表, 类型为：list

关键字参数

- **Name** (*str*) -- 端口捕获名称, 类型为：string
- **CaptureMode** (*str*) -- 捕获模式：, 类型为：string, 默认值：ALL, 支持参数

  ALL CTRL_PLANE RealTime_All
- **CacheCapacity** (*str*) -- 缓存容量, 类型为：string, 默认值：Cache_Max, 支持参数

  Cache_Max Cache_32KB Cache_64KB Cache_128KB Cache_256KB Cache_512KB Cache_1MB Cache_2MB Cache_4MB Cache_8MB Cache_16MB Cache_32MB Cache_64MB Cache_128MB Cache_256MB Cache_512MB Cache_1GB
- **FilterMode** (*str*) -- 筛选模式, 类型为：string, 默认值：BYTE, 支持选项有：

  BYTE PDU
- **BufferFullAction** (*str*) -- 缓存区满后执行动作, 类型为：string, 默认值：STOP, 支持选项有：

  STOP WRAP
- **StartingFrameIndex** -- 下载报文的起始编号, 类型为：number, 默认值：1
- **AttemptDownloadPacketCount** -- 下载报文数量, 类型为：number, 默认值：0, 0 表示下载所有报文
- **FcsError** -- Fec 错误, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Ipv4ChecksumError** -- Ipv4 Checksum 错误, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **PayloadError** -- Payload 错误, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableRealtimeCapture** -- 捕获模式为 Control Plane (Tx and Rx) 或 RealTime All(Control Plane tx/rx and data plane rx), 类型为：bool, 取值范围：True 或 False, 默认值：False

- **SliceMode** (*str*) -- 切片模式, 类类型为：string, 默认值：DISABLE, 支持选项有：

  DISABLE ENABLE

- **SliceByteSize** -- 切片字节大小, 类型为：number, 取值范围：32-16383, 默认值：128

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Capture | Ports=${Ports} | CaptureMode=CTRL_PLANE |
```

TesterLibrary.Port.capture.**edit_capture_event**(*Port*, *EventType='QUALIFY'*, ***kwargs*)

在指定端口上设置帧捕获条件

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：object
- **EventType** (*str*) -- 帧捕获类型：, 类型为：string, 支持参数：

  QUALIFY START START

关键字参数

- **LogicRelation** -- 指定捕获事件之间的逻辑关系 And 或 Or
- **PatternMatch** -- 指定捕获事件之间的逻辑关系 And 或 Or
- **FcsError** -- FCS 错误，支持 IGNORE、INCLUDE 或 EXCLUDE
- **PrbsError** -- FCS 错误，支持 IGNORE、INCLUDE 或 EXCLUDE
- **Ipv4ChecksumError** -- IPv4 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **TcpChecksumError** -- TCP 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **UdpChecksumError** -- UDP 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **IgmpChecksumError** -- Igmp 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **IcmpChecksumError** -- Icmp 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **SequenceError** -- 序列号错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **UndersizedFrame** -- 超短帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **OversizedFrame** -- 超长帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **JumboFrame** -- Jumbo 帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **FrameLength** -- 特定长度帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **FrameLengthValue** -- 特定帧长度，默认值：0
- **SignaturePresent** -- 带有签名的流帧，支持 IGNORE、INCLUDE 或 EXCLUDE

- **StreamIdMatch** -- 特定流号的流帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **StreamId** -- 特定流号，默认值：0
- **Ipv4Packets** -- IPv4 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **TcpPackets** -- TCP 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **UdpPackets** -- UDP 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **Ipv6Packets** -- IPv6 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **IgmpPackets** -- IGMP 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **PayloadError** -- PRBS 错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Capture Event | Port=${Port} | EventType=QUALIFY | LogicRelation=Or |␣
↪PrbsError=INCLUDE | FrameLength=INCLUDE | FrameLengthValue=128 |
```

TesterLibrary.Port.capture.**edit_capture_filter**(*Port*, *Expression*)

在指定端口上设置报文过滤逻辑表达式

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：object
- **Expression** (*str*) -- 过滤逻辑表达式：, 类型为：string

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${BytePattern} | Create Capture Byte Pattern | Port=${Port} |␣
↪CustomCapturePatternOperator=OR | CustomCapturePatternNot=True |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoReply |
| ${PduPattern} | Create Capture Pdu Pattern | Port=${Port} | HeaderTypes=$
↪{HeaderTypes} | FieldName=Icmpv4EchoReply_1.code | Value=4 | MaxValue=5 |␣
↪CustomCapturePatternOperator=OR | CustomCapturePatternNot=True |
| Edit Capture Filter | Port=${Port} | Expression=${BytePattern} && $
↪{PduPattern} |
```

TesterLibrary.Port.capture.**edit_capture_pattern**(*Pattern*, ***kwargs*)

修改 Capture Pattern 参数

参数 Pattern (*str*) -- Capture Pattern 的标识, 类型为：sting，例如：CaptureBytePattern_1 或 CapturePduPattern_1

关键字参数

- **Pattern** 支持的 **Args** (*Pdu*) -- CustomCapturePatternOperator (str): 表达式位运算符：, 类型为：string, 默认值：AND, 支持参数

  AND OR XOR

CustomCapturePatternNot (bool): 表达式取反：, 类型为：bool, 取值范围：True 或 False, 默认值：False

UseFrameLength (bool): 使用 Frame 长度：, 类型为：bool, 取值范围：True 或 False, 默认值：False

Data (str): 最小值：, 类型为：string, 取值范围：十六进制字符串, 默认值：0x0,

MaxData (str): 最大值：, 类型为：string, 取值范围：十六进制字符串, 默认值：0xff,

Mask (str): 掩码：, 类型为：string, 取值范围：十六进制字符串, 默认值：0xff,

Offset (int): 偏移位：, 类型为：number, 取值范围：0-16378, 默认值：0

MinFrameLength (int): 最小长度, 当 UseFrameLength 为 True 有效：, 类型为：number, 取值范围：64-16383, 默认值：64

MaxFrameLength (int): 最大长度, 当 UseFrameLength 为 True 有效：, 类型为：number, 取值范围：64-16383, 默认值：16383

- **Pattern 支持的 Args** -- CustomCapturePatternOperator (str): 表达式位运算符：, 类型为：string, 默认值：AND, 支持参数

  AND OR XOR

  CustomCapturePatternNot (bool): 表达式取反：, 类型为：bool, 取值范围：True 或 False, 默认值：False

  Value (str): 最小值：, 类型为：string

  MaxValue (str): 最大值：, 类型为：string

  Mask (str): 掩码：, 类型为：string

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${BytePattern} | Create Capture Byte Pattern | Port=${Port} |␣
↪CustomCapturePatternOperator=OR | CustomCapturePatternNot=True |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoReply |
| ${PduPattern} | Create Capture Pdu Pattern | Port=${Port} | HeaderTypes=$
↪{HeaderTypes} | FieldName=Icmpv4EchoReply_1.code | Value=4 | MaxValue=5 |␣
↪CustomCapturePatternOperator=OR | CustomCapturePatternNot=True |
| Edit Capture Pattern | Pattern=${BytePattern} |␣
↪CustomCapturePatternOperator=XOR|
| Edit Capture Pattern | Pattern=${PduPattern} |␣
↪CustomCapturePatternOperator=XOR|
```

TesterLibrary.Port.capture.**get_capture_info**(*Port*, *Items=None*)

在指定端口报文捕获信息

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：object
- **Items** (*list*) -- 端口报文捕获信息, 支持参数：CaptureState ElapsedTime CapturedPacketCount BufferFull DownloadedPacketCount CurrentDataFile

返回类型 dict

实际案例

robotframework:

```
| Get Capture Info | Port=${Port} |
```

TesterLibrary.Port.capture.**start_capture**(*Ports=None*)

启动测试仪表端口数据抓包

参数 **Ports** (*list*) -- 测试仪仪表端口 Port 对象列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 | //192.168.0.1/1/2 |
| ${Ports} | Reserve Ports | ${Ports} | ${Locations} |
| Start Capture | Ports=${Ports} |
```

TesterLibrary.Port.capture.**stop_capture**(*Ports=None*)

停止测试仪表端口数据抓包

参数 **Ports** (*list*) -- 测试仪仪表端口 Port 对象列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 | //192.168.0.1/1/2 |
| ${Ports} | Reserve Ports | ${Ports} | ${Locations} |
| Start Capture | Ports=${Ports} |
| Sleep | 30 |
| Stop Capture | Ports=${Ports} |
```

TesterLibrary.Port.common module

TesterLibrary.Port.common.**del_port**(*Ports=None*)

删除测试仪端口

参数 **Ports** -- 测试仪表端口对象列表, 类型为：list

返回 Returns：布尔值 Bool (范围：True / False)

实际案例

robotframework:

```
| Del Port | Ports=${Port_1} |
| Del Port |
```

TesterLibrary.Port.common.**edit_port**(*Ports*, ***kwargs*)

修改测试仪表端口参数

参数 **Ports** (list(Port)) -- 测试仪表端口列表

关键字参数

- **EnableLink** (*bool*) -- 设置端口 link Up 和 Down, 默认值: True
- **AutoNegotiation** (*bool*) -- 自协商, 默认值: False
- **Mtu** (*int*) -- 端口 MTU 值, 范围: 128-9600
- **FecType** (*str*) -- Fec 类型, 支持: TYPE_OFF TYPE_RS_FEC_CLAUSE91 TYPE_FEC_CLAUSE74 TYPE_RS_FEC_CLAUSE108 TYPE_RS_FEC_CONSORTIUM TYPE_RS_FEC_CLAUSE119
- **LineSpeed** (*str*) -- 端口速率切换, 支持: SPEED_UNKNOWN SPEED_10M SPEED_100M SPEED_1G SPEED_2_5G SPEED_5G SPEED_10G SPEED_25G SPEED_40G SPEED_50G SPEED_100G SPEED_200G SPEED_400G
- **Duplex** (*str*) -- 全双工半双工, 支持: HALF FULL
- **FlowControl** (*str*) -- 流控, 支持: DISABLE ENABLE AUTO
- **Media** (*str*) -- 媒介, 支持: COPPER FIBER FAKE
- **PhyMode** (*str*) -- Phy Mode, 支持: MODE_AUTO MODE_1000BASEX MODE_SGMII
- **PpmAdjust** (*int*) -- Ppm Adjust, 范围: -300-300
- **DataPathMode** (*str*) -- Data Path 模式, 支持: NORMAL LOOPBACK
- **RemoteFault** (*str*) -- 远端错误, 支持: NORMAL IGNORE
- **Master** (*str*) -- Master, 支持: ADVERTISE_SINGLE_PORT ADVERTISE_MULTI_PORT MANUAL_MASTER MANUAL_SLAVE
- **NoParam** (*bool*) -- 远端错误, 默认值: False

返回 字符串：string，返回保存的 DB 文件的绝对路径字符串

实际案例

robotframework:

```
| Edit Port | Ports=${Ports} | AutoNegotiation=True | FecType=TYPE_OFF |
```

TesterLibrary.Port.common.**edit_port_load_profile**(*Ports*, ***kwargs*)

编辑测试仪表负载配置文件参数

参数 **Ports** (list(IsisIpv4Router) -- 测试仪表端口对象 object 列表

关键字参数

- **TransmitMode** (*str*) -- 传输模式, 默认值：CONTINUOUS，取值范围：
  CONTINUOUS: 连续
  BURST: 突发
  TIME: 按时间突发
  STEP: 单步突发
  ONSTREAM: 基于流调速
- **BurstSize** (*int*) -- 突发报文数，默认值：1
- **InterFrameGap** (*int*) -- 突发间隔，默认值：12.0
- **InterFrameGapUnit** (*str*) -- 突发间隔单位，默认值：BYTES，取值范围：
  NS
  MS
  US
  SEC
  BYTES
- **BurstCount** (*int*) -- 突发次数，默认值：1
- **Seconds** (*int*) -- 发送时间，单位：sec，默认值：100
- **Frames** (*int*) -- 发送帧数，默认值：1
- **LoadProfileType** (*str*) -- 负载类型，默认值：PORT_BASE，取值范围：
  PORT_BASE:
  STREAM_BASE
  PRIORITY_BASE
  MANUAL_BASE
- **Rate** (*int*) -- 端口负载，默认值：10
- **Unit** (*str*) -- 端口负载单位，默认值：PERCENT，取值范围：
  PERCENT:
  FRAME_PER_SEC
  BYTE_PER_SEC
  LINEBIT_PER_SEC
  KLINEBIT_PER_SEC
  MLINEBIT_PER_SEC
  INTER_FRAME_GAP
- **GenerateError** (*str*) -- 报文造错，默认值：NO_ERROR，取值范围：
  NO_ERROR CRC
- **IgnoreLinkState** (*str*) -- 忽略连接状态，默认值：NO，取值范围：
  NO YES
- **TimeStampPosTx** (*str*) -- 发送时间戳位置，默认值：TIMESTAMP_HEAD，取值范围：
  TIMESTAMP_HEAD TIMESTAMP_TAIL

- **TimeStampPosRx** (*str*) -- 接收时间戳位置，默认值：TIMESTAMP_HEAD，取值范围：

  TIMESTAMP_HEAD TIMESTAMP_TAIL

- **LatencyCompensationTx** (*int*) -- 发送时延补偿，默认值：0
- **LatencyCompensationRx** (*int*) -- 接收时延补偿，默认值：0
- **LatencyCompensationOn** (*bool*) -- 时延补偿开启，默认值：True

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=CONTINUOUS |␣
↪Unit=PERCENT | Rate=100 |
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=BURST | BurstSize=10␣
↪| InterFrameGap=20 | InterFrameGapUnit=MS | BurstCount=100 |
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=TIME | Seconds=10 |
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=STEP | Frames=10 |
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=ONSTREAM | Rate=50 |␣
↪Unit=FRAME_PER_SEC |
```

TesterLibrary.Port.common.**edit_stream_load_profile**(*Streams*, ***kwargs*)

编辑测试仪表负载配置文件参数

参数 **Streams** (list(SreamTemplate)) -- 测试仪表流量对象列表，测试仪表流量对象 object 列表

:keyword：param Rate (int): 流量负载，默认值：10 :keyword：param Unit (int): 流量负载单位，默认值：PERCENT，取值范围：

PERCENT

FRAME_PER_SEC

BYTE_PER_SEC

LINEBIT_PER_SEC

KLINEBIT_PER_SEC

MLINEBIT_PER_SEC

INTER_FRAME_GAP

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Port Load Profile | Ports=${Ports} | LoadProfileType=STREAM_BASE |
| Edit Stream Load Profile | Streams=${Streams} | Rate=50 | Unit=FRAME_PER_
↪SEC |
```

TesterLibrary.Port.common.**get_port_speed**(*Ports*)

修改测试仪表端口参数

参数 **Ports** (list(Port)) -- 测试仪表端口列表

返回 端口速率列表

返回类型 list

实际案例

robotframework:

```
| Get Port Speed | Ports=${Ports} |
```

TesterLibrary.Port.common.**get_ports**()

获取当前测试仪表配置中所有的端口对象

返回 Port 对象列表

返回类型 list

实际案例

robotframework:

```
| ${result} | Get Ports |
```

TesterLibrary.Port.common.**release_port**(*Locations=None*, *Ports=None*)

释放测试仪表的端口

参数 **Locations** (*list*) -- 端口在测试仪表机箱硬件中的位置, //$(机箱 IP 地址)/$(板卡序号)/$(端口序号) (例如：[//192.168.0.1/1/1, //192.168.0.1/1/2])

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} == [//192.168.0.1/1/1, //192.168.0.1/1/2] |
| ${result} | Release Port | ${Locations} |
```

TesterLibrary.Port.common.**relocate_ports**(*Ports*, *Locations*)

逻辑端口迁移到执行的测试仪表的物理端口.

参数

- **Ports** (*list*) -- 端口对象的列表

- **Locations** (*list*) -- 端口在测试仪表机箱硬件中的位置, //$(机箱 IP 地址)/$(板卡序号)/$(端口序号) (例如: [//192.168.0.1/1/1, //192.168.0.1/1/2])

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} == [//192.168.0.1/1/1, //192.168.0.1/1/2] |
| ${result} | Relocate Ports | ${Ports} | ${Locations} |
```

TesterLibrary.Port.common.**reserve_port**(*Locations*, *Force=False*, *Debug=False*)

预约测试仪表的端口

参数

- **Locations** (*list*) -- 端口在测试仪表机箱硬件中的位置, //$(机箱 IP 地址)/$(板卡序号)/$(端口序号) (例如: [//192.168.0.1/1/1, //192.168.0.1/1/2])
- **Force** (*bool*) -- 强制占用测试仪表端口, 默认值: False
- **Debug** (*bool*) -- 调试模式, 只创建离线端口, 默认值: False

返回 端口对象列表

返回类型 list

实际案例

robotframework:

```
| ${Locations} == [//192.168.0.1/1/1, //192.168.0.1/1/2] |
| ${result} | Reserve Port | ${Locations} |
```

TesterLibrary.Port.common.**wait_port_state**(*Ports=None*, *State=None*, *Interval=1*, *TimeOut=60*)

等待测试仪表端口链路达到指定状态

参数

- **Ports** (list(Port)) -- 测试仪表端口对象列表
- **State** (*str*) -- 测试仪表连接端口状态, 默认值 UP: DOWN UP
- **Interval** (*int*) -- 状态查询间隔, 默认值:1
- **TimeOut** (*int*) -- 超时时间, 默认值:60

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Port | Ports=${Ports} | EnableLink=False |
| Wait Port State | Ports=${Ports} | State=DOWN |
```

TesterLibrary.Port.interface module

TesterLibrary.Port.interface.**create_interface**(*Port*, *Layers=None*)

在指定端口上創建接口

参数

- **Port** (Port) -- 测试仪仪表端口 Port 对象
- **Layers** (*list*) -- 接口封装类型, 支持的有:

  eth

  ipv4

  ipv6

返回 接口 interface 对象

返回类型 (Interface)

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 |
| ${Layers} | Create List | eth | ipv4 |
| ${Port} | Reserve Ports | ${Ports} | ${Location} |
| ${Interface} | Create Interface | ${Port} | ${Layers} |
```

TesterLibrary.Port.interface.**edit_interface**(*Interface*, *Layer=None*, *Level=None*, ***kwargs*)

修改测试仪表接口的参数

参数

- **Interface** (Interface) -- 测试仪仪表接口 Interface 对象
- **Layer** (*str*) -- 创建接口类型, 支持的有:

  EthIILayer

  VLANLayer

  IPv4Layer

  IPv6Layer

- **Level** (*int*) -- 要修改的 Layer 在 interface 的所有相同 Layer 的序号，默认值：None, 范围：0-1，为 None 表示修改所有 Layer

关键字参数

- **Count** --
- **EnableInterfaceCount** --
- **RouterIdMode** --

- **RouterId** --
- **RouterIdStep** --
- **RouterIdList** --
- **Ipv6RouterId** --
- **Ipv6RouterIdList** --
- **EnableVlansAssociation** --
- **EthIILayer** -- AddressMode

  Address

  Step

  AddressList

  EnableRandMac

  RandomSeed
- **VLANLayer** -- AddressMode

  VlanId

  Step

  VlanIdList

  Priority

  PriorityStep

  PriorityList

  Cfi

  Tpid
- **IPv4Layer** -- AddressMode

  Address

  Step

  AddressList

  PrefixLength

  Gateway

  GatewayStep

  GatewayList

  GatewayCount

  GatewayMac

  ResolvedMacList

  ResolvedState
- **IPv6Layer** -- AddressMode

  Address

  Step

  AddressList

  PrefixLength

  Gateway

GatewayStep

GatewayList

GatewayCount

GatewayMac

ResolvedMacList

ResolvedState

LinkLocalGenType

LinkLocal

LinkLocalStep

LinkLocalList

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 |
| ${Layers} | Create List | eth | ipv4 |
| ${Port} | Reserve Ports | ${Ports} | ${Location} |
| ${Interface} | Create Interface | ${Port} | ${Layers} |
| Edit Interface | Interfaces=${Interface} | Layer=IPv4Layer | Gateway=192.
↪168.1.1 |
```

TesterLibrary.Port.interface.**edit_interface_stack**(*Interfaces*, *Layers=None*, *Tops=None*)

修改测试仪表接口的结构

参数

- **Interfaces** (list(Interface)) -- 测试仪仪表接口 Interface 对象列表
- **Layers** (*list*) -- 接口封装类型，支持的有：

  eth

  pppoe

  ppp

  vlan

  ipv4

  ipv6
- **Tops** (*list*) -- 接口最上层封装类型，支持的有：

  eth

  pppoe

  ppp

  vlan

  ipv4

  ipv6

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 |
| ${Layers} | Create List | eth |
| ${Tops} | Create List | vlan | ipv4 |
| ${Port} | Reserve Ports | ${Ports} | ${Location} |
| ${Interface} | Create Interface | ${Port} | ${Layers} |
| Edit Interface Stack | ${Interface} | ${Layers} | ${Tops} |
```

TesterLibrary.Port.interface.**get_gateway_mac**(*Interface*)

获取测试仪表学习到的网关 Mac 地址

参数 **Interface** (Interface) -- 测试仪表接口对象

返回 Mac 地址列表 List

返回类型 list

实际案例

robotframework:

```
| Get Gateway Mac | Interface=${Interface} |
```

TesterLibrary.Port.interface.**get_interfaces**(*Ports=None*, *Types=None*)

获取测试仪表学习到的网关 Mac 地址

参数 **Ports** -- 测试仪表接口对象 object

Returns: 测试仪表接口对象列表 List

Examples: robotframework:

```
| Get Gateway Mac | Interface=${Interface} |
```

TesterLibrary.Port.interface.**get_layer_from_interfaces**(*Interfaces*, *Layer='ipv4'*)

获取测试仪表接口的封装层对象

:param：param Interfaces: 测试仪表接口对象列表:param：type Interfaces: 类型为：list

Returns: 测试仪表接口的封装层对象列表 List

Examples: robotframework:

```
| Get Layer From Interfaces | Interfaces=${Interface} | Layer=ipv4 |
```

TesterLibrary.Port.interface.**start_arp**(*Ports=None*, *Interfaces=None*)

测试仪表启动接口 ARP 功能

:param：param Ports: 端口对象的列表:param：type Ports: 类型为：list :param：param Interfaces: 接口对象的列表:param：type Interfaces: 类型为：list，当 Ports 和 Interfaces 都为 None 时，表示启动所有接口的 ARP

Returns: 布尔值 Bool (范围：True / False)

Examples: robotframework:

```
| Start Arp |
```

TesterLibrary.Port.interface.**stop_arp**(*Ports=None*, *Interfaces=None*)

测试仪表停止接口 ARP 功能

:param：param Ports: 端口对象的列表:param：type Ports: 类型为：list :param：param Interfaces: 接口对象的列表:param：type Interfaces: 类型为：list，当 Ports 和 Interfaces 都为 None 时，表示启动所有接口的 ARP

Returns: 布尔值 Bool (范围：True / False)

Examples: robotframework:

```
| Stop Arp |
```

Module contents

1.1.3 TesterLibrary.Protocol package

Submodules

TesterLibrary.Protocol.bfd module

TesterLibrary.Protocol.bfd.**bfd_admin_down**(*Sessions*)

设置 BFD 会话状态 AdminDown

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Admin Down | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bfd.**bfd_admin_up**(*Sessions*)

设置 BFD 会话状态 AdminUp

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Admin Up | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bfd.**bfd_disable_demand_mode**(*Sessions*)

关闭 BFD demand 模式

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Disable Demand Mode | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bfd.**bfd_enable_demand_mode**(*Sessions*)

开启 BFD Demand 模式

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Enable Demand Mode | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bfd.**bfd_initiate_poll**(*Sessions*)

发送 BFD poll Sequence

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Initiate Poll | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bfd.**bfd_resume_pdus**(*Sessions*)

恢复发送 BFD PDU

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Resume Pdus | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bfd.**bfd_set_diagnostic_state**(*Sessions*, *State*)

设置 BFD 状态诊断码

参数

- **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list
- **State** (*str*) -- 设置状态诊断码, 类型为：string, 默认值：NO_DIAGNOSTIC, 支持的状态诊断码：

  NO_DIAGNOSTIC

  CONTROL_DETECTION_TIME_EXPIRED

  ECHO_FUNCTION_FAILED

  NEIGHBOR_SIGNAL_SESSION_DOWN

  FORWARDING_PLANE_RESET

PATH_DOWN

CONCATENATED_PATH_DOWN

ADMINISTRATIVELY_DOWN

REVERSE_CONCATENATED_PATH_DOWN

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Set Diagnostic State | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bfd.**bfd_stop_pdus**(*Sessions*)

停止发送 BFD PDU

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Stop Pdus | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bfd.**create_bfd**(*Port*, ***kwargs*)

创建 BFD 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- BFD 协议会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 BFD 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterRole** (*str*) -- BFD 会话的角色, 类型为：string, 默认值：Active, 支持角色：

  Active

  Passive
- **TimeIntervalUnit** (*str*) -- 时间间隔的单位。类型为：string, 默认值：milliseconds, 支持单位：

  milliseconds

  microseconds
- **DesiredMinTXInterval** (*int*) -- 期望的最小发送时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：50
- **RequiredMinRXInterval** (*int*) -- 需要的最小接收时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：50
- **RequiredMinEchoRXInterval** (*int*) -- 需要的最小 Echo 报文接收时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：0

- **DetectMultiple** (*int*) -- 用于检测超时的时间因子, 类型为：number, 取值范围：2-100, 默认值：3
- **AuthenticationType** (*str*) -- 认证方式, 类型为：string, 默认值：None, 支持的方式：

  NONE

  SIMPLE_PASSWORD

  KEYED_MD5

  METICULOUS_KEYED_MD5

  KEYED_SHA1

  METICULOUS_KEYED_SHA1
- **Password** (*str*) -- 当认证方式不为 NONE 时，在该单元格输入认证密码。密码可以是数字、字母或者数字和字母的组合，最长为 16 位。类型为：string, 默认值：Xinertel
- **KeyID** (*int*) -- 当认证方式不为 NONE 时，在该单元格输入 Key ID, 类型为：number, 取值范围：0-255, 默认值：1

返回 BFD 协议会话对象, 类型：object

返回类型 (BfdRouter)

实际案例

```
| Create bfd | Port=${Port} | TimeIntervalUnit=microseconds |
```

TesterLibrary.Protocol.bfd.**create_bfd_ipv4_session**(*Session*, ***kwargs*)

创建 BFD IPv4 会话对象

参数 **Session** (BfdRouter) -- BFD 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BFD IPv4 会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 BFD IPv4 会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **NumberOfSessions** (*str*) -- BFD IPv4 会话的数目, 类型为：string, 取值范围：1-4294967295, 默认值：1
- **StartDestinationAddress** (*str*) -- 指定第一个目的 IPv4 地址, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：192.0.1.0
- **DestinationAddressIncrement** (*str*) -- 指定下一个目的 IPv4 地址的增量, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：0.0.0.1
- **EnableMyDiscriminator** (*bool*) -- 是否指定本地标识符, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MyDiscriminator** (*int*) -- 指定本地标识符的初始值。只有使能本地标识符被选中才可编辑, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **MyDiscriminatorIncrement** (*int*) -- 指定下一个本地标识符的增量。只有使能本地标识符被选中才可编辑。类型为：number, 取值范围：1-4294967295, 默认值：1
- **EnableYourDiscriminator** (*bool*) -- 是否指定对端标识符, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **YourDiscriminator** (*int*) -- 指定对端标识符的初始值。只有使能本地标识符被选中才可编辑, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **YourDiscriminatorIncrement** (*int*) -- 指定下一个对端标识符的增量。只有使能本地标识符被选中才可编辑。类型为：number, 取值范围：1-4294967295, 默认值：1

返回 BFD IPv4 会话对象, 类型：object

返回类型 (BfdIpv4SessionConfig)

实际案例

```
| ${Session} | Create Bfd | Port=${Port} |
| Create Bfd Ipv4 Session | Session=${Session} | NumberOfSessions=10 |
```

TesterLibrary.Protocol.bfd.**create_bfd_ipv6_session**(*Session*, ***kwargs*)

创建 BFD IPv6 路由对象

参数 **Session** (BfdRouter) -- BFD 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BFD IPv6 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 BFD IPv6 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **NumberOfSessions** (*str*) -- BFD IPv6 会话的数目, 类型为：string, 取值范围：1-4294967295, 默认值：1
- **StartDestinationAddress** (*str*) -- 指定第一个目的 IPv6 地址, 类型为：string, 取值范围：有效的 ipv6 地址, 默认值：2000::1
- **DestinationAddressIncrement** (*str*) -- 指定下一个目的 IPv4 地址的增量, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：::1
- **EnableMyDiscriminator** (*bool*) -- 是否指定本地标识符, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MyDiscriminator** (*int*) -- 指定本地标识符的初始值。只有使能本地标识符被选中才可编辑, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **MyDiscriminatorIncrement** (*int*) -- 指定下一个本地标识符的增量。只有使能本地标识符被选中才可编辑。类型为：number, 取值范围：1-4294967295, 默认值：1
- **EnableYourDiscriminator** (*bool*) -- 是否指定对端标识符, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **YourDiscriminator** (*int*) -- 指定对端标识符的初始值。只有使能本地标识符被选中才可编辑, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **YourDiscriminatorIncrement** (*int*) -- 指定下一个对端标识符的增量。只有使能本地标识符被选中才可编辑。类型为：number, 取值范围：1-4294967295, 默认值：1

返回 BFD IPv6 会话对象, 类型：object

返回类型 (BfdIpv6SessionConfig)

实际案例

```
| ${Session} | Create Bfd | Port=${Port} |
| Create Bfd Ipv6 Session | Session=${Session} |
```

TesterLibrary.Protocol.bfd.**edit_bfd**(*Session*, ***kwargs*)

编辑 BFD 协议会话对象参数

参数 **Session** (BfdRouter) -- BFD 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BFD 协议会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 BFD 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterRole** (*str*) -- BFD 会话的角色, 类型为：string, 默认值：Active, 支持角色：

  Active

  Passive
- **TimeIntervalUnit** (*str*) -- 时间间隔的单位。类型为：string, 默认值：milliseconds, 支持单位：

  milliseconds

  microseconds
- **DesiredMinTXInterval** (*int*) -- 期望的最小发送时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：50
- **RequiredMinRXInterval** (*int*) -- 需要的最小接收时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：50
- **DetectMultiple** (*int*) -- 用于检测超时的时间因子, 类型为：number, 取值范围：2-100, 默认值：3
- **AuthenticationType** (*str*) -- 认证方式, 类型为：string, 默认值：None, 支持的方式：

  NONE

  SIMPLE_PASSWORD

  KEYED_MD5

  METICULOUS_KEYED_MD5

  KEYED_SHA1

  METICULOUS_KEYED_SHA1
- **Password** (*str*) -- 当认证方式不为 NONE 时，在该单元格输入认证密码。密码可以是数字、字母或者数字和字母的组合，最长为 16 位。类型为：string, 默认值：Xinertel
- **KeyID** (*int*) -- 当认证方式不为 NONE 时，在该单元格输入 Key ID, 类型为：number, 取值范围：0-255, 默认值：1

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Bfd | Session=${Session} | EnableViewRoutes=True |
```

TesterLibrary.Protocol.bfd.**get_bfd_ipv4_session_result**(*Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 BFD IPV4 会话统计结果

参数

- **Session** (BfdIpv4SessionConfig) -- BFD IPV4 会话对象, 类型为:Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项:

  Ipv4SessionKeyID

  Ipv4SessionID

  SessionID

  SessionIndex

  Ipv4SourceAddress

  Ipv4DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

  BfdControlBits_PFCADM

  ReceiveCount

  TransmitCount

  TransmitInterval

  ReceivedRequiredMinRXInterval

  ReceivedRequiredMinEchoRXInterval

  FlapsDetected

  TimeoutsDetected

  RXAvgRate

  RXMaxRate

  RXMinRate

  TXAvgRate

  TXMaxRate

  TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BfdIpv4SessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Ipv4 Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bfd.**get_bfd_ipv6_session_result**(*Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 BFD IPV6 会话统计结果

参数

- **Session** (BfdIpv6SessionConfig) -- BFD IPV6 会话对象, 类型为:Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项:

  Ipv6SessionKeyID

  Ipv6SessionID

  SessionID

  SessionIndex

  Ipv6SourceAddress

  Ipv6DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

  BfdControlBits_PFCADM

  ReceiveCount

  TransmitCount

  TransmitInterval

  ReceivedRequiredMinRXInterval

  ReceivedRequiredMinEchoRXInterval

  FlapsDetected

  TimeoutsDetected

RXAvgRate

RXMaxRate

RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BfdIpv6SessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Ipv6 Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bfd.**get_bfd_isis_ipv6_session_result**(*BfdSession*, *IsisSession*, *SessionId*, *StaItems: Optional[list] = None*)

获取 ISIS BFD IPV6 会话统计结果

参数

- **BfdSession** (BfdRouter) -- BFD 会话对象, 类型为：Object
- **IsisSession** (IsisRouter) -- ISIS 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  IpSessionKeyID

  IpSessionID

  SessionID

  SessionIndex

  Ipv6SourceAddress

  Ipv6DestinationAddress

  BfdSessionState

  MyDiscriminator

YourDiscriminator

BfdDiagnostic

LastBfdDiagnosticErrorRx

BfdControlBits_PFCADM

ReceiveCount

TransmitCount

TransmitInterval

ReceivedRequiredMinRXInterval

ReceivedRequiredMinEchoRXInterval

FlapsDetected

TimeoutsDetected

RXAvgRate

RXMaxRate

RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=IsisBfdIpv6SessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Isis Ipv6 Session Statistic | BfdSession=${BfdSession}
↪| IsisSession=${IsisSession} | SessionId=${SessionId} | StaItems=@{StaItems}
↪ |
| Clear Result |
```

TesterLibrary.Protocol.bfd.**get_bfd_isis_session_result**(*BfdSession*, *IsisSession*, *SessionId*, *StaItems: Optional[list] = None*)

获取 ISIS BFD 会话统计结果

参数

- **BfdSession** (BfdRouter) -- BFD 会话对象, 类型为：Object
- **IsisSession** (IsisRouter) -- ISIS 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string

- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  IpSessionKeyID

  IpSessionID

  SessionID

  SessionIndex

  Ipv4SourceAddress

  Ipv4DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

  BfdControlBits_PFCADM

  ReceiveCount

  TransmitCount

  TransmitInterval

  ReceivedRequiredMinRXInterval

  ReceivedRequiredMinEchoRXInterval

  FlapsDetected

  TimeoutsDetected

  RXAvgRate

  RXMaxRate

  RXMinRate

  TXAvgRate

  TXMaxRate

  TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=IsisBfdSessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Isis Session Statistic | BfdSession=${BfdSession} |␣
↪IsisSession=${IsisSession} | SessionId=${SessionId} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bfd.**get_bfd_ospfv2_session_result**(*BfdSession*, *Ospfv2Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 OSPFV2 BFD 会话统计结果

参数

- **Ospfv2Session** (BfdRouter) -- OSPFv2 会话对象, 类型为：Object
- **BfdSession** (OspfRouter) -- BFD 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  Ipv4SessionKeyID

  Ipv4SessionID

  SessionID

  SessionIndex

  Ipv4SourceAddress

  Ipv4DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

  BfdControlBits_PFCADM

  ReceiveCount

  TransmitCount

  TransmitInterval

  ReceivedRequiredMinRXInterval

  ReceivedRequiredMinEchoRXInterval

  FlapsDetected

  TimeoutsDetected

  RXAvgRate

  RXMaxRate

  RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv2BfdSessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Ospfv2 Session Statistic | BfdSession=${BfdSession} |␣
↪Ospfv2Session=${Ospfv2Session} | SessionId=${SessionId} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bfd.**get_bfd_ospfv3_session_result**(*BfdSession*, *Ospfv3Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 OSPFV3 BFD 会话统计结果

参数

- **BfdSession** (BfdRouter) -- BFD 会话对象, 类型为：Object
- **Ospfv3Session** (Ospfv3Router) -- OSPFv3 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  Ipv6SessionKeyID

  Ipv6SessionID

  SessionID

  SessionIndex

  Ipv6SourceAddress

  Ipv6DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

BfdControlBits_PFCADM

ReceiveCount

TransmitCount

TransmitInterval

ReceivedRequiredMinRXInterval

ReceivedRequiredMinEchoRXInterval

FlapsDetected

TimeoutsDetected

RXAvgRate

RXMaxRate

RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv3BfdSessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Ospfv3 Session Statistic | BfdSession=${BfdSession} |␣
↪Ospfv3Session=${Ospfv3Session} | SessionId=${SessionId} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bfd.**get_bfd_session_result**(*Session*, *StaItems: Optional[list] = None*)

获取 BFD 协议会话统计结果

参数

- **Session** (BfdRouter) -- BFD 协议会话对象，类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  SessionID

  SessionState

  BfdSessionUpCount

  BfdSessionDownCount

TXBfdPackets

RXBfdPackets

TimeoutsDetected

FlapsDetected

返回

eg:

```
{
    'TXBfdPackets': 10,
    'RXBfdPackets': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BfdSessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bfd.**wait_bfd_state**(*Sessions*, *State='RUNNING'*, *Interval=1*, *TimeOut=60*)

等待 BFD 协议会话达到指定状态

参数

- **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BFD 协议会话达到的状态, 类型为：string, 默认值：达到 RUNNING, 支持下列状态：

  DISABLED

  NOT_STARTED

  IDLE

  RUNNING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bfd State | Sessions=${Sessions} | State=RUNNING | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.bgp module

TesterLibrary.Protocol.bgp.**advertise_bgp**(*Sessions*)

通告 BGP 协议会话 lsa

参数 **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Bgp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bgp.**advertise_bgp_route**(*Routes*)

通告 BGP 协议指定 lsa

参数 **Routes** (*list*) -- BGP 协议路由对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Bgp Route | Routes=${Routes} |
```

TesterLibrary.Protocol.bgp.**connect_bgp**(*Sessions*)

连接 BGP 协议会话

参数 **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Connect Bgp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bgp.**create_bgp**(*Port*, ***kwargs*)

创建 BGP 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- BGP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：BOTH, 支持版本：

  BOTH

  IPV4

  IPV6

- **BgpInitiator** (*bool*) -- BGP 会话发起者, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AsNumber** (*int*) -- 自治域, 类型为：number, 取值范围：1-65535, 默认值：1
- **AsNumberStep** (*int*) -- 自治域跳变, 类型为：number, 取值范围：0-65535, 默认值：1
- **Enable4ByteAs** (*bool*) -- 使能 4 字节自治域, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AsNumber4Byte** (*int*) -- 4 字节自治域, 类型为：number, 取值范围：0.1-65535.65535, 默认值：1.1
- **AsNumber4ByteStep** (*int*) -- 4 字节自治域跳变, 类型为：number, 取值范围：0.1-65535.65535, 默认值：0.1
- **DutAsNumber** (*int*) -- DUT 自治域, 类型为：number, 取值范围：1-65535, 默认值：1
- **DutAsNumberStep** (*int*) -- DUT 自治域跳变, 类型为：number, 取值范围：1-65535, 默认值：1
- **Enable4ByteDutAs** (*bool*) -- 使能 DUT4 字节自治域, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Dut4ByteAsNumber** (*int*) -- DUT4 字节自治域, 类型为：number, 取值范围：0.1-65535.65535, 默认值：1.1
- **Dut4ByteAsNumberStep** (*int*) -- DUT4 字节自治域跳变, 类型为：number, 取值范围：0.1-65535.65535, 默认值：0.1
- **BgpType** (*str*) -- BGP 类型, 类型为：string, 取值范围：EBGP, IBGP, 默认值：IBGP
- **UseGatewayAsDutIp** (*bool*) -- 使用网关地址作为 DUT 地址, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **BgpSessionIpAddressType** (*str*) -- 会话 IP 类型, 类型为：string, 取值范围：INTERFACE_IP, ROUTE_ID, 默认值：INTERFACE_IP
- **DutIpv4Address** (*str*) -- DUT IPv4 地址, 当 IP 版本为 IPv4，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID, 类型为：string, 取值范围：IPv4 地址, 默认值：2.1.1.1
- **DutIpv4AddressStep** (*str*) -- DUT IPv4 地址跳变, 当 IP 版本为 IPv4，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID 增量步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **DutIpv6Address** (*str*) -- DUT IPv4 地址, 当 IP 版本为 IPv6，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **DutIpv6AddressStep** (*str*) -- DUT IPv4 地址跳变, 当 IP 版本为 IPv6，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID 增量步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **HoldTime** (*int*) -- Hold Time 间隔 (sec), 类型为：number, 取值范围：3-65535, 默认值：90
- **KeepaliveTime** (*int*) -- Keep Alive 间隔 (sec), 类型为：number, 取值范围：1-65535, 默认值：30

- **ConnectRetryCount** (*int*) -- 重连次数, 取值范围：0-65535, 默认值：0
- **ConnectRetryInterval** (*int*) -- 重连间隔 (sec), 取值范围：10-300, 默认值：30
- **MaxRoutesPerUpdateMessage** (*int*) -- Update 报文中最大路由数量, 取值范围：10-300, 默认值：2000
- **RouteRefreshMode** (*str*) -- Route Refresh 模式, 类型为：string, 取值范围：None; Route Refresh, 默认值：None
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RestartTime** (*int*) -- 平滑重启时间（秒）, 取值范围：3-4095, 默认值：90
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Authentication** (*str*) -- 使用的认证类型, 类型为：string, 取值范围：None 或 MD5, 默认值：None
- **Password** (*str*) -- 认证密码, 类型为：类型为：string, 取值范围：字符串，由 1-255 个数字、字母或特殊字符组成, 默认值：xinertel
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableSr** (*bool*) -- 使能 SR, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 BGP 协议会话对, 类型：object

返回类型 (BgpRouter)

实际案例

```
| Create Bgp | Port=${Port} |
```

TesterLibrary.Protocol.bgp.**create_bgp_capability**(*Session*, ***kwargs*)

创建 BGP Capability 对象

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP Capability 名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP Capability, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **CapabilityCode** (*int*) -- Capability Code, 类型为：number, 默认值：1, 取值范围：1-255
- **CapabilityValue** (*str*) -- Capability 值类型为：list

返回 BGP Capability 对象, 类型：object

返回类型 (BgpCapabilityConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${CapabilityValue} | Create List | 1 | 2 | 3 | 4 | 5 |
| Create Bgp Capability | Session=${Session} | CapabilityCode=5 |␣
↪CapabilityValue=${CapabilityValue} |
```

TesterLibrary.Protocol.bgp.**create_bgp_evpn_ethernet_segment_routes**(*Session*, ***kwargs*)

创建 Bgp Evpn Ethernet Segment Routes 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **NextHopIpv6** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **EnableOriginatorId** (*bool*) -- 是否启用 Originator ID, 当仿真路由器作为 BGP 路由反射器时使用该属性, 类型为：bool, 默认值：False
- **OriginatorId** (*str*) -- 指定 originator ID 的值。该值用于标识路由发起者的 router id, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **EthernetSegmentType** (*str*) -- 类型为：string, 默认值：OPERATOR, 取值范围：

  OPERATOR

  IEEE802

  BRIDGEDLAN

  MACBASED

ROUTEID

AS

- **EthernetSegmentIdentifier** (*str*) -- 类型为：string，默认值：00:00:00:00:00:00:00:00:00:00
- **EviCount** (*int*) -- 创建的 EVI （EVPN instance，EVPN 实例）数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **DataPlanEncapsulation** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

  SRv6

- **EsImportRoute** (*str*) -- 类型为：有效的 mac 地址，默认值：00:00:00:00:00:00

返回 Bgp Evpn Ethernet Segment Routes 对象, 类型：object / list

返回类型 (EvpnRouteEthernetSegmentConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Ethernet Segment Routes | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_evpn_inclusive_multicast_routes**(*Session*, ***kwargs*)

创建 Bgp Evpn Inclusive Multicast Routes 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **NextHopIpv6** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **EnableOriginatorId** (*bool*) -- 是否启用 Originator ID, 当仿真路由器作为 BGP 路由反射器时使用该属性, 类型为：bool, 默认值：False

- **OriginatorId** (*str*) -- 指定 originator ID 的值。该值用于标识路由发起者的 router id, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **EviCount** (*int*) -- 创建的 EVI （EVPN instance，EVPN 实例）数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **EthernetTagId** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：1
- **PmsiTunnelType** (*str*) -- 指定多播报文传输所使用隧道的类型。只支持 INGRESS_REPLICATION（头端复制隧道。隧道标识携带本地 PE 的单播隧道目的端 IP 地址）。类型为：string, 默认值：INGRESS_REPLICATION
- **DataPlanEncapsulation** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

  SRv6
- **Label1** (*int*) -- 封装标签（VNI/VSID）, 数据平面封装为 VXLAN 时可见, 指定转发二层业务流量所使用封装标签的起始值, 类型为：number, 取值范围：1-16777215, 默认值：0
- **Label1Step** (*int*) -- 指定转发二层业务流量所使用封装标签的跳变步长, 类型为：number, 取值范围：1-16777215, 默认值：1
- **EnableCustomMplsLabel** (*bool*) -- 使能自定义 MPLS 标签, 类型为：bool, 默认值：False
- **MplsLabel** (*int*) -- MPLS 标签, 类型为：number, 取值范围：1-1048575, 默认值：0
- **MplsLabelStep** (*int*) -- MPLS 标签跳变, 类型为：number, 取值范围：1-1048575, 默认值：0

返回 Bgp Evpn Inclusive Multicast Routes 对象, 类型：object / list

返回类型 (EvpnRouteInclusiveMulticastConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Inclusive Multicast Routes | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_evpn_ip_prefix_routes**(*Session*, ***kwargs*)

创建 Bgp Evpn Ip Prefix Routes 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **NextHopIpv6** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **EnableOriginatorId** (*bool*) -- 是否启用 Originator ID, 当仿真路由器作为 BGP 路由反射器时使用该属性, 类型为：bool, 默认值：False
- **OriginatorId** (*str*) -- 指定 originator ID 的值。该值用于标识路由发起者的 router id, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **EthernetSegmentType** (*str*) -- 指定以太网段标识的类型，用于确定以太网段标识的格式, 类型为：string, 默认值：OPERATOR, 取值范围：

  OPERATOR

  IEEE802

  BRIDGEDLAN

  MACBASED

  ROUTEID

AS

- **EthernetSegmentIdentifier** (*str*) -- 指定 CE 和 PE 之间连接的标识, 类型为：string, 默认值：00:00:00:00:00:00:00:00:00:00
- **EthernetTagId** (*int*) -- 指定广播域（例如 VLAN）的标识, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **NetWorkCount** (*int*) -- 指定要创建的网络数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **EviCount** (*int*) -- 创建的 EVI （EVPN instance，EVPN 实例）数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **VpnDistributionType** (*str*) -- RT、RD 以及 VNI 值在 VPN 之间的分配方式, 类型为：string, 默认值：ROUNDROBIN, 取值范围：

  ROUNDROBIN

  LINEAR
- **DataPlanEncapsulation** (*str*) -- 封装有效负载所使用的头部类型, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

  SRv6
- **EnableCustomMplsLabel** (*bool*) -- 使能自定义 MPLS 标签, 类型为：bool, 默认值：False
- **Label1** (*int*) -- 封装标签（VNI/VSID）, 数据平面封装为 VXLAN 时可见, 指定转发二层业务流量所使用封装标签的起始值, 类型为：number, 取值范围：1-16777215, 默认值：0
- **Label1Step** (*int*) -- 指定转发二层业务流量所使用封装标签的跳变步长, 类型为：number, 取值范围：1-16777215, 默认值：1
- **IpType** (*str*) -- 类型为：string, 默认值：IPV4, 取值范围：

  IPV4

  IPV6
- **StartIpv4Address** (*str*) -- 起始 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：100.0.0.2
- **Ipv4Increment** (*str*) -- IPv4 地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **Ipv4PrefixLength** (*int*) -- IPv4 地址前缀, 类型为：number, 取值范围：1-32, 默认值：24
- **GatewayIp** (*str*) -- 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **StartIpv6Address** (*str*) -- 起始 IPv6 地址, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **Ipv6Increment** (*str*) -- IPv6 地址跳变, 类型为：有效的 ipv6 地址, 默认值：::1
- **Ipv6PrefixLength** (*int*) -- IPv6 地址前缀, 类型为：number, 取值范围：1-128, 默认值：64
- **GatewayIpv6** (*str*) -- 类型为：有效的 ipv6 地址, 默认值：'::'
- **EnableIncludeRouterMac** (*bool*) -- 是否包含路由 MAC, 类型为：bool, 默认值：False

- **RouterMac** (*str*) -- 路由 MAC 地址, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **MplsLabel** (*int*) -- MPLS 标签, 类型为：number, 取值范围：1-1048575, 默认值：0
- **MplsLabelStep** (*int*) -- MPLS 标签跳变, 类型为：number, 取值范围：1-1048575, 默认值：0

返回 Bgp Evpn Ip Prefix Routes 对象, 类型：object / list

返回类型 (EvpnRouteIpPrefixConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Ip Prefix Routes | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_evpn_mac_ip_routes**(*Session*, ***kwargs*)

创建 Bgp Evpn Mac Ip Routes 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为: string, 默认值: IGP, 取值范围:

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **NextHopIpv6** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **EnableOriginatorId** (*bool*) -- 是否启用 Originator ID, 当仿真路由器作为 BGP 路由反射器时使用该属性, 类型为：bool, 默认值：False
- **OriginatorId** (*str*) -- 指定 originator ID 的值。该值用于标识路由发起者的 router id, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为: string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为: string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1

- **EthernetSegmentType** (*str*) -- 指定以太网段标识的类型，用于确定以太网段标识的格式, 类型为：string, 默认值：OPERATOR, 取值范围：

  OPERATOR

  IEEE802

  BRIDGEDLAN

  MACBASED

  ROUTEID

  AS

- **EthernetSegmentIdentifier** (*str*) -- 指定 CE 和 PE 之间连接的标识, 类型为：string, 默认值：00:00:00:00:00:00:00:00:00:00

- **EthernetTagId** (*int*) -- 指定广播域（例如 VLAN）的标识, 类型为：number, 取值范围：1-4294967295, 默认值：0

- **NetWorkCount** (*int*) -- 指定要创建的网络数量, 类型为：number, 取值范围：1-4294967295, 默认值：1

- **StartMacAddress** (*str*) -- 指定路由块中的起始 MAC 地址, 类型为：有效的 mac 地址, 默认值：00:10:01:00:00:01

- **MacIncrement** (*str*) -- 指定路由块中 MAC 地址的跳变步长, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:01

- **EviCount** (*int*) -- 创建的 EVI （EVPN instance，EVPN 实例）数量, 类型为：number, 取值范围：1-4294967295, 默认值：1

- **VpnDistributionType** (*str*) -- RT、RD 以及 VNI 值在 VPN 之间的分配方式, 类型为：string, 默认值：ROUNDROBIN, 取值范围：

  ROUNDROBIN

  LINEAR

- **AssociatedIpType** (*str*) -- 指定通告路由中所携带主机 IP 路由的版本, 类型为：string, 默认值：IPV4, 取值范围：

  NONE

  IPV4

  IPV6

- **DataPlanEncapsulation** (*str*) -- 封装有效负载所使用的头部类型, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

  SRv6

- **EnableMacMobility** (*bool*) -- 使能 MAC 地址迁移, 类型为：bool, 默认值：False

- **StickyStatic** (*bool*) -- MAC 地址是静态 MAC 地址, 类型为：bool, 默认值：False

- **SequenceNumber** (*int*) -- 指定 MAC 迁移扩展团体属性 TLV 中的序列号起始值, 类型为：number, 取值范围：1-4294967295, 默认值：0

- **Label1** (*int*) -- 封装标签（VNI/VSID）, 数据平面封装为 VXLAN 时可见, 指定转发二层业务流量所使用封装标签的起始值, 类型为：number, 取值范围：1-16777215, 默认值：0

- **Label1Step** (*int*) -- 指定转发二层业务流量所使用封装标签的跳变步长, 类型为：number, 取值范围：1-16777215, 默认值：1
- **StartIpv4Address** (*str*) -- 起始 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：100.0.0.2
- **Ipv4Increment** (*str*) -- IPv4 地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **Ipv4PrefixLength** (*int*) -- IPv4 地址前缀, 类型为：number, 取值范围：1-32, 默认值：24
- **StartIpv6Address** (*str*) -- 起始 IPv6 地址, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **Ipv6Increment** (*str*) -- IPv6 地址跳变, 类型为：有效的 ipv6 地址, 默认值：::1
- **Ipv6PrefixLength** (*int*) -- IPv6 地址前缀, 类型为：number, 取值范围：1-128, 默认值：64
- **EnableLabel2** (*bool*) -- 使能 MPLS Label2, 类型为：bool, 默认值：False
- **Label2** (*int*) -- 封装 2 标签 (L3 VNI), 类型为：number, 取值范围：1-16777215, 默认值：2000
- **Label2Step** (*int*) -- 封装 2 标签跳变 (L3 VNI Step), 类型为：number, 取值范围：1-16777215, 默认值：1
- **EnableIncludeRouterMac** (*bool*) -- 是否包含路由 MAC, 类型为：bool, 默认值：False
- **RouterMac** (*str*) -- 路由 MAC 地址, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **EnableIncludeDefaultGateway** (*bool*) -- 指定默认网关, 类型为：bool, 默认值：False
- **EnableCustomMplsLabel** (*bool*) -- 使能自定义 MPLS 标签, 类型为：bool, 默认值：False
- **MplsLabel** (*int*) -- MPLS 标签, 类型为：number, 取值范围：1-1048575, 默认值：0
- **MplsLabelStep** (*int*) -- MPLS 标签跳变, 类型为：number, 取值范围：1-1048575, 默认值：0
- **EnableCustomMplsLabel2** (*bool*) -- 使能自定义 MPLS Label2, 类型为：bool, 默认值：False
- **MplsLabel2** (*int*) -- MPLS 标签 2, 类型为：number, 取值范围：1-1048575, 默认值：0
- **MplsLabel2Step** (*int*) -- MPLS 标签 2 跳变, 类型为：number, 取值范围：1-1048575, 默认值：0

返回 Bgp Evpn Mac Ip Routes 对象, 类型：object / list

返回类型 (EvpnRouteMacIpConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Mac Ip Routes | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_evpn_route_ad**(*Session*, ***kwargs*)

创建 BGP EVPN 以太自动发现路由对象

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP EVPN 以太自动发现路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP EVPN 以太自动发现路由, 类型为：bool, 默认值：True, 取值范围：True 或 False,
- **Origin** (*str*) -- ORIGIN, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- AS 路径的值, 类型为：list
- **AdRouteType** (*str*) -- 以太自动发现路由类型, 类型为：string, 默认值：ESI, 取值范围：

  EVI

  ESI

  VPWS
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True, 取值范围：True 或 False
- **NextHop** (*str*) -- 下一跳, 类型为：string, 默认值：100.0.0.1, 取值范围：IPv4 地址
- **NextHopIpv6** (*str*) -- IPv6 下一跳, 类型为：string, 默认值：2001::1, 取值范围：IPv6 地址
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳, 类型为：string, 默认值：fe80::1, 取值范围：IPv6 地址
- **EnableOriginatorId** (*bool*) -- 使能 Originator ID, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **OriginatorId** (*str*) -- Originator ID, 类型为：string, 默认值：192.0.0.1, 取值范围：IPv4 地址
- **VrfRouteTarget** (*str*) -- VRF 路由目标, 类型为：string, 默认值：100:1, 取值范围：AS:Number
- **VrfRouteTargetStep** (*str*) -- VRF 路由目标跳变, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1, 取值范围：AS:Number 或 IPv4:Number
- **VrfRouteDistinguisherStep** (*str*) -- VRF 路由标识跳变, 类型为：string, 默认值：0:1

- **EthernetSegmentType** (*str*) -- 以太网段类型, 类型为：string, 默认值：OPERATOR, 取值范围：

  OPERATOR

  IEEE802

  BRIDGEDLAN

  MACBASED

  ROUTEID

  AS

- **EthernetSegmentIdentifier** (*str*) -- 以太网段标识, 类型为：string, 默认值：OPERATOR
- **EthernetTagId** (*str*) -- 以太网标签 ID, 类型为：string, 默认值：00:00:00:00:00:00:00:00:00:00
- **EthernetTagIdStep** (*int*) -- 太网标签 ID 跳变, 类型为：number, 默认值：1, 取值范围：0-4294967295
- **EthernetTagCountPerEvi** (*int*) -- 每个 EVI 下以太网标签数, 类型为：number, 默认值：1, 取值范围：1-4294967295
- **ActiveStandbyMode** (*str*) -- 主备方式, 类型为：string, 默认值：SINGLE, 取值范围：

  ALL

  SINGLE

- **EviCount** (*int*) -- EVI 数, 类型为：number, 默认值：1, 取值范围：1-4294967295
- **DataPlanEncapsulation** (*str*) -- 数据平面封装, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

  SRv6

- **Label1** (*int*) -- 封装标签（VNI/VSID）, 类型为：number, 默认值：100, 取值范围：0-16777215
- **Label1Step** (*int*) -- 封装标签跳变（VNI/VSID Step）, 类型为：number, 默认值：1, 取值范围：0-16777215
- **IncludeLayer2AttributeExtendedCommunity** (*bool*) -- 以太自动发现路由类型为 VPWS 时可见, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **PBit** (*bool*) -- P Bit 多归单活场景中，该标志位置 1 表示该 PE 为主 PE, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **BBit** (*bool*) -- B Bit 多归单活场景中，该标志位置 1 表示该 PE 为备 PE, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **CBit** (*bool*) -- C Bit 置 1 时发送给该 PE 的 EVPN 报文必须携带控制字, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **L2Mtu** (*bool*) -- 指定最大传输单元, 单位是字节, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **EnableCustomMplsLabel** (*bool*) -- 使能自定义 MPLS 标签, 类型为：bool, 默认值：False, 取值范围：True 或 False

- **MplsLabel** (*int*) -- MPLS 标签值, 类型为：number, 默认值：0, 取值范围：0-1048575
- **MplsLabelStep** (*int*) -- MPLS 标签步长, 类型为：number, 默认值：0, 取值范围：0-1048575

返回 BGP EVPN 以太自动发现路由对象, 类型：object

返回类型 (EvpnRouteAdConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Route Ad | Session=${Session} | EnableCustomMplsLabel=True
↪| MplsLabel=1000 | MplsLabelStep=2 |
```

TesterLibrary.Protocol.bgp.**create_bgp_flow_spec_conponent_type**(*FlowSpec*, *Types*, ***kwargs*)

创建 Bgp Flow Specs Conponent Type 对象, 类型为：object / list

参数

- **FlowSpec** (BgpIpv4FlowSpecConfig) -- BGP Flow Spec 对象, 类型为：object / list
- **Types** (*int*) -- BGP IPv4 FLowSpec Type 序号, 类型为：number, 取值范围：1-12

关键字参数

- **IpValue** (*str*) -- 指定起始地址, 类型为有效的 ipv4 地址, 默认值: "192.0.1.0"
- **PrefixLength** (*int*) -- 指定前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **AddressList** (*str*) -- 类型为列表时, 指定地址列表, 类型为有效的 ipv4 地址, 默认值: ""
- **InputType** (*str*) -- 指定类型, 类型为：string, 默认值：RANGE, 取值范围：

  RANGE

  LIST

  RFC_4814
- **Count** (*int*) -- 地址数量, 类型为：number, 取值范围：1-99, 默认值：1
- **Step** (*int*) -- 地址跳变步长, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **EqualBit** (*bool*) -- 数据与指定值相等表示匹配, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LessThanBit** (*bool*) -- 数据小于指定值表示匹配, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MoreThanBit** (*bool*) -- 数据大于指定值表示匹配, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AndBit** (*bool*) -- 如果选中 And Bit，则该 {option/value} 组与前一个 {option/value} 组之间的关系是逻辑与（AND）；如果未选中 And Bit，则该 {option/value} 组与前一个 {option/value} 组之间的关系是逻辑或（OR）。类型为：bool, 取值范围：True 或 False, 默认值：False

- **ValueField** (*int*) -- 类型为：number, 取值范围：1-15, 默认值：1
- **Count** -- 类型为：number, 取值范围：0-99, 默认值：1
- **ValueIncrement** (*int*) -- 类型为：number, 取值范围：0-65535, 默认值：1
- **ValueList** (*int*) -- 类型为：number, 取值范围：0-65535, 默认值：""
- **ValueType** (*str*) -- 指定值的类型, 类型为：string, 默认值：Increment, 取值范围：

  Increment

  List
- **NotBit** (*bool*) -- 选中 Not Bit 时，对计算结果按位取反。类型为：bool, 取值范围：True 或 False, 默认值：False
- **MatchBit** (*bool*) -- 选中 Match Bit 时，(data & value) == value 表示按位匹配；未选中 Match Bit 时，如果数据中包含值掩码，则 (data & value) == True 表示匹配。其中，data 是发送的数据，value 是给定的值掩码。类型为：bool, 取值范围：True 或 False, 默认值：False
- **DSCPValue** (*int*) -- 类型为：number, 取值范围：0-63, 默认值：0

返回 Bgp Flow Spec Conponent Type 对象, 类型：object / list

返回类型 (BgpIpv4FlowSpecType1Component)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpec} | Create Bgp Ipv4 Flow Specs | Session=${Session} |
| Create Bgp Flow Spec Conponent Type | FlowSpec=${FlowSpec} | Type=1 |
```

TesterLibrary.Protocol.bgp.**create_bgp_flow_spec_custom_path_attribute**(*FlowSpec*, ***kwargs*)

创建 BGP Flow Spec Custom Path Attribute 对象, 类型为：object / list

参数 **FlowSpec** (BgpIpv4FlowSpecConfig) -- 所属的 Bgp Flow Spec 对象, 类型为：object / list

关键字参数

- **PathAttributeType** (*int*) -- 路径属性的类型, 类型为：number, 取值范围：1-255, 默认值：1
- **OptionalFlag** (*str*) -- 指定 Optional Flag 的值, 类型为：string, 默认值：OPTION, 取值范围：

  WELL_KNOWN

  OPTION
- **TransitiveFlag** (*str*) -- 指定 Transitive Flag 的值, 类型为：string, 默认值：NONTRANSITIVE, 取值范围：

  NONTRANSITIVE

  TRANSITIVE
- **PartialFlag** (*str*) -- 指定 Partial Flag 的值, 类型为：string, 默认值：PARTIAL, 取值范围：

  COMPLETE

  PARTIAL

- **ExtendedLengthFlag** (*bool*) -- 是否启用 Extended Length Flag, 类型为：bool, 默认值：False
- **AttributeValue** (*str*) -- 指定路径属性的值, 类型为：string, 默认值：""

返回 Bgp Route Pool Custom Path Attribute 对象, 类型：object / list

返回类型 (BgpFlowSpecPathAttributeConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpecs} | Create Bgp Ipv4 Flow Specs | Session=${Session} |
| Create Bgp Flow Spec Custom Path Attribute | FlowSpec=${FlowSpecs} |
```

TesterLibrary.Protocol.bgp.**create_bgp_flow_specs_actions**(*FlowSpec*, ***kwargs*)

创建 Bgp Ipv4 Flow Specs Actions 对象, 类型为：object / list

参数 **FlowSpec** (BgpIpv4FlowSpecConfig) -- BGP Flow Spec 对象, 类型为：object / list

关键字参数

- **EnableTrafficRate** (*bool*) -- 启用流量限速动作。类型为：bool, 取值范围：True 或 False, 默认值：True
- **TrafficRate** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：0
- **AsNum** (*int*) -- 指定 AS 号。类型为：number, 取值范围：1-4294967295, 默认值：1
- **EnableTrafficAction** (*bool*) -- 启用 Traffic action。类型为：bool, 取值范围：True 或 False, 默认值：True
- **SampleBit** (*bool*) -- 启用流量抽样记录。类型为：bool, 取值范围：True 或 False, 默认值：True
- **TerminateBit** (*bool*) -- 撤销已生效的匹配规则。类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableRedirect** -- 启用流量重定向到指定的路由目标动作。类型为：bool, 取值范围：True 或 False, 默认值：False
- **RouteTarget** (*str*) -- 指定重定向的路由目标。类型为：string, 默认值："100:1"
- **EnableTrafficMarking** (*bool*) -- 启用重新标记报文 DSCP 值的动作。类型为：bool, 取值范围：True 或 False, 默认值：True
- **DSCPValue** (*int*) -- 以十六进制形式指定重新标记报文所使用的 DSCP 值。类型为：number, 取值范围：1-4294967295, 默认值：0
- **EnableRedirectToIpNextHop** (*bool*) -- 启用重定向到下一跳动作, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NextHop** (*str*) -- 指定下一跳 IP, 类型为有效的 ipv4 地址, 默认值："0.0.1.0"
- **CopyBit** (*bool*) -- 复制一份规则匹配的流量，并执行重定向到下一跳动作。类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Bgp Ipv4 Flow Specs Actions 对象, 类型：object / list

返回类型 (BgpIpv4FlowSpecAction)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpec} | Create Bgp Ipv4 Flow Specs | Session=${Session} |
| Create Bgp Ipv4 Flow Specs Action | FlowSpec=${FlowSpec} |
```

TesterLibrary.Protocol.bgp.**create_bgp_ipv4_flow_specs**(*Session*, ***kwargs*)

创建 Bgp Ipv4 Flow Specs 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **RouteCount** (*int*) -- 类型为：number, 取值范围：1-8000000, 默认值：1
- **FlowSpecSubAfi** (*str*) -- 指定 SubAFI 的值, 类型为：string, 默认值：FlowSpec, 取值范围：

  FlowSpec

  FlowSpecVpn
- **Origin** (*str*) -- 指定 ORIGIN 的值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE
- **EnableLocalPref** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalPref** (*int*) -- 类型为：number, 默认值：10
- **EnableMed** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MultExitDisc** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：0
- **EnableClusterIdList** (*bool*) -- 是否启用 Cluster ID List, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ClusterIdList** (*str*) -- 指定 cluster ID list 的值, 类型为：有效的 ipv4 地址, 默认值：""
- **EnableCommunity** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CommunityType** (*str*) -- 团体类型, 类型为：string, 默认值：AA_NN, 取值范围：

  AA_NN

  NO_EXPORT

  NO_ADVERTISE

  LOCAL_AS
- **CommunityAsNumber** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 AS 号, 类型为：number, 取值范围：1-65535, 默认值：1

- **CommunityId** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 ID 值, 类型为：number, 取值范围：1-65535, 默认值：1
- **Community** (*str*) -- 当 Type 为 NO_EXPORT 时，团体值为 0xffffff01；当 Type 为 NO_ADVERTISE 时，团体值为 0xffffff02；当 Type 为 LOCAL_AS 时，团体值为 0xffffff03, 类型为：string, 默认值：""
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：string, 默认值：""
- **ComponentType** (*list*) -- 过滤规则, 类型为：list, 默认值：Type1, 取值范围：

  Type1

  Type2

  Type3

  Type4

  Type5

  Type6

  Type7

  Type8

  Type9

  Type10

  Type11

  Type12
- **VrfNum** (*int*) -- VRF 数量, 类型为：number, 默认值：1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：1:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1

返回 Bgp Ipv4 Flow Specs 对象, 类型：object / list

返回类型 (BgpIpv4FlowSpecConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Ipv4 Flow Specs | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_ipv4_route_pool**(*Session*, ***kwargs*)

创建 BGP IPv4 路由对象

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP IPv4 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP IPv4 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **SubAfi** (*str*) -- Sub-AFI, 类型为：string, 默认值：UNICAST, 支持类型：

  UNICAST

  MULTICAST

  VPN

  LABELED

- **Origin** (*str*) -- 路由属性中的 ORIGIN 值, 类型为：string, 取值范围：Incomplete; IGP; EGP, 默认值：IGP
- **AsPath** (*int*) -- AS Path, 类型为：number, 取值范围：1-255,
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 取值范围：Incomplete; IGP; EGP, 默认值：IGP
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableLocalPref** (*bool*) -- 使能 Local Preference, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalPref** (*int*) -- LocalPref, 当使能 Local Preference 为选中状态时配置该选项, 类型为：number, 取值范围：0-4294967295, 默认值：10
- **EnableMed** (*bool*) -- 使能 Multi Exit Discriminator, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MultExitDisc** (*int*) -- Multi Exit Discriminator, 当使能 Multi Exit Discriminator 为选中状态时配置该选项, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **AtomicAggregate** (*bool*) -- 使能 Atomic Aggregate, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableAggregator** (*bool*) -- 使能 Aggregator, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AggregatorAsNumber** (*int*) -- Aggregator 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **AggregatorIp** (*str*) -- Aggregator IP, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **EnableOriginatorId** (*bool*) -- 使能 Originator ID, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **OriginatorId** (*str*) -- Originator ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **EnableClusterIdList** (*bool*) -- 使能 Cluster ID List, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ClusterIdList** (*str*) -- Cluster ID List, 类型为：string, 取值范围：IPv4 地址, 默认值：空
- **EnableCommunity** (*bool*) -- 使能团体, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CommunityType** (*str*) -- 团体类型, 类型为：string 默认值：AA:NN, 取值范围：

  AA_NN

  NO_EXPORT

  NO_ADVERTISE

  LOCAL_AS

- **CommunityAsNumber** (*int*) -- 团体自治域, 当 Type 为 AA:NN 时，指定团体的 AS 号, 类型为：number, 取值范围：1-65535, 默认值：1
- **CommunityId** (*int*) -- 团体 ID, 当 Type 为 AA:NN 时，指定团体的 ID 值, 类型为：number, 取值范围：1-65535, 默认值：1
- **Community** (*str*) -- 团体, 类型为：list, 默认值：[]
- **CommunityIncrement** (*str*) -- 团体跳变, 类型为：list, 默认值：[]
- **CommunityPerBlockCount** (*int*) -- 每个路由组团体数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：list, 默认值：[], 例如：['0x00:0x02:1:1', '0x01:0x02:1:2', '0x02:0x02:1:3']
- **VrfRd** (*str*) -- VRF 路由标识, 类型为：string, 取值范围：AS:Number 或 IPv4:Number, 默认值：1:1
- **VrfRdStep** (*str*) -- VRF 路由标识跳变, 类型为：string, 取值范围：AS:Number 或 IPv4:Number, 默认值：0:1
- **VrfRt** (*str*) -- VRF 路由目标, 类型为：string, 取值范围：AS:Number, 默认值：100:1
- **VrfRtStep** (*str*) -- VRF 路由目标跳变, 类型为：string, 取值范围：AS:Number, 默认值：0:1
- **VrfCount** (*int) (int*) -- VRF 数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartingLabel** (*int*) -- 起始标签, 类型为：number, 取值范围：0-1048575, 默认值：16
- **LabelType** (*str*) -- 路由标签类型, 类型为：类型为：string, 取值范围：Fixed; Incrementa; Explicit Nul; Implicit Null, 默认值：Fixed
- **FirstRoute** (*str*) -- IPv4 路由起始值, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **RandomRoute** (*bool*) -- 随机路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RouteCount** (*int*) -- 每个会话路由数量, 类型为：number, 取值范围：1-8000000, 默认值：1
- **RouteStep** (*str*) -- IPv4 路由跳变步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.1.0
- **Ipv4RouteStep** (*int*) -- IPv4 路由跳变步长, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **PrefixLength** (*int*) -- IPv4 路由前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **NextHopAddrType** (*str*) -- 下一跳地址类型, 类型为：string, 取值范围：IPv4; IPv6, 默认值：IPv4
- **NextHop** (*str*) -- 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **NextHopStep** (*str*) -- 下一跳步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **IPv6NextHop** (*str*) -- IPv6 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：2000::1
- **IPv6NextHopStep** (*str*) -- IPv6 下一跳步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1

- **EnableLinkLocalNextHop** (*bool*) -- 使能 Link Local 地址作为下一跳, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Ipv6LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：2000::1
- **Ipv6LinkLocalNextHopStep** (*str*) -- IPv6 Link Local 下一跳步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **EncodeSrTlvs** (*list*) -- 编码 SR TLV, 类型为：list, 默认值：0, 取值范围：

  LABEL_INDEX

  SRGB

  SRV6_VPN_SID

  SRV6_SERVICES
- **OverrideGlobalSrgb** (*bool*) -- 覆盖全局 SRGB, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SrgbBase** (*int*) -- SRGB 起始值, 类型为：number, 取值范围：0-16777215, 默认值：16000
- **SrgbRange** (*int*) -- SRGB 范围, 类型为：number, 取值范围：0-16777215, 默认值：1000
- **LabelIndex** (*int*) -- 标签序号, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **LabelStep** (*int*) -- 标签步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **Srv6SidInfoSubTlvType** (*int*) -- SRv6 SID Information Sub TLV 类型, 类型为：number, 取值范围：1-255, 默认值：1
- **Srv6LocatorBlockLength** (*int*) -- SRv6 Locator Block 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **Srv6LocatorNodeLength** (*int*) -- SRv6 Locator Node 长度, 类型为：number, 取值范围：1-128, 默认值：32
- **Srv6FuncLength** (*int*) -- SRv6 Function 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **Srv6FuncOpcode** (*str*) -- SRv6 Function Opcode, 类型为：string, 取值范围：格式：ff:ff:ff, 默认值：0
- **Srv6ArguLength** (*int*) -- SRv6 Argument 长度, 类型为：number, 取值范围：1-128, 默认值：32
- **Srv6Argument** (*str*) -- SRv6 Argument, 类型为：string, 取值范围：格式：ff:ff:ff, 默认值：0
- **EncodedSrv6ServiceDataSubTlvs** (*list*) -- 编码 SRv6 Service Data Sub TLVs, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  SRV6_ID_STRUCTURE
- **Srv6TranspositionLength** (*int*) -- SRv6 Transposition 长度, 类型为：number, 取值范围：0-24, 默认值：0
- **Srv6TranspositionOffset** (*int*) -- SRv6 Transposition 偏移, 类型为：number, 取值范围：0-15, 默认值：0
- **Srv6Locator** (*str*) -- 使用的认证类型, 类型为：string, 取值范围：None 或 MD5, 默认值：None

- **Srv6LocatorStep** (*str*) -- 认证密码, 类型为：类型为：string, 取值范围：字符串，由 1-255 个数字、字母或特殊字符组成, 默认值：xinertel
- **Srv6EndpointBehavior** (*str*) -- SRv6 Endpoint Behavior, 类型为：string, 默认值：CUSTOM, 支持的值：

  END_DX6

  END_DX4

  END_DT6

  END_DT4

  END_DT46

  END_DX2

  END_DX2V

  END_DT2U

  END_DT2M

  CUSTOM
- **Srv6CustomEndpointBehavior** (*int*) -- 自定义 SRv6 Endpoint Behavior, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0xFFFF

返回 BGP IPv4 路由对象, 类型：object

返回类型 (BgpIpv4RoutepoolConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${Community} | Create List | AA_NN | NO_EXPORT | NO_ADVERTISE | LOCAL_AS |
| ${CommunityIncrement} | Create List | 1:1 | 1:2 | 1:3 | 1:4 |
| ${ExtendedCommunity} | Create List | 0x00:0x02:1:1 | 0x01:0x02:1:2 |␣
↪0x02:0x02:1:3 |
| ${RoutePool} | Create Bgp Ipv4 Route Pool | Session=${Session} | Community=$
↪{Community} | CommunityIncrement=${CommunityIncrement} | ExtendedCommunity=$
↪{ExtendedCommunity} |
| ${Community} | Create List | AA_NN | NO_EXPORT |
| ${CommunityIncrement} | Create List | 2:1 | 2:2 |
| Edit Configs | Configs=${RoutePool} | Community=${Community} |␣
↪CommunityIncrement=${CommunityIncrement} | CommunityPerBlockCount=2 |
```

TesterLibrary.Protocol.bgp.**create_bgp_ipv4_vpls**(*Session*, ***kwargs*)

创建 Bgp Ipv4 Vpls 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE

  CONFED_SEQUENCE

  CONFED_SET

- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **MultExitDisc** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LocalPreference** (*int*) -- Local 优先级, 类型为：number, 取值范围：1-4294967295, 默认值：10
- **VeId** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **VeIdStep** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：1
- **BlockOffset** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockOffsetStep** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockSize** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：5
- **MtuSize** (*int*) -- 类型为：number, 取值范围：64-9000, 默认值：1500
- **EncapType** (*str*) -- 封装类型, 类型为：string, 默认值：VLAN, 取值范围：

  VLAN

  ETHERNET

  VPLS
- **ControlFlag** (*int*) -- 控制标识。以十进制表示。类型为：number, 取值范围：1-255, 默认值：0
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **VrfCount** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：1

返回 Bgp Ipv4 Vpls 对象, 类型：object / list

返回类型 (BgpIpv4VplsConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Ipv4 Vpls | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_ipv6_flow_spec**(*Session*, ***kwargs*)

创建 BGP Ipv6 Flow Spec 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：object / list

关键字参数

- **RouteCount** (*int*) -- 类型为：number, 取值范围：1-8000000, 默认值：1

- **FlowSpecSubAfi** (*str*) -- 指定 SubAFI 的值, 类型为：string, 默认值：FlowSpec, 取值范围：

  FlowSpec

  FlowSpecVpn

- **FlowSpecActionType** (*str*) -- 指定 Optional Flag 的值, 类型为：string, 默认值：RedirectRT, 取值范围：

  RedirectRT

- **ComponentType** (*list*) -- 指定 Transitive Flag 的值, 类型为：list, 默认值：Type1, 取值范围：

  Type1

  Type2

- **DestinationPrefix** (*str*) -- 指定 Partial Flag 的值, 类型为：有效的 ipv6 地址, 默认值：2000::1

- **DestinationPrefixLength** (*bool*) -- 是否启用 Extended Length Flag, 类型为：bool, 默认值：False

- **DestinationPrefixIncrement** (*int*) -- 指定路径属性的长度, 类型为：number, 取值范围：1-128, 默认值：64

- **DestinationPrefixCount** (*int*) -- 指定路径属性的值, 类型为：number, 取值范围：1-65535, 默认值：1

- **DestinationPrefixOffset** (*int*) -- 目的 IP 前缀偏移, 类型为：number, 默认值：0

- **SourcePrefix** (*str*) -- 源 IP 前缀, 类型为：有效的 ipv6 地址, 默认值：2000::1

- **SourcePrefixLength** (*int*) -- 源 IP 前缀长度, 类型为：number, 取值范围：1-128, 默认值：64

- **SourcePrefixIncrement** (*int*) -- 源 IP 前缀偏移, 类型为：number, 取值范围：1-4294967295, 默认值：1

- **SourcePrefixCount** (*int*) -- 源 IP 前缀个数, 类型为：number, 取值范围：1-65535, 默认值：1

- **SourcePrefixOffset** (*int*) -- 源 IP 前缀偏移, 类型为：number, 取值范围：0-255, 默认值：0

- **Origin** (*str*) -- 指定 ORIGIN 的值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：””

- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE

- **EnableLocalPref** (*bool*) -- 是否启用 Local_PREF 属性。类型为：bool, 取值范围：True 或 False, 默认值：True

- **LocalPref** (*int*) -- 指定 Local_PREF 的值。类型为：number, 取值范围：1-4294967295, 默认值：10

- **EnableMed** (*bool*) -- 是否启用 MULTI_EXIT_DISC 属性。类型为：bool, 取值范围：True 或 False, 默认值：False
- **MultExitDisc** (*int*) -- 指定 Multi Exit Discriminator 的值。类型为：number, 取值范围：1-4294967295, 默认值：0
- **EnableClusterIdList** (*bool*) -- 是否启用 Cluster ID List, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ClusterIdList** (*str*) -- 指定 cluster ID list 的值, 类型为：有效的 ipv4 地址, 默认值：""
- **EnableCommunity** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CommunityType** (*str*) -- 团体类型, 类型为：string, 默认值：AA_NN, 取值范围：

  AA_NN

  NO_EXPORT

  NO_ADVERTISE

  LOCAL_AS
- **CommunityAsNumber** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 AS 号, 类型为：number, 取值范围：1-65535, 默认值：1
- **CommunityId** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 ID 值, 类型为：number, 取值范围：1-65535, 默认值：1
- **Community** (*str*) -- 当 Type 为 NO_EXPORT 时，团体值为 0xffffff01；当 Type 为 NO_ADVERTISE 时，团体值为 0xffffff02；当 Type 为 LOCAL_AS 时，团体值为 0xffffff03, 类型为：string, 默认值：""
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：string, 默认值：""
- **VrfNum** (*int*) -- VRF 数量, 类型为：number, 默认值：1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：1:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1

返回 Bgp Ipv6 Flow Spec 对象, 类型：object / list

返回类型 (BgpIpv6FlowSpecConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpecs} | Create Bgp Ipv6 Flow Specs | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_ipv6_flow_spec_action**(*FlowSpec*, ***kwargs*)

创建 Bgp Ipv6 Flow Specs Actions 对象, 类型为：object / list

参数 **FlowSpec** (BgpIpv6FlowSpecConfig) -- BGP Flow Spec 对象, 类型为：object / list

关键字参数

- **EnableTrafficRate** (*bool*) -- 启用流量限速动作。类型为：bool, 取值范围：True 或 False, 默认值：True
- **TrafficRate** (*int*) -- 指定流量的最大传输速率, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **AsNum** (*int*) -- 指定 AS 号。类型为：number, 取值范围：1-4294967295, 默认值：1
- **EnableTrafficAction** (*bool*) -- 启用 Traffic action。类型为：bool, 取值范围：True 或 False, 默认值：True
- **SampleBit** (*bool*) -- 启用流量抽样记录。类型为：bool, 取值范围：True 或 False, 默认值：True
- **TerminateBit** (*bool*) -- 撤销已生效的匹配规则。类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableRedirect** (*bool*) -- 启用流量重定向到指定的路由目标动作。类型为：bool, 取值范围：True 或 False, 默认值：False
- **RedirectIpv6RouteTarget** (*str*) -- 指定重定向的路由目标。类型为：string, 默认值："100:1"
- **EnableRedirectToIpv6NextHop** (*bool*) -- 启用重定向到下一跳动作, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Type** (*str*) -- 类型为：string, 默认值："TYPE_0x000c", 取值范围：

  TYPE_0x0800

  TYPE_0x000c
- **NextHop** (*str*) -- 指定下一跳 IP, 类型为有效的 ipv6 地址, 默认值: "2000::1"
- **CopyBit** (*bool*) -- 复制一份规则匹配的流量，并执行重定向到下一跳动作。类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Bgp Ipv6 Flow Specs Actions 对象, 类型：object / list

返回类型 (BgpIpv6FlowSpecAction)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpec} | Create Bgp Ipv6 Flow Specs | Session=${Session} |
| Create Bgp Ipv6 Flow Specs Action | FlowSpec=${FlowSpec} |
```

TesterLibrary.Protocol.bgp.**create_bgp_ipv6_route_pool**(*Session*, ***kwargs*)

创建 BGP IPv6 路由对象

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP IPv6 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP IPv6 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SubAfi** (*str*) -- Sub-AFI, 类型为：string, 默认值：UNICAST, 支持类型：

  UNICAST

  MULTICAST

  VPN

LABELED

- **Origin** (*str*) -- 路由属性中的 ORIGIN 值, 类型为：string, 取值范围：Incomplete; IGP; EGP, 默认值：IGP
- **AsPath** (*int*) -- AS Path, 类型为：number, 取值范围：1-255,
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 取值范围：Incomplete; IGP; EGP, 默认值：IGP
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableLocalPref** (*bool*) -- 使能 Local Preference, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalPref** (*int*) -- LocalPref, 当使能 Local Preference 为选中状态时配置该选项, 类型为：number, 取值范围：0-4294967295, 默认值：10
- **EnableMed** (*bool*) -- 使能 Multi Exit Discriminator, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MultExitDisc** (*int*) -- Multi Exit Discriminator, 当使能 Multi Exit Discriminator 为选中状态时配置该选项, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **AtomicAggregate** (*bool*) -- 使能 Atomic Aggregate, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableAggregator** (*bool*) -- 使能 Aggregator, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AggregatorAsNumber** (*int*) -- Aggregator 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **AggregatorIp** (*str*) -- Aggregator IP, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **EnableOriginatorId** (*bool*) -- 使能 Originator ID, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **OriginatorId** (*str*) -- Originator ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **EnableClusterIdList** (*bool*) -- 使能 Cluster ID List, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ClusterIdList** (*str*) -- Cluster ID List, 类型为：string, 取值范围：IPv4 地址, 默认值：空
- **EnableCommunity** (*bool*) -- 使能团体, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CommunityType** (*str*) -- 团体类型, 类型为：string, 取值范围：AA:NN; NO_EXPORT; NO_ADVERTISE; LOCAL_AS, 默认值：AA:NN
- **CommunityAsNumber** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 AS 号, 类型为：number, 取值范围：1-65535, 默认值：1
- **Community** (*str*) -- 团体, 类型为：list, 默认值：[], 取值范围：

  AA_NN

  NO_EXPORT

  NO_ADVERTISE

  LOCAL_AS

- **CommunityIncrement** (*str*) -- 团体跳变, 类型为：list, 默认值：[], 取值范围：AA:NN

- **CommunityPerBlockCount** (*int*) -- 每个路由组团体数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：list, 默认值：[], 例如：['0x00:0x02:1:1', '0x01:0x02:1:2', '0x02:0x02:1:3']
- **VrfRd** (*str*) -- VRF 路由标识, 类型为：string, 取值范围：AS:Number 或 IPv4:Number, 默认值：1:1
- **VrfRdStep** (*str*) -- VRF 路由标识跳变, 类型为：string, 取值范围：AS:Number 或 IPv4:Number, 默认值：0:1
- **VrfRt** (*str*) -- VRF 路由目标, 类型为：string, 取值范围：AS:Number, 默认值：100:1
- **VrfRtStep** (*str*) -- VRF 路由目标跳变, 类型为：string, 取值范围：AS:Number, 默认值：0:1
- **VrfCount** (*int*) -- VRF 数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartingLabel** (*int*) -- 起始标签, 类型为：number, 取值范围：0-1048575, 默认值：16
- **LabelType** (*str*) -- 路由标签类型, 类型为：类型为：string, 取值范围：Fixed; Incrementa; Explicit Nul; Implicit Null, 默认值：Fixed
- **FirstIpv6Route** (*str*) -- IPv6 路由起始值, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **RouteCount** (*int*) -- 每个会话路由数量, 类型为：number, 取值范围：1-8000000, 默认值：1
- **RouteStep** (*str*) -- IPv6 路由跳变步长, 类型为：string, 取值范围：IPv6 地址, 默认值：'0:0:0:1::'
- **Ipv6RouteStep** (*int*) -- IPv6 路由跳变步长, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **PrefixLength** (*int*) -- IPv4 路由前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **NextHopAddrType** (*str*) -- 下一跳地址类型, 类型为：string, 取值范围：IPv4; IPv6, 默认值：IPv4
- **NextHop** (*str*) -- 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **NextHopStep** (*str*) -- 下一跳步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **IPv6NextHop** (*str*) -- IPv6 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：2000::1
- **IPv6NextHopStep** (*str*) -- IPv6 下一跳步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 Link Local 地址作为下一跳, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Ipv6LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：2000::1
- **Ipv6LinkLocalNextHopStep** (*str*) -- IPv6 Link Local 下一跳步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **EncodeSrTlvs** (*list*) -- 编码 SR TLV, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

SRV6_VPN_SID

SRV6_SERVICES

- **OverrideGlobalSrgb** (*bool*) -- 覆盖全局 SRGB, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SrgbBase** (*int*) -- SRGB 起始值, 类型为：number, 取值范围：0-16777215, 默认值：16000
- **SrgbRange** (*int*) -- SRGB 范围, 类型为：number, 取值范围：0-16777215, 默认值：1000
- **LabelIndex** (*int*) -- 标签序号, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **LabelStep** (*int*) -- 标签步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **Srv6SidInfoSubTlvType** (*int*) -- SRv6 SID Information Sub TLV 类型, 类型为：number, 取值范围：1-255, 默认值：1
- **Srv6LocatorBlockLength** (*int*) -- SRv6 Locator Block 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **Srv6LocatorNodeLength** (*int*) -- SRv6 Locator Node 长度, 类型为：number, 取值范围：1-128, 默认值：32
- **Srv6FuncLength** (*int*) -- SRv6 Function 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **Srv6FuncOpcode** (*str*) -- SRv6 Function Opcode, 类型为：string, 取值范围：格式：ff:ff:ff, 默认值：0
- **Srv6ArguLength** (*int*) -- SRv6 Argument 长度, 类型为：number, 取值范围：1-128, 默认值：32
- **Srv6Argument** (*str*) -- SRv6 Argument, 类型为：string, 取值范围：格式：ff:ff:ff, 默认值：0
- **EncodedSrv6ServiceDataSubTlvs** (*list*) -- 编码 SRv6 Service Data Sub TLVs, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  SRV6_ID_STRUCTURE
- **Srv6TranspositionLength** (*int*) -- SRv6 Transposition 长度, 类型为：number, 取值范围：0-24, 默认值：0
- **Srv6TranspositionOffset** (*int*) -- SRv6 Transposition 偏移, 类型为：number, 取值范围：0-15, 默认值：0
- **Srv6Locator** (*str*) -- 使用的认证类型, 类型为：string, 取值范围：None 或 MD5, 默认值：None
- **Srv6LocatorStep** (*str*) -- 认证密码, 类型为：类型为：string, 取值范围：字符串，由 1-255 个数字、字母或特殊字符组成, 默认值：xinertel
- **Srv6EndpointBehavior** (*str*) -- SRv6 Endpoint Behavior, 类型为：string, 默认值：CUSTOM, 支持的值：

  END_DX6

  END_DX4

  END_DT6

  END_DT4

  END_DT46

END_DX2

END_DX2V

END_DT2U

END_DT2M

CUSTOM

- **Srv6CustomEndpointBehavior** (*int*) -- 自定义 SRv6 Endpoint Behavior, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0xFFFF

返回 BGP IPv6 路由对象, 类型：object

返回类型 (BgpIpv6RoutepoolConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${Community} | Create List | AA_NN | NO_EXPORT | NO_ADVERTISE | LOCAL_AS |
| ${CommunityIncrement} | Create List | 1:1 | 1:2 | 1:3 | 1:4 |
| ${ExtendedCommunity} | Create List | 0x00:0x02:1:1 | 0x01:0x02:1:2 |␣
↪0x02:0x02:1:3 |
| ${RoutePool} | Create Bgp Ipv6 Route Pool | Session=${Session} | Community=$
↪{Community} | CommunityIncrement=${CommunityIncrement} | ExtendedCommunity=$
↪{ExtendedCommunity} |
| ${Community} | Create List | AA_NN | NO_EXPORT |
| ${CommunityIncrement} | Create List | 2:1 | 2:2 |
| Edit Configs | Configs=${RoutePool} | Community=${Community} |␣
↪CommunityIncrement=${CommunityIncrement} | CommunityPerBlockCount=2 |
```

TesterLibrary.Protocol.bgp.**create_bgp_ipv6_vpls**(*Session*, ***kwargs*)

创建 Bgp Ipv6 Vpls 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **LinkLocalNextHop** (*str*) -- 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **MultExitDisc** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LocalPreference** (*int*) -- Local 优先级, 类型为：number, 取值范围：1-4294967295, 默认值：10
- **VeId** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：1
- **VeIdStep** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockOffset** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockOffsetStep** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockSize** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：5
- **MtuSize** (*int*) -- 类型为：number, 取值范围：64-65535, 默认值：1500

- **EncapType** (*str*) -- 封装类型, 类型为：string, 默认值：VLAN, 取值范围：

  VLAN

  ETHERNET

  VPLS

- **EnableRfc4761** (*bool*) -- 类型为：bool, 默认值：True
- **ControlFlag** (*int*) -- 控制标识。以十进制表示。类型为：number, 默认值：0
- **StripVlan** (*bool*) -- 类型为：bool, 默认值：False
- **VeFlooding** (*bool*) -- 类型为：bool, 默认值：False
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **VrfCount** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：1

返回 Bgp Ipv6 Vpls 对象, 类型：object / list

返回类型 (BgpIpv6VplsConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Ipv6 Vpls | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_link_states**(*Session*, ***kwargs*)

创建 Bgp Link States 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 路由产生源, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE

- **AsPath** (*str*) -- 自治域路径, 类型为：string, 默认值：""
- **AsPathType** (*str*) -- 自治域路径类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE

  CONFED_SEQUENCE

  CONFED_SET

- **NextHop** (*str*) -- 下一跳, 类型为：有效的 ipv4 地址, 默认值：10.0.0.1

- **IPv6NextHop** (*str*) -- IPv6 下一跳, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能下一跳本地链路地址, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LinkLocalNextHop** (*str*) -- 下一跳本地链路地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **LocalPreference** (*int*) -- 本地优先级, 类型为：number, 默认值：10
- **Community** (*str*) -- 团体, 当 Type 为 NO_EXPORT 时, 团体值为 0xffffff01；当 Type 为 NO_ADVERTISE 时，团体值为 0xffffff02；当 Type 为 LOCAL_AS 时，团体值为 0xffffff03, 类型为：string, 默认值：""
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：string, 默认值：""
- **ProtocolID** (*str*) -- 协议 ID, 类型为：string, 默认值：DIRECT, 取值范围：

  ISIS_LEVEL_1

  ISIS_LEVEL_2

  OSPFV2

  DIRECT

  STATIC

  OSPFV3

  BGP
- **IdentifierType** (*str*) -- 标识符类型, 类型为：string, 默认值：CUSTOMIZED, 取值范围：

  DEFAULT_LAYER3

  CUSTOMIZED
- **Identifier** (*int*) -- 标识符, 类型为：number, 取值范围：1-1048575, 默认值：0
- **EnableNodeNLRI** (*bool*) -- 使能节点 NLRI, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalNodeDescriptorFlag** (*list*) -- 本地节点描述符标志, 类型为：list, 默认值：IGP_ROUTER_ID, 取值范围：

  AS_NUMBER

  BGPLS_IDENTIFIER

  OSPF_AREA_ID

  IGP_ROUTER_ID

  BGP_ROUTER_ID

  MEMBER_ASN
- **AsNumber** (*int*) -- 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **BgpLsIdentifier** (*str*) -- BGP-LS 标识符, 类型为：有效的 ipv4 地址, 默认值：1.0.0.1
- **OspfAreaId** (*str*) -- OSPF 区域 ID, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0

- **IGPRouterIdType** (*str*) -- IGP 路由 ID 类型, 类型为：string, 默认值：OSPF_NONPSEUDONODE, 取值范围：

  ISIS_NONPSEUDONODE

  ISIS_PSEUDONODE

  OSPF_NONPSEUDONODE

  OSPF_PSEUDONODE

- **IsisNonPseud** (*str*) -- ISIS 非伪节点路由 ID, 类型为：string, 默认值：”00:10:12:00:00:01”
- **IsisPseud** (*str*) -- ISIS 伪节点路由 ID, 类型为：string, 默认值：”00:10:12:00:00:01.02”
- **OspfNonPseud** (*str*) -- OSPF 非伪节点路由 ID, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **OspfPseud** (*str*) -- OSPF 伪节点路由 ID, 类型为：string, 默认值：”192.0.0.1:192.0.0.1”
- **BgpRouterId** -- BGP 路由 ID, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **MemberAsn** (*int*) -- 自治域成员, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LocalNodeAttributeFlag** (*str*) -- 本地节点描述符标志类型为：string, 默认值：NODE_FLAG_BITS, 取值范围：

  MULTI_TOPO_ID

  NODE_FLAG_BITS

  NODE_NAME

  ISIS_AREA_ID

  IPV4_LOCAL_NODE_ROUTERID

  IPV6_LOCAL_NODE_ROUTERID

  SR_CAPABILITIES

  SR_ALGORITHM

  SR_LOCAL_BLOCK

  SR_SRMA_PREF

  SRV6_CAPABILITIES

  SRV6_NODE_MSD

- **MultiTopoId** (*str*) -- 多拓扑 ID, 类型为：string, 取值范围：1-4095, 默认值：””
- **NodeFlagBitIsis** (*str*) -- 节点标志（ISIS）, 类型为：string, 默认值：ATTACHED, 取值范围：

  OVERLOAD

  ATTACHED

- **NodeFlagBitOspfv2** (*list*) -- 节点标志（OSPFv2）, 类型为：list, 默认值：ABR, 取值范围：

  EXTERNAL

  ABR

- **NodeFlagBitOspfv3** (*list*) -- 节点标志（OSPFv3）, 类型为：list, 默认值：ABR, 取值范围：

  EXTERNAL ABR ROUTER V6

- **NodeName** (*str*) -- 节点名称, 类型为：string, 默认值：”“
- **IsisAreaId** (*int*) -- IS-IS 区域 ID, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LocalIpv4RouterIds** (*str*) -- 本端 IPv4 路由 IDs, 类型为：有效的 ipv4 地址, 默认值：”“
- **LocalIpv6RouterIds** (*str*) -- 本端 IPv6 路由 IDs, 类型为：有效的 ipv6 地址, 默认值：”“
- **SidLabelType** (*str*) -- SID/Label 类型, 类型为：string, 默认值：BIT20_LABEL, 取值范围：

  BIT20_LABEL

  BIT32_SID

- **SrCapabilitiesFlags** (*list*) -- SR 能力标志, 类型为：list, 默认值：MPLS_IPv4, 取值范围：

  MPLS_IPv4

  MPLS_IPv6

- **SrCapabilities** (*str*) -- SR 能力, 类型为：string, 默认值：”16:100”
- **SrAlgorithm** (*int*) -- SR 算法, 类型为：number, 取值范围：1-255, 默认值：0
- **SrLocalBlock** (*str*) -- SRLB 范围, 类型为：string, 默认值：”16:100”
- **SrmsPref** (*int*) -- SRMS 优先级, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6CapabilitiesFlags** (*list*) -- SRv6 能力标志, 类型为：list, 默认值：O_FLAG, 取值范围：

  UNUSED0

  O_FLAG

  UNUSED2

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

  UNUSED7

  UNUSED8

  UNUSED9

  UNUSED10

  UNUSED11

  UNUSED12

  UNUSED13

  UNUSED14

  UNUSED15

- **Srv6MsdFlags** (*list*) -- SRv6 节点 MSD 类型, 类型为:list, 默认值:NONE, 取值范围:

  NONE

  MAX_SEGMENTS_LELT

  MAX_END_POP

  MAX_T_INSERT_SRH

  MAX_T_ENCAPS_SRH

  MAX_END_D_SRH

- **Srv6MsdMaxSegmentLeft** (*int*) -- 最大 Segments Left, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEndPop** (*int*) -- 最大 End Pop, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxInsert** (*int*) -- 最大 T.Insert SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEncap** (*int*) -- 最大 T.Encaps SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEndD** (*int*) -- 最大 End D SRH, 类型为：number, 取值范围：1-255, 默认值：8

返回 Bgp Link States 对象, 类型：object / list

返回类型 (BgpLsNodeConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Link States | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_link_states_link**(*Session*, *LinkState*, ***kwargs*)

创建 Bgp Link States Link 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **LinkState** (BgpLsNodeConfig) -- Bgp Link States 对象列表, 类型为:object / list

关键字参数

- **LinkLocalRemoteIdFlag** (*str*) -- 链路本端/远端 ID 位置, 类型为：string, 默认值：NONE, 取值范围:

  NONE

  LINK_DESCRIPTOR

  LINK_ATTRIBUTE

- **LinkLocalId** (*int*) -- 链路本端 ID, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LinkRemoteId** (*int*) -- 链路远端 ID, 类型为：number, 取值范围：1-4294967295, 默认值：0

- **RemoteNodeDescriptorFlag** (*str*) -- 远端节点描述符标志, 类型为：string, 默认值：IGP_ROUTER_ID, 取值范围：

  AS_NUMBER

  BGPLS_IDENTIFIER

  OSPF_AREA_ID

  IGP_ROUTER_ID

  BGP_ROUTER_ID

  MEMBER_ASN

- **AsNumber** (*int*) -- 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **BgpLsIdentifier** (*str*) -- BGP-LS 标识符, 类型为：有效的 ipv4 地址, 默认值：1.0.0.1
- **OspfAreaId** (*str*) -- OSPF 区域 ID, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **IGPRouterIdType** (*str*) -- IGP 路由 ID 类型, 类型为：string, 默认值：OSPF_NONPSEUDONODE, 取值范围：

  ISIS_NONPSEUDONODE

  ISIS_PSEUDONODE

  OSPF_NONPSEUDONODE

  OSPF_PSEUDONODE

- **IsisNonPseud** (*str*) -- ISIS 非伪节点路由 ID, 类型为：string, 默认值：”00:10:12:00:00:01”
- **IsisPseud** (*str*) -- OSPF 伪节点路由 ID, 类型为：string, 默认值：”00:10:12:00:00:01.02”
- **OspfNonPseud** (*str*) -- OSPF 非伪节点路由 ID, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **OspfPseud** (*str*) -- OSPF 伪节点路由 ID, 类型为：string, 默认值：”192.0.0.1:192.0.0.1”
- **BgpRouterId** (*str*) -- BGP 路由 ID, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **MemberAsn** (*int*) -- 自治域成员, 类型为：number, 取值范围：1-1048575, 默认值：0
- **LinkDescriptorFlag** (*list*) -- 链路描述符标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  LINK_IDENTIFIES

  IPV4_INTERFACE

  IPV4_NEIGHBOR

  IPV6_INTERFACE

  IPV6_NEIGHBOR

  MULTI_TOPOLOGY

- **Ipv4InterfaceAddr** (*str*) -- IPv4 接口地址, 类型为：有效的 ipv4 地址, 默认值：192.168.0.1

- **Ipv4NeighborAddr** (*str*) -- IPv4 邻接地址, 类型为：有效的 ipv4 地址, 默认值：192.168.0.2
- **Ipv6InterfaceAddr** (*str*) -- IPv6 接口地址, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **Ipv6NeighborAddr** (*str*) -- IPv6 邻接地址, 类型为：有效的 ipv6 地址, 默认值：2001::2
- **MultiTopologyId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-4095, 默认值：“”
- **LinkAttributeFlag** (*list*) -- 链路属性标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  IPV4_LOCAL_NODE

  IPV4_REMOTE_NODE

  IPV6_LOCAL_NODE

  IPV6_REMOTE_NODE

  LINK_IDENTIFIES

  LINK_PROTECTION_TYPE

  IGP_METRIC

  SHARED_RISE

  ADI_SID

  LAN_ADJ_SID

  PEERNODE_SID

  PEERADJ_SID

  PEERSET_SID

  SRV6_LINK_MSD

  SRV6_END_X_SID

  SRV6_LAN_END_X_SID
- **LocalIpv4RouterIds** (*str*) -- 本端 IPv4 路由 IDs, 类型为：有效的 ipv4 地址, 默认值：””
- **RemoteIpv4RouterIds** (*str*) -- 远端 IPv4 路由 IDs, 类型为：有效的 ipv4 地址, 默认值：””
- **LocalIpv6RouterIds** (*str*) -- 本端 IPv6 路由 IDs, 类型为：有效的 ipv6 地址, 默认值：””
- **RemoteIpv6RouterIds** (*str*) -- 远端 IPv6 路由 IDs, 类型为：有效的 ipv6 地址, 默认值：””
- **IgpMetricType** (*str*) -- IGP 度量类型, 类型为：string, 默认值：UNKNOWN, 取值范围：

  ISIS_NARROW

  OSPFv2_LINK

  ISIS_WIDE
- **IgpMetricValue** (*int*) -- IGP 度量值, 类型为：number, 取值范围：IS-IS Narrow Metrics：0-255, OSPFv2 Link Metrics：0-65535, IS-IS Wide Metrics：0-16777215, 默认值：0

- **LinkProteType** (*list*) -- 链路保护类型, 类型为: list, 默认值: UNKNOWN, 取值范围:

  EXTRA_TRAFFIC

  UNPROTECTED

  SHARED

  DEDICATED1

  DEDICATED2

  ENHANCED

  RESERVED4

  RESERVED8

- **ShareRiskGroup** (*int*) -- SRLG 信息, 类型为: number, 取值范围: 1-4294967295, 默认值: ""
- **OspfAdjSidFlag** (*list*) -- OSPF SR 邻接 SID 标志, 类型为: list, 默认值: UNKNOWN, 取值范围:

  UNKNOWN

  BACKUP

  VALUE

  LOCAL

  GROUP

  PERSISTENT

- **IsisAdjSidFlag** (*list*) -- ISIS SR 邻接 SID 标志, 类型为: list, 默认值: NOSHOW, 取值范围:

  NOSHOW

  ADDRESS

  BACKUP

  VALUE

  LOCAL

  SET

  PERSISTENT

- **AdjSidWeight** (*int*) -- 权重, 类型为: number, 取值范围: 1-255, 默认值: 1
- **AdjSidLabel** (*int*) -- SID/Label, 类型为: number, 取值范围: 1-255, 默认值: 16
- **OspfNeighborId** (*str*) -- OSPF 邻居 ID, 类型为: 有效的 ipv4 地址, 默认值: 192.0.1.0
- **IsisSystemId** (*str*) -- ISIS 系统 ID, 类型为: 有效的 mac 地址, 默认值: 00:00:00:00:00:00
- **PeerNodeFlag** (*list*) -- Peer Node 标志, 类型为: list, 默认值: UNKNOWN, 取值范围:

  UNKNOWN

  VALUE

  LOCAL

BACKUP

PERSISTENT

- **PeerNodeWeight** (*int*) -- Peer Node 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **PeerNodeSidLabel** (*int*) -- Peer Node SID/Index/Label, 类型为：number, 取值范围：1-255, 默认值：16
- **PeerAdjFlag** (*list*) -- Peer Adj 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  VALUE

  LOCAL

  BACKUP

  PERSISTENT
- **PeerAdjWeight** (*int*) -- Peer Adj 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **PeerAdjSidLabel** (*int*) -- Peer Adj 权重, 类型为：number, 取值范围：1-255, 默认值：16
- **PeerSetFlag** (*list*) -- Peer Set 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  VALUE

  LOCAL

  BACKUP

  PERSISTENT
- **PeerSetWeight** (*int*) -- Peer Set 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **PeerSetSidLabel** (*int*) -- Peer Set SID/Index/Label, 类型为：number, 取值范围：1-255, 默认值：16
- **Srv6MsdFlags** (*list*) -- SRv6 链路 MSD 类型, 类型为：list, 默认值：NONE, 取值范围：

  NONE

  MAX_SEGMENTS_LELT

  MAX_END_POP

  MAX_T_INSERT_SRH

  MAX_T_ENCAPS_SRH

  MAX_END_D_SRH
- **Srv6MsdMaxSegmentLeft** (*int*) -- 最大 Segments Left, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEndPop** (*int*) -- 最大 End Pop, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxInsert** (*int*) -- 最大 T.Insert SRH, 类型为：number, 取值范围：1-255, 默认值：8

- **Srv6MsdMaxEncap** (*int*) -- 最大 T.Encaps SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEndD** (*int*) -- 最大 End D SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6EndXSidEndpointBehavior** (*str*) -- SRv6 Endpoint 功能指令类型, 类型为：string, 默认值：CUSTOM, 取值范围：

  END

  END_WITH_PSP

  END_WITH_USP

  END_WITH_PSP_USP

  END_X

  END_X_WITH_PSP

  END_X_WITH_USP

  END_X_WITH_PSP_USP

  END_T

  END_T_WITH_PSP

  END_T_WITH_USP

  END_T_WITH_PSP_USP

  END_B6_ENCAPS

  END_BM

  END_DX6

  END_DX4

  END_DT6

  END_DT4

  END_DT46

  END_DX2

  END_DX2V

  END_DT2U

  END_DT2M

  END_B6_ENCAPS_RED

  END_WITH_USD

  END_WITH_PSP_USD

  END_WITH_USP_USD

  END_WITH_PSP_USP_USD

  END_X_WITH_USD

  END_X_WITH_PSP_USD

  END_X_WITH_USP_USD

  END_X_WITH_PSP_USP_USD

  END_T_WITH_USD

  END_T_WITH_PSP_USD

END_T_WITH_USP_USD

END_T_WITH_PSP_USP_USD

CUSTOM

- **Srv6EndXSidFlag** (*list*) -- SRv6 End.X SID 标志, 类型为：list, 默认值：NONE, 取值范围：

  NONE

  BACKUP_FLAG

  SET_FLAG

  PERSISTENT_FLAG

- **Srv6EndXSidAlgorithm** (*int*) -- SRv6 End.X SID 算法, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6EndXSidWeight** (*int*) -- SRv6 End.X SID 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **Srv6EndXSidSid** (*str*) -- SRv6 End.X SID, 类型为：有效的 ipv6 地址, 默认值：::1
- **Srv6LanEndXSidSystemId** (*str*) -- ISIS 邻居 ID, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **Srv6LanEndXSidRouterId** (*str*) -- OSPFv3 邻居 ID, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **EnableInterfaceIp** (*bool*) -- 启用本端 IPv4 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InterfaceIp** (*str*) -- 本端 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **EnableNeighborIp** (*bool*) -- 启用远端 IPv4 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NeighborIp** (*str*) -- 远端 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **EnableInterfaceIpv6** (*bool*) -- 启用本端 IPv6 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InterfaceIpv6** (*str*) -- 本端 IPv6 地址, 类型为：有效的 ipv4 地址, 默认值：2000::1
- **EnableNeighborIpv6** (*bool*) -- 启用远端 IPv6 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NeighborIpv6** (*str*) -- 启用远端 IPv6 地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **EnableGroup** (*bool*) -- 启用组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Group** (*int*) -- 组, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **EnableUniLinkLoss** (*bool*) -- 启用单向链路损耗, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LinkLoss** (*int*) -- 链路损耗, 类型为：number, 取值范围：1-100, 默认值：3
- **LinkLossAflag** (*bool*) -- 链路损耗的 A-flag, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableUniDelay** (*bool*) -- 启用单向延迟, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **UniDelay** (*int*) -- 单向延迟, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UniAflag** (*bool*) -- 单向延迟的 A-flag, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableUniMinMaxDelay** (*bool*) -- 启用单向延迟最小/最大值, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniMinMaxAflag** (*bool*) -- 启用单向延迟最小/最大值的 A-flag 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniMinDelay** (*int*) -- 单向最小延迟, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UniMaxDelay** (*int*) -- 单向最大延迟, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUniDelayVariation** (*bool*) -- 启用单向延迟变化, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniVarDelay** (*int*) -- 单向延迟变化, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUniResidual** (*bool*) -- 启用单向剩余带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniResBandwidth** (*int*) -- 单向剩余带宽, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUniAva** (*bool*) -- 启用单向可用带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniAvaBandwidth** (*int*) -- 单向可用带宽, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUniUtilized** (*bool*) -- 启用单向已用带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniUtilized** (*int*) -- 单向已用带宽, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableMaximum** (*bool*) -- 启用最大带宽 (字节/秒), 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Maximum** (*int*) -- 最大带宽 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableReservable** (*bool*) -- 启用预留带宽 (字节/秒), 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Reservable** (*int*) -- 预留带宽 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUnreserved** (*bool*) -- 启用未预留带宽优先级 (字节/秒), 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UnreservedBandwidth0** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth1** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth2** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth3** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000

- **UnreservedBandwidth4** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为: number, 取值范围: 1-4294967295, 默认值: 100000
- **UnreservedBandwidth5** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为: number, 取值范围: 1-4294967295, 默认值: 100000
- **UnreservedBandwidth6** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为: number, 取值范围: 1-4294967295, 默认值: 100000
- **UnreservedBandwidth7** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为: number, 取值范围: 1-4294967295, 默认值: 100000
- **EnableTeDefaultMetric** (*bool*) -- 启用 TE 默认度量, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **TeDefaultValue** (*int*) -- TE 默认度量, 类型为: number, 取值范围: 1-4294967295, 默认值: 0

返回 Bgp Link States Link 对象, 类型: object / list

返回类型 (BgpLsLinkConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${LinkState} | Create Bgp Link States | Sessions=${Session} |
| Create Bgp Link States Link | Sessions=${Session} | LinkState=${LinkState} |
```

TesterLibrary.Protocol.bgp.**create_bgp_link_states_prefix**(*Session*, *LinkState*, ***kwargs*)

创建 Bgp Link States Prefix 对象, 类型为: object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为: object / list
- **LinkState** (BgpLsNodeConfig) -- Bgp Link States 对象列表, 类型为: object / list

关键字参数

- **PrefixDescriptorFlag** (*list*) -- LS 前缀描述符标志, 类型为: list, 默认值: IP_REACH_INFO, 取值范围:

  MULTI_TOPOLOGY

  OSPF_ROUTE_TYPE

  IP_REACH_INFO

- **OspfRouteType** (*str*) -- OSPF 路由类型, 类型为: string, 默认值: INTRA_AREA, 取值范围:

  INTRA_AREA

  INTER_AREA

  EXTERNAL1

  EXTERNAL2

  NSSA1

  NSSA2

- **PrefixCount** (*int*) -- 地址前缀个数, 类型为: number, 取值范围: 1-65535, 默认值: 1

- **PrefixType** (*str*) -- 地址前缀类型, 类型为：string, 默认值：IPV4, 取值范围：

  IPV4

  IPV6

- **Ipv4Prefix** (*str*) -- IPv4 地址前缀, 类型为：有效的 ipv4 地址, 默认值：1.0.0.0
- **Ipv4PrefixLength** (*int*) -- IPv4 地址前缀步长, 类型为：number, 取值范围：1-32, 默认值：24
- **Ipv4PrefixStep** (*int*) -- IPv4 地址前缀步长, 类型为：number, 取值范围：1-65535, 默认值：1
- **Ipv6Prefix** (*str*) -- IPv6 地址前缀, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Ipv6PrefixLength** (*int*) -- IPv6 地址前缀步长, 类型为：number, 取值范围：1-128, 默认值：64
- **Ipv6PrefixStep** (*int*) -- IPv6 地址前缀长度, 类型为：number, 取值范围：1-65535, 默认值：1
- **MultiTopologyId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-4095, 默认值：“”
- **OspfSrPrefixSidFlag** (*list*) -- OSPF SR 前缀 SID 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  NO_PHP

  MAPPING_SERVER

  EXPLICIT_NULL

  VALUE

  LOCAL

- **IsisSrPrefixSidFlag** (*list*) -- ISIS SR 前缀 SID 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  RE_ADVER

  NODESID

  NO_PHP

  EXPLICIT_NULL

  VALUE

  LOCAL

- **PrefixAttributeFlag** (*str*) -- LS 前缀描述符标志, 类型为：string, 默认值：UNKNOWN|IGP_FLAGS, 取值范围：

  UNKNOWN

  IGP_FLAGS

  PREFIX_METRIC

  OSPF_FORWARDING

  SR_PREFIX_SID

  SR_RANGE

SR_ATTRIBUTE_FLAG

SR_SOURCE

SRV6_LOCATOR_TLV

- **IgpFlag** (*str*) -- IGP 标志, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  ISIS_UP_DOWN

  OSPF_NO_UNICAST

  OSPF_LOCAL_ADDRESS

  OSPF_PROPAGATE_NSSA

- **PrefixMetric** (*int*) -- 前缀度量, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **OspfForwardtype** (*str*) -- OSPF 转发地址类型, 类型为：string, 默认值：OSPFV2, 取值范围：

  OSPFV2

  OSPFV3

- **Ospfv2ForwardAddr** (*str*) -- OSPFv2 转发地址, 类型为：有效的 ipv4 地址, 默认值：192.168.1.1
- **Ospfv3ForwardAddr** (*str*) -- OSPFv3 转发地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **OspfSrPrefixFlag** (*str*) -- OSPF SR 前缀 SID 标志, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  NO_PHP

  MAPPING_SERVER

  EXPLICIT_NULL

  VALUE

  LOCAL

- **IsisSrPrefixFlag** (*str*) -- ISIS SR 前缀 SID 标志, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  RE_ADVER

  NODESID

  NO_PHP

  EXPLICIT_NULL

  VALUE

  LOCAL

- **Algorithm** (*int*) -- 算法, 类型为：number, 取值范围：1-255, 默认值：0
- **SidLabelIndex** (*int*) -- SID/Label/Index, 类型为：number, 取值范围：0-1048575 (20 比特值), 0-4294967295 (32 比特值), 默认值：0

- **Ospfv2SrPrefixAttributeFlag** (*list*) -- OSPFv2 SR 前缀属性标志, 类型为：list, 默认值：NODE, 取值范围：

  ATTACH

  NODE

- **Ospfv3SrPrefixAttributeFlag** (*list*) -- OSPFv3 SR 前缀属性标志, 类型为：list, 默认值：NO_UNICAST, 取值范围：

  NO_UNICAST

  LOCAL_ADDRESS

  MULTICAST

  PROPAGATE

  RE_ADVER

  HOST

- **IsisSrPrefixAttributeFlag** (*list*) -- ISIS SR 前缀属性标志, 类型为：list, 默认值：NODE, 取值范围：

  EXTERNAL_PREFIX

  RE_ADVER

  NODE

- **SrSourceIpv4Id** (*str*) -- SR IPv4 源路由 ID, 类型为：有效的 ipv4 地址, 默认值：192.168.1.0

- **SrSourceIpv6Id** (*str*) -- SR IPv6 源路由 ID, 类型为：有效的 ipv6 地址, 默认值：2000::1

- **Ospfv2SrRangeFlag** (*list*) -- OSPFv2 SR 范围标志, 类型为：list, 默认值：INTER_AREA, 取值范围：

  INTER_AREA

- **IsisSrRangeFlag** (*list*) -- ISIS SR 范围标志, ISIS SR 前缀 SID 标志, 类型为：list, 默认值：ATTACHED, 取值范围：

  ADDRESS_FAMILY

  MIRROR_CONTEXT

  S_FLAG

  D_FLAG

  ATTACHED

- **SrRangeSubTlv** (*str*) -- SR 范围 Sub-TLVs, 类型为：string, 默认值：”0 Range Sub-TLV”

- **Srv6LocatorFlag** (*str*) -- SRv6 Locator 标志, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  D_FLAG

- **Srv6LocatorAlgorithm** (*int*) -- SRv6 Locator 算法, 类型为：number, 取值范围：1-255, 默认值：0

- **Srv6LocatorMetric** (*int*) -- SRv6 Locator 度量值, 类型为：number, 取值范围：1-4294967295, 默认值：0

返回 Bgp Link States Prefix 对象, 类型：object / list

返回类型 (BgpLsPrefixConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${LinkState} | Create Bgp Link States | Sessions=${Session} |
| Create Bgp Link States Prefix | Sessions=${Session} | LinkState=${LinkState}
↪ |
```

TesterLibrary.Protocol.bgp.**create_bgp_link_states_prefix_sr_range_sub_tlv**(*Session*, *LinkStatePrefix*, ***kwargs*)

创建 Bgp Link States Prefix Sr Range Sub Tlv 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **LinkStatePrefix** (BgpLsPrefixConfig) -- Bgp Link States prefix 对象列表, 类型为：object / list

关键字参数

- **Algorithm** (*int*) -- 算法, 类型为：number, 取值范围：1-255, 默认值：0
- **OspfSrPrefixSidFlag** (*list*) -- OSPF SR Prefix SID Flags, 类型为：list, 默认值：UNKNOWN, 取值范围:

  UNKNOWN

  NO_PHP

  MAPPING_SERVER

  EXPLICIT_NULL

  VALUE

  LOCAL

- **IsisSrPrefixSidFlag** (*list*) -- ISIS SR Prefix SID Flags, 类型为：list, 默认值：UNKNOWN, 取值范围:

  UNKNOWN

  RE_ADVER

  NODESID

  NO_PHP

  EXPLICIT_NULL

  VALUE

  LOCAL

- **SidLabelIndex** (*int*) -- SID/Label/Index, 类型为：number, 取值范围：1-255, 默认值：0

返回 Bgp Link States Prefix Sr Range Sub Tlv 对象, 类型：object / list

返回类型 (BgpLsSrRangeSubTlvConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${LinkState} | Create Bgp Link States | Sessions=${Session} |
| ${prefix} | Create Bgp Link States Prefix | LinkState=${LinkState} |
| Create Bgp Link States Prefix Sr Range Sub Tlv | Sessions=${Session} |␣
↪LinkStatePrefix=${prefix} |
```

TesterLibrary.Protocol.bgp.**create_bgp_link_states_srv6_sid**(*Session*, *LinkState*, ***kwargs*)

创建 Bgp Link States Srv6 Sid 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **LinkState** (BgpLsNodeConfig) -- Bgp Link States 对象列表, 类型为：object / list

关键字参数

- **Srv6AttributeFlag** (*list*) -- SRv6 SID 属性标志, 类型为：list, 默认值：SRV6_ENDPOINT_BEHAVIOR, 取值范围：

  SRV6_ENDPOINT_BEHAVIOR

  SRV6_BGP_PEER_NODE_SID

  SRV6_SID_STRUCTURE

- **Srv6EndpointBehavior** (*str*) -- SRv6 Endpoint 功能指令类型, 类型为：string, 默认值：CUSTOM, 取值范围：

  END

  END_WITH_PSP

  END_WITH_USP

  END_WITH_PSP_USP

  END_X

  END_X_WITH_PSP

  END_X_WITH_USP

  END_X_WITH_PSP_USP

  END_T

  END_T_WITH_PSP

  END_T_WITH_USP

  END_T_WITH_PSP_USP

  END_B6_ENCAPS

  END_BM

  END_DX6

  END_DX4

  END_DT6

  END_DT4

  END_DT46

  END_DX2

END_DX2V

END_DT2U

END_DT2M

END_B6_ENCAPS_RED

END_WITH_USD

END_WITH_PSP_USD

END_WITH_USP_USD

END_WITH_PSP_USP_USD

END_X_WITH_USD

END_X_WITH_PSP_USD

END_X_WITH_USP_USD

END_X_WITH_PSP_USP_USD

END_T_WITH_USD

END_T_WITH_PSP_USD

END_T_WITH_USP_USD

END_T_WITH_PSP_USP_USD

CUSTOM

- **Srv6EndpointBehaviorFlag** (*list*) -- SRv6 Endpoint 功能指令标志, 类型为：list, 默认值：NONE, 取值范围：

  NONE

  UNUSED0

  UNUSED1

  UNUSED2

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

  UNUSED7

- **Srv6EndpointBehaviorAlgorithm** (*int*) -- SRv6 Endpoint 功能指令算法, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BgpPeerNodeSidFlag** (*list*) -- EPE Peer Node SID 标志, 类型为：list, 默认值：0, 取值范围：

  NONE

  BACKUP_FLAG

  SET_FLAG

  PERSISTENT_FLAG

- **Srv6BgpPeerNodeSidWeight** (*int*) -- EPE Peer Node SID 权重, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BgpPeerNodeSidPeerAsNumber** (*int*) -- Peer 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1001

- **Srv6BgpPeerNodeSidPeerBgpId** (*int*) -- Peer BGP 标识符, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **Srv6SidStructureLbLength** (*int*) -- SRv6 Locator Block 长度, 类型为：number, 取值范围：1-255, 默认值：32
- **Srv6SidStructureLnLength** (*int*) -- SRv6 Locator Node 长度, 类型为：number, 取值范围：1-255, 默认值：32
- **Srv6SidStructureFunLength** (*int*) -- SRv6 Function 长度, 类型为：number, 取值范围：1-255, 默认值：32
- **Srv6SidStructureArgLength** (*int*) -- SRv6 Argument 长度, 类型为：number, 取值范围：1-255, 默认值：32
- **Srv6Sid** (*str*) -- 类型为：有效的 ipv6 地址, 默认值：::1
- **MultiTopologyId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-4095, 默认值：“”

返回 Bgp Link States Srv6 Sid 对象, 类型：object / list

返回类型 (BgpLsSrv6SidConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${LinkState} | Create Bgp Link States | Sessions=${Session} |
| Create Bgp Link States Srv6 Sid | Sessions=${Session} | LinkState=$
↪{LinkState} |
```

TesterLibrary.Protocol.bgp.**create_bgp_random_route**(*PortNumber*, *Type=None*, *Ipv4RouteNumber=300000*, *Ipv6RouteNumber=100000*, *Ipv4StartMask=23*, *Ipv6StartMask=32*, *Ipv4StartRoute='60.0.0.0'*, *Ipv6StartRoute='60::'*, ***kwargs*)

TesterLibrary.Protocol.bgp.**create_bgp_route_pool_custom_path_attribute**(*RoutePool*, ***kwargs*)

创建 BGP Route Pool Custom Path Attribute 对象, 类型为：object / list

参数 **RoutePool** (BgpIpv4RoutepoolConfig) -- 所属的 Bgp Route Pool 对象, 类型为：object / list

关键字参数

- **PathAttributeType** (*int*) -- 路径属性的类型, 类型为：number, 取值范围：1-255, 默认值：1
- **OptionalFlag** (*str*) -- 指定 Optional Flag 的值, 类型为：string, 默认值：OPTION, 取值范围：

  WELL_KNOWN

  OPTION
- **TransitiveFlag** (*str*) -- 指定 Transitive Flag 的值, 类型为：string, 默认值：NONTRANSITIVE, 取值范围：

  NONTRANSITIVE

  TRANSITIVE

- **PartialFlag** (*str*) -- 指定 Partial Flag 的值, 类型为：string, 默认值：PARTIAL, 取值范围：

  COMPLETE

  PARTIAL

- **ExtendedLengthFlag** (*bool*) -- 是否启用 Extended Length Flag, 类型为：bool, 默认值：False
- **AttributeExtendedLength** (*int*) -- 指定路径属性的长度, 类型为：number, 默认值：0
- **AttributeValue** (*str*) -- 指定路径属性的值, 类型为：string, 默认值：""

返回 Bgp Route Pool Custom Path Attribute 对象, 类型：object / list

返回类型 (BgpPathAttributeConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${RoutePool} | Create Bgp Ipv4 Route Pool | Session=${Session} |
| ${AttributeValue} | Set Variable |␣
↪0500220001001E00AAAA0001000100010000000000000001000013000100063010140000000 |
| Create Bgp Route Pool Custom Path Attribute | RoutePool=${RoutePool} |␣
↪AttributeValue=${AttributeValue} |
```

TesterLibrary.Protocol.bgp.**create_bgp_segement_sub_tlv**(*Session*, *SegementList*, *Types*, ***kwargs*)

创建 Bgp Segement Sub Tlv 对象, 类型为：object

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **SegementList** (BgpSrTePolicySegmentList) -- BGP Sr Te Policy Segement List 对象, 类型为：object
- **Types** (*str*) -- BGP Sr Te Policy Segement Sub Tlv 类型, 类型为 string, 默认值: A, 取值范围：A-K

关键字参数

- **Flags** (*list*) -- 选择一个或多个 Flag, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  V_FLAG

  A_FLAG

  S_FLAG

  B_FLAG

- **Label** (*int*) -- 指定标签, 类型为：number, 取值范围：1-1048575, 默认值：1600
- **LabelStep** (*int*) -- 指定标签跳变, 类型为：number, 取值范围：1-1048575, 默认值：1
- **Tc** (*int*) -- 指定 TC（Traffic Class，通信量类）值, 类型为：number, 取值范围：0-7, 默认值：0
- **Sbit** (*int*) -- 指定栈底标志（S）的值, 类型为：number, 取值范围：0-1, 默认值：1

- **Ttl** (*int*) -- 指定 TTL 值, 类型为：number, 取值范围：1-1048575, 默认值：255
- **Srv6Sid** (*str*) -- 指定 SRv6 SID, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Srv6SidStep** (*str*) -- 指定 SRv6 SID 的跳变步长, 类型为：有效的 ipv6 地址, 默认值：::1
- **EndpointBehavior** (*int*) -- 指定 SRv6 Endpoint Behavior, 类型为：number, 取值范围：1-1048575, 默认值：0
- **LbLength** (*int*) -- SRv6 SID Locator Block 长度, 类型为：number, 取值范围：1-1048575, 默认值：0
- **LnLength** (*int*) -- SRv6 SID Locator Node 长度, 类型为：number, 取值范围：1-1048575, 默认值：0
- **FunLength** (*int*) -- SRv6 SID Function 长度, 类型为：number, 取值范围：1-1048575, 默认值：0
- **ArgLength** (*int*) -- SRv6 SID 参数长度, 类型为：number, 取值范围：1-1048575, 默认值：0
- **SrAlgorithm** (*int*) -- SR 算法, 类型为：number, 取值范围：1-1048575, 默认值：0
- **Ipv4NodeAddress** (*str*) -- IPv4 节点地址, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv4NodeAddressStep** (*str*) -- IPv4 节点地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **Ipv6NodeAddress** (*str*) -- IPv6 节点地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Ipv6NodeAddressStep** (*str*) -- IPv6 节点地址跳变, 类型为：有效的 ipv6 地址, 默认值：::1
- **LocalInterfaceId** -- 本地 Interface ID, 类型为：number, 取值范围：1-1048575, 默认值：0
- **LocalIpv4Address** (*str*) -- IPv4 节点地址, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **LocalIpv4AddressStep** (*str*) -- IPv4 节点地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **RemoteIpv4Address** (*str*) -- 远端 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **RemoteIpv4AddressStep** (*str*) -- 远端 IPv4 地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **LocalIpv6NodeAddress** (*str*) -- IPv6 本地节点地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **LocalIpv6NodeAddressStep** (*str*) -- IPv6 本地节点地址跳变, 类型为：有效的 ipv6 地址, 默认值：::1
- **RemoteInterfaceId** (*int*) -- 远端 Interface ID, 类型为：number, 取值范围：1-1048575, 默认值：0
- **RemoteIpv6NodeAddress** (*str*) -- IPv6 远端节点地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **RemoteIpv6NodeAddressStep** (*str*) -- IPv6 远端节点地址跳变, 类型为：有效的 ipv6 地址, 默认值：::1

返回 Bgp Segement Sub Tlv 对象, 类型：object / list

返回类型 (BgpSegmentSubTlvTypeA)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${SrTe} | Create Bgp Sr Te Policy | Session=${Session} |
| ${SrTeSegmentList} | Create Bgp Sr Te Policy Segement List | Session=$
↪{Session} | SrTePolicys=${SrTe} |
| Create Bgp Segement Sub Tlv | Session=${Session} | SrTePolicys=${SrTe} |␣
↪Types=B |
```

TesterLibrary.Protocol.bgp.**create_bgp_sr_te_policy**(*Session*, ***kwargs*)

创建 Bgp Sr Te Policy 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

关键字参数

- **BindingSidCount** (*int*) -- Binding SID 数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **PolicyColor** (*int*) -- 指定 SR Policy 的起始 Color 值, 类型为：number, 默认值：0
- **PolicyColorStep** (*int*) -- 指定 Policy Color 的跳变步长, 类型为：number, 默认值：1
- **IpVersion** (*str*) -- 指定 IP 前缀类型, 类型为：string, 默认值：IPV4, 取值范围：

  IPV4

  IPV6
- **EndpointCount** (*int*) -- 指定目的节点的数量。该参数值不能大于 Binding SID 数量。类型为：number, 取值范围：1-4294967295, 默认值：1
- **Ipv4Endpoint** (*str*) -- 指定 Policy 块中目的节点的起始 IP 地址, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv4EndpointStep** (*str*) -- 指定 Policy 块中目的节点的 IP 地址的跳变步长, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **Ipv6Endpoint** (*str*) -- 指定 Policy 目的节点的 IP 地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Ipv6EndpointStep** (*str*) -- 指定 Policy 目的节点的 IP 地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **EndpointIncrementMode** (*str*) -- 指定 Policy 块中目的节点 IP 地址的生成方式, 类型为：string, 默认值：RoundRobin, 取值范围：

  RoundRobin

  Sequential
- **Origin** (*str*) -- 指定 ORIGIN 的值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：””

- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE

  CONFED_SEQUENCE

  CONFED_SET

- **LocalPref** (*int*) -- 指定 Local_PREF 的值, 类型为：number, 默认值：10
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **Ipv4NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv6NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **Distinguisher** (*int*) -- 输入 SR Policy 标识, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **RouteTarget** (*str*) -- 指定重定向的路由目标。类型为：string, 默认值：””
- **Community** (*str*) -- 当 Type 为 NO_EXPORT 时，团体值为 0xffffff01；当 Type 为 NO_ADVERTISE 时，团体值为 0xffffff02；当 Type 为 LOCAL_AS 时，团体值为 0xffffff03, 类型为：string, 默认值：””
- **ExtendedCommunity** (*str*) -- 扩展团体，类型为：string，默认值：”0x03:0x0b:0:0”
- **SrTePolicySubTlv** (*list*) -- 选择一个或多个 TLV, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  REMOTE_ENDPOINT

  COLOR

  PREFERENCE

  BINDING_SID

  ENLP

  PRIORITY

  EGRESS_ENDPOINT

  POLICY_CP_NAME

  SRV6_BSID

  POLICY_NAME

- **RemoteEndpointAsn** (*int*) -- 远端 Endpoint 自治域号, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **Ipv4RemoteEndpoint** (*str*) -- IPv6 远端 Endpoint, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv6RemoteEndpoint** (*str*) -- IPv6 远端 Endpoint, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Color** (*int*) -- 指定 Color 扩展团体中 Color Value 字段的值, 类型为：number, 取值范围：1-4294967295, 默认值：0

- **ColorFlags** (*list*) -- 指定 Color 扩展团体中 Flags 字段的值, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  C_FLAG

  O_FLAG

- **Preference** (*int*) -- 指定 SR Policy 中候选路径的优先级, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **BsidFlags** (*str*) -- Binding SID Flags, 类型为：string, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  S_FLAG

  I_FLAG

- **BsidLength** (*str*) -- Binding SID 长度, 类型为：string, 默认值：NO_SHOW, 取值范围：

  OCTET_0

  OCTET_4

  OCTET_16

- **BsidLabel** (*int*) -- Binding SID 标签, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **BsidLabelStep** (*int*) -- Binding SID 标签跳变, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **BsidTc** (*int*) -- 指定 BSID 中的 S 字段值, 类型为：number, 取值范围：0-7, 默认值：0
- **BsidS** (*int*) -- 指定 BSID 中的 S 字段值, 类型为：number, 取值范围：0-1, 默认值：0
- **BsidTtl** (*int*) -- 指定 BSID 中的 TTL 字段值, 类型为：number, 默认值：0
- **Ipv6Bsid** (*str*) -- 指定 IPv6 BSID, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Ipv6BsidStep** (*str*) -- 指定 IPv6 BSID 的跳变步长, 类型为：有效的 ipv6 地址, 默认值：::1
- **Enlp** (*str*) -- 指定显式空标签策略, 类型为：string, 默认值：IPV4, 取值范围：

  RESERVED0

  IPV4

  IPV6

- **PolicyPriority** (*int*) -- 指定拓扑改变后重新计算 SR Policy 时的 Policy 优先级, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BsidFlags** (*list*) -- 指定 SRv6 Binding SID sub-TLV 中包含的 Flag, 类型为：list, 默认值：NO_SHOW(0x0), 取值范围：

  NO_SHOW

  S_FLAG

  I_FLAG

  B_FLAG

- **Srv6Bsid** (*str*) -- 指定 SRv6 BSID, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Srv6BsidStep** (*str*) -- 指定 SRv6 BSID 的跳变步长, 类型为：有效的 ipv6 地址, 默认值：::1
- **Srv6BsidEndpointBehavior** (*int*) -- 指定 SRv6 SID Endpoint Behavior, 类型为：number, 取值范围：1-65535, 默认值：0
- **Srv6BsidLbLength** (*int*) -- 指定 SRv6 SID Function 长度。类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BsidLnLength** (*int*) -- 指定 SRv6 SID Locator Node 长度。单位：比特, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BsidFunLength** (*int*) -- 指定 SRv6 SID Function 长度。单位：比特, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BsidArgLength** (*int*) -- 指定 SRv6 SID 参数长度。单位：比特, 类型为：number, 取值范围：1-255, 默认值：0
- **CandidatePathName** (*str*) -- Policy 候选路径名称, 类型为：string, 默认值：""
- **PolicyName** (*str*) -- 指定 Policy 的名称, 类型为：string, 默认值：""
- **TunnelEgressEndpointAfi** (*str*) -- 隧道出口端点 AFI, 类型为：string, 默认值：IPV4, 取值范围：

  RESERVED0

  IPV4

  IPV6
- **Ipv4TunnelEgressEndpoint** (*str*) -- IPv4 远端 Endpoint, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv6TunnelEgressEndpoint** (*str*) -- IPv6 远端 Endpoint, 类型为：有效的 ipv6 地址, 默认值：2000::1

返回 Bgp Sr Te Policy 对象, 类型：object / list

返回类型 (BgpSrTePolicyConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Sr Te Policy | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**create_bgp_sr_te_policy_Segement_list**(*Session*, *SrTePolicy*, ***kwargs*)

创建 Bgp Sr Te Policy Segement List 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **SrTePolicy** (BgpSrTePolicyConfig) -- BGP Sr Te Policy 对象, 类型为：object / list

关键字参数

- **SubTlvs** (*list*) -- 选择一个或多个子 TLV, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

WEIGHT

- **Weight** (*int*) -- 指定 Segment List(SID 列表)对应的权重, 类型为:number, 取值范围: 1-4294967295, 默认值: 1

返回 Bgp Sr Te Policy Segement List 对象, 类型: object / list

返回类型 (BgpSrTePolicySegmentList)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${SrTe} | Create Bgp Sr Te Policy | Session=${Session} |
| Create Bgp Sr Te Policy Segement List | Session=${Session} | SrTePolicy=$
↪{SrTe} |
```

TesterLibrary.Protocol.bgp.**disconnect_bgp**(*Sessions*)

断开 BGP 协议会话连接

参数 **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Disconnect Bgp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bgp.**edit_bgp**(*Session*, ***kwargs*)

编辑 Bgp 协议会话对象参数

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为: object

关键字参数

- **Name** (*str*) -- BGP 协会话名称, 类型为: string
- **Enable** (*bool*) -- 使能 BGP 协议会话, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **BgpInitiator** (*bool*) -- BGP 会话发起者, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **AsNumber** (*int*) -- 自治域, 类型为: number, 取值范围: 1-65535, 默认值: 1
- **AsNumberStep** (*int*) -- 自治域跳变, 类型为: number, 取值范围: 0-65535, 默认值: 1
- **Enable4ByteAs** (*bool*) -- 使能 4 字节自治域, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **AsNumber4Byte** (*int*) -- 4 字节自治域, 类型为: number, 取值范围: 0.1-65535.65535, 默认值: 1.1
- **AsNumber4ByteStep** (*int*) -- 4 字节自治域跳变, 类型为: number, 取值范围: 0.1-65535.65535, 默认值: 0.1
- **DutAsNumber** (*int*) -- DUT 自治域, 类型为: number, 取值范围: 1-65535, 默认值: 1
- **DutAsNumberStep** (*int*) -- DUT 自治域跳变, 类型为: number, 取值范围: 1-65535, 默认值: 1

- **Enable4ByteDutAs** (*bool*) -- 使能 DUT4 字节自治域, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Dut4ByteAsNumber** (*int*) -- DUT4 字节自治域, 类型为：number, 取值范围：0.1-65535.65535, 默认值：1.1
- **Dut4ByteAsNumberStep** (*int*) -- DUT4 字节自治域跳变, 类型为：number, 取值范围：0.1-65535.65535, 默认值：0.1
- **BgpType** (*str*) -- BGP 类型, 类型为：string, 取值范围：EBGP, IBGP, 默认值：IBGP
- **UseGatewayAsDutIp** (*bool*) -- 使用网关地址作为 DUT 地址, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **BgpSessionIpAddressType** (*str*) -- 会话 IP 类型, 类型为：string, 取值范围：INTERFACE_IP, ROUTE_ID, 默认值：INTERFACE_IP
- **DutIpv4Address** (*str*) -- DUT IPv4 地址, 当 IP 版本为 IPv4，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID, 类型为：string, 取值范围：IPv4 地址, 默认值：2.1.1.1
- **DutIpv4AddressStep** (*str*) -- DUT IPv4 地址跳变, 当 IP 版本为 IPv4，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID 增量步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **DutIpv6Address** (*str*) -- DUT IPv4 地址, 当 IP 版本为 IPv6，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **DutIpv6AddressStep** (*str*) -- DUT IPv4 地址跳变, 当 IP 版本为 IPv6，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID 增量步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **HoldTime** (*int*) -- Hold Time 间隔 (sec), 类型为：number, 取值范围：3-65535, 默认值：90
- **KeepaliveTime** (*int*) -- Keep Alive 间隔 (sec), 类型为：number, 取值范围：1-65535, 默认值：30
- **ConnectRetryCount** (*int*) -- 重连次数, 取值范围：0-65535, 默认值：0
- **ConnectRetryInterval** (*int*) -- 重连间隔 (sec), 取值范围：10-300, 默认值：30
- **MaxRoutesPerUpdateMessage** (*int*) -- Update 报文中最大路由数量, 取值范围：10-300, 默认值：2000
- **RouteRefreshMode** (*str*) -- Route Refresh 模式, 类型为：string, 取值范围：None; Route Refresh, 默认值：None
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RestartTime** (*int*) -- 平滑重启时间（秒）, 取值范围：3-4095, 默认值：90
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Authentication** (*str*) -- 使用的认证类型, 类型为：string, 取值范围：None 或 MD5, 默认值：None
- **Password** (*str*) -- 认证密码, 类型为：类型为：string, 取值范围：字符串，由 1-255 个数字、字母或特殊字符组成, 默认值：xinertel
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **EnableSr** (*bool*) -- 使能 SR, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Bgp | Session=${Session} |
```

TesterLibrary.Protocol.bgp.**edit_bgp_port_config**(*Ports*, ***kwargs*)

修改 Bgp 端口统计对象

参数 **Ports** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **BgpUpdateMessageRate** (*int*) -- BGP Update Messages Tx Rate (messages/sec), 取值范围：1-1000, 默认值：10
- **SessionConnectDelay** (*int*) -- BGP Session Connect Delay (ms), 取值范围：1-10000, 默认值：100
- **SessionDisconnectDelay** (*int*) -- BGP Session Disconnect Delay (ms), 取值范围：1-10000, 默认值：100
- **BgpSrVersion** (*str*) -- BGP SR Version, 默认值：RFC8669, 取值范围：RFC8669
- **SrgbBase** (*int*) -- Global SRGB Base, 取值范围：0-16777215, 默认值：16000
- **SrgbRange** (*int*) -- Global SRGB Range, 取值范围：0-16777215, 默认值：1000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Bgp Port Config | Ports=${Ports} | BgpUpdateMessageRate=100 |
```

TesterLibrary.Protocol.bgp.**establish_bgp**(*Sessions*)

建立 BGP 协议会话

参数 **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Establish Bgp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bgp.**get_bgp_evpn_routes_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 Bgp Evpn Routes 统计结果

参数

- **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxAdRouteCount

  RxAdRouteCount

  TxMacIpRouteCount

  RxMacIpRouteCount

  TxInclusiveMcastRouteCount

  RxInclusiveMcastRouteCount

  TxEthernetSegmentRouteCount

  RxEthernetSegmentRouteCount

  TxIpPrefixRouteCount

  RxIpPrefixRouteCount

  TxWithdrawnAdRouteCount

  RxWithdrawnAdRouteCount

  TxWithdrawnMacIpRouteCount

  RxWithdrawnMacIpRouteCount

  TxWithdrawnInclusiveMcastRouteCount

  RxWithdrawnInclusiveMcastRouteCount

  TxWithdrawnEthernetSegmentRouteCount

  RxWithdrawnEthernetSegmentRouteCount

  TxWithdrawnIpPrefixRouteCount

  RxWithdrawnIpPrefixRouteCount

返回

eg:

```
{
    'TxAdRouteCount': 10,
    'RxAdRouteCount': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=EvpnRoutesStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bgp Evpn Routes Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bgp.**get_bgp_link_state_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 Bgp Evpn Routes 统计结果

参数

- **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxAdvertisedNodeCount

  RxAdvertisedNodeCount

  TxWithdrawnNodeCount

  RxWithdrawnNodeCount

  TxAdvertisedLinkCount

  RxAdvertisedLinkCount

  TxWithdrawnLinkCount

  RxWithdrawnLinkCount

  TxAdvertisedIpv4PrefixCount

  RxAdvertisedIpv4PrefixCount

  TxWithdrawnIpv4PrefixCount

  RxWithdrawnIpv4PrefixCount

  TxAdvertisedIpv6PrefixCount

  RxAdvertisedIpv6PrefixCount

  TxWithdrawnIpv6PrefixCount

  RxWithdrawnIpv6PrefixCount

  TxAdvertisedSrv6SidCount

  RxAdvertisedSrv6SidCount

  TxWithdrawnSrv6SidCount

  RxWithdrawnSrv6SidCount

返回

eg:

```
{
    'TxAdRouteCount': 10,
    'RxAdRouteCount': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BgpLinkStateStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bgp Link State Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bgp.**get_bgp_router_from_route_pool**(*Configs*, *Type='ipv4'*)

获取 BGP Route Pool 对应的绑定流源或目的端点对象

参数

- **Configs** (list(BgpIpv4RoutepoolConfig)) -- 测试仪表 BGP Route Pool 对象列表
- **Type** (*str*) -- Route Pool 类型支持 ipv4 和 ipv6

返回 BGP Route Pool 对应的绑定流源或目的端点对象列表

返回类型 (list(BgpIpv4RoutepoolConfig))

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${RouterPool} | Create Bgp Ipv4 Route Pool | Session=${Session} |
| ${Point} | Get Router From Route Pool | Configs=${RouterPool} |
```

TesterLibrary.Protocol.bgp.**get_bgp_session_block_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 Bgp Session Block 统计结果

参数

- **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  PeerState

  TxOpen

  RxOpen

  TxKeepalive

  RxKeepalive

  TxUpdate

  RxUpdate

  TxAdvertisedUpdate

  RxAdvertisedUpdate

  TxWithdrawnUpdate

  RxWithdrawnUpdate

  TxAdvertisedRoutes

RxAdvertisedRoutes

TxWithdrawnRoutes

RxWithdrawnRoutes

LastTxUpdateRoutes

LastRxUpdateRoutes

TxNotification

RxNotification

TxRefresh

RxRefresh

返回

eg:

```
{
    'TxAdRouteCount': 10,
    'RxAdRouteCount': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BgpSessionBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bgp Session Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bgp.**get_bgp_session_statistic**(*Session*, *Id*, *StaItems=None*)

获取 Bgp Session 统计结果

参数

- **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：Object
- **Id** (*int*) -- Bgp Peer Id, 类型为：number
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  PeerState

  TxOpen

  RxOpen

  TxKeepalive

  RxKeepalive

  TxUpdate

  RxUpdate

  TxAdvertisedUpdate

  RxAdvertisedUpdate

TxWithdrawnUpdate

RxWithdrawnUpdate

TxAdvertisedRoutes

RxAdvertisedRoutes

TxWithdrawnRoutes

RxWithdrawnRoutes

LastTxUpdateRoutes

LastRxUpdateRoutes

TxNotification

RxNotification

TxRefresh

RxRefresh

返回

eg:

```
{
    'TxAdRouteCount': 10,
    'RxAdRouteCount': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BgpSessionStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bgp Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.bgp.**wait_bgp_ipv4_router_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 BGP IPv4 会话组达到指定状态

参数

- **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BGP IPv4 会话组达到的状态, 类型为：string, 默认值：达到 ESTABLISHED, 支持下列状态：

  NOT_START

  IDLE

  CONNECT

  ACTIVE

  OPEN_SENT

  OPEN_CONFIRM

ESTABLISHED

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bgp IPv4 Router State | Sessions=${Sessions} | State=ESTABLISHED |␣
↪Interval=2 | TimeOut=120 |
```

TesterLibrary.Protocol.bgp.**wait_bgp_ipv6_router_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 BGP IPv6 会话组达到指定状态

参数

- **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BGP IPv6 会话组达到的状态, 类型为：string, 默认值：达到 ESTABLISHED, 支持下列状态：

  NOT_START

  IDLE

  CONNECT

  ACTIVE

  OPEN_SENT

  OPEN_CONFIRM

  ESTABLISHED
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bgp IPv6 Router State | Sessions=${Sessions} | State=ESTABLISHED |␣
↪Interval=2 | TimeOut=120 |
```

TesterLibrary.Protocol.bgp.**wait_bgp_router_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 BGP 会话组达到指定状态

参数

- **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

- **State** (*str*) -- 等待 BGP IPv4 会话组达到的状态, 类型为：string, 默认值：达到 ESTABLISHED, 支持下列状态：

  NOT_START

  IDLE

  CONNECT

  ACTIVE

  OPEN_SENT

  OPEN_CONFIRM

  ESTABLISHED

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bgp Router State | Sessions=${Sessions} | State=ESTABLISHED |␣
↪Interval=2 | TimeOut=120 |
```

TesterLibrary.Protocol.bgp.**wait_bgp_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 BGP 协议会话达到指定状态

参数

- **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BGP 协议会话达到的状态, 类型为：string, 默认值：达到 RUNNING, 支持下列状态：

  DISABLED

  NOT_START

  RUNNING

  STARTING

  STOPPING

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bgp State | Sessions=${Sessions} | State=DR | Interval=2 | TimeOut=120␣
↪|
```

TesterLibrary.Protocol.bgp.**withdraw_bgp**(*Sessions*)

撤销 BGP 协议会话 lsa

参数 **Sessions** (BgpRouter) -- OSPFv3 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Bgp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.bgp.**withdraw_bgp_route**(*Routes*)

撤销 BGP 协议指定 lsa

参数 **Routes** (*list*) -- BGP 协议路由对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Bgp Route | Routes=${Routes} |
```

TesterLibrary.Protocol.common module

TesterLibrary.Protocol.common.**get_sessions**(*Ports=None*, *Protocols=None*)

获取当前测试仪表配置中所有的协议对象

参数

- **Ports** (*list*) -- 端口对象的列表
- **Protocols** (*list*) -- 指定需要启动的协议类型列表，目前支持的协议类型：

  bgp

  bfd

  isis

  ospfv2

  ospfv3

  pim

  rip

  dot1x

  dhcpv4server

  dhcpv4

  dhcpv6server

dhcpv6

vxlan

saa

IGMP

igmpquery

mld

mldquery

l2tp

pppoe

ldp

lspping

pcep

返回 协议会话对象列表

实际案例

```
| ${result} | Get Sessions |
```

TesterLibrary.Protocol.common.**select_interface**(*Session*, *Interface*)

协议绑定测试仪表接口

参数

- **Session** (*object*) -- 测试仪表协议对象
- **Interface** (*object*) -- 测试仪表接口对象

返回 布尔值 Bool (范围：True / False)

实际案例

```
| Select Interface | Session=${Session} | Interface=${Interface} |
```

TesterLibrary.Protocol.common.**start_protocol**(*Ports=None*, *Protocols=None*, *Objects=None*)

测试仪表启动协议

参数

- **Ports** (*list*) -- 端口对象的列表
- **Protocols** (*list*) -- 指定需要启动的协议类型列表, 目前支持的协议类型：

  ospfv2

  ospfv3

  dhcpv4

  dhcpv6

  vxlan
- **Objects** (*list*) -- 当 Ports 和 Protocols 都为 None 时，可以指定协议会话对象列表

返回 布尔值 Bool (范围：True / False)

实际案例

```
| Start Protocol |
```

TesterLibrary.Protocol.common.**stop_protocol**(*Ports=None, Protocols=None, Objects=None*)

测试仪表停止协议

参数

- **Ports** (*list*) -- 端口对象的列表
- **Protocols** (*list*) -- 指定需要停止的协议类型列表，目前支持的协议类型：

  ospfv2

  ospfv3

  dhcpv4

  dhcpv6

  vxlan
- **Objects** (*list*) -- 当 Ports 和 Protocols 都为 None 时，可以指定协议会话对象列表

返回 布尔值 Bool (范围：True / False)

实际案例

```
| Stop Protocol |
```

TesterLibrary.Protocol.dhcpv4 module

TesterLibrary.Protocol.dhcpv4.**create_dhcp_client**(*Port, **kwargs*)

创建 DHCPv4 客户端协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象

关键字参数

- **Name** (*str*) -- DHCP 协议会话名称
- **Mode** (*str*) -- DHCPv4 客户端模式，默认值：CLIENT, 支持的参数：

  CLIENT

  RELAY_AGENT
- **HostName** (*str*) -- 主机名字，默认值：XINERTEL
- **ParameterRequests** (*list*) -- 主机请求选项, 默认值：['NONEOPTION', 'SUBNET_MASK', 'DOMAIN_NAME_SERVERS', 'DOMAIN_NAME', 'STATIC_ROUTES'], 支持的参数:

  SUBNET_MASK

  ROUTERS

  DOMAIN_NAME_SERVERS

DOMAIN_NAME

STATIC_ROUTES

IP_LEASE_TIME

SERVER_IDENTIFIER

T1

T2

- **EnableRouterOption** (*bool*) -- 启用路由选项, 默认值：False
- **VendorClassIdentifier** (*str*) -- 供应商识别, 默认值：XINERTEL
- **BroadcastFlag** (*str*) -- 默认值：BROADCAST, 支持参数：

  UNICAST

  BROADCAST
- **RelayAgentIp** (*str*) -- 代理端 IP, 取值范围：IPv4 地址, 默认值：1.1.1.1
- **ServerIp** (*str*) -- DHCPv4 服务端 IP, 取值范围：IPv4 地址, 默认值：2.1.1.3
- **EnableRelayAgentCircuitID** (*bool*) -- 使能代理电路标识, 默认值：False
- **RelayAgentCircuitID** (*str*) -- 代理电路标识, 默认值：""
- **EnableRelayAgentRemoteID** (*str*) -- 使能代理远程标识, 默认值：False
- **RelayAgentRemoteID** (*str*) -- 代理远程标识, 默认值：""
- **EnableSyncAddressToInterface** (*bool*) -- 使能同步地址到接口, 默认值：True
- **HostInterface** (Interface) -- 客户端接口对象, 类型：object

返回 DHCP 协议会话对象 Object

返回类型 (DhcpClient)

实际案例

```
| ${Session} | Create Dhcp Client | Port=${Port} | Name=DHCP_Client_1 |
```

TesterLibrary.Protocol.dhcpv4.**create_dhcp_client_custom_option**(*Session*, ***kwargs*)

创建测试仪表 DHCP 协议会话 option 对象

参数 **Session** (DhcpClient) -- DHCPv4 Client 协议对象, 类型为：object / list

关键字参数

- **OptionTag** (*number*) -- 可选项类型标识, 默认值：0, 取值范围：0-255
- **OptionType** (*str*) -- 可选项数据类型, 默认值：HEX, 支持以下参数：

  HEX

  STRING

  BOOLEAN

  INT8

  INT16

  INT32

  IP

- **EnableOptionValueList** (*bool*) -- 可选项值列表, 默认值：False
- **OptionValue** (*str*) -- 可选项值, 默认值：""
- **MessageType** (*str*) -- 携带 Option 消息类型, 默认值：DISCOVER, 支持以下参数：

  DISCOVER

  REQUEST

返回 测试仪表 DHCP 协议会话 option 对象 Object

返回类型 (Dhcpv4ClientOption)

实际案例

```
| Create Dhcp Client Custom Option | Session=${Sessions} |
```

TesterLibrary.Protocol.dhcpv4.**create_dhcp_server**(*Port*, ***kwargs*)

创建 DHCP Server 会话对象

参数 **Port** (Port) -- 测试仪表端口对象

关键字参数

- **Name** (*str*) -- DHCP Server 协议会话名称
- **LeaseTime** (*number*) -- 租约时间（秒）, 默认值：600, 范围：1-4294967295
- **RenewTime** (*number*) -- T1 租约更新时间 (%), 默认值：50, 范围：0-200
- **RebindTime** (*number*) -- T2 租约更新时间 (%), 默认值：87.5, 范围：0-200
- **MinLeaseTime** (*number*) -- 最小允许租约时间 (秒), 默认值：10, 范围：1-4294967295
- **DeclineReserveTime** (*number*) -- 资源释放等待时间 (秒), 默认值：10, 范围：1-600
- **OfferReserveTime** (*number*) -- 租约申请超时 (秒), 默认值：10, 范围：1-600
- **ServerHostName** (*str*) -- 服务端名字
- **DuplicateAddressDetection** (*bool*) -- 重复地址检测（DAD）, 默认值：False
- **DuplicateAddressDetectionTimeout** (*number*) -- DAD 超时时间, 默认值：0.5, 范围：0-60

返回 DHCP Server 会话对象 Object

返回类型 (DhcpServer)

实际案例

```
| Create Dhcp Server | Port=${Port} | RenewTime=100 |
```

TesterLibrary.Protocol.dhcpv4.**create_dhcp_server_address_pool**(*Sessions*, ***kwargs*)

创建 DHCP Server 会话对象地址池

参数 **Sessions** (DhcpServer) -- DHCPv4 Client 协议对象, 类型为：object / list

关键字参数

- **Name** (*str*) -- DHCP Server 协议会话名称

- **PoolAddressStart** (*str*) -- 开始地址, 取值范围：IPv4 地址, 默认值：1.1.1.2
- **PoolAddressStep** (*str*) -- 地址步长, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **PrefixLength** (*number*) -- 前缀长度, 默认值：24, 范围：0-32
- **RouterList** (*str*) -- 网关列表
- **LimitPoolCount** (*bool*) -- 限制地址池个数, 默认值：True
- **PoolAddressCount** (*number*) -- 地址池地址个数, 默认值：255, 范围：0-4294967295
- **DomainName** (*str*) -- 域名
- **DnsList** (*str*) -- DNS 地址列表

返回 DHCP 地址池对象列表 list

返回类型 (Dhcpv4AddressPool)

实际案例

```
| Create Dhcp Server Address Pool | Sessions=${Sessions} | PoolAddressStart=2.
↪2.2.2 |
```

TesterLibrary.Protocol.dhcpv4.**create_dhcp_server_custom_option**(*Session*, ***kwargs*)

创建测试仪表 DHCP 协议会话 option 对象

参数 **Session** (DhcpClient) -- DHCPv4 Client 协议对象, 类型为：object / list

关键字参数

- **OptionTag** (*number*) -- 可选项类型标识, 默认值：0, 取值范围：0-255
- **OptionType** (*str*) -- 可选项数据类型, 默认值：HEX, 支持以下参数：

  HEX

  STRING

  BOOLEAN

  INT8

  INT16

  INT32

  IP
- **EnableOptionValueList** (*bool*) -- 可选项值列表, 默认值：False
- **OptionValue** (*str*) -- 可选项值, 默认值：""
- **MessageType** (*str*) -- 携带 Option 消息类型, 默认值：DISCOVER, 支持以下参数：

  DISCOVER

  REQUEST

返回 测试仪表 DHCP 协议会话 option 对象 Object

返回类型 (Dhcpv4ClientOption)

实际案例

```
| Create Dhcp Server Custom Option | Session=${Sessions} |
```

TesterLibrary.Protocol.dhcpv4.**dhcp_abort**(*Sessions*)

终止测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Abort | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv4.**dhcp_bind**(*Sessions*)

Bind 测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Bind | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv4.**dhcp_rebind**(*Sessions*)

重新 Bind 测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Rebind | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv4.**dhcp_reboot**(*Sessions*)

重新启动测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Reboot | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv4.**dhcp_release**(*Sessions*)

释放测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Release | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv4.**dhcp_renew**(*Sessions*)

Renew 测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Renew | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv4.**edit_dhcp_client**(*Session*, ***kwargs*)

编写 DHCP 协议会话对象参数

参数 **Session** (DhcpClient) -- DHCPv4 Client 协议对象

关键字参数

- **Name** (*str*) -- DHCP 协议会话名称
- **Mode** (*str*) -- DHCPv4 客户端模式，默认值：CLIENT, 支持的参数：

  CLIENT

  RELAY_AGENT
- **HostName** (*str*) -- 主机名字，默认值：XINERTEL
- **ParameterRequests** (*list*) -- 主机请求选项, 默认值：['NONEOPTION', 'SUBNET_MASK', 'DOMAIN_NAME_SERVERS', 'DOMAIN_NAME', 'STATIC_ROUTES'], 支持的参数:

  SUBNET_MASK

  ROUTERS

  DOMAIN_NAME_SERVERS

  DOMAIN_NAME

  STATIC_ROUTES

  IP_LEASE_TIME

  SERVER_IDENTIFIER

  T1

T2

- **EnableRouterOption** (*bool*) -- 启用路由选项, 默认值：False
- **VendorClassIdentifier** (*str*) -- 供应商识别, 默认值：XINERTEL
- **BroadcastFlag** (*str*) -- 默认值：BROADCAST, 支持参数：

  UNICAST

  BROADCAST
- **RelayAgentIp** (*str*) -- 代理端 IP, 取值范围：IPv4 地址, 默认值：1.1.1.1
- **ServerIp** (*str*) -- DHCPv4 服务端 IP, 取值范围：IPv4 地址, 默认值：2.1.1.3
- **EnableRelayAgentCircuitID** (*bool*) -- 使能代理电路标识, 默认值：False
- **RelayAgentCircuitID** (*str*) -- 代理电路标识, 默认值：""
- **EnableRelayAgentRemoteID** (*str*) -- 使能代理远程标识, 默认值：False
- **RelayAgentRemoteID** (*str*) -- 代理远程标识, 默认值：""
- **EnableSyncAddressToInterface** (*bool*) -- 使能同步地址到接口, 默认值：True
- **HostInterface** (Interface) -- 客户端接口对象, 类型：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Dhcp Client | Session=${Session} | RelayAgentIp=2.2.2.2 |
```

TesterLibrary.Protocol.dhcpv4.**edit_dhcp_client_port_config**(*Ports*, ***kwargs*)

修改 DHCP 客户端端口配置对象

Args：

Ports (Port): 测试仪表端口对象, 类型为：object / list

关键字参数

- **SetupRate** (*int*) -- 建立速率, 默认值: 100, 范围: 1-65535
- **TeardownRate** -- 拆除速率, 默认值: 100, 范围: 1-65535
- **MaxOutstanding** -- 最大同时请求个数速率, 默认值: 100, 范围: 1-65535
- **LeaseTime** -- 期望租约时间（秒）, 默认值: 600, 范围: 1-4294967295
- **SessionRetryCount** -- 创建会话尝试次数, 默认值: 0, 范围: 0-65535
- **MessageRetryCount** -- 消息发送超时尝试次数, 默认值: 5, 范围: 0-65535
- **MessageTimeout** -- 消息超时时间（秒）, 默认值: 10, 范围: 1-99999
- **MaxMessageSize** -- 允许最大有效负荷（字节）, 默认值: 576, 范围: 291-1500

返回 Dhcpv4 Port Config 对象, 类型：object / list

返回类型 (Dhcpv4PortConfig)

实际案例

```
| Edit Dhcp Client Port Config | Port=${Port} | SetupRate=65535 |
```

TesterLibrary.Protocol.dhcpv4.**edit_dhcp_server**(*Session*, ***kwargs*)

编写 DHCP Server 会话对象参数

参数 **Session** (DhcpServer) -- DHCPv4 Client 协议对象

关键字参数

- **Name** (*str*) -- DHCP Server 协议会话名称
- **LeaseTime** (*number*) -- 租约时间（秒）, 默认值: 600, 范围: 1-4294967295
- **RenewTime** (*number*) -- T1 租约更新时间 (%), 默认值: 50, 范围: 0-200
- **RebindTime** (*number*) -- T2 租约更新时间 (%), 默认值: 87.5, 范围: 0-200
- **MinLeaseTime** (*number*) -- 最小允许租约时间 (秒), 默认值: 10, 范围: 1-4294967295
- **DeclineReserveTime** (*number*) -- 资源释放等待时间 (秒), 默认值: 10, 范围: 1-600
- **OfferReserveTime** (*number*) -- 租约申请超时 (秒), 默认值: 10, 范围: 1-600
- **ServerHostName** (*str*) -- 服务端名字
- **DuplicateAddressDetection** (*bool*) -- 重复地址检测（DAD）, 默认值: False
- **DuplicateAddressDetectionTimeout** (*number*) -- DAD 超时时间, 默认值: 0.5, 范围: 0-60

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Edit Dhcp Server | Session=${Session} | LeaseTime=1000 |
```

TesterLibrary.Protocol.dhcpv4.**edit_dhcp_server_port_config**(*Ports*, ***kwargs*)

修改 DHCP 服务器端口配置

Args:

Ports (Port): 测试仪表端口对象, 类型为: object / list

关键字参数

- **RenewRate** (*int*) -- 强制单播续租速度, 默认值: 100, 范围: 1-65535
- **MaxOutstanding** -- 最大请求个数, 默认值: 1000, 范围: 1-65535

返回 Dhcpv4 Server Port Config 对象, 类型: object / list

返回类型 (Dhcpv4ServerPortConfig)

实际案例

```
| Edit Dhcp Server Port Config | Port=${Port} | RenewRate=1000 |
```

TesterLibrary.Protocol.dhcpv4.**get_dhcp_client_block_statistic**(*Session*, *StaItems=None*)

获取 Dhcp Client Block Statistic 统计结果

参数

- **Session** (DhcpClient) -- Dhcp 客户端会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  BlockState

  AttemptRate

  SetupRate

  CurrentAttempt

  CurrentBound

  TotalAttempt

  TotalBound

  TotalFailed

  TotalReboot

  TotalRenew

  TotalRebind

  TotalRetry

  TxDiscover

  RxOffer

  TxRequest

  RxAck

  RxNak

  TxRenew

  TxRebind

  TxReboot

  TxRelease

  TxDecline

  RxForceRenew

返回

eg:

```
{
    'CurrentAttempt': 10,
    'CurrentBound': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | CurrentAttempt | CurrentBound |
| Subscribe Result | Types=Dhcpv4ClientBlockStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Client Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv4.**get_dhcp_client_statistic**(*Session*, *Id=1*, *StaItems=None*)

获取 Dhcp Client Statistic 统计结果

参数

- **Session** (DhcpClient) -- Dhcp 客户端会话对象, 类型为：Object
- **Id** (*int*) -- Dhcp 客户端会话 Index, 默认值: 1
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  ClientState

  IpAddress

  LeaseTime

  LeaseLeft

  ErrorStatus

  DiscoverResponseTime

  RequestResponseTime

返回

eg:

```
{
    'LeaseTime': 10,
    'DiscoverResponseTime': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | LeaseTime | DiscoverResponseTime |
| Subscribe Result | Types=Dhcpv4ClientStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Client Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv4.**get_dhcp_port_statistic**(*Port*, *StaItems=None*)

获取 Dhcp Port Statistic 统计结果

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：object

- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  CurrentAttempt

  CurrentBound

  TotalAttempt

  TotalBound

  TotalFailed

  TotalReboot

  TotalRenew

  TotalRebind

  TotalRetry

  TxDiscover

  RxOffer

  TxRequest

  RxAck

  RxNak

  TxRenew

  TxRebind

  TxReboot

  TxRelease

  RxForceRenew

返回

eg:

```
{
    'CurrentAttempt': 10,
    'CurrentBound': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | CurrentAttempt | CurrentBound |
| Subscribe Result | Types=Dhcpv4PortStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Port Statistic | Port=${Port} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv4.**get_dhcp_server_lease_statistic**(*Session*, *ClientId*, *StaItems=None*)

获取 Dhcp Server Lease Statistic 统计结果

参数

- **Session** (DhcpServer) -- DHCP 服务端会话对象, 类型为：object

- **ClientId** (*str*) -- DHCP Client Mac Address
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  ClientIp

  LeaseTime

  LeaseLeft

返回

eg:

```
{
    'LeaseTime': 100,
    'LeaseLeft': 50,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | LeaseTime | LeaseLeft |
| Subscribe Result | Types=Dhcpv4ServerStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Server Lease Statistic | Session=${Session} |␣
↪ClientId=00:00:12:01:01:03 | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv4.**get_dhcp_server_statistic**(*Session*, *StaItems=None*)

获取 Dhcp Server Statistic 统计结果

参数

- **Session** (DhcpServer) -- DHCP 服务端会话对象, 类型为：object / list
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  CurrentBound

  TotalBound

  TotalExpire

  TotalReboot

  TotalRenew

  TotalRebind

  TotalRelease

  RxDiscover

  TxOffer

  RxRequest

  TxAck

  TxNak

  RxDecline

RxRelease

TxForceRenew

返回

eg:

```
{
    'CurrentBound': 10,
    'TotalBound': 20,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | CurrentBound | TotalBound |
| Subscribe Result | Types=Dhcpv4ServerStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Server Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv4.**wait_dhcp_client_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 DHCP 协议会话达到指定状态

参数

- **Sessions** (*list*) -- DHCP 协议会话对象列表
- **State** (*str*) -- 等待 DHCP 协议会话达到的状态，默认值：BOUND，支持下列状态：

  NOT_READY

  IDLE

  BINDING

  BOUND

  RELEASING

  RENEWING

  REBINDING

  REBOOTING
- **Interval** (*number*) -- 查询 DHCP 协议会话的间隔，默认值：1 sec
- **TimeOut** (*number*) -- 等待 DHCP 协议会话状态的超时时间，默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcp Client State | Sessions=${Sessions} | State=${State} | Interval=$
↪{Interval} | TimeOut=${TimeOut} |
```

TesterLibrary.Protocol.dhcpv4.**wait_dhcp_server_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 DHCP 服务器协议会话达到指定状态

参数

- **Sessions** (*list*) -- DHCP Server 协议会话对象列表
- **State** (*str*) -- 等待 DHCP Server 协议会话达到的状态，默认值：UP，支持下列状态：

  NOT_READY

  DOWN

  UP
- **Interval** (*number*) -- 查询 DHCP 协议会话的间隔，默认值：1 sec
- **TimeOut** (*number*) -- 等待 DHCP 协议会话状态的超时时间，默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcp Server State | Sessions=${Sessions} | State=${State} | Interval=$
↪{Interval} | TimeOut=${TimeOut} |
```

TesterLibrary.Protocol.dhcpv6 module

TesterLibrary.Protocol.dhcpv6.**create_dhcpv6_client**(*Port*, ***kwargs*)

创建 DHCPv6 客户端会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- DHCPv6 客户端会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 DHCPv6 客户端会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EmulationMode** (*str*) -- 会话模式, 默认值：DHCPV6, 取值范围：

  DHCPV6

  DHCPV6PD

  DHCPV6ANDPD
- **EnableRenewMsg** (*bool*) -- 使能 Renew 消息, 默认值：True, 取值范围：True 或 False
- **EnableRebindMsg** (*bool*) -- 使能 Rebind 消息, 默认值：True, 取值范围：True 或 False
- **EnableReconfigAccept** (*bool*) -- 使能 Reconfigure 消息, 默认值：True, 取值范围：True 或 False

- **EnableSyncAddressInterface** (*bool*) -- 默认值：True, 取值范围：True 或 False
- **T1Timer** (*int*) -- T1 时刻（秒）, 取值范围：0-2147483647, 默认值：302400
- **T2Timer** (*int*) -- T2 时刻（秒）, 取值范围：0-2147483647, 默认值：483840
- **PreferredLifetime** (*int*) -- 首选生命周期（秒）, 取值范围：0-2147483647, 默认值：604800
- **ValidLifetime** (*int*) -- 有效生命周期（秒）, 取值范围：0-2147483647, 默认值：2592000
- **RapidCommitOptMode** (*str*) -- 快速交互模式, 默认值：DISABLE, 取值范围：

  DISABLE

  ENABLE

  SERVERSET
- **DuidType** (*str*) -- DUID 类型, 默认值：LL, 取值范围：

  LLT

  EN

  LL

  CUSTOM
- **DuidCustomValue** (*int*) -- 自定义 DUID 编号, 取值范围：0-65535, 默认值：1
- **DuidEnterpriseNumber** (*int*) -- DUID 企业编号, 取值范围：0-4294967295, 默认值：3456
- **DuidStartValue** (*str*) -- DUID 企业编号, 默认值：3456, 取值范围：匹配正则表达式”^([0-9a-fA-F]{1,256})$”
- **DuidStepValue** (*int*) -- DUID 标识符跳变, 取值范围：0-4294967295, 默认值：0x1
- **DestinationAddress** (*str*) -- 目的地址, 默认值：ALL, 取值范围：

  ALL

  SERVER
- **EnableRelayAgent** (*bool*) -- 使能中继代理, 默认值：False, 取值范围：True 或 False
- **RelayAgentIp** (*str*) -- 中继代理 IP, 默认值：2000::1, 取值范围：有效的 ipv6 地址
- **ServerIp** (*str*) -- 服务 IP, 默认值：2001::2, 取值范围：有效的 ipv6 地址
- **EnableUseRelayAgentMacForTraffic** (*bool*) -- 默认值：True
- **RequestPrefixLength** (*int*) -- 请求前缀长度, 取值范围：0-128, 默认值：64
- **RequestPrefixStartAddress** (*str*) -- 请求前缀地址, 默认值：’::’, 取值范围：有效的 ipv6 地址
- **ControlPlaneSrcIPv6Addr** (*str*) -- 默认值：LINKLOCAL, 取值范围：

  LINKLOCAL

  ROUTEADVERTISEMENT

- **RequestStartAddress** (*str*) -- 请求前缀地址, 默认值：'::', 取值范围：有效的 ipv6 地址
- **EnableAuthentication** (*bool*) -- 使能认证, 默认值：False, 取值范围：True 或 False
- **AuthenticationProtocol** (*str*) -- 认证协议, 默认值：DELAYED, 取值范围：

  DELAYED

  RECONFIGURATION
- **DhcpRealm** (*str*) -- 认证域名称, 默认值：'xinertel.com'
- **AuthenticationKeyId** (*int*) -- 取值范围：0-4294967295, 默认值：0
- **AuthenticationKey** (*str*) -- 认证秘钥 ID, 默认值：''
- **AuthenticationKeyType** (*str*) -- 认证秘钥类型, 默认值：ASCII, 取值范围：

  ASCII

  HEX
- **EnableDad** (*bool*) -- 使能重复地址检测（DAD）, 默认值：False, 取值范围：True 或 False
- **DadTimeout** (*int*) -- DAD 超时时间（秒）, 取值范围：1-4294967295, 默认值：2
- **DadTransmits** (*int*) -- DAD 传输次数, 取值范围：1-4294967295, 默认值：1
- **HostInterface** (Interface) -- 接口对象, 类型：object

返回 DHCPv6 客户端会话对象, 类型：object

返回类型 (Dhcpv6Client)

实际案例

```
| Create Dhcpv6 Client | Port=${Port} | DadTransmits=10 |
```

TesterLibrary.Protocol.dhcpv6.**create_dhcpv6_client_custom_options**(*Sessions*, ***kwargs*)

创建 DHCPv6 Client Custom Options 对象

Args：

Session (Dhcpv6Client): DHCPv6 客户端会话对象, 类型为：object / list

关键字参数

- **OptionVal** (*int*) -- 选项标识, 取值范围：0-255, 默认值：0
- **IncludeMsg** (*list*) -- 包含选项的消息类型, 默认值：['SOLICIT', 'REQUEST'], 取值范围：

  SOLICIT

  REQUEST

  CONFIRM

  RENEW

  REBIND

RELEASE

INFOREQUEST

RELAYFORWARD

- **Wildcards** (*bool*) -- 使能通配符, 默认值：False, 取值范围：True 或 False
- **StringIsHexadecimal** (*bool*) -- 使能十六进制字符, 默认值：False, 取值范围：True 或 False
- **OptionPayload** (*str*) -- 选项载荷, 默认值：”, 取值范围：string length in [0,512]
- **OptionHexPayload** (*int*) -- 十六进制选项载荷, 默认值：””
- **RemoveOption** (*bool*) -- 默认值：False, 取值范围：True 或 False

返回 DHCPv6 Client Custom Options 对象, 类型：object / list

返回类型 (Dhcpv6ClientCustomOptionsConfig)

实际案例

```
| ${Dhcpv6} | Create Dhcpv6 Client | Port=${Port} |
| Create Dhcpv6 Client Custom Options | Sessions=${Dhcpv6} | Wildcards=True |
```

TesterLibrary.Protocol.dhcpv6.**create_dhcpv6_server**(*Port*, ***kwargs*)

创建 DHCPv6 服务端会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- DHCPv6 服务端会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 DHCPv6 服务端会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EmulationMode** (*str*) -- 默认值：DHCPV6, 取值范围：

  DHCPV6

  DHCPV6PD

- **RenewalTimer** (*int*) -- T1 租约更新时间（%）, 取值范围：1-200, 默认值：50
- **RebindingTimer** (*int*) -- T2 租约更新时间（%）, 取值范围：1-200, 默认值：80
- **DnsList** (*str*) -- DNS 地址列表, 取值范围：IPv6 地址, 默认值：”::”
- **EnableDelayedAuth** (*bool*) -- 使能延时认证, 取值范围：True 或 False, 默认值：False
- **DhcpRealm** (*str*) -- DHCP 认证域名, 默认值：”xinertel.com”
- **AuthenticationKeyId** (*int*) -- 认证秘钥 ID, 取值范围：0-4294967295, 默认值：0
- **AuthenticationKey** (*str*) -- 认证秘钥, 默认值：””
- **AuthenticationKeyType** (*str*) -- 认证秘钥类型, 默认值：ASCII, 取值范围：

  ASCII

  HEX

- **EnabledReconfigureKey** (*bool*) -- 使能重新配置认证, 取值范围：True 或 False, 默认值：False
- **ReconfigureKey** (*str*) -- 重新配置秘钥类型, 默认值：""
- **ReconfigureKeyType** (*str*) -- 重新配置秘钥类型, 默认值：ASCII, 取值范围：

  ASCII

  HEX
- **EnabledDhcpv6Only** (*bool*) -- 使能单独 DHCPv6, 取值范围：True 或 False, 默认值：False
- **EnabledTcp** (*bool*) -- 使能 TCP, 取值范围：True 或 False, 默认值：False
- **TcpPort** (*int*) -- TCP 端口号, 取值范围：1-65535, 默认值：547
- **LeaseQueryStatusCode** (*str*) -- 租借查询应答码, 默认值：SUCCESS, 取值范围：

  SUCCESS

  UNKNOWN_QUERY_TYPE

  MALFORMED_QUERY

  NOT_CONFIGURED

  NOT_ALLOWED
- **BulkLeaseQueryStatusCode** (*str*) -- 批量租借查询应答码, 默认值：SUCCESS, 取值范围：

  SUCCESS

  UNKNOWN_QUERY_TYPE

  MALFORMED_QUERY

  NOT_CONFIGURED

  NOT_ALLOWED

  QUERY_TERMINATED
- **ActiveLeaseQueryStatusCode** (*str*) -- 活动租借查询应答码, 默认值：SUCCESS, 取值范围：

  SUCCESS

  UNKNOWN_QUERY_TYPE

  MALFORMED_QUERY

  NOT_CONFIGURED

  NOT_ALLOWED

  QUERY_TERMINATED

  DATA_MISSING

  CATCH_UP_COMPLETE

  NOT_SUPPORTED
- **StartTlsStatusCode** (*str*) -- 启动 TLS 应答码, 默认值：SUCCESS, 取值范围：

  SUCCESS

  TLS_CONNECTION_REFUSED

返回 DHCPv6 服务端会话对象, 类型：object

返回类型 (Dhcpv6Server)

实际案例

```
| Create Dhcpv6 Server | Port=${Port} | DadTransmits=10 |
```

TesterLibrary.Protocol.dhcpv6.**create_dhcpv6_server_address_pool**(*Sessions*, ***kwargs*)

创建 DHCPv6 Server Address Pool 对象

Args：

Session (Dhcpv6Server): DHCPv6 服务端会话对象, 类型为：object / list

关键字参数

- **PrefixLength** (*int*) -- 地址池前缀长度, 取值范围：0-128, 默认值：64
- **AssignMode** (*str*) -- 默认值：SUCCESS, 取值范围：

  CUSTOM

  EUI64
- **StartAddress** (*str*) -- 地址池起始地址, 取值范围：IPv6 地址, 默认值：”2001::1”
- **HostStep** (*str*) -- 地址池地址跳变, 取值范围：IPv6 地址, 默认值：”::1”
- **AddressCount** (*int*) -- 地址池地址总数, 取值范围：1-4294967295, 默认值：65535
- **PreferredLifetime** (*int*) -- 首选生命周期（秒）, 取值范围：0-4294967295, 默认值：604800
- **ValidLifetime** (*int*) -- 有效生命周期（秒）, 取值范围：0-4294967295, 默认值：2592000
- **MinIaidValue** (*int*) -- 最小 IAID 值, 取值范围：0-4294967295, 默认值：0
- **MaxIaidValue** (*int*) -- 最大 IAID 值, 取值范围：0-4294967295, 默认值：4294967295

返回 DHCPv6 Server Address Pool 对象, 类型：object / list

返回类型 (Dhcpv6AddressPoolsConfig)

实际案例

```
| ${Dhcpv6} | Create Dhcpv6 Server | Port=${Port} |
| Create Dhcpv6 Server Address Pool | Sessions=${Dhcpv6} | MinIaidValue=10 |
```

TesterLibrary.Protocol.dhcpv6.**create_dhcpv6_server_custom_options**(*Sessions*, ***kwargs*)

创建 DHCPv6 Server Custom Options 对象

Args：

Session (Dhcpv6Server): DHCPv6 服务端会话对象, 类型为：object / list

关键字参数

- **OptionVal** (*int*) -- 选项标识, 取值范围：0-255, 默认值：0
- **IncludeMsg** (*list*) -- 包含选项的消息类型, 默认值：['ADVERTISE', 'REPLY'], 取值范围：

  ADVERTISE

  REPLY

  RECONFIGURE

  RELAYREPLY
- **Wildcards** (*bool*) -- 使能通配符, 取值范围：True 或 False, 默认值：False
- **StringIsHexadecimal** (*bool*) -- 使能十六进制字符串, 取值范围：True 或 False, 默认值：False
- **OptionPayload** (*str*) -- 选项负载, 默认值：""
- **OptionHexPayload** (*str*) -- 选项负载, 默认值：""

返回 DHCPv6 Server Custom Options 对象, 类型：object / list

返回类型 (Dhcpv6ServerCustomOptionsConfig)

实际案例

```
| ${Dhcpv6} | Create Dhcpv6 Server | Port=${Port} |
| Create Dhcpv6 Server Custom Options | Sessions=${Dhcpv6} | Wildcards=True |
```

TesterLibrary.Protocol.dhcpv6.**create_dhcpv6_server_prefix_pool**(*Sessions*, ***kwargs*)

创建 DHCPv6 Server Prefix Pool 对象

Args：

Session (Dhcpv6Server): DHCPv6 服务端会话对象, 类型为：object / list

关键字参数

- **PrefixLength** (*int*) -- 前缀池前缀长度, 取值范围：0-128, 默认值：64
- **PrefixPoolStart** (*str*) -- 前缀池起始地址, 取值范围：IPv6 地址, 默认值："2001::1"
- **PrefixPoolStep** (*str*) -- 前缀池地址跳变, 取值范围：IPv6 地址, 默认值："0:0:0:1::"
- **PrefixAddressCount** (*int*) -- 前缀池地址总数, 取值范围：1-65535, 默认值：16
- **PreferredLifetime** (*int*) -- 最优租期（秒）, 取值范围：0-4294967295, 默认值：604800
- **ValidLifetime** (*int*) -- 有效租期（秒）, 取值范围：0-4294967295, 默认值：2592000
- **MinIaidValue** (*int*) -- 最小 IAID 值, 取值范围：0-4294967295, 默认值：0
- **MaxIaidValue** (*int*) -- 最大 IAID 值, 取值范围：0-4294967295, 默认值：4294967295

返回 DHCPv6 Server Prefix Pool 对象, 类型：object / list

返回类型 (Dhcpv6PrefixPoolsConfig)

实际案例

```
| ${Dhcpv6} | Create Dhcpv6 Server | Port=${Port} |
| Create Dhcpv6 Server Prefix Pool | Sessions=${Dhcpv6} | MinIaidValue=10 |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_abort**(*Sessions*)

中断 DHCPv6/PD 客户端

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Abort | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_active_lease_query**(*Sessions*)

DHCPv6 客户端活动租借查询

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Active Lease Query | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_bind**(*Sessions*)

DHCPv6 客户端绑定地址

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Bind | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_bulk_lease_query**(*Sessions*, ***kwargs*)

DHCPv6 客户端批量租借查询

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

Keyword Args:

QueryType (str): Bulk Leasequery 消息中 query-type 类型, 默认值: QUERY_BY_ADDRESS, 取值范围:

QUERY_BY_ADDRESS

QUERY_BY_CLIENTID

QUERY_BY_RELAY_ID

QUERY_BY_LINK_ADDRESS

QUERY_BY_REMOTE_ID

ClientAddress (str): 指定客户端 IPv6 地址, 默认值: '2000::1', 取值范围: 有效的 ipv6 地址

ClientId (str): 客户端 ID, 默认值: '', 取值范围: 匹配正则表达式"^([0-9a-fA-F]{0,512})$"

RelayIdentifier (str): 中继器 ID, 默认值: '', 取值范围: 匹配正则表达式"^([0-9a-fA-F]{0,512})$"

LinkAddress (str): 链路地址, 默认值: '2000::1', 取值范围: 有效的 ipv6 地址

RemoteId (str): 中继代理 Remote-ID 值, 默认值: ''

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Bulk Lease Query | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_confirm**(*Sessions*)

DHCPv6 客户端确认参数

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Confirm | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_info_request**(*Sessions*)

DHCPv6 客户端请求信息

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Info Request | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_lease_query**(*Sessions*, ***kwargs*)

DHCPv6 客户端租借查询

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

关键字参数

- **QueryType** (*str*) -- Leasequery 消息中 query-type 类型, 默认值: QUERY_BY_ADDRESS, 取值范围:

  QUERY_BY_ADDRESS

  QUERY_BY_CLIENTID

- **ClientAddress** (*str*) -- 客户端 IPv6 地址, 默认值: '2000::1', 取值范围: 有效的 ipv6 地址
- **ClientId** (*str*) -- 客户端 ID, 默认值: '', 取值范围: 匹配正则表达式"^([0-9a-fA-F]{0,512})$"

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Lease Query | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_rebind**(*Sessions*)

DHCPv6 客户端广播续租地址

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Rebind | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_release**(*Sessions*)

DHCPv6 客户端释放地址

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Release | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_renew**(*Sessions*)

DHCPv6 客户端单播续租地址

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Renew | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_client_start_tls**(*Sessions*)

DHCPv6 客户端启动 TLS

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Start Tls | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_server_abort**(*Sessions*)

中断 DHCPv6/PD 服务器

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Abort | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_server_reconfigure_rebind**(*Sessions*)

DHCPv6 服务端重新配置 Rebind

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Reconfigure Rebind | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_server_reconfigure_renew**(*Sessions*)

DHCPv6 服务端重新配置 Renew

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Reconfigure Renew | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_server_start**(*Sessions*)

启动 DHCPv6 服务端

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Start | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**dhcpv6_server_stop**(*Sessions*)

停止 DHCPv6 服务端

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Stop | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dhcpv6.**edit_dhcpv6_client_port_config**(*Ports*, ***kwargs*)

修改 DHCPv6 端口配置对象

Args：

Ports (Port): 测试仪表端口对象, 类型为：object

关键字参数

- **RequestRate** (*int*) -- Request 速率（会话/秒）, 取值范围：1-10000, 默认值：100
- **ReleaseRate** (*int*) -- Release 速率（会话/秒）, 取值范围：1-10000, 默认值：100
- **RenewRate** (*int*) -- Renew 速率（会话/秒）, 取值范围：1-10000, 默认值：100
- **MaxOutstanding** (*int*) -- 最大会话数量, 取值范围：1-2048, 默认值：1000
- **SolicitInitialTimeout** (*int*) -- Solicit 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：5
- **SolicitMaxTimeout** (*int*) -- Solicit 消息最大超时时间（秒）, 取值范围：1-99999, 默认值：120
- **SolicitRetryCount** (*int*) -- Solicit 消息重发次数, 取值范围：0-32, 默认值：10
- **SolicitIndefiniteRetry** (*bool*) -- Solicit 消息无限次重发, 默认值：False, 取值范围：True 或 False

- **SolicitDisableRetries** (*bool*) -- Solicit 消息无限次重发, 默认值: False, 取值范围: True 或 False
- **RequestInitialTimeout** (*int*) -- Request 消息初始超时时间（秒）, 取值范围: 1-99999, 默认值: 2
- **RequestMaxTimeout** (*int*) -- Request 消息最大超时时间（秒）, 取值范围: 1-99999, 默认值: 30
- **RequestRetryCount** (*int*) -- Request 消息重发次数, 取值范围: 0-32, 默认值: 10
- **RequestIndefiniteRetry** (*bool*) -- Request 消息无限次重发, 默认值: False, 取值范围: True 或 False
- **RequestDisableRetries** (*bool*) -- Request 消息禁止重发, 默认值: False, 取值范围: True 或 False
- **ConfirmInitialTimeout** (*int*) -- Confirm 消息初始超时时间（秒）, 取值范围: 1-99999, 默认值: 2
- **ConfirmMaxTimeout** (*int*) -- Confirm 消息最大超时时间（秒）, 取值范围: 1-99999, 默认值: 4
- **ConfirmMaxDuration** (*int*) -- Confirm 消息最大重发次数, 取值范围: 0-32, 默认值: 5
- **RenewInitialTimeout** (*int*) -- Renew 消息初始超时时间（秒）, 取值范围: 0-32, 默认值: 10
- **RenewMaxTimeout** (*int*) -- Renew 消息最大超时时间（秒）, 取值范围: 1-99999, 默认值: 600
- **RenewRetryCount** (*int*) -- Renew 消息重发次数, 取值范围: 0-32, 默认值: 5
- **RenewIndefiniteRetry** (*bool*) -- Renew 消息无限次重发, 默认值: False, 取值范围: True 或 False
- **RenewDisableRetries** (*bool*) -- Renew 消息禁止重发, 默认值: False, 取值范围: True 或 False
- **RebindInitialTimeout** (*int*) -- Rebind 消息初始超时时间（秒）, 取值范围: 1-99999, 默认值: 10
- **RebindMaxTimeout** (*int*) -- Rebind 消息最大超时时间（秒）, 取值范围: 1-99999, 默认值: 600
- **RebindRetryCount** (*int*) -- Rebind 消息重发次数, 取值范围: 0-32, 默认值: 5
- **RebindIndefiniteRetry** (*bool*) -- Rebind 消息无限次重发, 默认值: False, 取值范围: True 或 False
- **RebindDisableRetries** (*bool*) -- Rebind 消息禁止重发, 默认值: False, 取值范围: True 或 False
- **ReleaseInitialTimeout** (*int*) -- Release 消息初始超时时间（秒）, 取值范围: 1-99999, 默认值: 1
- **ReleaseRetryCount** (*int*) -- Release 消息重发次数, 取值范围: 0-32, 默认值: 3
- **ReleaseIndefiniteRetry** (*bool*) -- Release 消息无限次重发, 默认值: False, 取值范围: True 或 False
- **ReleaseDisableRetries** (*bool*) -- Release 消息禁止重发, 默认值: False, 取值范围: True 或 False

- **DeclineInitialTimeout** (*int*) -- Decline 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：1
- **DeclineRetryCount** (*int*) -- Decline 消息重发次数, 取值范围：0-32, 默认值：5
- **DeclineIndefiniteRetry** (*bool*) -- Decline 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **DeclineDisableRetries** (*bool*) -- Decline 消息禁止重发, 默认值：False, 取值范围：True 或 False
- **InfoRequestInitialTimeout** (*int*) -- Information-Request 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：1
- **InfoRequestMaxTimeout** (*int*) -- Information-Request 消息最大超时时间（秒）, 取值范围：1-99999, 默认值：120
- **InfoRequestRetryCount** (*int*) -- Information-Request 消息重发次数, 取值范围：0-32, 默认值：5
- **InfoRequestIndefiniteRetry** (*bool*) -- Information-Request 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **InfoRequestDisableRetries** (*bool*) -- Information-Request 消息禁止重发, 默认值：False, 取值范围：True 或 False
- **TcpServerPort** (*int*) -- TCP 服务端口号, 取值范围: 1-65535, 默认值: 547

返回 DHCPv6 Client Custom Options 对象, 类型：object / list

返回类型 (Dhcpv6PortRateConfig)

实际案例

```
| Edit Dhcpv6 Client Port Config | Ports=${Port} | TcpServerPort=10 |
```

TesterLibrary.Protocol.dhcpv6.**get_dhcpv6_client_block_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Client Block Statistic 统计结果

参数

- **Session** (Dhcpv6Client) -- Dhcpv6 客户端会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  BlockSessionState

  CurrentlyAttempting

  CurrentlyIdl

  CurrentlyBound

  AttemptRate

  BindRate

  RebindRate

  ReleaseRate

  RenewRate

AverageRebindToReplyTime
AverageReleaseToReplyTime
AverageRenewToReplyTime
AverageRequestToReplyTime
AverageSolicitToAdvertiseTime
AverageSolicitToReplyTime
MaxRebindToReplyTime
MaxReleaseToReplyTime
MaxRenewToReplyTime
MaxRequestToReplyTime
MaxSolicitToAdvertiseTime
MaxSolicitToReplyTime
MinRebindToReplyTime
MinReleaseToReplyTime
MinRenewToReplyTime
MinRequestToReplyTime
MinSolicitToAdvertiseTime
MinSolicitToReplyTime
AdvertiseRxCount
ReplyRxCount
ReconfigureRxCount
SolicitTxCount
RequestTxCount
ReleaseTxCount
RenewTxCount
RebindTxCount
ConfirmTxCount
InfoRequestTxCount
TotalAttempted
TotalBound
TotalFailed
TotalRebound
TotalReleased
TotalReleaseRetried
TotalRenewed
TotalRenewedRetried
TotalRetired

返回

eg:

```
{
    'TotalRenewedRetried': 10,
    'TotalRetired': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ClientBlockStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Client Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv6.**get_dhcpv6_client_statistic**(*Session*, *Id*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Client Statistic 统计结果

参数

- **Session** (Dhcpv6Client) -- Dhcpv6 客户端会话对象, 类型为：Object
- **Id** (*int*) -- Dhcpv6 客户端会话 Index
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  IaidValue

  MacAddr

  LeaseRx

  AddressType

  SessionState

  StateCode

  IpAddress

  LeaseRemaining

  PrefixLength

  RequestResponseTime

  SolicitResponseTime

返回

eg:

```
{
    'RequestResponseTime': 10,
    'SolicitResponseTime': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ClientStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Client Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv6.**get_dhcpv6_pd_client_statistic**(*Session*, *Id*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 pd Client Statistic 统计结果

参数

- **Session** (Dhcpv6Client) -- Dhcpv6 客户端会话对象, 类型为：Object
- **Id** (*int*) -- Dhcpv6 客户端会话 Index
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  Dhcpv6PdClientId

  IaidValue

  SessionIndex

  MacAddr

  VlanId

  LeaseRx

  AddressType

  SessionState

  StateCode

  IpAddress

  LeaseRemaining

  PrefixLength

  RequestResponseTime

  SolicitResponseTime

返回

eg:

```
{
    'RequestResponseTime': 10,
    'SolicitResponseTime': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ClientStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Client Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv6.**get_dhcpv6_port_statistic**(*Port*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Port Statistic 统计结果

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  CurrentlyAttempting

  CurrentlyIdl

  CurrentlyBound

  AverageSetupTime

  MaxSetupTime

  MinSetupTime

  SolicitTxCount

  RequestTxCount

  ReleaseTxCount

  RenewTxCount

  RebindTxCount

  ConfirmTxCount

  InfoRequestTxCount

  AdvertiseRxCount

  ReconfigureRxCount

  ReplyRxCount

  SuccessPercentage

  TotalAttempted

  TotalBound

  TotalBoundFailed

  TotalRebound

  TotalReleased

  TotalReleaseRetried

  TotalRenewed

  TotalRenewedRetried

  TotalRetired

返回

eg:

```
{
    'TotalRenewedRetried': 10,
    'TotalRetired': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6PortStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Port Statistic | Port=${Port} | StaItems=@{StaItems}
↪|
| Clear Result |
```

TesterLibrary.Protocol.dhcpv6.**get_dhcpv6_server_lease_statistic**(*Session*, *Pool*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Server Lease Statistic 统计结果

参数

- **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象，类型为：object / list
- **Pool** (Dhcpv6AddressPoolsConfig) -- DHCPv6 Server Address Pool 对象
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  ServerState

  CurrentlyBound

  ReconfigureRebindTxCount

  ReconfigureRenewTxCount

  ReconfigureTxCount

  AdvertiseTxCount

  ReplyTxCount

  SolicitRxCount

  RequestRxCount

  ReleaseRxCount

  RenewRxCount

  RebindRxCount

  TotalBound

  TotalExpired

  TotalReleased

TotalRenewed

返回

eg:

```
{
    'TotalReleased': 10,
    'TotalRenewed': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ServerStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Server Lease Statistic | Session=${Session} | Pool=$
↪{Pool} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv6.**get_dhcpv6_server_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Server Statistic 统计结果

参数

- **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象，类型为：object / list
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  ServerState

  CurrentlyBound

  ReconfigureRebindTxCount

  ReconfigureRenewTxCount

  ReconfigureTxCount

  AdvertiseTxCount

  ReplyTxCount

  SolicitRxCount

  RequestRxCount

  ReleaseRxCount

  RenewRxCount

  RebindRxCount

  TotalBound

  TotalExpired

  TotalReleased

  TotalRenewed

返回

eg:

```
{
    'TotalReleased': 10,
    'TotalRenewed': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ServerStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Server Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dhcpv6.**wait_dhcpv6_client_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Dhcpv6 客户端会话达到指定状态

参数

- **Session** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list
- **State** (*str*) -- 等待 DHCPv6 客户端会话组达到的状态, 类型为：string, 默认值：达到 BOUND, 支持下列状态：

  DISABLED

  IDLE

  BOUND

  SOLICITING

  REQUESTING

  RELEASING

  RENEWING

  REBINDING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcpv6 Client State | Sessions=${Sessions} | State=BOUND | Interval=2␣
↪| TimeOut=120 |
```

TesterLibrary.Protocol.dhcpv6.**wait_dhcpv6_pd_client_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Dhcpv6 PD 客户端会话达到指定状态

参数

- **Session** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list
- **State** (*str*) -- 等待 DHCPv6 客户端会话组达到的状态, 类型为：string, 默认值：达到 BOUND, 支持下列状态：

  DISABLED

  IDLE

  BOUND

  SOLICITING

  REQUESTING

  RELEASING

  RENEWING

  REBINDING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcpv6 Pd Client State | Sessions=${Sessions} | State=BOUND |␣
↪Interval=2 | TimeOut=120 |
```

TesterLibrary.Protocol.dhcpv6.**wait_dhcpv6_server_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Dhcpv6 服务端会话达到指定状态

参数

- **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list
- **State** (*str*) -- 等待 DHCPv6 客户端会话组达到的状态, 类型为：string, 默认值：达到 UP, 支持下列状态：

  NOTSTART

  UP

  DISABLED

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcpv6 Server State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.dot1x module

TesterLibrary.Protocol.dot1x.**abort_dot1x**(*Sessions*)

中断 802.1x 会话

参数 **Sessions** (Dot1x) -- 802.1x 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Abort Dot1x | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dot1x.**create_dot1x**(*Port*, ***kwargs*)

创建 802.1x 会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- 802.1x 会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 801.1x 会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AuthMode** (*str*) -- 默认值：MD5, 取值范围：

  MD5

  TLS

  TTLS
- **Identity** (*str*) -- 默认值：xinertel, 取值范围：string length in [1,255]
- **Password** (*str*) -- 默认值：xinertel, 取值范围：string length in [1,255]
- **UseAuthenticatorMac** (*bool*) -- 使能 801.1x 会话, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AuthenticatorMac** (*str*) -- 默认值：01:80:c2:00:00:03, 取值范围：有效的 mac 地址
- **RetryCount** (*int*) -- 默认值：5, 取值范围：uint32
- **RetryTimeout** (*int*) -- 默认值：5, 取值范围：1-4294967295
- **RetransmitCount** (*int*) -- 默认值：5, 取值范围：uint32

- **RetransmitTimeout** (*int*) -- 默认值：5, 取值范围：1-4294967295
- **SupplicantCertificateName** (*str*) -- 默认值：‘’, 取值范围：string length in [1,255]
- **CertificatePassword** (*str*) -- 默认值：‘’, 取值范围：string length in [1,255]
- **DuplicateUserInfoToInner** (*bool*) -- 使能 802.1x 会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **InnerIdentity** (*str*) -- 默认值：xinertel, 取值范围：string length in [1,255]
- **InnerPassword** (*str*) -- 默认值：xinertel, 取值范围：string length in [1,255]
- **InnerTunnelAuthMode** (*str*) -- 默认值：AUTO, 取值范围：

  AUTO

  GTC

  MS_CHAPV2

  MD5
- **EnableClientCertificate** (*bool*) -- 使能 802.1x 会话, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 802.1x 会话对象, 类型：object

返回类型 (Dot1x)

实际案例

```
| Create Dot1x | Port=${Port} | DadTransmits=10 |
```

TesterLibrary.Protocol.dot1x.**dot1x_delete_certificate**(*Sessions*)

删除 802.1x 证书

参数 **Sessions** (Dot1x) -- 802.1x 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dot1x Upload Certificate | Sessions=${Sessions} |
```

TesterLibrary.Protocol.dot1x.**dot1x_upload_certificate**(*Sessions*, *Folder*)

上传 802.1x 证书

参数

- **Sessions** (Dot1x) -- 802.1x 会话对象, 类型为：object / list
- **Folder** (*str*) -- 证书所在路径，e.g.’c:/CertificateFolder’

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dot1x Upload Certificate | Sessions=${Sessions} | Folder=${Folder} |
```

TesterLibrary.Protocol.dot1x.**edit_dot1x_port_config**(*Ports*, ***kwargs*)

修改 802.1x 端口配置对象

Args：

Ports (Port): 测试仪表端口对象, 类型为：object

关键字参数

- **AuthenticationRate** (*int*) -- 默认值：100, 取值范围：1-16384
- **LogoutRate** (*int*) -- 默认值：100, 取值范围：1-16384
- **OutstandingSessions** (*int*) -- 默认值：100, 取值范围：1-10000

返回 802.1x 端口配置对象, 类型：object / list

返回类型 (Dot1xPortConfig)

实际案例

```
| Edit dot1x Port Config | Ports=${Port} | OutstandingSessions=10 |
```

TesterLibrary.Protocol.dot1x.**get_dot1x_block_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 802.1x session block 统计结果

参数

- **Session** (Dot1x) -- 802.1x 会话对象, 类型为：object / list
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  BlockState

  CurrentAuthenticatedAttempt

  CurrentAuthenticated

  CurrentFailed

  CurrentLogoff

  AuthenticatedAttemptRate

  AuthenticatedRate

  LogoffRate

  TotalAttempt

  TotalAuthenticated

  TotalFailed

  TotalLogoff

  TotalRetry

  TotalRetransmit

  RxEapFailure

  RxEapRequest

RxEapSucess

TxEapResponse

MaxAuthenticatedTime

MaxLogoffTime

返回

eg:

```
{
    'MaxAuthenticatedTime': 10,
    'MaxLogoffTime': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dot1xBlockStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dot1x Block Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dot1x.**get_dot1x_port_statistic**(*Port*, *StaItems: Optional[list] = None*)

获取 802.1x port block 统计结果

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  CurrentAuthenticatedAttempt

  CurrentAuthenticated

  CurrentFailed

  CurrentLogoff

  TotalAttempt

  TotalAuthenticated

  TotalFailed

  TotalLogoff

  TotalRetry

  TotalRetransmit

返回

eg:

```
{
    'TotalRetry': 10,
    'TotalRetransmit': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dot1xPortStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dot1x Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.dot1x.**get_dot1x_statistic**(*Session*, *Index*, *StaItems: Optional[list] = None*)

获取 802.1x 统计结果

参数

- **Session** (Dot1x) -- 802.1x 会话对象, 类型为：object / list
- **Index** (*int*) -- Session Index
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  State

  ReqIdentity

  RespIdentity

  ReqChallenge

  RespChallenge

  TLSEstablish

  ReceiveOK

  ReceiveFail

返回

eg:

```
{
    'ReceiveOK': 10,
    'ReceiveFail': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dot1xStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dot1x Statistic | Session=${Session} | StaItems=@{StaItems}␣
↪|
| Clear Result |
```

TesterLibrary.Protocol.dot1x.**wait_dot1x_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 802.1x 会话达到指定状态

参数

- **Session** (Dot1x) -- 802.1x 会话对象, 类型为：object / list
- **State** (*str*) -- 等待 802.1x 会话组达到的状态, 类型为：string, 默认值：达到 AUTHENTICATED, 支持下列状态：

  DISABLED

  DOWN

  UNAUTHORIZED

  AUTHENTICATING

  AUTHENTICATED

  FAILED

  LOGGING_OFF
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dot1x State | Sessions=${Sessions} | State=AUTHENTICATED | Interval=2
↪| TimeOut=120 |
```

TesterLibrary.Protocol.igmp module

TesterLibrary.Protocol.igmp.**apply_igmp_querier**(*Sessions*)

IGMP Querier 增量配置下发到后台

参数 **Sessions** (list (IgmpQuerier)) -- IGMP Querier 协会话对象, 类型为: object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Apply Igmp Querier |
```

TesterLibrary.Protocol.igmp.**create_igmp**(*Port*, ***kwargs*)

创建 IGMP 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- IGMP 协会话名称, 类型为：string

- **Enable** (*bool*) -- 使能 ICMP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：IGMPV2, 支持版本：

  IGMPV1

  IGMPV2

  IGMPV3
- **PackReports** (*bool*) -- 合并报告报文, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InitialJoin** (*bool*) -- 单个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustJoin** (*bool*) -- 多个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustnessVariable** (*int*) -- Robust 值, 类型为：number, 取值范围：2-255, 默认值：2
- **UnsolicitedReportInterval** (*int*) -- 发送初始报文的时间间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **ForceLeave** (*bool*) -- 强制发送 Leave 报文, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterPresentTimeout** (*int*) -- IGMPv1 路由器存在的超时时间 (秒), 类型为：number, 取值范围：0-65535, 默认值：400
- **NotFragment** (*bool*) -- 设置 IP 头报文分片标志位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **TosValue** (*bool*) -- 设置 IP 头 TOS 值 (Hex), 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 IGMP 协议会话对象, 类型：object

返回类型 (Igmp)

实际案例

```
| Create Igmp | Port=${Port} | Version=IGMPV3 |
```

TesterLibrary.Protocol.igmp.**create_igmp_querier**(*Port*, ***kwargs*)

创建 IGMP Querier 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- IGMP Querier 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 IGMP Querier 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：IGMPV2, 支持版本：

  IGMPV1

  IGMPV2

  IGMPV3
- **RobustnessVariable** (*int*) -- 健壮系数, 取值范围：2-255, 默认值: 2

- **Interval** (*int*) -- 查询时间间隔（秒）, 取值范围：0-4294967295, 默认值: 125
- **QueryResponseInterval** (*int*) -- 查询响应时间间隔（毫秒）, 取值范围: 0-4294967295, 默认值: 10000
- **StartupQueryCount** (*int*) -- 初始查询报文个数, 取值范围：1-255, 默认值: 2
- **LastMemberQueryInterval** (*int*) -- 最后成员查询时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 1000
- **LastMemberQueryCount** (*bool*) -- 最后成员查询次数, 取值范围：0-255, 默认值: 2
- **IPv4DoNotFragment** (*bool*) -- 设置 IP 头报文分片标志位, 取值范围：True 或 False, 默认值: False
- **IPv4TosValue** (*str*) -- 设置 IP 头 TOS 值, 类型为：bool, 取值范围：0x0-0xff, 默认值: 0xc0

返回 IGMP 协议会话对象, 类型：object

返回类型 (IgmpQuerier)

实际案例

```
| Create Igmp Querier| Port=${Port} | Version=IGMPV3 |
```

TesterLibrary.Protocol.igmp.**edit_igmp**(*Session*, ***kwargs*)

编辑 IGMP 协议会话对象

参数 **Session** (Igmp) -- IGMP 协会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- IGMP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ICMP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：IGMPV2, 支持版本：

  IGMPV1

  IGMPV2

  IGMPV3
- **PackReports** (*bool*) -- 合并报告报文, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InitialJoin** (*bool*) -- 单个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustJoin** (*bool*) -- 多个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustnessVariable** (*int*) -- Robust 值, 类型为：number, 取值范围：2-255, 默认值：2
- **UnsolicitedReportInterval** (*int*) -- 发送初始报文的时间间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **ForceLeave** (*bool*) -- 强制发送 Leave 报文, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **RouterPresentTimeout** (*int*) -- IGMPv1 路由器存在的超时时间 (秒), 类型为：number, 取值范围：0-65535, 默认值：400
- **NotFragment** (*bool*) -- 设置 IP 头报文分片标志位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **TosValue** (*bool*) -- 设置 IP 头 TOS 值 (Hex), 类型为: bool, 取值范围: True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Igmp | Port=${Port} | Version=IGMPV1 | RouterPresentTimeout=500 |
```

TesterLibrary.Protocol.igmp.**edit_igmp_querier**(*Session*, ***kwargs*)

编辑 IGMP Querier 协议会话对象

参数 **Session** (IgmpQuerier) -- IGMP 协会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- IGMP Querier 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ICMP Querier 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：IGMPV2, 支持版本：

  IGMPV1

  IGMPV2

  IGMPV3
- **RobustnessVariable** (*int*) -- 健壮系数, 取值范围：2-255, 默认值: 2
- **Interval** (*int*) -- 查询时间间隔（秒）, 取值范围：0-4294967295, 默认值: 125
- **QueryResponseInterval** (*int*) -- 查询响应时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 10000
- **StartupQueryCount** (*int*) -- 初始查询报文个数, 取值范围：1-255, 默认值: 2
- **LastMemberQueryInterval** (*int*) -- 最后成员查询时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 1000
- **LastMemberQueryCount** (*bool*) -- 最后成员查询次数, 取值范围：0-255, 默认值: 2
- **IPv4DoNotFragment** (*bool*) -- 设置 IP 头报文分片标志位, 取值范围：True 或 False, 默认值: False
- **IPv4TosValue** (*str*) -- 设置 IP 头 TOS 值, 类型为：bool, 取值范围：0x0-0xff, 默认值: 0xc0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Igmp Querier | Port=${Port} | Version=IGMPV3 | IPv4TosValue=0xff |
```

TesterLibrary.Protocol.igmp.**get_igmp_host_statistic**(*Session*, *StaItems=None*)

获取 Igmp 协议会话统计结果

参数

- **Session** (Igmp) -- Igmp 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  IgmpTxFrames

  IgmpRxFrames

  IgmpRxUnknownTypes

  IgmpRxChecksumErrors

  IgmpRxLengthErrors

返回

eg:

```
{
    'IgmpTxFrames': 8,
    'IgmpRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | IgmpTxFrames | IgmpRxFrames |
| Subscribe Result | Types=IgmpHostResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Igmp Host Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.igmp.**get_igmp_port_statistic**(*Port*, *StaItems=None*)

获取 Igmp Port 统计结果

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  IgmpTxFrames

  IgmpRxFrames

  IgmpTxV1Reports

  IgmpTxV2Reports

  IgmpTxLeaveGroups

  IgmpTxV3Reports

IgmpTxV3ModeInclude

IgmpTxV3ModeExclude

IgmpTxV3ModeChangeToInclude

IgmpTxV3ModeChangeToExclude

IgmpTxV3ModeAllowNewSources

IgmpTxV3ModeBlockOldSources

IgmpRxV1Queries

IgmpRxV2Queries

IgmpRxV3Queries

IgmpRxGeneralQueries

IgmpRxGroupSpecificQueries

IgmpRxGroupAndSourceSpecificQueries

IgmpRxUnknownTypes

IgmpRxChecksumErrors

IgmpRxLengthErrors

返回

eg:

```
{
    'IgmpTxFrames': 8,
    'IgmpRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | IgmpTxFrames | IgmpRxFrames |
| Subscribe Result | Types=IgmpPortAggregatedResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Igmp Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.igmp.**get_igmp_querier_statistic**(*Session*, *StaItems=None*)

获取 Igmp Querier 协议会话统计结果

参数

- **Session** (IgmpQuerier) -- Igmp 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  QuerierTxFrames

  QuerierRxFrames

  QuerierRxUnknownTypes

  QuerierRxChecksumErrors

QuerierRxLengthErrors

返回

eg:

```
{
    'QuerierTxFrames': 8,
    'QuerierRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | QuerierTxFrames | QuerierRxFrames |
| Subscribe Result | Types=IgmpQuerierResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Igmp Querier Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.igmp.**select_source_interface**(*Session*, *Memberships*, *Interface*)

将协议会话组播组过滤源地址绑定到指定接口

参数

- **Session** (Igmp) -- IGMP/MLD 协会话对象, 类型为：object
- **Memberships** (MldMembershipsConfig) -- 组播协议和组播组绑定关系对象, 类型为：object
- **Interface** (Interface) -- 测试仪表接口对象, 类型为：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Interface} |  create_interface | Port=${Port} |
| ${Group} | Create Multicast Group | Version=IPV4 | Start=225.0.1.1 |␣
↪Number=20 |
| ${Session} | Create Igmp | Port=${Port} | Version=IGMPV3 |
| ${Memberships} | Create Memberships | Session=${Session} | Start=225.0.1.1␣
↪| DeviceGroupMapping=ONETOONE |
| binding_multicast_group | Session=${Session} | Memberships=${Memberships} |␣
↪MulticastGroup=${Group} |
| Select Source Interface | Session=${Session} | Memberships=${Memberships} |␣
↪Interface=${Interface} |
```

TesterLibrary.Protocol.igmp.**wait_igmp_querier_state**(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 Igmp Querier 协议会话达到指定状态

参数

- **Sessions** (list (IgmpQuerier)) -- Igmp Querier 协议会话对象列表

- **State** (*str*) -- 等待 Igmp Querier 协议会话达到的状态, 默认值：达到 UP, 支持下列状态：

  NOTSTARTED

  UP

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Igmp Querier State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.igmp.**wait_igmp_state**(*Sessions*, *State='MEMBER'*, *Interval=1*, *TimeOut=60*)

等待 Igmp 协议会话达到指定状态

参数

- **Sessions** (list (Igmp)) -- Igmp 协议会话对象列表
- **State** (*str*) -- 等待 Igmp 协议会话达到的状态, 默认值：达到 MEMBER, 支持下列状态：

  NONMEMBER JOINING MEMBER LEAVING

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Igmp State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.isis module

TesterLibrary.Protocol.isis.**advertise_isis**(*Lsps*)

通告 Isis 协议会话 lsp

参数 **Lsps** (IsisLspConfig) -- ISIS LSP 对象, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Isis | Lsp=${Lsp} |
```

TesterLibrary.Protocol.isis.**create_isis**(*Port*, ***kwargs*)

创建 ISIS 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- ISIS 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ISIS 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPV4, 支持版本：

  IPV4

  IPV6

  IPV4IPV6
- **Level** (*str*) -- 区域类型, 类型为：string, 默认值：L2, 支持版本：

  L1

  L2

  L1L2
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 默认值：BROADCAST, 支持参数：

  BROADCAST

  P2P
- **SystemId** (*str*) -- 系统 ID, 类型为：string, 取值范围：MAC 地址, 默认值：00:00:00:00:00:01
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-127, 默认值：0
- **AuthMethod** (*str*) -- 认证方式, 类型为：string, 默认值：NONE, 支持参数：

  NONE

  SIMPLE

  MD5
- **Password** (*str*) -- 4 字节自治域跳变, 类型为：string, 默认值：Xinertel
- **CircuitId** (*int*) -- 电路 ID, 类型为：number, 取值范围：0-255, 默认值：1
- **Area1** (*str*) -- 区域 ID 1, 类型为：hex number, 取值范围：0x0-0xff, 默认值：0x10
- **Area2** (*str*) -- 区域 ID 2, 类型为：hex number, 取值范围：0x0-0xff, 默认值：空
- **Area3** (*str*) -- 区域 ID 3, 类型为：hex number, 取值范围：0x0-0xff, 默认值：空
- **MetricMode** (*str*) -- 度量模式, 类型为：string, 默认值：NARROWWIDE, 支持参数：

  NARROW

  WIDE

NARROWWIDE

- **TeRouterId** (*str*) -- TE 路由器 ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1
- **TeRouterIdIpv6** (*str*) -- IPv6 TE 路由器 ID, 类型为：string, 取值范围：IPv6 地址, 默认值：3000::1
- **HelloInterval** (*int*) -- Hello PDU 发送间隔 (秒), 类型为：number, 取值范围：1-300, 默认值：10
- **HelloMultiplier** (*int*) -- Hello 时间间隔倍数, 类型为：number, 取值范围：1-100, 默认值：3
- **PsnInterval** (*int*) -- PSNP 发送间隔 (秒), 类型为：number, 取值范围：1-20, 默认值：2
- **LspRefreshTime** (*int*) -- LSP 刷新时间 (秒), 类型为：number, 取值范围：1-65535, 默认值：900
- **RetransInterval** (*int*) -- LSP 重传间隔 (秒), 类型为：number, 取值范围：1-100, 默认值：5
- **HelloPadding** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LspSize** (*int*) -- LSP 大小, 类型为：number, 取值范围：100-1492, 默认值：1492
- **ValidateIpAddr** (*bool*) -- 使能接口校验, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableBFD** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MtParams** (*int*) -- 多拓扑参数数量, 类型为：number, 取值范围：0-2, 默认值：0
- **PerPduAuthentication** (*int*) -- Per PDU 认证数量, 类型为：number, 取值范围：0-4, 默认值：0
- **ReportLabel** (*bool*) -- 使能 ReportLabel, 类型为：bool, 默认值：True
- **LearnRoute** (*bool*) -- 使能 LearnRoute, 类型为：bool, 默认值：True
- **RecordLspNextSequenceNum** (*bool*) -- 使能 Record Lsp Next Sequence Number, 类型为：bool, 默认值：True
- **L1NarrowMetric** (*int*) -- L1 Narrow Metric, 类型为：number, 取值范围：0-63, 默认值：1
- **L1WideMetric** (*int*) -- L1 Wild Metric, 类型为：number, 取值范围：0-16777214, 默认值：1
- **L2NarrowMetric** (*int*) -- L2 Narrow Metric, 类型为：number, 取值范围：0-63, 默认值：1
- **L2WideMetric** (*int*) -- L2 Wide Metric, 类型为：number, 取值范围：0-16777214, 默认值：1

返回 ISIS 协议会话对象

返回类型 (IsisRouter)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${MtId} | Create List | IPV4 | IPV6 |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True | MtParams=1 |
```

TesterLibrary.Protocol.isis.**create_isis_binding_sr_sid_sub_tlv**(*Binding*, ***kwargs*)

创建 ISIS Capability Srms Preference Sub Tlv 对象

参数 **Binding** (IsisSrBindingTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **ValueType** (*str*) -- 选择标识符（SID 或标签）, 默认值：BIT32, 取值范围：

  BIT20

  BIT32

- **Sid** (*int*) -- 值类型为 20bit 时，指定起始标签; 值类型为 32bit 时，指定起始 SID, 默认值：12000

返回 ISIS Capability Srms Preference Sub TLV 对象

返回类型 (IsisSrSRMSPrefSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Binding} | Create Isis Binding Tlv | Lsp=${LSP} | Ipv4Version=False |
| Create Isis Binding Sr sid Sub Tlv | Binding=${Binding}
```

TesterLibrary.Protocol.isis.**create_isis_capability_sr_algorithm_sub_tlv**(*Capability*, ***kwargs*)

创建 ISIS Capability Sr Algorithm Sub Tlv 对象

参数 **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数 **Algorithm** (*list*) -- 算法值, 默认值：0, 取值范围：int

返回 ISIS Capability Sr Algorithm Sub TLV 对象

返回类型 (IsisSrAlgorithmSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.1.1 |
| Create Isis Capability Sr Algorithm Sub Tlv | Capability=${Capability} |
```

TesterLibrary.Protocol.isis.**create_isis_capability_sr_capability_sub_tlv**(*Session*, *Capability*, ***kwargs*)

创建 ISIS Capability Sr Capability Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值: ['NOSHOW', 'IPv4_CAPABLE'], 取值范围:

  NOSHOW

  IPv4_CAPABLE

  IPv6_CAPABLE

- **ValueType** (*str*) -- 选择标识符（SID 或标签）, 默认值：BIT32, 取值范围:

  BIT20

  BIT32

返回 ISIS Capability Sr Capability Sub TLV 对象

返回类型 (IsisSrCapabilitySubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.1.1 |
| Create Isis Capability Sr Capability Sub Tlv | Session={Session} |␣
↪Capability=${Capability} |
```

TesterLibrary.Protocol.isis.**create_isis_capability_sr_fad_sub_tlv**(*Session*, *Capability*, ***kwargs*)

创建 ISIS Capability Sr Fad Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **FlexAlgo** (*int*) -- 灵活算法 ID, 默认值：128, 取值范围：128-255
- **MetricType** (*str*) -- 指定算路使用的度量类型, 默认值：IGP_METRIC, 取值范围:

  IGP_METRIC

  MIN_LINK_DELAY

  TE_METRIC

- **CalType** (*int*) -- 指定特定 IGP 算法的计算类型, 默认值：0, 取值范围：0-255
- **Priority** (*int*) -- 指定该 Sub TLV 的优先级, 默认值：0
- **FlexAlgoSubTlv** (*list*) -- 选择灵活算法路径计算要遵循的约束条件, 默认值：['UNKNOWN'], 取值范围:

  UNKNOWN

  EXCLUDE_ADMIN

INCLUDE__ANY_ADMIN

INCLUDE_ALL_ADMIN

DEFINITION_FLAGS

EXCLUDE_SRLG

- **ExcludeAdmin** (*int*) -- 类型为：number，默认值：0，取值范围：0-4294967295
- **IncludeAnyAdmin** (*int*) -- 类型为：number，默认值：0，取值范围：0-4294967295
- **IncludeAllAdmin** (*int*) -- 类型为：number，默认值：0，取值范围：0-4294967295
- **DefinitionFlags** (*list*) -- 类型为：hex int，默认值：0x80，取值范围：0-FF
- **ExcludeSRLG** (*list*) -- 类型为：hex int，默认值：0x10020000，取值范围：0-4294967295

返回 ISIS Capability Sr Fad Sub TLV 对象

返回类型 (IsisFelxAlgoDefinitionSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.1.1 |
| Create Isis Capability Sr Fad Sub Tlv | Session={Session} | Capability=$
↪{Capability} |
```

TesterLibrary.Protocol.isis.**create_isis_capability_sr_node_msd_sub_tlv**(*Session*, *Capability*, ***kwargs*)

创建 ISIS Capability Sr Node Msd Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表，类型为：object
- **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象，类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位，默认值：UNKNOWN，取值范围：

  UNKNOWN

  MAX_SEG_LELT

  MAX_END_POP

  MAX_T_INSERT

  MAX_T_ENCAPS

  MAX_END_D

- **MaxSegmentLeft** (*int*) -- 在应用与 SID 关联的 SRv6 Endpoint Function 指令之前，指定接收报文的 SRH 中 SL（Segment Left）字段的最大值，默认值：8，取值范围：0-255

- **MaxEndPop** (*int*) -- 指定 SRH 栈的顶端 SRH 中 SID 的最大数量, 默认值: 8, 取值范围：0-255
- **MaxInsert** (*int*) -- 指定执行 T.Insert 行为时可包含 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxEncap** (*int*) -- 指定执行 T.Encap 行为时可包含 SID 的最大数量, 默认值: 8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 默认值：8, 取值范围：0-255

返回 ISIS Capability Sr Node Msd Sub TLV 对象

返回类型 (IsisSrMsdSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.1.1 |
| Create Isis Capability Sr Node Msd Sub Tlv | Capability=${Capability} |
```

TesterLibrary.Protocol.isis.**create_isis_capability_srms_preference_sub_tlv**(*Capability*, ***kwargs*)

创建 ISIS Capability Srms Preference Sub Tlv 对象

参数 **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数 **Preference** (*int*) -- 指定本节点作为 SR Mapping Server 的优先级, 取值范围：0-255, 默认值：0

返回 ISIS Capability Srms Preference Sub TLV 对象

返回类型 (IsisSrSRMSPrefSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.1.1 |
| Create Isis Capability Srms Preference Sub Tlv | Capability=${Capability} |
```

TesterLibrary.Protocol.isis.**create_isis_capability_srv6_capability_sub_tlv**(*Session*, *Capability*, ***kwargs*)

创建 ISIS Capability Srv6 Capability Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数 **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：UNKNOWN, 取值范围：

UNKNOWN

UNUSED0

O_BIT

UNUSED2

UNUSED3

UNUSED4

UNUSED5

UNUSED6

UNUSED7

UNUSED8

UNUSED9

UNUSED10

UNUSED11

UNUSED12

UNUSED13

UNUSED14

UNUSED15

返回 ISIS Capability Srv6 Capability Sub TLV 对象

返回类型 (IsisSrv6CapabilitySubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.1.1 |
| Create Isis Capability Srv6 Capability Sub Tlv | Session=Session |␣
↪Capability=${Capability} |
```

TesterLibrary.Protocol.isis.**create_isis_capability_tlv**(*Session*, *Lsp*, ***kwargs*)

创建 ISIS Capability TLV 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **Option** (*list*) -- 选项, 默认值：['NOSHOW', 'SBIT'], 取值范围：

  NOSHOW

  SBIT

  DBIT

- **RouterId** (*str*) -- 路由器 ID, 默认值："192.0.0.1", 取值范围：有效 IPv4 地址

返回 ISIS Neighbor TLV 对象

返回类型 (IsisCapabilityTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| Create Isis Capability Tlv | Session=${Session} | Lsp=${LSP} | RouterId=1.1.
↪1.1 |
```

TesterLibrary.Protocol.isis.**create_isis_ipv4_tlv**(*Lsp*, ***kwargs*)

创建 ISIS IPv4 TLV 对象

参数 **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **Name** (*str*) -- ISIS IPv4 TLV 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 ISIS IPv4 TLV, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- 路由类型, 类型为：string, 默认值：INTERNAL, 支持参数：

  INTERNAL

  EXTERNAL
- **RouteCount** (*int*) -- 路由数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **MetricType** (*str*) -- 度量类型, 类型为：string, 默认值：INTERNAL, 支持参数：

  INTERNAL

  EXTERNAL
- **WideMetric** (*int*) -- 扩展度量, 类型为：number, 取值范围：0-16777214, 默认值：10
- **UpDownBit** (*bool*) -- Up/Down 位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **StartIpv4Prefix** (*str*) -- 起始 IPv4 路由前缀, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1
- **PrefixLength** (*int*) -- 前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **NarrowMetric** (*int*) -- 默认度量, 类型为：number, 取值范围：0-63, 默认值：10

返回 ISIS IPv4 TLV 对象

返回类型 (IsisIpv4TlvConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| Create Isis Ipv4 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
```

TesterLibrary.Protocol.isis.**create_isis_ipv6_tlv**(*Lsp*, ***kwargs*)

创建 ISIS IPv6 TLV 对象

参数 **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **Name** (*str*) -- ISIS IPv6 TLV 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 ISIS IPv6 TLV, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- 路由类型, 类型为：string, 默认值：INTERNAL, 支持参数：

  INTERNAL

  EXTERNAL
- **RouteCount** (*int*) -- 路由数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **MetricType** (*str*) -- 度量类型, 类型为：string, 默认值：INTERNAL, 支持参数：

  INTERNAL

  EXTERNAL
- **WideMetric** (*int*) -- 扩展度量, 类型为：number, 取值范围：0-16777214, 默认值：10
- **UpDownBit** (*bool*) -- Up/Down 位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **StartIpv6Prefix** (*str*) -- 起始 IPv4 路由前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- 前缀长度, 类型为：number, 取值范围：1-32, 默认值：24

返回 ISIS IPv6 TLV 对象

返回类型 (IsisIpv6TlvConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| Create Isis Ipv6 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
```

TesterLibrary.Protocol.isis.**create_isis_lsp**(*Session*, ***kwargs*)

创建 ISIS LSP 对象

参数 **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- ISIS LSP 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 ISIS LSP, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SystemId** (*str*) -- 系统 ID, 类型为：string, 取值范围：MAC 地址, 默认值：00:00:00:00:00:01
- **Level** (*str*) -- 区域类型, 类型为：string, 默认值：L2, 支持版本：

  L1

  L2
- **PseudonodeId** (*int*) -- 伪节点 ID, 类型为：number, 取值范围：1-100, 默认值：0
- **TeRouterId** (*str*) -- TE 路由器 ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1
- **TeRouterIdIpv6** (*str*) -- IPv6 TE 路由器 ID, 类型为：string, 取值范围：IPv6 地址, 默认值：3000::1
- **SequenceNumber** (*int*) -- 序列号, 类型为：number, 取值范围：1-300, 默认值：10
- **RemainingLifeTime** (*int*) -- 剩余生存时间, 类型为：number, 取值范围：1-100, 默认值：3
- **Checksum** (*int*) -- 使能正确校验和, 类型为：number, 取值范围：1-20, 默认值：2
- **AttachedBit** (*int*) -- 区域关联位, 类型为：number, 取值范围：1-65535, 默认值：900
- **OverloadBit** (*int*) -- 过载位, 类型为：number, 取值范围：1-100, 默认值：5

返回 ISIS LSP 对象

返回类型 (IsisLspConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
```

TesterLibrary.Protocol.isis.**create_isis_neighbor_custom_sub_tlv**(*SubTlv*, ***kwargs*)

创建 Isis Neighbor Custom Sub Tlv 对象

参数 **SubTlv** (IsisSrLinkMsdSubTlv) -- ISIS Neighbor Sr Link Msd Sid Sub TLV 对象, 类型为：object

关键字参数

- **SubType** (*int*) -- 该 Sub-TLV 的 Type 字段值, 默认值：0, 取值范围：0-255
- **SubValue** (*int*) -- 该 Sub-TLV 的 Value 字段值, 取值范围：十六进制值。默认值：08

返回 Isis Neighbor Sr Link Msd Sid Sub Tlv 对象

返回类型 (IsisSrLinkMsdSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Msd} | Create Isis Neighbor Sr Link Msd Sid Sub Tlv | Neighbor=${Neighbor}
↪ |
| Isis Neighbor Custom Sub Tlv | SubTlv=${Msd} |
```

TesterLibrary.Protocol.isis.**create_isis_neighbor_sr_adj_sid_sub_tlv**(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Sr Adj Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为: object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为: object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值: ['NOSHOW', 'VALUE', 'LOCAL'], 取值范围:

  NOSHOW

  ADDRESS

  BACKUP

  VALUE

  LOCAL

  SET

  PERSISTENT

- **Sid** (*int*) -- Flags 中包含 L.Local 和 V.Value 时，指定标签值; Flags 中不包含 Value/Index 时，指定 SID/Label 范围内的标签偏移值, 默认值: 0, 取值范围: 0-4294967295
- **Weight** (*int*) -- 指定 Adj-SID 权重，用于负载分担, 默认值: 0, 取值范围: 0-255

返回 Isis Neighbor Sr Adj Sid Sub Tlv 对象

返回类型 (IsisSrAdjSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Sr Adj Sid Sub Tlv | Neighbor=${Neighbor} |
```

TesterLibrary.Protocol.isis.**create_isis_neighbor_sr_lan_adj_sid_sub_tlv**(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Sr Lan Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为: object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为: object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值: ['NOSHOW', 'VALUE', 'LOCAL'], 取值范围:

  NOSHOW

  ADDRESS

  BACKUP

  VALUE

  LOCAL

  SET

  PERSISTENT

- **Sid** (*int*) -- Flags 中包含 L.Local 和 V.Value 时，指定标签值; Flags 中不包含 Value/Index 时，指定 SID/Label 范围内的标签偏移值, 默认值: 0, 取值范围: 0-4294967295
- **Weight** (*int*) -- 指定 Adj-SID 权重，用于负载分担, 默认值: 0, 取值范围: 0-255
- **SystemId** (*str*) -- 指定 LAN 上邻居的系统 ID, 默认值: "00:00:00:00:00:01"

返回 Isis Neighbor Sr Adj Sid Sub Tlv 对象

返回类型 (IsisSrAdjSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Sr Lan Adj Sid Sub Tlv | Neighbor=${Neighbor} |
```

TesterLibrary.Protocol.isis.**create_isis_neighbor_sr_link_msd_sub_tlv**(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Sr Link Msd Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为: object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为: object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位，默认值：UNKNOWN, 取值范围：

  UNKNOWN

  MAX_SEG_LELT

  MAX_END_POP

  MAX_T_INSERT

  MAX_T_ENCAPS

  MAX_END_D

- **MaxSegmentLeft** (*int*) -- 在应用与 SID 关联的 SRv6 Endpoint Function 指令之前，指定接收报文的 SRH 中 SL（Segment Left）字段的最大值, 默认值：8, 取值范围：0-255
- **MaxEndPop** (*int*) -- 指定 SRH 栈的顶端 SRH 中 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxInsert** (*int*) -- 指定执行 T.Insert 行为时可包含 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxEncap** (*int*) -- 指定执行 T.Encap 行为时可包含 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 默认值：8, 取值范围：0-255

返回 Isis Neighbor Sr Link Msd Sid Sub Tlv 对象

返回类型 (IsisSrLinkMsdSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Sr Link Msd Sid Sub Tlv | Neighbor=${Neighbor} |
```

TesterLibrary.Protocol.isis.**create_isis_neighbor_srv6_endx_sid_sub_tlv**(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Srv6 EndX Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位，默认值：UNKNOWN, 取值范围：

  UNKNOWN

  BACKUP

  SET

  PERSISTENT

UNUSED3

UNUSED4

UNUSED5

UNUSED6

UNUSED7

- **Algorithm** (*int*) -- 指定 SID 关联的算法, 默认值：0, 取值范围：0-255
- **Weight** (*int*) -- 指定 End.X SID 的权重，用于负载分担, 默认值：100, 取值范围：0-255
- **EndpointFunc** (*str*) -- 端点行为, 默认值：END_NO, 取值范围：

  END_NO

  END_PSP

  END_USP

  END_PSP_USP

  END_X_NO

  END_X_PSP

  END_X_USP

  END_X_PSP_USP

  END_T_NO

  END_T_PSP

  END_T_USP

  END_T_PSPS_USP

  END_B6

  END_B6_ENCAPS

  END_BM

  END_DX6

  END_DX4

  EDN_DT6

  END_DT4

  END_DT46

  END_DX2

  END_DX2V

  END_DX2U

  END_DX2M

  END_S

  END_B6_RED

  END_B6_ENCAPS_RED

  END_WITH_USD

  END_PSP_USD

  END_USP_USD

END_PSP_USP_USD

END_X_USD

END_X_PSP_USD

END_X_USP_USD

END_X_PSP_USP_USD

END_T_USD

END_T_PSP_USD

END_T_USP_USD

END_T_PSP_USP_USD

- **EnableCustom** (*bool*) -- 使能自定义端点行为, 默认值：False
- **CustomFunc** (*int*) -- 自定义端点行为, 默认值：0, 取值范围：0-65535
- **SID** (*str*) -- 指定通告的 SRv6 SID, 默认值：”::1”, 取值范围：有效 IPv6 地址

返回 Isis Neighbor Srv6 EndX Sid Sub Tlv 对象

返回类型 (IsisSrv6EndXSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Srv6 EndX Sid Sub Tlv | Neighbor=${Neighbor} |
```

TesterLibrary.Protocol.isis.**create_isis_neighbor_srv6_lan_endx_sid_sub_tlv**(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Srv6 Lan EndX Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为：object

关键字参数

- **SystemIdLan** (*str*) -- LAN 系统标识, 默认值：”00:10:96:00:00:01”
- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：UNKNOWN, 取值范围：

UNKNOWN

BACKUP

SET

PERSISTENT

UNUSED3

UNUSED4

UNUSED5

UNUSED6

UNUSED7

- **Algorithm** (*int*) -- 指定 SID 关联的算法, 默认值：0, 取值范围：0-255
- **Weight** (*int*) -- 指定 End.X SID 的权重，用于负载分担, 默认值：100, 取值范围：0-255
- **EndpointFunc** (*list*) -- 端点行为, 默认值：END_NO, 取值范围：

  END_NO

  END_PSP

  END_USP

  END_PSP_USP

  END_X_NO

  END_X_PSP

  END_X_USP

  END_X_PSP_USP

  END_T_N

  END_T_PSP

  END_T_USP

  END_T_PSPS_USP

  END_B6

  END_B6_ENCAPS

  END_BM

  END_DX6

  END_DX4

  EDN_DT6

  END_DT4

  END_DT46

  END_DX2

  END_DX2V

  END_DX2U

  END_DX2M

  END_S

  END_B6_RED

  END_B6_ENCAPS_RED

  END_WITH_USD

  END_PSP_USD

  END_USP_USD

  END_PSP_USP_USD

  END_X_USD

  END_X_PSP_USD

END_X_USP_USD

END_X_PSP_USP_USD

END_T_USD

END_T_PSP_USD

END_T_USP_USD

END_T_PSP_USP_USD

- **EnableCustom** (*bool*) -- 使能自定义端点行为, 默认值：False
- **CustomFunc** (*int*) -- 自定义端点行为, 默认值：0
- **SID** (*str*) -- 指定通告的 SRv6 SID, 默认值：”::1”, 取值范围：有效 IPv6 地址

返回 Isis Neighbor Srv6 Lan EndX Sid Sub Tlv 对象

返回类型 (IsisSrv6LanEndXSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Srv6 Lan EndX Sid Sub Tlv | Neighbor=${Neighbor} |
```

TesterLibrary.Protocol.isis.**create_isis_neighbor_te_config**(*Neighbor*, ***kwargs*)

创建 ISIS 邻居 TLV 的 Te Config 对象

参数 **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为：object

关键字参数

- **EnableInterfaceIp** (*bool*) -- 是否包含本地 IPv4 地址, 默认值：False
- **InterfaceIp** (*str*) -- 本地 IPv4 地址, 取值范围：有效的 ip 地址, 默认值：’0.0.0.0’
- **EnableNeighborIp** (*bool*) -- 是否包含邻居 IPv4 地址, 默认值：False
- **NeighborIp** (*int*) -- 邻居 IPv4 地址, 取值范围：有效的 ip 地址, 默认值：10
- **EnableInterfaceIpv6** (*bool*) -- 是否包含本地 IPv6 地址, 默认值：False
- **InterfaceIpv6** (*str*) -- 本地 IPv6 地址, 取值范围：有效的 ipv6 地址, 默认值：’2000::1’
- **EnableNeighborIpv6** (*bool*) -- 是否包含邻居 IPv6 地址, 默认值：False
- **NeighborIpv6** (*str*) -- 邻居 IPv6 地址, 取值范围：有效的 ipv6 地址, 默认值：’2000::1’
- **EnableTeGroup** (*bool*) -- 是否包含 TE 组, 默认值：False
- **TeGroup** (*int*) -- TE 组, 取值范围：0-4294967295, 默认值：1
- **EnableMaxBandwidth** (*bool*) -- 是否包含最大带宽值, 默认值：False
- **MaximunLink** (*int*) -- 最大带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：1000
- **EnableResBandwidth** (*bool*) -- 是否包含预留带宽值, 默认值：False

- **MaximumReservableLink** (*int*) -- 最大预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 1000
- **EnableUnresBandwidth** (*bool*) -- 是否包含未预留带宽优先级, 默认值: False
- **UnreservedBandwidth0** (*int*) -- 优先级 0 的未预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 0
- **UnreservedBandwidth1** (*int*) -- 优先级 1 的未预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 0
- **UnreservedBandwidth2** (*int*) -- 优先级 2 的未预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 0
- **UnreservedBandwidth3** (*int*) -- 优先级 3 的未预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 0
- **UnreservedBandwidth4** (*int*) -- 优先级 4 的未预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 0
- **UnreservedBandwidth5** (*int*) -- 优先级 5 的未预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 0
- **UnreservedBandwidth6** (*int*) -- 优先级 6 的未预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 0
- **UnreservedBandwidth7** (*int*) -- 优先级 7 的未预留带宽值 (字节/秒), 取值范围: 0-4294967295, 默认值: 0

返回 ISIS Neighbor TLV Te Config 对象

返回类型 (IsisTEConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Te Config | Neighbor=${Neighbor} |
```

TesterLibrary.Protocol.isis.**create_isis_neighbor_tlv**(*Lsp*, ***kwargs*)

创建 ISIS 邻居 TLV 对象

参数 **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为: object

关键字参数

- **SystemId** (*str*) -- 邻居系统 ID, 取值范围: 有效的 MAC 地址, 默认值: "00:00:00:00:00:01"
- **PseudonodeSystemId** (*int*) -- 伪节点 ID, 取值范围: 0-255, 默认值: 0
- **NarrowMetric** (*int*) -- 默认度量, 取值范围: 0-63, 默认值: 1
- **WideMetric** (*int*) -- 扩展度量, 取值范围: 0-16777214, 默认值: 10

返回 ISIS Neighbor TLV 对象

返回类型 (IsisNeighborConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
```

TesterLibrary.Protocol.isis.**create_isis_sr_binding_tlv**(*Session*, *Lsp*, ***kwargs*)

创建 ISIS Binding TLV 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **EnableMt** (*bool*) -- 使能多拓扑, 默认值：False
- **MtId** (*str*) -- 多拓扑 ID, 默认值：STANDARD, 取值范围：

  STANDARD

  IPV6_ROUTING
- **Flags** (*list*) -- 标签, 默认值：['NOSHOW'], 取值范围：

  NOSHOW

  FBIT

  MBIT

  SBIT

  DBIT

  ABIT
- **Weight** (*int*) -- 权重, 默认值：0, 取值范围：0-255
- **Range** (*int*) -- 范围, 默认值：1, 取值范围：0-65535
- **Ipv4Version** (*bool*) -- 默认值：True
- **Ipv4Prefix** (*str*) -- IPv4 前缀, 默认值："192.0.0.1"
- **Ipv4PrefixLength** (*int*) -- IPv4 前缀长度, 默认值：1, 取值范围：1-32
- **Ipv6Prefix** (*str*) -- IPv6 前缀, 默认值："2000::1
- **Ipv6PrefixLength** (*int*) -- IPv6 前缀长度, 默认值：64, 取值范围：1-128

返回 ISIS Neighbor TLV 对象

返回类型 (IsisSrBindingTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| Create Isis Binding Tlv | Session={Session} | Lsp=${LSP} |␣
↪Ipv4Version=False |
```

TesterLibrary.Protocol.isis.**create_isis_srv6_end_sid_sub_tlv**(*Session*, *Location*, ***kwargs*)

创建 ISIS Capability Srms Preference Sub Tlv 对象

参数 **Location** (IsisSrv6LocatorTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：['UNKNOWN'], 取值范围：

  UNKNOWN

  UNUSED0

  UNUSED1

  UNUSED2

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

  UNUSED7

- **EndpointFunc** (*str*) -- 端点行为, 默认值：END_NO, 取值范围：

  END_NO

  END_PSP

  END_USP

  END_PSP_USP

  END_X_NO

  END_X_PSP

  END_X_USP

  END_X_PSP_USP

  END_T_NO

  END_T_PSP

  END_T_USP

  END_T_PSPS_USP

  END_B6

  END_B6_ENCAPS

  END_BM

  END_DX6

  END_DX4

EDN_DT6
END_DT4
END_DT46
END_DX2
END_DX2V
END_DX2U
END_DX2M
END_S
END_B6_RED
END_B6_ENCAPS_RED
END_WITH_USD
END_PSP_USD
END_USP_USD
END_PSP_USP_USD
END_X_USD
END_X_PSP_USD
END_X_USP_USD
END_X_PSP_USP_USD
END_T_USD
END_T_PSP_USD
END_T_USP_USD
END_T_PSP_USP_USD

- **EnableCustom** (*bool*) -- 使能自定义端点行为, 默认值：False
- **CustomFunc** (*int*) -- 自定义端点行为, 默认值：0, 取值范围：0-65535
- **SID** (*str*) -- 指定通告的 SRv6 SID, 默认值：”::1”, 取值范围：IPv6 地址

返回 ISIS Capability Srms Preference Sub TLV 对象

返回类型 (IsisSrv6EndSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Location} | Create Isis Srv6 Location Tlv | Lsp=${LSP} | Algorithm=1 |
| Create Isis Srv6 End Sid Sub Tlv | Session={Session} | Location=${Location}
↪|
```

TesterLibrary.Protocol.isis.**create_isis_srv6_location_tlv**(*Session*, *Lsp*, ***kwargs*)

创建 ISIS Binding TLV 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **MtId** (*str*) -- 多拓扑 ID, 默认值：STANDARD, 取值范围：
- **Metric** (*int*) -- 指定度量值, 默认值：0, 取值范围：0-4294967295
- **Flags** (*list*) -- 标签, 默认值：['UNKNOWN'], 取值范围：

  UNKNOWN

  D_BIT

  A_BIT

  UNUSED2

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

  UNUSED7
- **Algorithm** (*int*) -- Locator 关联算法, 类型为：number, 取值范围：0-255, 默认值：0
- **NumLocator** (*int*) -- Locator 数量, 取值范围：0-4294967295, 默认值：1
- **LocatorSize** (*int*) -- 定位器大小, 取值范围：1-128, 默认值：64
- **Locator** (*str*) -- 定位器, 默认值："aaaa:1:1:1::", 取值范围：IPv6 地址
- **LocatorStep** (*int*) -- 定位器步长, 默认值：1, 取值范围：0-65535

返回 ISIS Srv6 Location TLV 对象

返回类型 (IsisSrv6LocatorTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| Create Isis Srv6 Location Tlv | Session={Session} | Lsp=${LSP} |␣
↪Algorithm=1 |
```

TesterLibrary.Protocol.isis.**create_isis_tlv_bier_Mpls_sub_sub_tlv**(*Bier*, ***kwargs*)

创建 ISIS Tlv Bier Mpls Sub Sub Tlv 对象

参数 **Bier** (IsisBierSubTlv) -- ISIS Bierv6 Sub Tlv 对象, 类型为：object

关键字参数

- **MaxSI** (*int*) -- 指定最大 Set ID, 默认值：1, 取值范围：0-255
- **LabelorBiftId** (*int*) -- 指定标签范围中的起始标签值, 默认值：100, 取值范围：0-4294967295
- **BSLength** (*int*) -- 指定本地比特串的长度, 默认值：1, 取值范围：0-15

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisBierMplsSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Ipv6} | Create Isis Ipv6 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Bier Sub Tlv | Tlv=${Ipv6} |
| Create Isis Tlv Bier Mpls Sub Sub Tlv | Bier=${Ipv6} |
```

TesterLibrary.Protocol.isis.**create_isis_tlv_bier_sub_tlv**(*Tlv*, ***kwargs*)

创建 ISIS Tlv Bier Sub Tlv 对象

参数 **Tlv** (IsisIpv4TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **BFRId** (*int*) -- 指定 BFR（Bit Forwarding Router，比特转发路由器）ID, 取值范围：1-65535, 默认值：1
- **SubDomainId** (*int*) -- 指定 BIER 子域 ID, 取值范围：0-255, 默认值：1
- **IgpAlgorithm** (*int*) -- IGP 算法, 取值范围：0-255, 默认值：0
- **BierAlgorithm** (*int*) -- BIER 算法, 取值范围：0-255, 默认值：0

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisBierSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Ipv4} | Create Isis Ipv4 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Bier Sub Tlv | Tlv=${Ipv4} |
```

TesterLibrary.Protocol.isis.**create_isis_tlv_bierv6_bift_id_sub_tlv**(*Bier*, ***kwargs*)

创建 ISIS Tlv Bierv6 Bift Id Sub Tlv 对象

参数 **Bier** (IsisIpv6TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Type** (*int*) -- 指定 Type 字段值, 默认值：7, 取值范围：0-255
- **MPRA** (*str*) -- 指定 MPRA 地址, 默认值：'::1', 取值范围：有效 IPv6 地址

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisBierBiftIdSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Ipv6} | Create Isis Ipv6 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Bierv6 Bift Id Sub Tlv | Tlv=${Ipv6} |
```

TesterLibrary.Protocol.isis.**create_isis_tlv_bierv6_sub_sub_tlv**(*Bier*, ***kwargs*)

创建 ISIS Tlv Bierv6 Sub Sub Tlv 对象

参数 **Bier** (IsisBierSubTlv) -- ISIS Bierv6 Sub Tlv 对象, 类型为：object

关键字参数

- **MaxSI** (*int*) -- 指定最大 Set ID, 默认值：1, 取值范围：0-255
- **LabelorBiftId** (*int*) -- 指定标签范围中的起始标签值, 默认值：100, 取值范围：0-4294967295
- **BSLength** (*int*) -- 指定本地比特串的长度, 默认值：1, 取值范围：0-15

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisBierMplsSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Ipv6} | Create Isis Ipv6 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Bier Sub Tlv | Tlv=${Ipv6} |
| Create Isis Tlv Bierv6 Sub Sub Tlv | Bier=${Ipv6} |
```

TesterLibrary.Protocol.isis.**create_isis_tlv_end_bier_sub_tlv**(*Bier*, ***kwargs*)

创建 ISIS Tlv End Bier Sub Tlv 对象

参数 **Bier** (IsisIpv6TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Type** (*int*) -- 指定 Type 字段值。取值范围：0-255, 默认值：3
- **EndBierAddr** (*str*) -- 指定 End.BIER SID, 默认值：”::1”, 取值范围：有效 IPv6 地址

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisEndBierSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Ipv6} | Create Isis Ipv6 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
| Create Isis Tlv End Bier Sub Tlv | Tlv=${Ipv6} |
```

TesterLibrary.Protocol.isis.**create_isis_tlv_flex_algorithm_prefix_metric_sub_tlv**(*Tlv*, ***kwarg*

创建 ISIS Tlv Flex Algorithm Prefix Metric Sid Sub Tlv 对象

参数 **Tlv** (IsisIpv4TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Algorithm** (*int*) -- Locator 关联算法, 类型为：number, 取值范围：128-255, 默认值：128
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisFlexAlgoPrefixMetricSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Ipv4} | Create Isis Ipv4 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Flex Algorithm Prefix Metric Sub Tlv | Tlv=${Ipv4} |
```

TesterLibrary.Protocol.isis.**create_isis_tlv_prefix_sid_sub_tlv**(*Session*, *Tlv*, ***kwargs*)

创建 ISIS Tlv Prefix Sid Sub Tlv 对象

参数 **Tlv** (IsisIpv4TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值: ['NOSHOW', 'NOPHP'], 取值范围：

  NOSHOW

  ADVERTISEMENT

  NODESID

  NOPHP

  EXPLICIT

  VALUE

  LOCAL

- **Sid** (*int*) -- SID/Label, 默认值：0, 取值范围：0-4294967295
- **Algorithm** (*int*) -- 指定计算到其他节点/前缀的可达信息的算法, 指定 SID 关联的算法, 默认值：0, 取值范围：0-255
- **PrefixSidStep** (*int*) -- 默认值：1

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisSrPrefixSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${Ipv4} | Create Isis Ipv4 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Prefix Sid Sub Tlv | Tlv=${Ipv4} |
```

TesterLibrary.Protocol.isis.**edit_isis**(*Session*, ***kwargs*)

编辑 ISIS 协议会话对象参数

参数 **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- ISIS 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ISIS 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPV4, 支持版本：

  IPV4

  IPV6

IPV4IPV6

- **Level** (*str*) -- 区域类型, 类型为：string, 默认值：L2, 支持版本：

  L1

  L2

  L1L2

- **NetworkType** (*str*) -- 网络类型, 类型为：string, 默认值：BROADCAST, 支持参数：

  BROADCAST

  P2P

- **SystemId** (*str*) -- 系统 ID, 类型为：string, 取值范围：MAC 地址, 默认值：00:00:00:00:00:01

- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-127, 默认值：0

- **AuthMethod** (*str*) -- 认证方式, 类型为：string, 默认值：NONE, 支持参数：

  NONE

  SIMPLE

  MD5

- **Password** (*str*) -- 4 字节自治域跳变, 类型为：string, 默认值：Xinertel

- **CircuitId** (*int*) -- 电路 ID, 类型为：number, 取值范围：0-255, 默认值：1

- **Area1** (*str*) -- 区域 ID 1, 类型为：hex number, 取值范围：0x0-0xff, 默认值：0x10

- **Area2** (*str*) -- 区域 ID 2, 类型为：hex number, 取值范围：0x0-0xff, 默认值：空

- **Area3** (*str*) -- 区域 ID 3, 类型为：hex number, 取值范围：0x0-0xff, 默认值：空

- **MetricMode** (*str*) -- 度量模式, 类型为：string, 默认值：NARROWWIDE, 支持参数：

  NARROW

  WIDE

  NARROWWIDE

- **TeRouterId** (*str*) -- TE 路由器 ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1

- **TeRouterIdIpv6** (*str*) -- IPv6 TE 路由器 ID, 类型为：string, 取值范围：IPv6 地址, 默认值：3000::1

- **HelloInterval** (*int*) -- Hello PDU 发送间隔 (秒), 类型为：number, 取值范围：1-300, 默认值：10

- **HelloMultiplier** (*int*) -- Hello 时间间隔倍数, 类型为：number, 取值范围：1-100, 默认值：3

- **PsnInterval** (*int*) -- PSNP 发送间隔 (秒), 类型为：number, 取值范围：1-20, 默认值：2

- **LspRefreshTime** (*int*) -- LSP 刷新时间 (秒), 类型为：number, 取值范围：1-65535, 默认值：900

- **RetransInterval** (*int*) -- LSP 重传间隔 (秒), 类型为：number, 取值范围：1-100, 默认值：5

- **HelloPadding** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LspSize** (*int*) -- LSP 大小, 类型为：number, 取值范围：100-1492, 默认值：1492
- **ValidateIpAddr** (*bool*) -- 使能接口校验, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableBFD** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MtParams** (*int*) -- 多拓扑参数数量, 类型为：number, 取值范围：0-2, 默认值：0
- **PerPduAuthentication** (*int*) -- Per PDU 认证数量, 类型为：number, 取值范围：0-4, 默认值：0
- **ReportLabel** (*bool*) -- 使能 ReportLabel, 类型为：bool, 默认值：True
- **LearnRoute** (*bool*) -- 使能 LearnRoute, 类型为：bool, 默认值：True
- **RecordLspNextSequenceNum** (*bool*) -- 使能 Record Lsp Next Sequence Number, 类型为：bool, 默认值：True
- **L1NarrowMetric** (*int*) -- L1 Narrow Metric, 类型为：number, 取值范围：0-63, 默认值：1
- **L1WideMetric** (*int*) -- L1 Wild Metric, 类型为：number, 取值范围：0-16777214, 默认值：1
- **L2NarrowMetric** (*int*) -- L2 Narrow Metric, 类型为：number, 取值范围：0-63, 默认值：1
- **L2WideMetric** (*int*) -- L2 Wide Metric, 类型为：number, 取值范围：0-16777214, 默认值：1

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${MtId} | Create List | IPV4 | IPV6 |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True | MtParams=1 |
```

TesterLibrary.Protocol.isis.**edit_isis_mt_params**(*Session*, *Index=0*, *MtId=None*, *MtFlags=None*)

编辑 ISIS 协议会话 MT 参数

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Index** (*int*) -- ISIS 协议会话 MT 参数对象序号，类型为：number, 取值范围：0-1, 默认值：0

- **MtId** (*str*) -- 多拓扑 ID, 类型为：string, 默认值：IPV4, 支持参数：

  IPV4

  IPV6

- **MtFlags** (*list*) -- 多拓扑 Flags, 类型为：list, 默认值：NOSHOW, 支持参数：

  NOSHOW

  ABIT

  OBIT

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True | MtParams=1 |
| Edit Isis Mt Params | Session=${Session} | MtId=IPV6 | MtFlags=${{MtFlags} |
```

TesterLibrary.Protocol.isis.**edit_isis_per_pdu**(*Session*, *Index=0*, ***kwargs*)

编辑 ISIS 协议会话 Per Pdu Authentication 参数

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Index** (*int*) -- ISIS 协议会话 Per Pdu Authentication 参数对象序号，类型为：number, 取值范围：0-1, 默认值：0

关键字参数

- **PdusType** (*str*) -- PDU 类型, 类型为：string, 默认值：L1_HELLO, 支持参数：

  L1_HELLO

  L2_HELLO

  L1_AREA_PDUS

  L2_DOMAIN_PDUS

- **AuthMethod** (*str*) -- 认证类型, 类型为：string, 默认值：NONE, 支持参数：

  NONEReportLabel

  SIMPLE

  MD5

- **Password** (*str*) -- 认证密码, 类型为：string, 默认值：Xinertel.

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True |␣
↪PerPduAuthentication=1 |
| Edit Isis Per Pdu | Session=${Session} | PdusType=L2_HELLO |␣
↪AuthMethod=SIMPLE | Password=Test |
```

TesterLibrary.Protocol.isis.**edit_isis_port_config**(*Ports*, ***kwargs*)

修改 Isis 端口统计对象

参数 **Ports** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数 **UpdateRoutesTransmitRate** (*int*) -- IS-IS Tx Hello Rate(messages/second) , 取值范围：1-10000, 默认值：1000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Isis Port Config | Ports=${Ports} | UpdateRoutesTransmitRate=100 |
```

TesterLibrary.Protocol.isis.**get_isis_mt_params**(*Session*, *Index=0*)

获取 ISIS 协议会话 MT 参数

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Index** (*int*) -- ISIS 协议会话 MT 参数对象序号，类型为：number, 取值范围：0-1, 默认值：0

返回

eg:

```
{
    'MtId': 'IPV4',
    'MtFlags': '['ABIT', 'OBIT'],
}
```

返回类型 dict

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True | MtParams=2 |
| Edit Isis Mt Params | Session=${Session} | Index=0 | MtId=IPv4 | MtFlags=${
↪{MtFlags} |
| Edit Isis Mt Params | Session=${Session} | Index=1 | MtId=IPv6 | MtFlags=${
↪{MtFlags} |
| Get Isis Mt Params | Session=${Session} | Index=0 |
| Get Isis Mt Params | Session=${Session} | Index=1 |
```

TesterLibrary.Protocol.isis.**get_isis_per_pdu**(*Session*, *Index=0*)

获取 ISIS 协议会话 Per Pdu Authentication 参数

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Index** (*int*) -- ISIS 协议会话 Per Pdu Authentication 参数对象序号，类型为：number, 取值范围：0-4, 默认值：0

返回

eg:

```
{
    'PdusType': 'L1_HELLO',
    'AuthMethod': 'SIMPLE',
    'Password': 'Xinertel',
}
```

返回类型 dict

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True |␣
↪PerPduAuthentication=1 |
| Edit Isis Per Pdu Authentication | Session=${Session} | PdusType=L2_HELLO |␣
↪AuthMethod=SIMPLE | Password=Test |
| Get Isis Per Pdu Authentication | Session=${Session} | Index=0 |
```

TesterLibrary.Protocol.isis.**get_isis_router_from_tlv**(*Configs*)

获取 ISIS TLV 对应的绑定流源或目的端点对象

参数 **Configs** (list(IsisIpv4TlvConfig, *IsisIpv6TlvConfig*)) -- 测试仪表 ISIS TLV 对象列表, 类型为：list

返回 ISIS TLV 对应的绑定流源或目的端点对象列表

返回类型 (list(IsisIpv4Router)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
| ${TLV} | Create Isis Ipv4 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
| Get Isis Router From Tlv | Configs=${TLV} |
```

TesterLibrary.Protocol.isis.**get_isis_session_stats**(*Session*, *StaItems=None*)

获取 Isis Session 统计结果

参数

- **Session** (IsisRouter) -- Isis 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxP2pHello

RxP2pHello

TxLanL1Hello

RxLanL1Hello

TxLanL2Hello

RxLanL2Hello

TxL1Lsp

RxL1Lsp

TxL2Lsp

RxL2Lsp

TxL1Csnp

RxL1Csnp

TxL2Csnp

RxL2Csnp

TxL1Psnp

RxL1Psnp

TxL2Psnp

RxL2Psnp

返回

eg:

```
{
    'TxL1Lsp': 10,
    'RxL1Lsp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=IsisSessionStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Isis Session Stats | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.isis.**get_isis_tlv_stats**(*Session*, *StaItems=None*)

获取 Isis Session 统计结果

参数

- **Session** (IsisRouter) -- Isis 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxPrefixSid

  RxPrefixSid

  TxAdjSid

RxAdjSid

TxLanAdjSid

RxLanAdjSid

TxSidBinding

RxSidBinding

TxSrv6Loc

RxSrv6Loc

TxSrv6EndX

RxSrv6EndX

TxSrv6LanEndX

RxSrv6LanEndX

返回

eg:

```
{
    'TxPrefixSid': 10,
    'RxPrefixSid': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=IsisTlvStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Isis Tlv Stats | Session=${Session} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.isis.**graceful_restart_isis**(*Sessions*)

平滑重启 Isis 协议会话

参数 **Sessions** (list (IsisRouter)) -- ISIS 会话对象列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Graceful Restart Isis | Session=${Session} |
```

TesterLibrary.Protocol.isis.**wait_isis_l1_broadcast_adj_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 ISIS 协议会话 l1_broadcast_adj_state 达到指定状态

参数

- **Sessions** (list (IsisRouter)) -- ISIS 协议会话对象列表, 类型为：list

- **State** (*str*) -- 等待 ISIS 协议会话 l1_broadcast_adj_state 达到的状态, 类型为：string, 默认值：达到 DISOTHER 或 DIS, 支持下列状态：

  NOTSTART

  IDLE

  INIT

  DISOTHER

  DIS

  GR

  GRHELPER

  NA

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Isis L1 Broadcast Adj State | Sessions=${Sessions} | State=RUNNING |␣
↪Interval=2 | TimeOut=120 |
```

TesterLibrary.Protocol.isis.**wait_isis_l2_broadcast_adj_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 ISIS 协议会话 l2_broadcast_adj_state 达到指定状态

参数

- **Sessions** (list (IsisRouter)) -- ISIS 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 ISIS 协议会话 l2_broadcast_adj_state 达到的状态, 类型为：string, 默认值：达到 DISOTHER 或 DIS, 支持下列状态：

  NOTSTART

  IDLE

  INIT

  DISOTHER

  DIS

  GR

  GRHELPER

  NA

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Isis L2 Broadcast Adj State | Sessions=${Sessions} | State=RUNNING |␣
↪Interval=2 | TimeOut=120 |
```

TesterLibrary.Protocol.isis.**wait_isis_state**(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 ISIS 协议会话达到指定状态

参数

- **Sessions** (list (IsisRouter)) -- ISIS 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 ISIS 协议会话达到的状态, 类型为：string, 默认值：达到 UP, 支持下列状态：

  NOTSTART

  IDLE

  INIT

  UP

  GR

  GRHELPER

  DISABLE
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Isis State | Sessions=${Sessions} | State=GR | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.isis.**wait_isis_three_way_p2p_adj_state**(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 ISIS 协议会话 three_way_p2p_adj 达到指定状态

参数

- **Sessions** (list (IsisRouter)) -- ISIS 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 ISIS 协议会话 three_way_p2p_adj 达到的状态, 类型为：string, 默认值：达到 UP, 支持下列状态：

  UP

  INIT

  DOWN

  NOTSTART

  NA

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Isis Three Way P2p Adj State | Sessions=${Sessions} | State=INIT |␣
↪Interval=2 | TimeOut=120 |
```

TesterLibrary.Protocol.isis.**withdraw_isis**(*Lsps*)

通告 Isis 协议会话 lsp

参数 **Lsps** (IsisLspConfig) -- ISIS LSP 对象, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Isis | Lsp=${Lsp} |
```

TesterLibrary.Protocol.l2tp module

TesterLibrary.Protocol.l2tp.**abort_l2tp**(*Sessions*)

中断 L2tp 协议会话

参数 **Sessions** (L2tp) -- L2tp 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Abort L2tp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.l2tp.**connect_l2tp**(*Sessions*)

建立 L2tp 协议会话

参数 **Sessions** (L2tp) -- L2tp 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Connect L2tp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.l2tp.**create_l2tp**(*Port*, ***kwargs*)

创建 L2tp 协议会话对象

参数 Port (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- L2tp 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 L2tp 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EmulationMode** (*str*) -- L2TP 角色, 默认值：LAC, 取值范围：

  LAC

  LNS
- **TunnelCountPerNode** (*int*) -- 每 LAC/LNS 隧道数, 取值范围：1-32768, 默认值：1
- **SessionCountPerTunnel** (*int*) -- 每隧道会话数, 取值范围：0-65535, 默认值：0
- **TunnelStartingId** (*int*) -- 隧道起始 ID, 取值范围：1-65535, 默认值：1
- **SessionStartingId** (*int*) -- 会话起始 ID, 取值范围：1-65535, 默认值：1
- **UdpSourcePort** (*int*) -- UDP 源端口, 取值范围：1-65535, 默认值：1701
- **UdpChecksumEnabled** (*bool*) -- 使能 UDP 校验和, 默认值：True
- **RetryTunnelCreationEnabled** (*bool*) -- 使能隧道重试, 默认值：False
- **TunnelCreationTimeout** (*int*) -- 隧道建立超时 (secs), 取值范围：1-65535, 默认值：5
- **MaxTunnelCreationTimes** (*int*) -- 隧道建立最大尝试次数, 取值范围：1-65535, 默认值：5
- **HostName** (*str*) -- 主机名, 取值范围：string length in [1,255], 默认值：xinertel
- **EnableAuthentication** (*bool*) -- 使能认证, 默认值：True
- **IncomingTunnelPassword** (*str*) -- Incoming 隧道密码, 取值范围：string length in [1,255], 默认值：xinertel
- **OutgoingTunnelPassword** (*str*) -- Outgoing 隧道密码, 取值范围：string length in [1,255], 默认值：xinertel
- **HelloEnabled** (*bool*) -- 使能 Hello, 默认值：False
- **HelloInterval** (*int*) -- Hello 间隔 (secs), 取值范围：1-255, 默认值：60
- **TxBitRate** (*int*) -- 发送 bps 速率 (bits/sec), 取值范围：1-65535, 默认值：56000
- **BearerCapabilities** (*str*) -- 负载能力, 默认值：ANALOG, 取值范围：

  DIGITAL

  ANALOG

  BOTH

- **BearerType** (*str*) -- 负载类型, 默认值：ANALOG, 取值范围：

  DIGITAL

  ANALOG
- **FrameCapabilities** (*str*) -- 帧能力, 默认值：SYNC, 取值范围：

  SYNC

  ASYNC

  BOTH
- **FrameType** (*str*) -- 帧类型, 默认值：SYNC, 取值范围：

  SYNC

  ASYNC
- **CallingNumberEnabled** (*bool*) -- 使能 Calling Number, 默认值：False
- **CallingNumber** (*str*) -- 隧道的 Calling Number, 默认值：xinertel
- **RxWindowSize** (*int*) -- 接收窗口大小, 取值范围：1-65535, 默认值：4
- **UseGatewayAsRemoteIp** (*bool*) -- 使用网关作为远端地址, 默认值：True
- **RemoteIpv4Address** (*str*) -- 远端 IPv4 地址, 取值范围：IPv4 地址, 默认值：2.1.1.1
- **RemoteIpv4AddressStep** (*str*) -- 远端 IPv4 地址跳变, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **RemoteIpv6Address** (*str*) -- 远端 IPv6 地址, 取值范围：IPv6 地址, 默认值：2000::1
- **RemoteIpv6AddressStep** (*str*) -- 远端 IPv6 地址跳变, 取值范围：IPv6 地址, 默认值：::1
- **LcpProxyMode** (*str*) -- LCP 代理模式, 默认值：NONE, 取值范围：

  NONE

  LCP

  LCP_AUTH
- **ForceLcpRenegotiation** (*bool*) -- 强制 LCP 重协商, 默认值：False
- **Ipv4TosValue** (*hex int*) -- IPv4 TOS 值, 取值范围：1-65535, 默认值：0xc0
- **Ipv6TrafficClassValue** (*hex int*) -- IPv6 Traffic Class Value, 取值范围：1-65535, 默认值：0x0
- **HideFramingCapabilities** (*bool*) -- 默认值：False
- **HideBearerCapabilities** (*bool*) -- 默认值：False
- **HideAssignedTunnelId** (*bool*) -- 默认值：False
- **HideChallenge** (*bool*) -- 默认值：False
- **HideChallengeResponse** (*bool*) -- 默认值：False
- **HideAssignedSessionId** (*bool*) -- 默认值：False
- **HideCallSerialNumber** (*bool*) -- 默认值：False
- **HideFramingType** (*bool*) -- 默认值：False
- **HideCallingNumber** (*bool*) -- 默认值：False
- **HideTxConnectSpeed** (*bool*) -- 默认值：False

- **HideLastSentLcpConfReq** (*bool*) -- 默认值：False
- **HideLastReceivedLcpConfReq** (*bool*) -- 默认值：False
- **HideProxyAuthenType** (*bool*) -- 默认值：False
- **HideProxyAuthenName** (*bool*) -- 默认值：False
- **HideProxyAuthenChallenge** (*bool*) -- 默认值：False
- **HideProxyAuthenId** (*bool*) -- 默认值：False
- **HideProxyAuthenResponse** (*bool*) -- 默认值：False

返回 L2tp 协议会话对, 类型：object

返回类型 (L2tp)

实际案例

```
| Create L2tp | Port=${Port} |
```

TesterLibrary.Protocol.l2tp.**disconnect_l2tp**(*Sessions*)

断开 L2tp 协议会话

参数 **Sessions** (L2tp) -- L2tp 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Disconnect L2tp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.l2tp.**edit_l2tp**(*Session*, ***kwargs*)

修改 L2tp 协议会话对象

参数 **Session** (L2tpProtocolConfig) -- L2tp 协议会话对, 类型为：object

关键字参数

- **Name** (*str*) -- L2tp 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 L2tp 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EmulationMode** (*str*) -- L2TP 角色, 默认值：LAC, 取值范围：

  LAC

  LNS
- **TunnelCountPerNode** (*int*) -- 每 LAC/LNS 隧道数, 取值范围：1-32768, 默认值：1
- **SessionCountPerTunnel** (*int*) -- 每隧道会话数, 取值范围：0-65535, 默认值：0
- **TunnelStartingId** (*int*) -- 隧道起始 ID, 取值范围：1-65535, 默认值：1
- **SessionStartingId** (*int*) -- 会话起始 ID, 取值范围：1-65535, 默认值：1
- **UdpSourcePort** (*int*) -- UDP 源端口, 取值范围：1-65535, 默认值：1701
- **UdpChecksumEnabled** (*bool*) -- 使能 UDP 校验和, 默认值：True
- **RetryTunnelCreationEnabled** (*bool*) -- 使能隧道重试, 默认值：False

- **TunnelCreationTimeout** (*int*) -- 隧道建立超时 (secs), 取值范围：1-65535, 默认值：5
- **MaxTunnelCreationTimes** (*int*) -- 隧道建立最大尝试次数, 取值范围：1-65535, 默认值：5
- **HostName** (*str*) -- 主机名, 取值范围：string length in [1,255], 默认值：xinertel
- **EnableAuthentication** (*bool*) -- 使能认证, 默认值：True
- **IncomingTunnelPassword** (*str*) -- Incoming 隧道密码, 取值范围：string length in [1,255], 默认值：xinertel
- **OutgoingTunnelPassword** (*str*) -- Outgoing 隧道密码, 取值范围：string length in [1,255], 默认值：xinertel
- **HelloEnabled** (*bool*) -- 使能 Hello, 默认值：False
- **HelloInterval** (*int*) -- Hello 间隔 (secs), 取值范围：1-255, 默认值：60
- **TxBitRate** (*int*) -- 发送 bps 速率 (bits/sec), 取值范围：1-65535, 默认值：56000
- **BearerCapabilities** (*str*) -- 负载能力, 默认值：ANALOG, 取值范围：

  DIGITAL

  ANALOG

  BOTH
- **BearerType** (*str*) -- 负载类型, 默认值：ANALOG, 取值范围：

  DIGITAL

  ANALOG
- **FrameCapabilities** (*str*) -- 帧能力, 默认值：SYNC, 取值范围：

  SYNC

  ASYNC

  BOTH
- **FrameType** (*str*) -- 帧类型, 默认值：SYNC, 取值范围：

  SYNC

  ASYNC
- **CallingNumberEnabled** (*bool*) -- 使能 Calling Number, 默认值：False
- **CallingNumber** (*str*) -- 隧道的 Calling Number, 默认值：xinertel
- **RxWindowSize** (*int*) -- 接收窗口大小, 取值范围：1-65535, 默认值：4
- **UseGatewayAsRemoteIp** (*bool*) -- 使用网关作为远端地址, 默认值：True
- **RemoteIpv4Address** (*str*) -- 远端 IPv4 地址, 取值范围：IPv4 地址, 默认值：2.1.1.1
- **RemoteIpv4AddressStep** (*str*) -- 远端 IPv4 地址跳变, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **RemoteIpv6Address** (*str*) -- 远端 IPv6 地址, 取值范围：IPv6 地址, 默认值：2000::1
- **RemoteIpv6AddressStep** (*str*) -- 远端 IPv6 地址跳变, 取值范围：IPv6 地址, 默认值：::1

- **LcpProxyMode** (*str*) -- LCP 代理模式, 默认值：NONE, 取值范围：

  NONE

  LCP

  LCP_AUTH

- **ForceLcpRenegotiation** (*bool*) -- 强制 LCP 重协商, 默认值：False
- **Ipv4TosValue** (*hex int*) -- IPv4 TOS 值, 取值范围：1-65535, 默认值：0xc0
- **Ipv6TrafficClassValue** (*hex int*) -- IPv6 Traffic Class Value, 取值范围：1-65535, 默认值：0x0
- **HideFramingCapabilities** (*bool*) -- 默认值：False
- **HideBearerCapabilities** (*bool*) -- 默认值：False
- **HideAssignedTunnelId** (*bool*) -- 默认值：False
- **HideChallenge** (*bool*) -- 默认值：False
- **HideChallengeResponse** (*bool*) -- 默认值：False
- **HideAssignedSessionId** (*bool*) -- 默认值：False
- **HideCallSerialNumber** (*bool*) -- 默认值：False
- **HideFramingType** (*bool*) -- 默认值：False
- **HideCallingNumber** (*bool*) -- 默认值：False
- **HideTxConnectSpeed** (*bool*) -- 默认值：False
- **HideLastSentLcpConfReq** (*bool*) -- 默认值：False
- **HideLastReceivedLcpConfReq** (*bool*) -- 默认值：False
- **HideProxyAuthenType** (*bool*) -- 默认值：False
- **HideProxyAuthenName** (*bool*) -- 默认值：False
- **HideProxyAuthenChallenge** (*bool*) -- 默认值：False
- **HideProxyAuthenId** (*bool*) -- 默认值：False
- **HideProxyAuthenResponse** (*bool*) -- 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit L2tp | Session=${Session} |
```

TesterLibrary.Protocol.l2tp.**edit_l2tp_port_config**(*Ports*, ***kwargs*)

修改 DHCPv6 端口配置对象

Args：

Ports (Port): 测试仪表端口对象, 类型为：object / list

关键字参数 **TunnelConnectRate** (*int*) -- Request 速率（会话/秒）, 取值范围：1-1000, 默认值：100

返回 L2tp 端口对象, 类型：object / list

返回类型 (L2tpPortConfig)

实际案例

```
| Edit L2tp Port Config | Ports=${Port} | TcpServerPort=10 |
```

TesterLibrary.Protocol.l2tp.**get_l2tp_block_statistic**(*Session*, *StaItems=None*)

获取 L2tp Block Statistic 统计结果

参数

- **Session** (L2tp) -- L2tp 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TunnelCount

  SessionCount

  TunnelUp

  TunnelDown

  SessionUp

  SessionDown

  TunnelSetupRate

  SessionSetupRate

  AverageTunnelSetupTime

  MaxTunnelSetupTime

  MinTunnelSetupTime

  AverageSessionSetupTime

  MaxSessionSetupTime

  MinSessionSetupTime

  TxPackets

  RxPackets

  TxSccrq

  RxSccrq

  TxSccrp

  RxSccrp

  TxScccn

  RxScccn

  TxIcrq

  RxIcrq

  TxIcrp

  RxIcrp

  TxIccn

  RxIccn

  TxSli

  RxSli

TxStopCcn

RxStopCcn

TxWen

RxWen

TxHello

RxHello

TxCdn

RxCdn

TxZlb

RxZlb

返回

eg:

```
{
    'TxZlb': 10,
    'RxZlb': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=L2tpBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get L2tp Block Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.l2tp.**get_l2tp_port_statistic**(*Port*, *StaItems=None*)

获取 L2tp Session Statistic 统计结果

参数

- **Port** (Port) -- 测试仪表端口对象，类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  LacCount

  LnsCount

  TunnelCount

  SessionCount

  TunnelUp

  TunnelDown

  SessionUp

  SessionDown

返回

eg:

```
{
    'TunnelUp': 10,
    'SessionUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=L2tpPortStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get L2tp Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.l2tp.**get_l2tp_session_statistic**(*Session*, *NodeIndexInBlock*, *StaItems=None*)

获取 L2tp Session Statistic 统计结果

参数

- **Session** (L2tp) -- L2tp 协议会话对象, 类型为：Object
- **NodeIndexInBlock** (*int*) -- Session Index, 类型为：int
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  LocalTunnelId

  RemoteTunnelId

  LocalSessionId

  RemoteSessionId

  SessionState

  LocalTunnelIpAddress

  RemoteTunnelIpAddress

  LocalTunnelIpv6Address

  RemoteTunnelIpv6Address

  TxIcrq

  RxIcrq

  TxIcrp

  RxIcrp

  TxIccn

  RxIccn

  TxCdn

  RxCdn

  ResultCode

  ErrorCode

ErrorMessage

返回

eg:

```
{
    'TxIcrq': 10,
    'RxIcrq': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=L2tpSessionStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get L2tp Session Statistic | Session=${Session} |␣
↪NodeIndexInBlock=1 | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.l2tp.**get_l2tp_tunnel_statistic**(*Session*, *NodeIndexInBlock*, *StaItems=None*)

获取 L2tp Tunnel Statistic 统计结果

参数

- **Session** (L2tp) -- L2tp 协议会话对象, 类型为：Object
- **NodeIndexInBlock** (*int*) -- Session Index, 类型为：int
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  LocalTunnelId

  RemoteTunnelId

  TunnelState

  UdpSourcePort

  UdpDestinationPort

  LocalIpAddress

  RemoteIpAddress

  LocalIpv6Address

  RemoteIpv6Address

  SessionCount

  SessionUp

  SessionDown

  TxPackets

  RxPackets

  TxSccrq

  RxSccrq

TxSccrp

RxSccrp

TxScccn

RxScccn

TxSli

RxSli

TxStopCcn

RxStopCcn

TxWen

RxWen

TxHello

RxHello

返回

eg:

```
{
    'TxHello': 10,
    'RxHello': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=L2tpTunnelStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get L2tp Tunnel Statistic | Session=${Session} |␣
↪NodeIndexInBlock=1 | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.l2tp.**wait_l2tp_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 L2tp 协议会话达到指定状态

参数

- **Sessions** (L2tp) -- L2tp 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 L2tp 协议会话达到的状态, 类型为：string, 默认值：达到 CONNECTED, 支持下列状态：

  NONE IDLE CONNECTING CONNECTED DISCONNECTING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait L2tp State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.ldp module

TesterLibrary.Protocol.ldp.**abort_request_ldp_label**(*Configs*)

中止 LDP 协议会话 LSP 请求标签

参数 **Configs** (*list*) -- LDP LSP 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Abort Request Ldp Label | Configs=${Configs} |
```

TesterLibrary.Protocol.ldp.**advertise_ldp_label**(*Configs*)

通告 LDP 协议会话 LSP 标签

参数 **Configs** (*list*) -- LDP LSP 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Ldp Label | Configs=${Configs} |
```

TesterLibrary.Protocol.ldp.**create_ldp**(*Port*, ***kwargs*)

创建 LDP 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **HelloType** (*str*) -- Hello 类型, 类型为：string, 默认值：DIRECT, 取值范围:

  DIRECT

  TARGETED

  DIRECT_TARGETED
- **LabelAdvertType** (*str*) -- 标签分配方式, 类型为：string, 默认值：DIRECT, 取值范围:

  DU

  DOD

- **TransportMode** (*str*) -- Transport Address TLV 模式, 类型为：string, 默认值：TESTER_IP, 取值范围:

  TESTER_IP

  ROUTER_ID

  NONE

- **DutIpv4Address** (*int*) -- DUT IPv4 地址, 类型为：number, 型为：string, 默认值：2.1.1.1, 取值范围: IPv4 地址
- **DirectHelloInterval** (*int*) -- 直连 Hello 发送间隔 (sec), 类型为：number, 默认值：5, 取值范围：1-21845
- **TargetedHelloInterval** (*int*) -- 远端 Hello 发送间隔 (sec), 类型为：number, 默认值：15, 取值范围：1-21845
- **KeepAliveInterval** (*int*) -- 保活间隔 (sec), 类型为：number, 默认值：60, 取值范围：1-21845
- **LabelReqRetryCount** (*int*) -- 标签请求间隔 (sec), 类型为：number, 默认值：10, 取值范围：1-65535
- **LabelReqRetryInterval** (*int*) -- 标签请求重试次数, 类型为：number, 默认值：60, 取值范围：1-65535
- **Authentication** (*str*) -- 鉴权类型, 类型为：string, 默认值：DIRECT, 取值范围:

  NONE

  MD5

- **Password** (*str*) -- 密码, 类型为：string, 默认值：xinertel
- **EgressLabel** (*str*) -- 出标签方式, 类型为：string, 默认值：DIRECT, 取值范围:

  NEXT_AVAILABLE

  IMPLICIT

  EXPLICIT

- **MinLabel** (*int*) -- 最小标签值, 类型为：number, 默认值：16, 取值范围：0-1048575
- **EnableLspResult** (*bool*) -- LSP 统计使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnablePseudowireLspResult** (*bool*) -- 伪线 LSP 统计使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LspBindMode** (*str*) -- LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

- **VcLspBindMode** (*str*) -- 虚拟电路 LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

NONE

- **GeneralizedLspBindMode** (*str*) -- 通用伪线 LSP 绑定模式, 类型为: string, 默认值: TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

返回 LDP 协议会话对象, 类型: object

返回类型 (Ldp)

实际案例

```
| Create Ldp | Port=${Port} |
```

TesterLibrary.Protocol.ldp.**create_ldp_fec_128**(*Session*, ***kwargs*)

创建 LDP FEC 128 对象

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为: object

关键字参数

- **Name** (*str*) -- LDP FEC 128 对象名称, 类型为: string
- **Enable** (*bool*) -- 使能 LDP FEC 128, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **ControlWordEnable** (*bool*) -- 控制字使能, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **Encapsulation** (*str*) -- 封装类型 (hex), 类型为: string, 默认值: PREFIX_FEC, 取值范围:

  ETHERNET_TAGGED_MODE

  ETHERNET

  CEM

- **GroupId** (*int*) -- 组 ID, 类型为: number, 默认值: 1, 取值范围: 1-65535
- **InterfaceMtu** (*int*) -- 接口 MTU, 类型为: number, 默认值: 1500, 取值范围: 1-65535
- **IncludePwStatusTlv** (*bool*) -- 伪线状态码使能, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **PwStatusCode** (*list*) -- 伪线状态码, 类型为: list, 默认值: PW_NOT_FORWARDING, 取值范围:

  PW_NOT_FORWARDING

  LOCAL_AC_RX_FAULT

  LOCAL_AC_TX_FAULT

  LOCAL_PSN_PW_RX_FAULT

  LOCAL_PSN_PW_TX_FAULT

- **UseCustomPwStatusTlv** (*bool*) -- 自定义伪线状态码使能, 类型为: bool, 取值范围: True 或 False, 默认值: False

- **CustomPwStatusCode** (*int*) -- 自定义伪线状态码, 类型为：number, 默认值：0, 取值范围: 0-4294967295
- **VcCount** (*int*) -- VC 数量, 类型为：number, 默认值：1, 取值范围：0-4294967295
- **StartVcId** (*int*) -- 起始 VC, 类型为：number, 默认值：1, 取值范围：0-4294967295
- **VcIdStep** (*int*) -- VC 跳变步长, 类型为：number, 默认值：1, 取值范围: 0-4294967295

返回 LDP FEC 128 对象列表, 类型: list

返回类型 (LdpFec128LspConfig)

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Ingress} | Create Ldp Fec 128 | Session=${Session} |
```

TesterLibrary.Protocol.ldp.**create_ldp_fec_129**(*Session*, ***kwargs*)

创建 LDP FEC 129 对象

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP FEC 129 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP FEC 129, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **ControlWordEnable** (*bool*) -- 控制字使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Encapsulation** (*str*) -- 封装类型 (hex), 类型为：string, 默认值：PREFIX_FEC, 取值范围:

  ETHERNET_TAGGED_MODE

  ETHERNET

  CEM
- **GroupId** (*int*) -- 组 ID, 类型为：number, 默认值：1, 取值范围：1-65535
- **InterfaceMtu** (*int*) -- 接口 MTU, 类型为：number, 默认值：1500, 取值范围: 1-65535
- **IncludePwStatusTlv** (*bool*) -- 伪线状态码使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **PwStatusCode** (*list*) -- 伪线状态码, 类型为：list, 默认值：PW_NOT_FORWARDING, 取值范围:

  PW_NOT_FORWARDING

  LOCAL_AC_RX_FAULT

  LOCAL_AC_TX_FAULT

  LOCAL_PSN_PW_RX_FAULT

  LOCAL_PSN_PW_TX_FAULT
- **UseCustomPwStatusTlv** (*bool*) -- 自定义伪线状态码使能, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **CustomPwStatusCode** (*int*) -- 自定义伪线状态码, 类型为：number, 默认值：0, 取值范围：0-4294967295
- **PwCount** (*int*) -- PW 数量, 类型为：number, 默认值：1, 取值范围：0-65535
- **Agi** (*str*) -- 起始 Agi, 型为：string, 默认值：100:1, 取值范围：IPv6 地址
- **AgiStep** (*str*) -- Agi 跳变步长, 型为：string, 默认值：0:1, 取值范围：IPv6 地址
- **Saii** (*str*) -- 起始 Saii, 型为：string, 默认值：10.0.0.1, 取值范围：IPv4 地址
- **SaiiStep** (*str*) -- Saii 跳变步长, 型为：string, 默认值：0.0.0.1, 取值范围：IPv4 地址
- **Taii** (*str*) -- 起始 Taii, 型为：string, 默认值：192.0.0.1, 取值范围：IPv4 地址
- **TaiiStep** (*str*) -- Taii 跳变步长, 型为：string, 默认值：0.0.0.1, 取值范围：IPv4 地址

返回 LDP FEC 129 对象列表, 类型：list

返回类型 (LdpFec129LspConfig)

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Ingress} | Create Ldp Fec 129 | Session=${Session} |
```

TesterLibrary.Protocol.ldp.**create_ldp_ipv4_egress**(*Session*, ***kwargs*)

创建 LDP IPv4 Egress 对象

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP IPv4 Egress 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP IPv4 Egress, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **FecType** (*str*) -- Fec 类型, 类型为：string, 默认值：PREFIX_FEC, 取值范围：

  PREFIX_FEC

  HOST_FEC
- **LspCount** (*int*) -- Lsp 数量, 类型为：number, 默认值：1, 取值范围：1-65535
- **StartIpv4Prefix** -- Lsp IPv4 前缀地址, 型为：string, 默认值：192.0.1.0, 取值范围：IPv4 地址
- **PrefixLength** (*int*) -- Lsp IPv4 前缀长度, 类型为：number, 默认值：24, 取值范围：1-32
- **PrefixStep** (*int*) -- Lsp IPv4 前缀跳变步长, 类型为：number, 默认值：1, 取值范围：1-65535
- **Ipv4PrefixStep** -- Lsp IPv4 前缀地址跳变步长, 型为：string, 默认值：0.0.1.0, 取值范围：IPv4 地址

返回 LDP IPv4 Egress 对象列表, 类型：list

返回类型 (LdpIpv4EgressLspConfig)

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Ldp Ipv4 Egress | Session=${Session} |
```

TesterLibrary.Protocol.ldp.**create_ldp_ipv4_ingress**(*Session*, ***kwargs*)

创建 LDP IPv4 Ingress 对象

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP IPv4 Ingress 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP IPv4 Ingress, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **FecType** (*str*) -- Fec 类型, 类型为：string, 默认值：PREFIX_FEC, 取值范围:

  PREFIX_FEC

  HOST_FEC
- **LspCount** (*int*) -- Lsp 数量, 类型为：number, 默认值：1, 取值范围：1-65535
- **StartIpv4Prefix** -- Lsp IPv4 前缀地址, 型为：string, 默认值：192.0.1.0, 取值范围：IPv4 地址
- **PrefixLength** (*int*) -- Lsp IPv4 前缀长度, 类型为：number, 默认值：24, 取值范围：1-32
- **PrefixStep** (*int*) -- Lsp IPv4 前缀跳变步长, 类型为：number, 默认值：1, 取值范围：1-65535
- **Ipv4PrefixStep** -- Lsp IPv4 前缀地址跳变步长, 型为：string, 默认值：0.0.1.0, 取值范围：IPv4 地址

返回 LDP IPv4 Ingress 对象列表, 类型：list

返回类型 (LdpIpv4IngressLspConfig)

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Ingress} | Create Ldp Ipv4 Ingress | Session=${Session} |
```

TesterLibrary.Protocol.ldp.**edit_ldp**(*Session*, ***kwargs*)

编辑 LDP 协议会话对象参数

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **HelloType** (*str*) -- Hello 类型, 类型为：string, 默认值：DIRECT, 取值范围:

  DIRECT

  TARGETED

DIRECT_TARGETED

- **LabelAdvertType** (*str*) -- 标签分配方式, 类型为: string, 默认值: DIRECT, 取值范围:

  DU

  DOD

- **TransportMode** (*str*) -- Transport Address TLV 模式, 类型为: string, 默认值: TESTER_IP, 取值范围:

  TESTER_IP

  ROUTER_ID

  NONE

- **DutIpv4Address** (*int*) -- DUT IPv4 地址, 类型为: number, 型为: string, 默认值: 2.1.1.1, 取值范围: IPv4 地址

- **DirectHelloInterval** (*int*) -- 直连 Hello 发送间隔 (sec), 类型为: number, 默认值: 5, 取值范围: 1-21845

- **TargetedHelloInterval** (*int*) -- 远端 Hello 发送间隔 (sec), 类型为: number, 默认值: 15, 取值范围: 1-21845

- **KeepAliveInterval** (*int*) -- 保活间隔 (sec), 类型为: number, 默认值: 60, 取值范围: 1-21845

- **LabelReqRetryCount** (*int*) -- 标签请求间隔 (sec), 类型为: number, 默认值: 10, 取值范围: 1-65535

- **LabelReqRetryInterval** (*int*) -- 标签请求重试次数, 类型为: number, 默认值: 60, 取值范围: 1-65535

- **Authentication** (*str*) -- 鉴权类型, 类型为: string, 默认值: DIRECT, 取值范围:

  NONE

  MD5

- **Password** (*str*) -- 密码, 类型为: string, 默认值: xinertel

- **EgressLabel** (*str*) -- 出标签方式, 类型为: string, 默认值: DIRECT, 取值范围:

  NEXT_AVAILABLE

  IMPLICIT

  EXPLICIT

- **MinLabel** (*int*) -- 最小标签值, 类型为: number, 默认值: 16, 取值范围: 0-1048575

- **EnableLspResult** (*bool*) -- LSP 统计使能, 类型为: bool, 取值范围: True 或 False, 默认值: False

- **EnablePseudowireLspResult** (*bool*) -- 伪线 LSP 统计使能, 类型为: bool, 取值范围: True 或 False, 默认值: False

- **LspBindMode** (*str*) -- LSP 绑定模式, 类型为: string, 默认值: TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

- **VcLspBindMode** (*str*) -- 虚拟电路 LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

- **GeneralizedLspBindMode** (*str*) -- 通用伪线 LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
```

TesterLibrary.Protocol.ldp.**edit_ldp_port_config**(*Ports*, ***kwargs*)

修改 LDP 端口配置对象

Args：

Ports (Port): 测试仪表端口对象, 类型为：object

关键字参数

- **EstablishRate** (*int*) -- LDP 发送速率 (messages/sec), 取值范围：1-10000, 默认值：100
- **AdvertiseRate** (*int*) -- 会话建立速率 (sessions/sec), 取值范围：1-10000, 默认值：100
- **ReleaseRate** (*int*) -- 会话释放速率 (sessions/sec), 取值范围：1-10000, 默认值：100
- **FecPerLdpMsg** (*int*) -- 取值范围：1-65535, 默认值：65535

返回 LDP Port Config 对象, 类型：object / list

返回类型 (LdpPortConfig)

实际案例

```
| Edit LDP Port Config | Ports=${Port} | TcpServerPort=10 |
```

TesterLibrary.Protocol.ldp.**establish_ldp**(*Sessions*)

建立 LDP 协议会话

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Establish Ldp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ldp.**get_ldp_lsp_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 Ldp Lsp 统计结果

参数

- **Session** (Ldp) -- LDP 会话对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  FECInfo

  FECType

  LabelValue

  LspMode

  LspState

  LspType

返回

eg:

```
{
    'LabelValue': 16,
    'LspMode': DU,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LdpLspStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Ldp Lsp Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.ldp.**get_ldp_point_from_lsp**(*Configs*)

获取 LDP LSP 对应的绑定流源或目的端点对象

参数 **Configs** (*list*) -- 测试仪表 LDP LSP 对象列表, 类型为：list

返回 LDP LSP 对应的绑定流源或目的端点对象列表, 类型：list

返回类型 (LdpIpv4EgressLspConfig, *LdpIpv4IngressLspConfig*, *LdpFec128LspConfig*, *LdpFec129LspConfig*)

实际案例

```
| Get Ldp Point From Lsp | Configs=${IPv4EgressLsp} |
```

TesterLibrary.Protocol.ldp.**get_ldp_session_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 Ldp Session 统计结果

参数

- **Session** (Ldp) -- LDP 会话对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxAddressWithdraw

  RxAddressWithdraw

  TxAddress

  RxAddress

  TxDirectHello

  RxDirectHello

  TxInitialization

  RxInitialization

  TxKeepAlive

  RxKeepAlive

  TxLabelAbort

  RxLabelAbort

  TxLabelMapping

  RxLabelMapping

  TxLabelRelease

  RxLabelRelease

  TxLabelRequest

  RxLabelRequest

  TxLabelWithdraw

  RxLabelWithdraw

  TxNotification

  RxNotification

  TxTargetHello

RxTargetHello

TxIPv6DirectHello

RxIPv6DirectHello

TxIPv6TargetHello

RxIPv6TargetHello

返回

eg:

```
{
    'TxIPv6TargetHello': 10,
    'RxIPv6TargetHello': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LdpSessionStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Ldp Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.ldp.**request_ldp_label**(*Configs*)

LDP 协议会话 LSP 请求标签

参数 **Configs** (*list*) -- LDP LSP 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Request Ldp Label | Configs=${Configs} |
```

TesterLibrary.Protocol.ldp.**restart_ldp**(*Sessions*)

重启 LDP 协议会话

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Restart Ldp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ldp.**resume_ldp_hello**(*Sessions*)

恢复 LDP 协议会话 Hello 发送

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Ldp Hello | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ldp.**resume_ldp_keepalive**(*Sessions*)

恢复 LDP 协议会话 Keepalive 发送

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Ldp Keepalive | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ldp.**start_ldp**(*Sessions*)

启动 LDP 协议会话

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Start Ldp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ldp.**stop_ldp**(*Sessions*)

停止 LDP 协议会话

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Ldp | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ldp.**stop_ldp_hello**(*Sessions*)

停止 LDP 协议会话 Hello 发送

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Ldp Hello | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ldp.**stop_ldp_keepalive**(*Sessions*)

停止 LDP 协议会话 Keepalive 发送

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Ldp Keepalive | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ldp.**wait_ldp_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 LDP 协议会话达到指定状态

参数

- **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 LDP 协议会话达到的状态, 类型为：string, 默认值：达到 OPERATIONAL, 支持下列状态：

  DISABLED

  NOT_STARTED

  NON_EXISTENT

  INITIAL

  OPEN_SENT

  OPEN_REC

  OPERATIONAL

  RESTARTING

  HELPING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ldp State | Sessions=${Sessions} | State=RESTARTING | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.ldp.**withdraw_ldb_label**(*Configs*)

撤销 LDP 协议会话 LSP 标签

参数 **Configs** (*list*) -- LDP LSP 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Ldp Label | Configs=${Configs} |
```

TesterLibrary.Protocol.lsp_ping module

TesterLibrary.Protocol.lsp_ping.**create_lsp_ping**(*Port*, ***kwargs*)

创建 Lsp Ping 会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- Lsp Ping 会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 Lsp Ping 会话, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Lsp Ping 会话对象, 类型：object

返回类型 (LspPing)

实际案例

```
| Create Lsp Ping | Port=${Port} |
```

TesterLibrary.Protocol.lsp_ping.**create_lsp_ping_echo_request**(*Sessions*, ***kwargs*)

创建 Lsp Ping Echo Request 对象

Args：

Session (LspPing): Lsp Ping 会话对象, 类型为：object / list

关键字参数

- **OperationMode** (*list*) -- Operation 模式, 默认值：['PING'], 取值范围：

  PING

  TRACE

- **ReplyMode** (*str*) -- Echo Reply 模式, 默认值：REPLYVIAUDP, 取值范围：

  NOTREPLY

  REPLYVIAUDP
- **PingInterval** (*int*) -- Ping 发送测试包的时间间隔（秒）, 默认值：4, 取值范围：1-65535
- **PingTimeOut** (*int*) -- Ping 探测超时时间（秒）, 默认值：2, 取值范围：1-60
- **TraceInterval** (*int*) -- Trace 发送测试包的时间间隔（秒）, 默认值：120, 取值范围：1-65535
- **TraceTimeOut** (*int*) -- Trace 探测超时时间（秒）, 默认值：2, 取值范围：1-60
- **InnerLabel** (*str*) -- 标签, 默认值：NONE, 取值范围：

  NONE

  LDPIPv4

  VPNIPv4

  SEGMENT_ROUTING
- **OuterLabel** (*str*) -- 标签, 默认值：NONE, 取值范围：

  NONE

  LDPIPv4

  VPNIPv4

  SEGMENT_ROUTING
- **TimeToLive** (*int*) -- 生存时间, 默认值：255, 取值范围：1-255
- **ExpBits** (*int*) -- 实验比特位的值, 默认值：0, 取值范围：0-7
- **PadMode** (*str*) -- 填充模式, 默认值：WITHOUT_PAD, 取值范围：

  WITHOUT_PAD

  DROP_PAD

  COPY_PAD
- **Data** (*int*) -- 填充数据, 默认值：”, 取值范围：0-255
- **DesIpv4Addr** (*str*) -- 目的地址, 默认值：”127.0.0.1”，取值范围：有效的 ipv4 地址
- **ValidateFecStack** (*bool*) -- 校验 FEC Stack, 默认值：False, 取值范围：True 或 False
- **DownstreamMappingTlvType** (*str*) -- Downstream Mapping TLV 类型, 默认值：DOWNSTREAM_DETAILED_MAPPING_TLV, 取值范围：

  DOWNSTREAM_MAPPING_TLV

  DOWNSTREAM_DETAILED_MAPPING_TLV

返回 Lsp Ping Echo Request 对象, 类型：object / list

返回类型 (LspPingEchoRequestConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| Create Lsp Ping Echo Request | Sessions=${LspPing} |
```

TesterLibrary.Protocol.lsp_ping.**create_lsp_ping_fec_ldp_ipv4**(*EchoRequests*, ***kwargs*)

创建 Lsp Ping Fec Ldp Ipv4 对象

Args:

EchoRequests (LspPingEchoRequestConfig): Lsp Ping Echo Request 对象, 类型为: object / list

关键字参数

- **Count** (*int*) -- 数量, 默认值: 1, 取值范围: 1-65535
- **StartAddr** (*str*) -- IPv4 地址, 默认值: "172.0.0.1", 取值范围: 有效的 ipv4 地址
- **PrefixLength** (*int*) -- 前缀长度, 默认值: 24, 取值范围: 1-32
- **Step** (*int*) -- 步长, 默认值: 1, 取值范围: 1-255

返回 Lsp Ping Fec Ldp Ipv4 对象, 类型: object / list

返回类型 (LspPingFecLdpIpv4PrefixConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| Create Lsp Ping Fec Ldp Ipv4 | EchoRequests=${EchoRequest} |
```

TesterLibrary.Protocol.lsp_ping.**create_lsp_ping_fec_segment_routing**(*EchoRequests*, ***kwargs*)

创建 Lsp Ping Fec Segment Routing 对象

Args:

EchoRequests (LspPingEchoRequestConfig): Lsp Ping Echo Request 对象, 类型为: object / list

关键字参数

- **IgpProtocol** (*str*) -- FEC 校验使用的 IGP 协议, 默认值: ISIS, 取值范围:

  OSPF

  ISIS
- **PrefixCount** (*int*) -- 前缀数量, 默认值: 1, 取值范围: 1-4294967295
- **PrefixAddrIncrement** (*int*) -- 地址步长, 默认值: 1, 取值范围: 1-4294967295

返回 Lsp Ping Fec Segment Routing 对象, 类型: object / list

返回类型 (LspPingFecSrConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| Create Lsp Ping Fec Segment Routing | EchoRequests=${EchoRequest} |
```

TesterLibrary.Protocol.lsp_ping.**create_lsp_ping_fec_sr_adjacency**(*Srs*, ***kwargs*)

创建 Lsp Ping Fec Sr Adjacency 对象

Args：

Srs (LspPingFecSrConfig): Lsp Ping Fec Segment Routing 对象, 类型为：object / list

关键字参数

- **IsisSystemId** (*str*) -- ISIS 系统 ID, 默认值："00:00:94:00:00:01"，取值范围：有效的 mac 地址
- **IsisLanSystemId** (*str*) -- ISIS LAN 系统 ID, 默认值："00:00:00:00:00:00"，取值范围：有效的 mac 地址
- **IsisNeighborId** (*str*) -- ISIS 邻居 ID, 默认值："00:00:94:00:00:01"，取值范围：有效的 mac 地址
- **IsisNodeId** (*int*) -- ISIS 节点 ID, 默认值：0, 取值范围：uint8
- **OspfLinkType** (*str*) -- OSPF 链路类型, 默认值：P2P, 取值范围：

  P2P

  TRANSITNETWORK

  STUBNETWORK

  VIRTUALLINK
- **OspfLinkId** (*str*) -- OSPF 链路 ID, 默认值："0.0.0.0"，取值范围：有效的 ipv4 地址
- **OspfLinkData** (*str*) -- OSPF 链路数据, 默认值："0.0.0.0"，取值范围：有效的 ipv4 地址
- **LocalRouterId** (*str*) -- 本地路由器 ID, 默认值："192.168.1.1"，取值范围：有效的 ipv4 地址
- **RemoteRouterId** (*str*) -- 远端路由器 ID, 默认值："192.168.1.1"，取值范围：有效的 ipv4 地址
- **LocalInterfaceId** (*str*) -- 本地接口 ID, 默认值："0.0.0.0"，取值范围：有效的 ipv4 地址
- **RemoteInterfaceId** (*str*) -- 远端接口 ID, 默认值："0.0.0.0"，取值范围：有效的 ipv4 地址

返回 Lsp Ping Fec Sr Detail 对象, 类型：object / list

返回类型 (LspPingFecSrDetailConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| ${Sr} | Create Lsp Ping Fec Segment Routing | EchoRequests=${EchoRequest} |
| Create Lsp Ping Fec Sr Ajacency | Srs=${Sr} |
```

TesterLibrary.Protocol.lsp_ping.**create_lsp_ping_fec_sr_prefix**(*Srs*, ***kwargs*)

创建 Lsp Ping Fec Sr Prefix 对象

Args:

Srs (LspPingFecSrConfig): Lsp Ping Fec Segment Routing 对象, 类型为: object / list

关键字参数

- **Prefix** (*str*) -- 前缀地址, 默认值: "192.0.0.1", 取值范围: 有效的 ipv4 地址
- **Length** (*int*) -- 前缀地址长度, 默认值: 24, 取值范围: 1-32
- **Algorithm** (*int*) -- 算法, 默认值: 0, 取值范围: uint8

返回 Lsp Ping Fec Sr Detail 对象, 类型: object / list

返回类型 (LspPingFecSrDetailConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| ${Sr} | Create Lsp Ping Fec Segment Routing | EchoRequests=${EchoRequest} |
| Create Lsp Ping Fec Sr Prefix | Srs=${Sr} |
```

TesterLibrary.Protocol.lsp_ping.**create_lsp_ping_fec_vpn_ipv4**(*EchoRequests*, ***kwargs*)

创建 Lsp Ping Fec Vpn Ipv4 对象

Args:

EchoRequests (LspPingEchoRequestConfig): Lsp Ping Echo Request 对象, 类型为: object / list

关键字参数

- **Count** (*int*) -- 数量, 默认值: 1, 取值范围: 1-65535
- **StartAddr** (*str*) -- IPv4 地址, 默认值: "172.0.0.1", 取值范围: 有效的 ipv4 地址
- **PrefixLength** (*int*) -- 前缀长度, 默认值: 24, 取值范围: 1-32
- **Step** (*int*) -- 步长, 默认值: 1, 取值范围: 1-255
- **RouteDistinguisher** (*str*) -- 路由标识, 默认值: "100:1", 取值范围: 匹配格式"uint16:uint32 | ipv4:uint16 | uint32:uint16 | uint16.uint16:uint16"

返回 Lsp Ping Fec Vpn Ipv4 对象, 类型: object / list

返回类型 (LspPingFecVPNIpv4PrefixConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| Create Lsp Ping Fec Vpn Ipv4 | EchoRequests=${EchoRequest} |
```

TesterLibrary.Protocol.lsp_ping.**edit_lsp_ping_port_config**(*Ports*, ***kwargs*)

修改 Lsp Ping 端口配置对象

Args：

Ports (Port): 测试仪表端口对象, 类型为：object

关键字参数

- **UpdateTransmitRate** (*int*) -- 默认值：1000, 取值范围：1-10000
- **FrequencyPing** (*int*) -- 执行 Ping 测试时间间隔（秒）, 默认值：60, 取值范围：1-2147483647
- **FrequencyTrace** (*int*) -- 执行 Trace 测试时间间隔（秒）, 默认值：60, 取值范围：60-2147483647

返回 Lsp Ping 协议端口对象, 类型：object / list

返回类型 (LspPingPortConfig)

实际案例

```
| Edit Lsp Ping Port Config | Ports=${Port} | FrequencyTrace=10 |
```

TesterLibrary.Protocol.lsp_ping.**get_lsp_ping_echo_request_statistic**(*Session*, *EchoRequest*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Port Statistic 统计结果

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object
- **EchoRequest** (LspPingEchoRequestConfig) -- Lsp Ping Echo Request 对象, 类型为：object
- **StaItems** (*list*) -- 需要获取的统计项目，类型为：list，目前支持的统计项

  FailCount

  SuccessCount

  FecInfo

  MaxPingLatency

  AvgPingLatency

  MinPingLatency

  RxReturnCode

返回

eg:

```
{
    'MinPingLatency': 10,
    'RxReturnCode': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LspPingEchoRequestStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Lsp Ping Echo Request Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.lsp_ping.**get_lsp_ping_session_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Port Statistic 统计结果

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object / list
- **StaItems** (*list*) -- 需要获取的统计项目，类型为：list，目前支持的统计项

  TxEchoRequest

  RxEchoRequest

  TxEchoReply

  RxEchoReply

返回

eg:

```
{
    'TxEchoReply': 10,
    'RxEchoReply': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LspPingSessionStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Lsp Ping Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.lsp_ping.**get_lsp_trace_echo_request_statistic**(*Session*, *EchoRequest*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Port Statistic 统计结果

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object
- **EchoRequest** (LspPingEchoRequestConfig) -- Lsp Ping Echo Request 对象, 类型为：object
- **StaItems** (*list*) -- 需要获取的统计项目，类型为：list，目前支持的统计项

  FailCount

  SuccessCount

  FecInfo

  MaxPingLatency

  AvgPingLatency

  MinPingLatency

  RxReturnCode

返回

eg:

```
{
    'MinPingLatency': 10,
    'RxReturnCode': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LspPingEchoRequestStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Lsp Trace Echo Request Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.lsp_ping.**pause_lsp_ping**(*Sessions*)

暂停发送 LSP Ping 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pause Lsp Ping | Sessions=${Sessions} |
```

TesterLibrary.Protocol.lsp_ping.**pause_lsp_trace**(*Sessions*)

暂停发送 LSP Trace 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pause Lsp Trace | Sessions=${Sessions} |
```

TesterLibrary.Protocol.lsp_ping.**resume_lsp_ping**(*Sessions*)

继续发送 LSP Ping 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Lsp Ping | Sessions=${Sessions} |
```

TesterLibrary.Protocol.lsp_ping.**resume_lsp_trace**(*Sessions*)

继续发送 LSP Trace 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Lsp Trace | Sessions=${Sessions} |
```

TesterLibrary.Protocol.lsp_ping.**start_lsp_ping**(*Sessions*)

开始发送 LSP Ping 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Start Lsp Ping | Sessions=${Sessions} |
```

TesterLibrary.Protocol.lsp_ping.**stop_lsp_ping**(*Sessions*)

停止发送 LSP Ping 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Lsp Ping | Sessions=${Sessions} |
```

TesterLibrary.Protocol.lsp_ping.**wait_lsp_ping_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Lsp Ping 会话的 Ping 消息达到指定状态

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object / list
- **State** (*str*) -- 等待 Lsp Ping 会话组的 Ping 消息达到的状态, 类型为：string, 默认值：PAUSE_SEND, 支持下列状态：

  IDLE

  PAUSE_SEND

  RESUME_SEND
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Lsp Ping State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.lsp_ping.**wait_lsp_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Lsp Ping 会话达到指定状态

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object / list
- **State** (*str*) -- 等待 Lsp Ping 会话组达到的状态, 类型为：string, 默认值：UP, 支持下列状态：

  DISABLE

  NOTSTART

  UP

  DOWN
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Lsp State | Sessions=${Sessions} | State=UP | Interval=2 | TimeOut=120␣
↪|
```

TesterLibrary.Protocol.lsp_ping.**wait_lsp_trace_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Lsp Ping 会话的 Trace 消息达到指定状态

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object / list
- **State** (*str*) -- 等待 Lsp Ping 会话组的 Trace 消息达到的状态, 类型为：string, 默认值：PAUSE_SEND, 支持下列状态：

  IDLE

  PAUSE_SEND

  RESUME_SEND
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Lsp Trace State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.mld module

TesterLibrary.Protocol.mld.**apply_mld_querier**(*Sessions*)

MLD Querier 增量配置下发到后台

参数 **Sessions** (list (MldQuerier)) -- MLD Querier 协会话对象, 类型为：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Apply Mld Querier |
```

TesterLibrary.Protocol.mld.**create_mld**(*Port*, ***kwargs*)

创建 MLD 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- MLD 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 MLD 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：MLDV1, 支持版本：

  MLDV1

  MLDV2
- **PackReports** (*bool*) -- 合并报告报文, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InitialJoin** (*bool*) -- 单个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustJoin** (*bool*) -- 多个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustnessVariable** (*int*) -- Robust 值, 类型为：number, 取值范围：2-255, 默认值：2
- **UnsolicitedReportInterval** (*int*) -- 发送初始报文的时间间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **ForceLeave** (*bool*) -- 强制发送 Leave 报文, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TrafficClass** (*hex int*) -- IP 头的 Traffic Class 值, 型为：string, 取值范围：0x0-0xff, 默认值：0xc0

返回 MLD 协议会话对象, 类型：object

返回类型 (Mld)

实际案例

```
| Create Mld | Port=${Port} | Version=MLDV2 |
```

TesterLibrary.Protocol.mld.**create_mld_querier**(*Port*, ***kwargs*)

创建 MLD Querier 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- MLD Querier 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 MLD Querier 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：MLDV1, 支持版本：

  MLDV1

  MLDV2
- **RobustnessVariable** (*int*) -- 健壮系数, 取值范围：2-255, 默认值: 2
- **Interval** (*int*) -- 查询时间间隔（秒）, 取值范围：0-4294967295, 默认值: 125
- **QueryResponseInterval** (*int*) -- 查询响应时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 10000
- **StartupQueryCount** (*int*) -- 初始查询报文个数, 取值范围：1-255, 默认值: 2
- **LastMemberQueryInterval** (*int*) -- 最后成员查询时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 1000
- **LastMemberQueryCount** (*bool*) -- 最后成员查询次数, 取值范围：0-255, 默认值: 2
- **IPv6TrafficClassValue** (*str*) -- 设置 IPv6 头 TrafficClass 值, 取值范围：0x0-0xff, 默认值: 0x0

返回 MLD 协议会话对象, 类型：object

返回类型 (MldQuerier)

实际案例

```
| Create Mld Querier| Port=${Port} | Version=MLDV3 |
```

TesterLibrary.Protocol.mld.**edit_mld**(*Session*, ***kwargs*)

创建 MLD 协议会话对象

参数 **Session** (Mld) -- MLD 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- MLD 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 MLD 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：MLDV1, 支持版本：

  MLDV1

  MLDV2

- **PackReports** (*bool*) -- 合并报告报文, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InitialJoin** (*bool*) -- 单个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustJoin** (*bool*) -- 多个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustnessVariable** (*int*) -- Robust 值, 类型为：number, 取值范围：2-255, 默认值：2
- **UnsolicitedReportInterval** (*int*) -- 发送初始报文的时间间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **ForceLeave** (*bool*) -- 强制发送 Leave 报文, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TrafficClass** (*hex int*) -- IP 头的 Traffic Class 值, 型为：string, 取值范围：0x0-0xff, 默认值：0xc0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Mld | Port=${Port} | Version=MLDV2 |
```

TesterLibrary.Protocol.mld.**edit_mld_querier**(*Session*, ***kwargs*)

编辑 MLD Querier 协议会话对象

参数 **Session** (MldQuerier) -- MLD 协会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- MLD Querier 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ICMP Querier 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：MLDV1, 支持版本：

  MLDV1

  MLDV2
- **RobustnessVariable** (*int*) -- 健壮系数, 取值范围：2-255, 默认值: 2
- **Interval** (*int*) -- 查询时间间隔（秒）, 取值范围：0-4294967295, 默认值: 125
- **QueryResponseInterval** (*int*) -- 查询响应时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 10000
- **StartupQueryCount** (*int*) -- 初始查询报文个数, 取值范围：1-255, 默认值: 2
- **LastMemberQueryInterval** (*int*) -- 最后成员查询时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 1000
- **LastMemberQueryCount** (*bool*) -- 最后成员查询次数, 取值范围：0-255, 默认值: 2
- **IPv6TrafficClassValue** (*str*) -- 设置 IPv6 头 TrafficClass 值, 取值范围：0x0-0xff, 默认值: 0x0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Mld Querier | Port=${Port} | Version=MLDV2 |␣
↪IPv6TrafficClassValue=0xff |
```

TesterLibrary.Protocol.mld.**get_mld_host_statistic**(*Session*, *StaItems=None*)

获取 Mld 协议会话统计结果

参数

- **Session** (Mld) -- Mld 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  MldTxFrames

  MldRxFrames

  MldRxUnknownTypes

  MldRxChecksumErrors

  MldRxLengthErrors

返回

eg:

```
{
    'MldTxFrames': 8,
    'MldRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | MldTxFrames | MldRxFrames |
| Subscribe Result | Types=MldHostResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Mld Host Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.mld.**get_mld_port_statistic**(*Port*, *StaItems=None*)

获取 Mld Port 统计结果

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  MldTxFrames

  MldRxFrames

  MldTxV1Reports

  MldStopListenGroups

MldTxV2Reports

MldTxV2ModeInclude

MldTxV2ModeExclude

MldTxV2ModeChangeToInclude

MldTxV2ModeChangeToExclude

MldTxV2ModeAllowNewSources

MldTxV2ModeBlockOldSources

MldRxV1Queries

MldRxV2Queries

MldRxGeneralQueries

MldRxGroupSpecificQueries

MldRxGroupAndSourceSpecificQueries

MldRxUnknownTypes

MldRxChecksumErrors

MldRxLengthErrors

返回

eg:

```
{
    'MldTxFrames': 8,
    'MldRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | MldTxFrames | MldRxFrames |
| Subscribe Result | Types=MldPortAggregatedResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Mld Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.mld.**get_mld_querier_statistic**(*Session*, *StaItems=None*)

获取 Mld Querier 协议会话统计结果

参数

- **Session** (MldQuerier) -- Mld 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  QuerierTxFrames

  QuerierRxFrames

  QuerierRxUnknownTypes

  QuerierRxChecksumErrors

  QuerierRxLengthErrors

返回

eg:

```
{
    'QuerierTxFrames': 8,
    'QuerierRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | QuerierTxFrames | QuerierRxFrames |
| Subscribe Result | Types=MldQuerierResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Mld Querier Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.mld.**wait_mld_querier_state**(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 Mld Querier 协议会话达到指定状态

参数

- **Sessions** (list (MldQuerier)) -- Mld Querier 协议会话对象列表
- **State** (*str*) -- 等待 Mld Querier 协议会话达到的状态, 默认值：达到 UP, 支持下列状态：

  NOTSTARTED

  UP
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Mld Querier State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.mld.**wait_mld_state**(*Sessions*, *State='MEMBER'*, *Interval=1*, *TimeOut=60*)

等待 Mld 协议会话达到指定状态

参数

- **Sessions** (list (Mld)) -- Mld 协议会话对象列表
- **State** (*str*) -- 等待 Mld 协议会话达到的状态, 默认值：达到 MEMBER, 支持下列状态：

  NONMEMBER JOINING MEMBER LEAVING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec

- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Mld State | Sessions=${Sessions} | State=UP | Interval=2 | TimeOut=120␣
↪|
```

TesterLibrary.Protocol.multicast module

TesterLibrary.Protocol.multicast.**binding_multicast_group**(*Session*, *Memberships*, *MulticastGroup*)

将全局组播组绑定到组播协议会话上

参数

- **Session** (Mld, *Igmp*) -- IGMP/MLD 协会话对象, 类型为：object
- **Memberships** (MldMembershipsConfig) -- 组播协议和组播组绑定关系对象, 类型为：object
- **MulticastGroup** (MldSelectMulticastGroupCommand) -- 全局组播组对象, 类型为：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Group} | Create Multicast Group | Version=IPV4 | Start=225.0.1.1 |␣
↪Number=20 |
| ${Session} | Create Igmp | Port=${Port} | Version=IGMPV3 |
| ${Memberships} | Create Memberships | Session=${Session} | Start=225.0.1.1␣
↪| DeviceGroupMapping=ONETOONE |
| binding_multicast_group | Session=${Session} | Memberships=${Memberships} |␣
↪MulticastGroup=${Group} |
```

TesterLibrary.Protocol.multicast.**create_memberships**(*Session*, ***kwargs*)

创建组播协议和组播组绑定关系对象

参数 **Session** (Mld, *Igmp*) -- IGMP/MLD 协会话对象, 类型为：object

关键字参数

- **DeviceGroupMapping** (*str*) -- 主机和组播组映射关系, 类型为：str, 默认值：MANYTOMANY, 取值范围：

  MANYTOMANY

  ONETOONE

  ROUNDROBIN

- **SourceFilterMode** (*str*) -- 源地址过滤模式, 类型为: str, 默认值: EXCLUDE, 取值范围:

  INCLUDE

  EXCLUDE

- **UserDefinedSources** (*bool*) -- 自定义源地址, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **SpecifySourcesAsList** (*bool*) -- 配置离散源地址, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **SourceAddressList** (*list*) -- 离散源地址列表, 类型为: list, 取值范围: ipv4 or ipv6 string list
- **NumberOfSources** (*int*) -- 组播组地址掩码, 类型为: number, 取值范围: 0-16777215 默认值: 1
- **StartingSourceIp** (*str*) -- 组播组起始源地址, 类型为: string, 取值范围: ipv4 or ipv6 string list，, 默认值 ipv4: 192.0.1.0 , ipv6: 2000::1
- **PrefixLength** (*int*) -- 组跳变位, 类型为: number, 取值范围: ipv4: 1-32 默认值: 32, ipv6: 1-128 默认值: 128
- **Increment** (*int*) -- 跳变步长, 类型为: number, 取值范围: 0-16777215 默认值: 1

返回 组播协议和组播组绑定关系对象, 类型: object

返回类型 (MldMembershipsConfig)

实际案例

```
| ${Session} | Create Mld | Port=${Port} |
| Create Memberships | Session=${Session} | Start=225.0.1.1 |␣
↪DeviceGroupMapping=ONETOONE |
```

TesterLibrary.Protocol.multicast.**create_multicast_group**(*Version='IPv4'*, ***kwargs*)

创建全局组播组对象

参数 **Version** (*str*) -- 组播组 IP 版本，类型 string, 支持 ipv4 和 ipv6

关键字参数

- **Count** (*int*) -- 组播组数量, 类型为: number, 取值范围: 0-65535, 默认值: 1
- **Mode** (*str*) -- 组播组地址模式, 类型为: string, 默认值: RANGE, 取值范围:

  RANGE

  LIST

  RFC_4814

- **Start** (*str*) -- 组播组地址起始值, 类型为: ipv4/ipv6 string, 默认值: 225.0.0.1 或 ff1e::1
- **Number** (*int*) -- 组播组地址数量, 类型为: number, 取值范围: 1-268435456, 默认值: 1
- **Increment** (*int*) -- 组播组地址步长, 类型为: number, 取值范围: 1-268435456, 默认值: 1
- **Prefix** (*int*) -- 组播组地址掩码, 类型为: number, 取值范围: ipv4: 1-32 默认值: 32, ipv6: 1-128 默认值: 128

返回 全局组播组对象, 类型：object

返回类型 (MldSelectMulticastGroupCommand)

实际案例

```
| Create Multicast Group | Version=IPV4 | Start=225.0.1.1 | Number=20 |
```

TesterLibrary.Protocol.ospfv2 module

TesterLibrary.Protocol.ospfv2.**advertise_ospf_lsa**(*Sessions=None*, *Type=None*, *Lsa=None*)

通告 OSPFv2 协议会话 lsa

参数

- **Sessions** (list(OspfRouter)) -- OSPFv2 协议会话对象列表, 类型为：list
- **Type** (*str*) -- OSPFv2 lsa 类型, 类型为：string, 支持的 lsa 类型：

  Router

  Network

  Summary

  AsbrSummary

  External
- **Lsa** (*list*) -- OSPFv2 lsa 列表, 类型为：list, 当 Type=None 时参数生效

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Ospf Lsa | Sessions=${Sessions} | Type=router |
| Advertise Ospf Lsa | Sessions=${Sessions} | Lsa=${Lsas} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf**(*Port*, ***kwargs*)

创建 OSPFv2 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- OSPFv2 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 OSPFv2 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AreaId** (*str*) -- 区域 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 取值范围：Broadcast 或 P2P, 默认值：Broadcast
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-255, 默认值：0

- **Cost** (*int*) -- 接口开销, 类型为：number, 取值范围：1-65535, 默认值：10
- **AuthenticationType** (*str*) -- 类型为：string, 取值范围：None Simple 或 MD5, 默认值：None
- **Password** (*str*) -- 密码, 类型为：string, 默认值：Xinertel
- **Md5KeyId** (*int*) -- MD5 密钥, 类型为：number, 取值范围：0-255, 默认值：1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT
- **EnableOspfv2Mtu** (*bool*) -- 使能 OSPF MTU, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GracefulRestartReason** (*str*) -- 平滑重启原因, 类型为：string, 默认值：UNKNOWN, 支持的原因：

  UNKNOWN

  SOFTWARE

  RELOADORUPGRADE

  SWITCH
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **HelloInterval** (*int*) -- Hello 包间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **RouterDeadInterval** (*int*) -- 路由器失效间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：40
- **LsaRetransInterval** (*int*) -- LSA 重传间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：5
- **LsaRefreshTime** (*int*) -- LSA 刷新间隔 (秒), 类型为：number, 取值范围：1-1800, 默认值：1800
- **EnableSrManagement** (*bool*) -- 启用 SR, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 OSPFv2 协议会话对象, 类型：object

返回类型 (OspfRouter)

实际案例

```
| Create Ospf | Port=${Port} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_adj_sid_sub_tlv**(*Session*, *ExtendedLinkTlv*, ***kwargs*)

创建 OSPFv2 Adj Sid Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **ExtendedLinkTlv** (Ospfv2ExtendedLinkTlvConfig) -- OSPFv2 Extended Link Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Adj Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- 选项, 类型为：list, 默认值：['ValueIndexFlag', 'LocalGlobalFlag', 'NONE'], 支持选项有：

  BackupFlag

  ValueIndexFlag

  LocalGlobalFlag

  GroupFlag

  PersistentFlag

  NONE
- **MultiTopologyId** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：0-255, 默认值：0
- **Weight** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：0-255, 默认值：0
- **SidLabel** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：1-255, 默认值：1

返回 OSPFv2 Adj Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv2AdjSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} | Age=20␣
↪|
| ${Tlv} | | Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa} |
| Create Ospf Adj Sid Sub Tlv | Session=${Session} | ExtendedLinkTlv=${Tlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_asbr_summary_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 Asbr Summary LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Summary LSA 的名称, 类型为：string

- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **AsbrRouterId** (*int*) -- 路由个数, 类型为：number, 取值范围：1-1000000, 默认值：1
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Asbr Summary LSA 对象, 类型：object

返回类型 (Ospfv2AsbrSummaryLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Asbr Summary Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_bier_mpls_encap_sub_tlv**(*Tlv*, ***kwargs*)

创建 OSPFv2 Bier Mpls Encap Sub Tlv 对象

参数 **Tlv** (Port) -- OSPFv2 Bier Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Bier Mpls Encap Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TlvType** (*int*) -- Type 字段值, 类型为：number, 取值范围：0-255, 默认值：10
- **MaxSi** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：1-255, 默认值：1

- **Label** (*int*) -- 标签范围中的起始标签值, 类型为：number, 取值范围：0-1048575, 默认值：100

返回 OSPFv2 Bier Mpls Encap Sub Tlv 对象, 类型：object

返回类型 (Ospfv2BierMplsEncapSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| ${SubTlv}| Create Ospf Bier Sub Tlv | Tlv=${Tlv} |
| Create Ospf Bier Mpls Encap Sub Tlv | Tlv=${SubTlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_bier_sub_tlv**(*Tlv*, ***kwargs*)

创建 OSPFv2 Bier Sub Tlv 对象

参数 **Tlv** (Port) -- OSPFv2 Ext Prefix Range Tlv / Ospfv2 Ext Prefix Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Bier Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TlvType** (*int*) -- Type 字段值, 类型为：number, 取值范围：0-255, 默认值：9
- **SubDomainId** (*int*) -- BIER 子域 ID, 类型为：number, 取值范围：1-255, 默认值：1
- **MtId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-255, 默认值：1
- **BfrId** (*int*) -- BFR（Bit Forwarding Router，比特转发路由器）ID, 类型为：number, 取值范围：1-65535, 默认值：1
- **Bar** (*int*) -- BIER 算法, 类型为：number, 取值范围：0-255, 默认值：0
- **Ipa** (*int*) -- IGP 算法, 类型为：number, 取值范围：0-255, 默认值：0

返回 OSPFv2 Bier Sub Tlv 对象, 类型：object

返回类型 (Ospfv2BierSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| Create Ospf Bier Sub Tlv | Tlv=${Tlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_custom_sub_tlv**(*SrLinkMsdSubTlv*, ***kwargs*)

创建 OSPFv2 Custom Sub Tlv 对象

参数 **SrLinkMsdSubTlv** (Ospfv2SrLinkMsdSubTlvConfig) -- OSPFv2 Sr Link Msd Sub Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Custom Sub Tlv 的名称, 类型为：string

- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SubType** (*int*) -- 类型为：number, 取值范围：0-255, 默认值：0
- **SubValue** (*int*) -- 类型为：number, 取值范围：0-255, 默认值：8

返回 OSPFv2 Custom Sub Tlv 对象, 类型：object

返回类型 (Ospfv2CustomMsdSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} | Age=20␣
↪|
| ${Tlv} | | Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa} |
| ${SubTlv} | Create Ospf Sr Link Msd Sub Tlv | SrLinkMsdSubTlv=${Tlv} |
| Create Ospf Custom Sub Tlv | SrLinkMsdSubTlv = ${SubTlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_ext_prefix_range_tlv**(*Session*, *OpaqueExtendedPrefixLsa*, ***kwargs*)

创建 OSPFv2 Ext Prefix Range Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **OpaqueExtendedPrefixLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Extended Prefix LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Ext Prefix Range Tlv 的名称, 类型为: string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PrefixLength** (*int*) -- 前缀长度, 类型为：number, 取值范围：0-32, 默认值：24
- **AF** (*str*) -- 前缀的地址族, 类型为：string, 默认值：IPv4Unicast, 取值范围：

  IPv4Unicast
- **ExtendedPrefixRange** (*int*) -- 要生成的前缀的数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **ExtendedPrefixFlags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NoneFlag, 支持选项有：

  NoneFlag

  IAInterArea
- **AddressPrefix** (*str*) -- 起始地址前缀, 类型为：string, 默认值：192.0.1.0, 取值范围：有效的 ipv4 地址

返回 OSPFv2 Ext Prefix Range Tlv 对象, 类型：object

返回类型 (Ospfv2ExtPrefixRangeTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Prefix Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Ext Prefix Range Tlv | Session=${Session} |␣
↪OpaqueExtendedPrefixLsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_ext_prefix_tlv**(*Session*, *OpaqueExtendedPrefixLsa*, ***kwargs*)

创建 OSPFv2 Ext Prefix Range Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **OpaqueExtendedPrefixLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Extended Prefix LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Ext Prefix Range Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- 起始地址前缀, 类型为：string, 默认值：Unspecified, 取值范围：

  Unspecified

  IntraArea

  InterArea

  AsExternal

  NssaExternal
- **AddressPrefix** (*str*) -- 起始地址前缀, 类型为：string, 默认值：192.0.1.0, 取值范围：有效的 ipv4 地址
- **PrefixLength** (*int*) -- 要生成的前缀的数量, 类型为：number, 取值范围：0-32, 默认值：24
- **PrefixTlvBlockCount** (*int*) -- 要生成的前缀的数量, 类型为：number, 取值范围：0-32, 默认值：1
- **AF** (*str*) -- 起始地址前缀, 类型为：string, 默认值：IPv4Unicast, 取值范围：

  IPv4Unicast
- **ExtendedPrefixFlags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NoneFlag, 支持选项有：

  NoneFlag

  AttachFlag

  NodeFlag

返回 OSPFv2 Ext Prefix Tlv 对象, 类型：object

返回类型 (Ospfv2ExtPrefixTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Prefix Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Ext Prefix Range Tlv | Session=${Session} |␣
↪OpaqueExtendedPrefixLsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_extended_link_tlv**(*OpaqueExtendedLinkLsa*, ***kwargs*)

创建 OSPFv2 Extended link Tlv 对象

参数 **OpaqueExtendedLinkLsa** (Ospfv2OpaqueSrExtLinkLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Extended link Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LinkType** (*str*) -- 前缀的地址族, 类型为：string, 默认值：P2P, 取值范围：

  P2P

  TRANSITNETWORK

  STUBNETWORK

  VIRTUALLINK
- **LinkId** (*str*) -- 起始地址前缀, 类型为：string, 默认值：0.0.0.0, 取值范围：有效的 ipv4 地址
- **LinkData** (*str*) -- 起始地址前缀, 类型为：string, 默认值：0.0.0.0, 取值范围：有效的 ipv4 地址

返回 OSPFv2 Extended link Tlv 对象, 类型：object

返回类型 (Ospfv2ExtendedLinkTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} | Age=20␣
↪|
| Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_external_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 External LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 External LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1

- **LsType** (*str*) -- LSA 类型, 类型为：string, 默认值：ExtLsaLsType1, 支持选项有：

  ExtLsaLsType1：AS-External(5)

  ExtLsaLsType2：NSSA(7)

- **RouteCount** (*int*) -- 路由个数, 类型为：number, 取值范围：1-1000000, 默认值：1
- **StartNetworkPrefix** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：192.0.1.0
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-32, 默认值：24
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **MetricType** (*str*) -- 选项, 类型为：string, 默认值：ExtLsaLsMetricType1, 支持选项有：

  ExtLsaLsMetricType1

  ExtLsaLsMetricType2

- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：10
- **ForwardingAddress** (*str*) -- 转发地址, 即：LSA 中携带的转发地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：192.0.1.0
- **RouterTag** (*int*) -- 路由标签, 类型为：number, 取值范围：0-2147483647, 默认值：0
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **LsaAutomaticConversion** (*bool*) -- LSA 自动转换, 即：当配置的会话为 NSSA 会话时,Renix 会自动将此处配置的外部 LSA 转换为 NSSA-LSA 进行发送; 当配置的会话为非 NSSA 会话时,Renix 会自动将此处配置的 NSSA-LSA 转换为外部 LSA 进行发送, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 External LSA 对象, 类型：object

返回类型 (Ospfv2ExternalLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf External Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_lan_adj_sid_sub_tlv**(*Session*, *ExtendedLinkTlv*, ***kwargs*)

创建 OSPFv2 Lan Adj Sid Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **ExtendedLinkTlv** (Ospfv2ExtendedLinkTlvConfig) -- OSPFv2 Extended Link Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Lan Adj Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- 选项, 类型为：list, 默认值：['ValueIndexFlag', 'LocalGlobalFlag', 'NONE'], 支持选项有：

  BackupFlag

  ValueIndexFlag

  LocalGlobalFlag

  GroupFlag

  PersistentFlag

  NONE
- **MultiTopologyId** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：0-255, 默认值：0
- **Weight** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：0-255, 默认值：0
- **NeighborId** (*str*) -- 起始地址前缀, 类型为：string, 默认值：0.0.0.0, 取值范围：有效的 ipv4 地址
- **SidLabel** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：1-255, 默认值：1

返回 OSPFv2 Lan Adj Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv2LanSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} | Age=20␣
↪|
| ${Tlv} | | Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa} |
| Create Ospf Lan Adj Sid Sub Tlv | Session=${Session} | Tlv=${Tlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_network_atch_router**(*NetworkLsa*, ***kwargs*)

创建 OSPFv2 Network LSA Atch Router 对象

参数 **NetworkLsa** (Ospfv2NetworkLsaConfig) -- 测试仪表 OSPFv2 Network LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Network LSA Atch Router 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AttachedRouter** (*str*) -- 附加路由器的 IP 地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0

返回 OSPFv2 Network LSA Atch Router 对象, 类型：object

返回类型 (Ospfv2NetworkAtchRouterConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${NetworkLsa} | Create Ospf Network Lsa | Session=${Session} | Age=20 |
| Create Ospf Network Lsa Atch Router | NetworkLsa=${NetworkLsa} |␣
↪Metric=65535 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_network_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 Network LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Network LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **DrIpAddress** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-32, 默认值：24
- **Options** (*list*) -- 选项, 类型为：list, 默认值：NONTBIT | EBIT, 支持选项有：

  NONTBIT

  TOSBIT

EBIT

MCBIT

NPBIT

EABIT

DCBIT

OBIT

DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Network LSA 对象, 类型：object

返回类型 (Ospfv2NetworkLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Network Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_opaque_extended_link_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 Opaque Extended Link LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 Opaque Extended Link LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Scope** (*str*) -- Tlv 类型, 类型为：string, 默认值：AreaLocal, 支持选项有：

  LinkLocal

  AreaLocal

  AreaSystemWide
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：192.0.0.1
- **Instance** (*int*) -- 实例, 类型为：number, 取值范围：0-16777215, 默认值：1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

EABIT

DCBIT

OBIT

DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Opaque Extended Link LSA 对象, 类型：object

返回类型 (Ospfv2OpaqueSrExtLinkLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Opaque Extended Link LSA | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_opaque_extended_prefix_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 Opaque Extended Prefix LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Opaque Extended Prefix LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Scope** (*str*) -- LSA 的泛洪区域, 类型为：string, 默认值：AreaLocal, 取值范围：

  LinkLocal

  AreaLocal

  AreaSystemWide

- **AdvertisingRouterId** (*str*) -- 通告路由器 ID, 类型为：string, 默认值：192.0.0.1, 取值范围：有效的 ipv4 地址
- **Instance** (*int*) -- 指定 LSA 中 Instance 字段的值, 类型为：number, 默认值：1, 取值范围：0-16777215
- **Options** (*list*) -- 类型为：list, 默认值：['NONTBIT', 'EBIT'], 取值范围：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

OBIT

DNBIT

- **Age** (*int*) -- LSA 的老化时间。单位为秒, 类型为：number, 默认值：1, 取值范围：0-3600
- **SequenceNumber** (*int*) -- LSA 的序列号, 类型为：number, 默认值：0x80000001, 取值范围：0-4294967295
- **Checksum** (*bool*) -- LSA 的校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Opaque Extended Prefix LSA 对象, 类型：object

返回类型 (OspfV2OpaqueSrExtPrefixLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Opaque Extended Prefix Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_opaque_router_info_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 Opaque Router Info LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Opaque Router Info LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Opaque Router Info LSA 对象, 类型：object

返回类型 (Ospfv2OpaqueRouterInfoLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_prefix_sid_sub_tlv**(*Session*, *Tlv*, ***kwargs*)

创建 OSPFv2 Prefix Sid Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **Tlv** (Ospfv2ExtPrefixTlvConfig) -- OSPFv2 Ext Prefix Range Tlv / Ospfv2 Ext Prefix Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Prefix Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PrefixSidTlvFlags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NoPhp, 取值范围：

  NoPhp

  MappingServer

  ExplicitNull

  ValueIndex

  LacalGlobal
- **MultiTopologyId** (*int*) -- 指定 MT-ID 的值, 类型为：number, 取值范围：0-255, 默认值：0
- **Algorithm** (*int*) -- 计算到其他节点/前缀的可达信息的算法, 类型为：number, 默认值：0
- **SidIndexLabel** (*int*) -- Flags 中包含 Value/Index 时，指定标签值; Flags 中不包含 Value/Index 时，指定 SID/Label 范围内的标签偏移值, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **SidIndexLabelStep** (*int*) -- SidIndexLabel 跳变, 类型为：number, 取值范围：0-4294967295, 默认值：1

返回 OSPFv2 Prefix Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv2PrefixSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| Create Ospf Prefix Sid Sub Tlv | Session=${Session} | Tlv=${Tlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_router_info_capability_tlv**(*OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Router Info Capability Tlv 对象

参数 **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Router Info Capability Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **InfoCapability** (*int*) -- 指定 TLV 值, 类型为：number, 默认值：1, 取值范围：0-255

返回 OSPFv2 Router Info Capability Tlv 对象, 类型：object

返回类型 (Ospfv2RouterInfoCapabilityTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| Create Ospf Router Info Capability Tlv | OpaqueRouterInfoLsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_router_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 Router LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Router LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：192.0.0.1
- **RouterType** (*list*) -- 路由器类型, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  ABR

  ASBR

  VLE

- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

TOSBIT

EBIT

MCBIT

NPBIT

EABIT

DCBIT

OBIT

DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Router LSA 对象, 类型：object

返回类型 (Ospfv2RouterLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Router Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_router_lsa_link**(*RouterLsa*, ***kwargs*)

创建 OSPFv2 Router LSA Link 对象

参数 **RouterLsa** (Ospfv2RouterLsaConfig) -- 测试仪表 OSPFv2 Router LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Router LSA Link 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LinkType** (*str*) -- 链路类型, 类型为：string, 默认值：P2P, 支持选项有：

  P2P

  TRANSITNETWORK

  STUBNETWORK

  VIRTUALLINK

- **LinkId** (*str*) -- 链路状态 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **LinkData** (*str*) -- 链路数据, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：0-65535, 默认值：1

返回 OSPFv2 Router LSA Link 对象, 类型：object

返回类型 (Ospfv2RouterLsaLinksConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${RouterLsa} | Create Ospf Router Lsa | Session=${Session} | Age=20 |
| Create Ospf Router Lsa Link | RouterLsa=${RouterLsa} | Metric=65535 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_sid_label_binding_sub_tlv**(*Session*, *Tlv*, ***kwargs*)

创建 OSPFv2 Sid Label Binding Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **Tlv** (Ospfv2ExtPrefixTlvConfig) -- OSPFv2 Ext Prefix Range Tlv / Ospfv2 Ext Prefix Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sid Label Binding Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SidLabelBindingTlvFlags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NoneFlag, 取值范围：

  NoneFlag

  MirroringContext
- **Weight** (*int*) -- 进行负载均衡时的权重, 类型为：number, 取值范围：0-255, 默认值：0
- **MultiTopologyId** (*int*) -- 指定 MT-ID 的值, 类型为：number, 取值范围：0-255, 默认值：0
- **SidLabelType** (*str*) -- 标识符（SID 或者标签）, 类型为：string, 默认值：Bit20, 取值范围：

  Bit20

  Bit32
- **SidLabel** (*int*) -- SID/Label Type 为 20-Bit Label 时，指定标签值; SID/Label Type 为 32-Bit SID 时，指定 SID, 类型为：number, 取值范围：0-255, 默认值：16

返回 OSPFv2 Sid Label Binding Sub Tlv 对象, 类型：object

返回类型 (Ospfv2SidLabelBindingSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| Create Ospf Sid Label Binding Sub Tlv | Session=${Session} | Tlv=${Tlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_sr_algorithm_tlv**(*OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Algorithm Tlv 对象

参数 **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Algorithm Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Algorithms** (*int*) -- 类型为：number, 默认值：0

返回 OSPFv2 Sr Algorithm Tlv 对象, 类型：object

返回类型 (Ospfv2SrAlgorithmTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| Create Ospf Sr Algorithm Tlv | OpaqueRouterInfoLsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_sr_fad_tlv**(*Session*, *OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Fad Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Fad Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **FlexAlgo** (*int*) -- 灵活算法 ID, 类型为：number, 默认值：128, 取值范围：128-255
- **MetricType** (*str*) -- 指定算路使用的度量类型, 类型为：str, 默认值：IGP_METRIC, 取值范围：

  IGP_METRIC

  MIN_LINK_DELAY

  TE_METRIC
- **CalcType** (*int*) -- 指定特定 IGP 算法的计算类型, 类型为：number, 默认值：0, 取值范围：0-127
- **Priority** (*int*) -- 指定该 Sub-TLV 的优先级, 类型为：number, 默认值：0, 取值范围：0-255
- **FlexAlgoSubTlv** (*list*) -- 选择灵活算法路径计算要遵循的约束条件, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  EXCLUDE_ADMIN

  INCLUDE_ANY_ADMIN

  INCLUDE_ALL_ADMIN

DEFINITION_FLAGS

EXCLUDE_SRLG

- **ExcludeAdmin** (*int*) -- 类型为：number，默认值：0，取值范围：0-4294967295
- **IncludeAnyAdmin** (*int*) -- 类型为：number，默认值：0，取值范围：0-4294967295
- **IncludeAllAdmin** (*int*) -- 类型为：number，默认值：0，取值范围：0-4294967295
- **DefinitionFlags** (*list*) -- 类型为：hex int，默认值：0x80，取值范围：0-FF
- **ExcludeSRLG** (*list*) -- 类型为：hex int，默认值：0x10020000，取值范围：0-4294967295

返回 OSPFv2 Sr Fad Tlv 对象，类型：object

返回类型 (Ospfv2FlexAlgoDefinitionTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| Create Ospf Sr Fad Tlv | Session=${Session} | OpaqueRouterInfoLsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_sr_fapm_sub_tlv**(*Tlv*, ***kwargs*)

创建 OSPFv2 Sr Fapm Sub Tlv 对象

参数 **Tlv** (Port) -- OSPFv2 Ext Prefix Range Tlv / Ospfv2 Ext Prefix Tlv 对象，类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Fapm Sub Tlv 的名称，类型为：string
- **Enable** (*bool*) -- 使能，类型为：bool，取值范围：True 或 False，默认值：True
- **Algorithm** (*int*) -- 灵活算法 ID，类型为：number，取值范围：128-255，默认值：128
- **Flags** (*int*) -- 单字节值，类型为：number，取值范围：0-255，默认值：0
- **Metric** (*int*) -- 度量值，类型为：number，取值范围：0-4294967295，默认值：0

返回 OSPFv2 Sr Fapm Sub Tlv 对象，类型：object

返回类型 (Ospfv2SrFapmSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| Create Ospf Sr Fapm Sub Tlv | Tlv=${Tlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_sr_link_msd_sub_tlv**(*Session*, *ExtendedLinkTlv*, ***kwargs*)

创建 OSPFv2 Sr Link Msd Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **ExtendedLinkTlv** (Ospfv2ExtendedLinkTlvConfig) -- OSPFv2 Extended Link Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Link Msd Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：UNKNOWN, 支持选项有：

  UNKNOWN

  MAX_SEG_LEFT

  MAX_END_POP

  MAX_T_INSERT

  MAX_T_ENCAPS

  MAX_END_D
- **MaxSegmentLeft** (*int*) -- 指定接收报文的 SRH 中 SL（Segment Left）字段的最大值, 类型为：number, 取值范围：0-255, 默认值：8
- **MaxEndPop** (*int*) -- 指定 SRH 栈的顶端 SRH 中 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaxInsert** (*int*) -- 指定执行 T.Insert 行为时可包含 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaxEncap** (*int*) -- 指定执行 T.Encap 行为时可包含 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaxEndD** (*int*) -- 指定执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8

返回 OSPFv2 Sr Link Msd Sub Tlv 对象, 类型：object

返回类型 (Ospfv2SrLinkMsdSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} | Age=20␣
↪|
| ${Tlv} | | Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa} |
| Create Ospf Sr Link Msd Sub Tlv | Session=${Session} | Tlv=${Tlv} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_sr_node_msd_tlv**(*Session*, *OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Node Msd Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Node Msd Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- TLV 中的标志位, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  MAX_SEG_LEFT

  MAX_END_POP

  MAX_T_INSERT

  MAX_T_ENCAPS

  MAX_END_D
- **MaxSegmentLeft** (*int*) -- 在应用与 SID 关联的 SRv6 Endpoint Function 指令之前，指定接收报文的 SRH 中 SL（Segment Left）字段的最大值, 类型为：number, 默认值：0, 取值范围：0-255
- **MaxEndPop** (*int*) -- 指定 SRH 栈的顶端 SRH 中 SID 的最大数量, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxInsert** (*int*) -- 指定执行 T.Insert 行为时可包含 SID 的最大数量, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxEncap** (*int*) -- 指定执行 T.Encap 行为时可包含 SID 的最大数量, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 类型为：number, 默认值：8, 取值范围：0-255

返回 OSPFv2 Sr Node Msd Tlv 对象, 类型：object

返回类型 (Ospfv2SrNodeMsdTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| Create Ospf Sr Node Msd Tlv | Session=${Session} | OpaqueRouterInfoLsa=$
↪{Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_sr_sid_label_range_tlv**(*OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Sid Label Range Tlv 对象

参数 **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Sid Label Range Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SidLabelType** (*str*) -- 类型为：str, 默认值：Bit20, 取值范围：

  Bit20

Bit32

- **SidLabelBase** (*int*) -- SID/Label Type 为 20-Bit Label 时，指定起始标签; SID/Label Type 为 32-Bit SID 时，指定起始 SID, 类型为：number, 默认值：0, 取值范围：1-4294967295
- **SidLabelRange** (*int*) -- 指定要创建的 SID/标签的数量, 类型为：number, 默认值：0, 取值范围：1-16777215

返回 OSPFv2 Sr Sid Label Range Tlv 对象, 类型：object

返回类型 (Ospfv2SidLabelRangeTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| Create Ospf Sr Sid Label Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_sr_srms_preference_tlv**(*OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Srms Preference Tlv 对象

参数 **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Srms Preference Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Preference** (*int*) -- 指定路由器作为 SR Mapping Server 的优先级, 类型为：number, 默认值：1, 取值范围：0-255

返回 OSPFv2 Sr Srms Preference Tlv 对象, 类型：object

返回类型 (Ospfv2SrmsPreferenceTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
| Create Ospf Sr Srms Preference Tlv | OpaqueRouterInfoLsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_summary_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 Summary LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Summary LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1

- **RouteCount** (*int*) -- 路由个数, 类型为：number, 取值范围：1-1000000, 默认值：1
- **StartNetworkPrefix** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：192.0.1.0
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-32, 默认值：24
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Summary LSA 对象, 类型：object

返回类型 (Ospfv2SummaryLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Summary Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**create_ospf_te_lsa**(*Session*, ***kwargs*)

创建 OSPFv2 Te LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Te LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1

- **TlvType** (*str*) -- Tlv 类型, 类型为：string, 默认值：LsaLink, 支持选项有：

  LsaRouter

  LsaLink

- **RouterId** (*str*) -- 路由器 ID, 类类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **LinkId** (*str*) -- Link ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **LinkType** (*str*) -- Link 类型, 类型为：string, 默认值：LsaLink, 支持选项有：

  LinkP2P

  LinkMultiaccess

- **Instance** (*int*) -- 实例, 类型为：number, 取值范围：0-16777215, 默认值：1
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：0-16777215, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Te LSA 对象, 类型：object

返回类型 (Ospfv2TeLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Te Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv2.**edit_ospf**(*Session*, ***kwargs*)

编辑 OSPFv2 协议会话对象参数

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- OSPFv2 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 OSPFv2 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AreaId** (*str*) -- 区域 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 取值范围：Broadcast 或 P2P, 默认值：Broadcast
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **Cost** (*int*) -- 接口开销, 类型为：number, 取值范围：1-65535, 默认值：10
- **AuthenticationType** (*str*) -- 类型为：string, 取值范围：None Simple 或 MD5, 默认值：None
- **Password** (*str*) -- 密码, 类型为：string, 默认值：Xinertel
- **Md5KeyId** (*int*) -- MD5 密钥, 类型为：number, 取值范围：0-255, 默认值：1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT
- **EnableOspfv2Mtu** (*bool*) -- 使能 OSPF MTU, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GracefulRestartReason** (*str*) -- 平滑重启原因, 类型为：string, 默认值：UNKNOWN, 支持的原因：

  UNKNOWN

  SOFTWARE

  RELOADORUPGRADE

  SWITCH
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **HelloInterval** (*int*) -- Hello 包间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **RouterDeadInterval** (*int*) -- 路由器失效间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：40

- **LsaRetransInterval** (*int*) -- LSA 重传间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：5
- **LsaRefreshTime** (*int*) -- LSA 刷新间隔 (秒), 类型为：number, 取值范围：1-1800, 默认值：1800
- **EnableSrManagement** (*bool*) -- 启用 SR, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Ospf | Session=${Session} |
```

TesterLibrary.Protocol.ospfv2.**edit_ospf_port_config**(*Ports*, ***kwargs*)

修改 Ospf 端口统计对象

参数 **Ports** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **TransmitRate** (*int*) -- OSPFv2 Message Tx Rate (messages/second), 取值范围：1-9000, 默认值：100
- **SessionOutstanding** (*int*) -- OSPFv2 Session Outstanding, 取值范围：1-1000, 默认值：20
- **UpdateMsgTransmitRate** (*int*) -- Deprecated. OSPFv2 Update Message Tx Rate (messages/second), 取值范围：1-9000, 默认值：10
- **EnableLoop** (*bool*) -- Enable Loop Back, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Ospf Port Config | Ports=${Ports} | TransmitRate=100 |
```

TesterLibrary.Protocol.ospfv2.**edit_ospf_te_lsa_link**(*TeLsa*, ***kwargs*)

编辑 OSPFv2 Te LSA Link 参数

参数 **TeLsa** (Ospfv2TeLsaConfig) -- OSPFv2 Te LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Te LSA Link 对象的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableLocalIp** (*bool*) -- 使能本端 IPv4 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LocalIp** (*str*) -- 本端 IPv4 地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableRemoteIp** (*bool*) -- 使能远端 IPv4 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **RemoteIp** (*str*) -- 远端 IPv4 地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableGroup** (*bool*) -- 启动组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Group** (*int*) -- 组 ID, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **EnableMaxBandwidth** (*bool*) -- 启动最大带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MaximumBandwidth** (*int*) -- 最大带宽, 类型为：number, 取值范围：0-16777215, 默认值：1000
- **EnableReservedBandwidth** (*bool*) -- 启动预留带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ReservableBandwidth** (*int*) -- 预留带宽, 类型为：number, 取值范围：0-4294967295, 默认值：1000
- **EnableUnreservedBandwidth** (*bool*) -- 启动未预留带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UnreservedBandwidth0** (*int*) -- 未预留带宽优先级 0, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth1** (*int*) -- 未预留带宽优先级 1, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth2** (*int*) -- 未预留带宽优先级 2, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth3** (*int*) -- 未预留带宽优先级 3, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth4** (*int*) -- 未预留带宽优先级 4, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth5** (*int*) -- 未预留带宽优先级 5, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth6** (*int*) -- 未预留带宽优先级 6, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth7** (*int*) -- 未预留带宽优先级 7, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${TeLsa} | Create Ospf Te Lsa | Session=${Session} | Age=20 |
| Edit Ospf Te Lsa Link| TeLsa=${TeLsa} | LocalIp=2.2.2.2 |
```

TesterLibrary.Protocol.ospfv2.**establish_ospf**(*Sessions*)

建立 OSPFv2 协议会话

参数 **Sessions** (list(OspfRouter)) -- OSPFv2 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Establish Ospf | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ospfv2.**get_ospf_router_from_lsa**(*Lsa*)

获取 OSPF LSA 对应的绑定流源或目的端点对象

参数 **Lsa** (Port) -- 测试仪表 OSPFv2 或 OSPFv3 LSA 对象, 类型为：object

返回 OSPFv2 或 OSPFv3 LSA 对应的绑定流源或目的端点对象, 类型：object

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${RouterLsa} | Create Ospf Router Lsa | Session=${Session} | Age=20 |
| ${Point} | Get Ospf Router From Lsa | Lsa=${RouterLsa} |
```

TesterLibrary.Protocol.ospfv2.**get_ospf_statistic**(*Session*, *StaItems=None*)

获取 OSPFv2 协议会话统计结果

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  RouterState

  AdjacencyState

  TxHello

  RxHello

  TxDd

  RxDd

  TxRouterLsa

  RxRouterLsa

  TxNetworkLsa

  RxNetworkLsa

  TxSummaryLsa

  RxSummaryLsa

  TxAsbrSummaryLsa

  RxAsbrSummaryLsa

  TxAsExternalLsa

  RxAsExternalLsa

  TxNssaLsa

  RxNssaLsa

  TxTeLsa

  RxTeLsa

  TxOpaqueRouterInfoLsa

  RxOpaqueRouterInfoLsa

TxOpaqueExtendedPrefixLsa

RxOpaqueExtendedPrefixLsa

TxOpaqueExtendedLinkLsa

RxOpaqueExtendedLinkLsa

TxRequest

RxRequest

TxUpdate

RxRequest

TxAck

RxAck

返回

eg:

```
{
    'AdjacencyState': 'Full',
    'TxUpdate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv2SessionResultPropertySet |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Ospf Statistic | Session=${Session} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.ospfv2.**grace_restart_ospf**(*Sessions*)

平滑重启 OSPFv2 协议会话

参数 **Sessions** (list(OspfRouter)) -- OSPFv2 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Grace Restart Ospf | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ospfv2.**wait_ospf_adjacency_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 OSPFv2 或 OSPFv3 协议会话达到指定邻接状态

参数

- **Sessions** (list(OspfRouter)) or (list(Ospfv3Router)) -- OSPFv2 或 OSPFv3 协议会话对象列表, 类型为：list

- **State** (*str*) -- 等待 OSPFv2 协议会话达到的邻接状态, 类型为：string, 默认值：FULL, 支持下列状态：

  DOWN

  INIT

  TWOWAY

  EXSTART

  EXCHANGE

  LOADING

  FULL

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话邻接状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ospf Adjacency State | Sessions=${Sessions} | State=FULL | Interval=2␣
↪| TimeOut=120 |
```

TesterLibrary.Protocol.ospfv2.**wait_ospf_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 OSPFv2 或 OSPFv3 协议会话达到指定状态

参数

- **Sessions** (list(OspfRouter)) or (list(Ospfv3Router)) -- OSPFv2 或 OSPFv3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 OSPFv2 协议会话达到的状态, 类型为：string, 默认值：达到 DR 或 BACKUP 或 DROTHER, 支持下列状态：

  NOTSTART

  P2P

  WAITING

  DR

  BACKUP

  DROTHER

  DISABLE

  DOWN

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ospf State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.ospfv2.**withdraw_ospf_lsa**(*Sessions=None*, *Type=None*, *Lsa=None*)

撤销 OSPFv2 协议会话 lsa

参数

- **Sessions** (list(OspfRouter)) -- OSPFv2 协议会话对象列表, 类型为：list
- **Type** (*str*) -- OSPFv2 lsa 类型, 类型为：string, 支持的 lsa 类型：

  Router

  Network

  Summary

  AsbrSummary

  External
- **Lsa** (*list*) -- OSPFv2 lsa 列表, 类型为：list, 当 Type=None 时参数生效

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Ospf Lsa | Sessions=${Sessions} | Type=router |
| Withdraw Ospf Lsa | Sessions=${Sessions} | Lsa=${Lsas} |
```

TesterLibrary.Protocol.ospfv3 module

TesterLibrary.Protocol.ospfv3.**advertise_ospfv3_lsa**(*Sessions=None*, *Type=None*, *Lsa=None*)

通告 OSPFv3 协议会话 lsa

参数

- **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list
- **Type** (*str*) -- OSPFv3 lsa 类型, 类型为：string, 支持的 lsa 类型：

  Router

  Network

  InterAreaPrefix

  InterAreaRouter

  AsExternal

  Link
- **Lsa** (*int*) -- OSPFv3 lsa 列表, 类型为：list, 当 Type=None 时参数生效

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Ospfv3 Lsa | Sessions=${Sessions} | Type=router |
| Advertise Ospfv3 Lsa | Sessions=${Sessions} | Lsa=${Lsas} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3**(*Port*, ***kwargs*)

创建 OSPFv3 协议会话对象

参数 Port (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- OSPFv3 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 OSPFv3 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **InstanceId** (*int*) -- 实例 ID, 类型为：number, 取值范围：0-255, 默认值：0
- **AreaId** (*str*) -- 区域 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableExtendedLsa** (*bool*) -- 使能扩展 LSA, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ExtendedLsaMode** (*str*) -- 扩展 LSA 模式, 类型为：string, 默认值：Full, 取值范围：

  NONE

  MixedModeOriginateOnly

  MixedModeOriginateSPF

  Full

- **AreaExtendedLsaMode** (*str*) -- 扩展区域 LSA 模式, 类型为：string, 默认值：InheritGlobal, 取值范围：

  InheritGlobal

  NONE

  MixedModeOriginateOnly

  MixedModeOriginateSPF

  Full

- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 取值范围：Broadcast 或 P2P, 默认值：Broadcast
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **InterfaceId** (*int*) -- 接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：10
- **Cost** (*int*) -- 接口开销, 类型为：number, 取值范围：1-65535, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'V6BIT', 'EBIT', 'RBIT'], 支持选项有：

  NONTBIT

  V6BIT

EBIT

MCBIT

NBIT

RBIT

DCBIT

Unused17

Unused16

Unused15

Unused14

Unused13

Unused12

Unused11

Unused10

Unused9

Unused8

Unused7

Unused6

Unused5

Unused4

Unused3

Unused2

Unused1

Unused0

- **EnableOspfv3Mtu** (*bool*) -- 使能 OSPFv3 MTU, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GracefulRestartReason** (*str*) -- 平滑重启原因, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  SOFTWARE

  RELOADORUPGRADE

  SWITCH
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **HelloInterval** (*int*) -- Hello 包间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **RouterDeadInterval** (*int*) -- 路由器失效间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：40
- **LsaRetransInterval** (*int*) -- LSA 重传间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：5

- **LsaRefreshTime** (*int*) -- LSA 刷新间隔 (秒), 类型为：number, 取值范围：1-1800, 默认值：1800

返回 OSPFv3 协议会话对象, 类型：object

返回类型 (Ospfv3Router)

实际案例

```
| Create Ospfv3 | Port=${Port} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_as_external_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 As External LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 As External LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **PrefixCount** (*int*) -- 前缀个数, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-128, 默认值：64
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  NUBIT

  LABIT

  MCBIT

  PBIT

  DNBit

  NBit

  Unused1

  Unused0

- **IsExternalMetric** (*bool*) -- 是否外部度量值, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1

- **EnableForwardingAddress** (*bool*) -- 使能转发地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ForwardingAddress** (*str*) -- 转发地址, 即：LSA 中携带的转发地址, 类型为：string, 取值范围：IPv6 地址, 默认值：'::'
- **AdminTag** (*int*) -- 管理标签, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ReferencedLsType** (*int*) -- 参考链路状态类型, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0x0
- **ReferencedLinkStateId** (*int*) -- 参考链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **LsaAutomaticConversion** (*bool*) -- LSA 自动转换, 即：当配置的会话为 NSSA 会话时,Renix 会自动将此处配置的外部 LSA 转换为 NSSA-LSA 进行发送; 当配置的会话为非 NSSA 会话时,Renix 会自动将此处配置的 NSSA-LSA 转换为外部 LSA 进行发送, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 As External LSA 对象, 类型：object

返回类型 (Ospfv3AsExternalLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 As External Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_bier_mpls_encap_sub_tlv**(*SubTlv*, ***kwargs*)

创建 OSPFv3 Bier Mpls Encap Sub Tlv 对象

参数 **SubTlv** (Ospfv3BierSubTlvConfig) -- Ospfv3 Bier Sub Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Bier Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TlvType** (*int*) -- Type 字段值, 类型为：number, 取值范围：0-255, 默认值：10
- **MaxSi** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：1-255, 默认值：1
- **Label** (*int*) -- 标签范围中的起始标签值, 类型为：number, 取值范围：0-1048575, 默认值：100
- **BsLen** (*int*) -- 本地比特串的长度, 类型为：number, 取值范围：0-15, 默认值：4

返回 Ospfv3 Bier Mpls Encap Sub Tlv 对象, 类型：object

返回类型 (Ospfv3BierMplsEncapSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Inter Area Prefix Lsa | Session=${Session} | Age=20 |
| ${SubTlv} | Create Ospfv3 Bier Sub Tlv | Lsa=${Lsa} |
| Create Ospfv3 Bier Mpls Encap Sub Tlv | SubTlv=${SubTlv} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_bier_sub_tlv**(*Lsa*, ***kwargs*)

创建 OSPFv3 Bier Sub Tlv 对象

参数 **Lsa** (Ospfv3InterAreaRouterLsaConfig) -- Ospfv3 Inter Area Router LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Bier Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TlvType** (*int*) -- Type 字段值, 类型为：number, 取值范围：0-255, 默认值：9
- **SubDomainId** (*int*) -- BIER 子域 ID, 类型为：number, 取值范围：1-255, 默认值：1
- **MtId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-255, 默认值：1
- **BfrId** (*int*) -- BFR（Bit Forwarding Router，比特转发路由器）ID, 类型为：number, 取值范围：1-65535, 默认值：1
- **Bar** (*int*) -- BIER 算法, 类型为：number, 取值范围：0-255, 默认值：0
- **Ipa** (*int*) -- IGP 算法, 类型为：number, 取值范围：0-255, 默认值：0

返回 Ospfv3 Bier Sub Tlv 对象, 类型：object

返回类型 (Ospfv3BierSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Inter Area Prefix Lsa | Session=${Session} | Age=20 |
| Create Ospfv3 Bier Sub Tlv | Lsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_endx_sid_structure_sub_tlv**(*SubTlv*, ***kwargs*)

创建 OSPFv3 Endx Sid Structure Sub Tlv 对象

参数 **SubTlv** (Ospfv3Srv6EndXSidSubTlvConfig) -- OSPFv3 Srv6 EndX Sid Sub Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Link Msd Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LbLength** (*int*) -- SRv6 SID Locator Block 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **LnLength** (*int*) -- SRv6 SID Locator Node 长度, 类型为：number, 取值范围：0-128, 默认值：32

- **FunctionLength** (*int*) -- SRv6 SID Function 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **ArgumentLength** (*int*) -- SRv6 SID Argument 长度, 类型为：number, 取值范围：0-128, 默认值：32

返回 Ospfv3 Endx Sid Structure Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6SidStructureSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| ${SubTlv} | Create Ospfv3 Srv6 Msd Sid Sub Tlv | RouterLsaLink=${LsaLink} |
| Create Ospfv3 Endx Sid Structure Sub Tlv | SubTlv=${SubTlv} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_inter_area_prefix_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Inter Area Prefix LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Inter Prefix LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ExtendedLsaTlvs** (*list*) -- 扩展 LSA TLVs, 类型为：list, 默认值：Ipv6InterAreaPrefix, 支持选项有：

  NONE

  Ipv6InterAreaPrefix
- **PrefixCount** (*int*) -- 前缀个数, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-128, 默认值：64
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  NUBIT

  LABIT

MCBIT

PBIT

- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Ospfv3 Inter Area Prefix LSA 对象, 类型：object

返回类型 (Ospfv3InterAreaPrefixLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Inter Area Prefix Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_inter_area_router_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Inter Area Router LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Router LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **AsbrId** (*str*) -- ASBR ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：NONTBIT | V6BIT | EBIT, 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

Unused16

Unused15

Unused14

Unused13

Unused12

Unused11

Unused10

Unused9

Unused8

Unused7

Unused6

Unused5

Unused4

Unused3

Unused2

Unused1

Unused0

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Ospfv3 Inter Area Router LSA 对象, 类型：object

返回类型 (Ospfv3InterAreaRouterLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Inter Area Router Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_intra_area_prefix_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Intra Area Prefix LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Intra Prefix LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

- **ReferencedLsType** (*str*) -- 参考 LS 类型, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0x0
- **ReferencedAdvertisingRouterId** (*str*) -- 参考通告路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **ReferencedLinkStateId** (*int*) -- 参考链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ExtendedLsaTlvs** (*list*) -- 扩展 LSA TLVs, 类型为：list, 默认值：Ipv6IntraAreaPrefix, 支持选项有：

  NONE

  Ipv6IntraAreaPrefix
- **PrefixCount** (*int*) -- 前缀个数, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-128, 默认值：64
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  NUBIT

  LABIT

  MCBIT

  PBIT

  DNBit

  NBit

  Unused1

  Unused0
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Ospfv3 Intra Area Prefix LSA 对象, 类型：object

返回类型 (Ospfv3IntraAreaPrefixLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Intra Area Prefix Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_lan_endx_sid_structure_sub_tlv**(*SubTlv*, ***kwargs*)

创建 OSPFv3 Lan Endx Sid Structure Sub Tlv 对象

参数 **SubTlv** (Ospfv3Srv6LanEndXSidSubTlvConfig) -- OSPFv3 Srv6 Lan EndX Sid Sub Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Link Msd Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LbLength** (*int*) -- SRv6 SID Locator Block 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **LnLength** (*int*) -- SRv6 SID Locator Node 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **FunctionLength** (*int*) -- SRv6 SID Function 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **ArgumentLength** (*int*) -- SRv6 SID Argument 长度, 类型为：number, 取值范围：0-128, 默认值：32

返回 Ospfv3 Lan Endx Sid Structure Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6LanEndXSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| ${SubTlv} | Create Ospfv3 Srv6 Msd Sid Sub Tlv | RouterLsaLink=${LsaLink} |
| Create Ospfv3 Lan Endx Sid Structure Sub Tlv | SubTlv=${SubTlv} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_link_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Link LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Link LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

- **ExtendedLsaTlvs** (*list*) -- 扩展 LSA TLVs, 类型为: list, 默认值: Ipv6IntraAreaPrefix, 支持选项有:

  NONE

  Ipv6IntraAreaPrefix

  Ipv6LinkLocalAddr

  Ipv4LinkLocalAddr

- **PrefixCount** (*int*) -- 前缀个数, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为: string, 取值范围: IPv6 地址, 默认值: 2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为: number, 取值范围: 1-128, 默认值: 64
- **Increment** (*int*) -- 步长, 类型为: number, 取值范围: 0-4294967295, 默认值: 1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为: list, 默认值: NONTBIT, 支持选项有:

  NONTBIT

  NUBIT

  LABIT

  MCBIT

  PBIT

  DNBit

  NBit

  Unused1

  Unused0

- **LinkLocalInterfaceAddress** (*str*) -- 本地链路接口地址, 类型为: string, 取值范围: IPv6 地址, 默认值: fe80::1
- **Ipv4LinkLocalInterfaceAddress** (*str*) -- 本地链路接口地址, 类型为: string, 取值范围: IPv4 地址, 默认值: 0.0.0.0
- **RouterPriority** (*int*) -- 路由优先级, 类型为: number, 取值范围: 1-255, 默认值: 1
- **Options** (*list*) -- 选项, 类型为: list, 默认值: NONTBIT | V6BIT | EBIT, 支持选项有:

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

Unused15

Unused14

Unused13

Unused12

Unused11

Unused10

Unused9

Unused8

Unused7

Unused6

Unused5

Unused4

Unused3

Unused2

Unused1

Unused0

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Link LSA 对象, 类型：object

返回类型 (Ospfv3LinkLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Link Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_network_atch_router**(*Lsa*, ***kwargs*)

创建 OSPFv3 Network LSA Atch Router 对象

参数 **Lsa** (Ospfv3NetworkLsaConfig) -- 测试仪表 OSPFv3 Network LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Network LSA Atch Router 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AttachedRouter** (*str*) -- 附加路由器的 IP 地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0

返回 OSPFv3 Network LSA Atch Router 对象, 类型：object

返回类型 (Ospfv3NetworkAtchRouterConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Network Lsa | Session=${Session} | Age=20 |
| Create Ospfv3 Network Lsa Atch Router | Lsa=${RouterLsa} | Metric=65535 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_network_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Network LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Network LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

  Unused15

  Unused14

  Unused13

  Unused12

  Unused11

  Unused10

  Unused9

  Unused8

  Unused7

  Unused6

  Unused5

  Unused4

  Unused3

  Unused2

Unused1

Unused0

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Network LSA 对象, 类型：object

返回类型 (Ospfv3NetworkLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Network Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_nssa_external_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Nssa External LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Nssa External LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ExtendedLsaTlvs** (*list*) -- 扩展 LSA TLVs, 类型为：list, 默认值：Ipv6ExternalPrefix, 支持选项有：

  NONE

  Ipv6ExternalPrefix
- **ExtendedLsaSubTlvs** (*list*) -- 扩展 LSA Sub-TLV, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONE

  Ipv6ForwardingAddr

  Ipv4ForwardingAddr

  RouteTag
- **PrefixCount** (*int*) -- 前缀个数, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-128, 默认值：64

- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  NUBIT

  LABIT

  MCBIT

  PBIT

  DNBit

  NBit

  Unused1

  Unused0
- **IsExternalMetric** (*bool*) -- 是否外部度量值, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **EnableForwardingAddress** (*bool*) -- 使能转发地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ForwardingAddress** (*str*) -- Ipv6 转发地址, 即：LSA 中携带的转发地址, 类型为：string, 取值范围：IPv6 地址, 默认值：'::'
- **Ipv4ForwardingAddress** (*str*) -- IPv4 转发地址, 取值范围：IPv4 地址, 默认值：0.0.0.0
- **AdminTag** (*int*) -- 管理标签, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ReferencedLsType** (*int*) -- 参考链路状态类型, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0x0
- **ReferencedLinkStateId** (*int*) -- 参考链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LsaAutomaticConversion** (*bool*) -- LSA 自动转换, 即：当配置的会话为 NSSA 会话时,Renix 会自动将此处配置的外部 LSA 转换为 NSSA-LSA 进行发送; 当配置的会话为非 NSSA 会话时,Renix 会自动将此处配置的 NSSA-LSA 转换为外部 LSA 进行发送, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Nssa External LSA 对象, 类型：object

返回类型 (Ospfv3NssaExternalLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Nssa External Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_opaque_router_info_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Opaque Router Info LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Opaque Router Info LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Scope** (*str*) -- 起始网络前缀, 类型为：string, 默认值：AreaLocal, 取值范围：

  LinkLocal

  AreaLocal

  AreaSystemWide
- **AdvertisingRouterId** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.0.1
- **LinkStateId** (*int*) -- 路由优先级, 类型为：number, 取值范围：1-255, 默认值：0
- **TlvsFlag** (*list*) -- 前缀选项, 类型为：list, 默认值：['NONTBIT', 'InfoCapabilities'], 支持选项有：

  NONEBIT

  InfoCapabilities

  FuncCapabilities
- **InformationalCapabilities** (*list*) -- 前缀选项, 类型为：list, 默认值：NONEBIT, 支持选项有：

  NONEBIT

  RcBit

  RhBit

  SrsBit

  TesBit

  PolBit

  Etbit

  MiBit

  SrhBit

  Unused8

  Unused9

  Unused10

  Unused11

Unused12

Unused13

Unused14

Unused15

Unused16

Unused17

Unused18

Unused19

Unused20

Unused22

Unused21

Unused23

Unused24

Unused25

Unused26

Unused27

Unused28

Unused29

Unused30

- **FunctionalCapabilities** (*list*) -- 前缀选项，类型为：list，默认值：NONEBIT, 支持选项有：

  NONEBIT

  Unused0

  Unused1

  Unused2

  Unused3

  Unused4

  Unused5

  Unused6

  Unused7

  Unused8

  Unused9

  Unused10

  Unused11

  Unused12

  Unused13

  Unused14

  Unused15

  Unused16

Unused17

Unused18

Unused19

Unused20

Unused22

Unused21

Unused23

Unused24

Unused25

Unused26

Unused27

Unused28

Unused29

Unused30

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Opaque Router Info LSA 对象, 类型：object

返回类型 (Ospfv3OpaqueRouterInfoLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Opaque Router Info LSA | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_router_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Router LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Router LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.0.0.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **RouterType** (*str*) -- 路由器类型, 类型为：string, 默认值：NONTBIT, 支持选项有：

  NONEBIT

  RouterTypeABR

RouterTypeASBR

RouterTypeVirtype

- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

  Unused15

  Unused14

  Unused13

  Unused12

  Unused11

  Unused10

  Unused9

  Unused8

  Unused7

  Unused6

  Unused5

  Unused4

  Unused3

  Unused2

  Unused1

  Unused0

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Router LSA 对象, 类型：object

返回类型 (Ospfv3RouterLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_router_lsa_link**(*RouterLsa*, ***kwargs*)

创建 OSPFv3 Router LSA Link 对象

参数 **RouterLsa** (Ospfv3RouterLsaConfig) -- 测试仪表 OSPFv3 Router LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Router LSA Link 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LinkType** (*str*) -- 链路类型, 类型为：string, 默认值：P2P, 支持选项有：

  P2P

  TRANSITNETWORK

  VIRTUALLINK
- **InterfaceId** (*int*) -- 接口 ID, 即该 ID 用于唯一标识 simulated router 的接口, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **NeighborInterfaceId** (*int*) -- 邻居接口 ID, 即该 ID 用于唯一标识邻居路由器的接口, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **NeighborRouterId** (*str*) -- 邻居路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.0.0.1
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-65535, 默认值：1

返回 OSPFvv3 Router LSA Link 对象, 类型：object

返回类型 (Ospfv3RouterLsaLinksConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} | Metric=65535 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_sr_algorithm_tlv**(*Lsa*, ***kwargs*)

创建 OSPFv3 Sr Algorithm Tlv 对象

参数 **Lsa** (Ospfv3OpaqueRouterInfoLsaConfig) -- Ospfv3 Opaque Router Info LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Sr Algorithm Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Algorithms** (*int*) -- 算法, 类型为：number, 默认值：0

返回 Ospfv3 Sr Algorithm Tlv 对象, 类型：object

返回类型 (Ospfv3SrAlgorithmTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Opaque Router Info LSA | Session=${Session} | Age=20␣
↪|
| Create Ospfv3 Sr Algorithm Tlv | Lsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_sr_fad_tlv**(*Session*, *Lsa*, ***kwargs*)

创建 OSPFv3 Sr Fad Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **Lsa** (Ospfv3OpaqueRouterInfoLsaConfig) -- Ospfv3 Opaque Router Info LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Sr Fad Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **FlexAlgorithm** (*int*) -- 灵活算法 ID, 类型为：number, 取值范围：128-255, 默认值：128
- **MetricType** (*str*) -- 度量类型, 类型为：string, 默认值：IGPMetric, 取值范围：

  IGPMetric

  MinUnidirectionalLinkDelay

  TEDefaultMetric
- **CalculationType** (*int*) -- 特定 IGP 算法的计算类型, 类型为：number, 取值范围：0-127, 默认值：0
- **Priority** (*int*) -- 该 TLV 的优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **FlexAlgorithmSubTlvs** (*list*) -- 灵活算法路径计算要遵循的约束条件, 类型为：list, 默认值：NONEBIT, 支持选项有：

  NONEBIT

  ExcludeAdminGroups

  IncludeAnyAdminGroups

  IncludeAllAdminGroups

  DefinitionFlags

  ExcludeSRLG
- **ExcludeAdminGroups** (*int*) -- 算法, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **IncludeAnyAdminGroups** (*int*) -- 算法, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **IncludeAllAdminGroups** (*int*) -- 算法, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **DefinitionFlags** (*int*) -- 算法, 类型为：number, 取值范围：0-FF, 默认值：0x80

- **ExcludeSRLG** (*int*) -- 算法, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Ospfv3 Sr Fad Tlv 对象, 类型：object

返回类型 (Ospfv3SrFadTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Opaque Router Info LSA | Session=${Session} | Age=20␣
↪|
| Create Ospfv3 Sr Fad Tlv | Lsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_sr_fapm_sub_tlv**(*Lsa*, ***kwargs*)

创建 OSPFv3 Sr Fapm Sub Tlv 对象

参数 **Lsa** (Ospfv3InterAreaRouterLsaConfig) -- Ospfv3 Inter Area Router LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Sr Fapm Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Algorithm** (*int*) -- 灵活算法 ID, 类型为：number, 取值范围：128-255, 默认值：128
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：128-255, 默认值：0

返回 Ospfv3 Sr Fapm Sub Tlv 对象, 类型：object

返回类型 (Ospfv3SrFapmSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Inter Area Prefix Lsa | Session=${Session} | Age=20 |
| Create Ospfv3 Sr Fapm Sub Tlv | Lsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_srv6_capabilities_tlv**(*Session*, *Lsa*, ***kwargs*)

创建 OSPFv3 Srv6 Capabilities Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **Lsa** (Ospfv3OpaqueRouterInfoLsaConfig) -- Ospfv3 Opaque Router Info LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Capabilities Tlv 的名称, 类型为: string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- 类型为：list, 默认值：NONEBIT, 支持选项有：

  NONEBIT

Unused0

OFlag

Unused2

Unused3

Unused4

Unused5

Unused6

Unused7

Unused8

Unused9

Unused10

Unused11

Unused12

Unused13

Unused14

Unused15

返回 Ospfv3 Srv6 Capabilities Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6CapabilitiesTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Opaque Router Info LSA | Session=${Session} | Age=20␣
↪|
| Create Ospfv3 Srv6 Capabilities Tlv | Lsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_srv6_endx_sid_sub_tlv**(*Session*, *RouterLsaLink*, ***kwargs*)

创建 OSPFv3 Srv6 EndX Sid Sub Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **RouterLsaLink** (Ospfv3RouterLsaLinksConfig) -- OSPFv3 Router Lsa LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 EndX Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EndpointBehaviorId** (*int*) -- SRv6 SID 的端点行为 ID, 类型为：number, 默认值：0

- **Flags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NONEBIT, 取值范围：

  NONEBIT

  Unused0

  Unused1

  Unused2

  Unused3

  Unused4

  PersistentFlag

  SetFlag

  BackupFlag

- **Algorithm** (*int*) -- SID 关联的算法, 类型为：number, 取值范围：0-255, 默认值：0
- **Weight** (*int*) -- END.X SID / LAN END.X SID 的权重，用于负载分担, 类型为：number, 取值范围：0-255, 默认值：1
- **Sid** (*str*) -- 通告的 SRv6 SID, 邻居路由器 ID, 类型为：string, 取值范围：有效的 ipv6 地址, 默认值：'aaaa:1:1:1::'

返回 OSPFv3 Srv6 EndX Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6EndXSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| Create Ospfv3 Srv6 Endx Sid Sub Tlv | RouterLsaLink=${LsaLink} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_srv6_lan_endx_sid_sub_tlv**(*Session*, *RouterLsaLink*, ***kwargs*)

创建 OSPFv3 Srv6 Lan EndX Sid Sub Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **RouterLsaLink** (Ospfv3RouterLsaLinksConfig) -- OSPFv3 Router Lsa LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Lan EndX Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EndpointBehaviorId** (*int*) -- SRv6 SID 的端点行为 ID, 类型为：number, 默认值：0

- **Flags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NONEBIT, 取值范围：

  NONEBIT

  Unused0

  Unused1

  Unused2

  Unused3

  Unused4

  PersistentFlag

  SetFlag

  BackupFlag

- **Algorithm** (*int*) -- SID 关联的算法, 类型为：number, 取值范围：0-255, 默认值：0
- **Weight** (*int*) -- END.X SID / LAN END.X SID 的权重，用于负载分担, 类型为：number, 取值范围：0-255, 默认值：1
- **Sid** (*str*) -- 通告的 SRv6 SID, 邻居路由器 ID, 类型为：string, 取值范围：有效的 ipv6 地址, 默认值：'aaaa:1:1:1::'

返回 OSPFv3 Srv6 Lan EndX Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6LanEndXSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| Create Ospfv3 Srv6 Lan Endx Sid Sub Tlv | RouterLsaLink=${LsaLink} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_srv6_link_msd_sub_tlv**(*Session*, *RouterLsaLink*, ***kwargs*)

创建 OSPFv3 Srv6 Link Msd Sub Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **RouterLsaLink** (Ospfv3RouterLsaLinksConfig) -- OSPFv3 Router Lsa LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Link Msd Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Msds** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NONEBIT, 取值范围：

  NONTBIT

MaxiSegmentLeft

MaxiEndPop

MaxiTInsert

MaxiTEncaps

MaxiEndD

- **MaximumEndDSrh** (*int*) -- 接收报文的 SRH 中 SL（Segment Left）字段的最大值, 类型为：number, 取值范围：0-255, 默认值：8
- **MaximumEndPop** (*int*) -- SRH 栈的顶端 SRH 中 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaximumSegmentsLeft** (*int*) -- 执行 T.Insert 行为时可包含 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaximumTEncapSrh** (*int*) -- 执行 T.Encap 行为时可包含 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaximumTInsertSrh** (*int*) -- 执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8

返回 OSPFv3 Srv6 Link Msd Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6MsdSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| Create Ospfv3 Srv6 Msd Sid Sub Tlv | RouterLsaLink=${LsaLink} |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_srv6_location_lsa**(*Session*, ***kwargs*)

创建 OSPFv3 Srv6 Location LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Location LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Scope** (*str*) -- 起始网络前缀, 类型为：string, 默认值：AreaLocal, 取值范围：

  LinkLocal

  AreaLocal

  AreaSystemWide

- **AdvertisingRouterId** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.0.1
- **LinkStateId** (*int*) -- 路由优先级, 类型为：number, 取值范围：1-255, 默认值：0
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0

- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Srv6 Location LSA 对象, 类型：object

返回类型 (Ospfv3Srv6LocatorLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Srv6 Location LSA | Session=${Session} | Age=20 |
```

TesterLibrary.Protocol.ospfv3.**create_ospfv3_srv6_location_tlv**(*Session*, *Lsa*, ***kwargs*)

创建 OSPFv3 Srv6 Location Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **Lsa** (Ospfv3Srv6LocatorLsaConfig) -- Ospfv3 Srv6 Location LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Location Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterType** (*str*) -- 路由器类型, 类型为：string, 默认值：IntraArea, 支持选项有：

  IntraArea

  InterArea

  ASExternal

  NSSAExternal
- **Algorithm** (*int*) -- Locator 关联算法, 类型为：number, 取值范围：0-255, 默认值：0
- **LocatorLength** (*int*) -- Locator 前缀长度, 类型为：number, 取值范围：0-128, 默认值：64
- **Flags** (*list*) -- 类型为：list, 默认值：NONEBIT, 支持选项有：

  NONEBIT

  Unused0

  Unused1

  Unused2

  Unused3

  Unused4

  Unused5

  ABit

  NFlag

- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **Locator** (*str*) -- 通告的 Locator, 类型为：string, 取值范围：有效的 ipv6 地址, 默认值：'aaaa:1:1:1::'

返回 Ospfv3 Srv6 Location Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6LocatorTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Opaque Router Info LSA | Session=${Session} | Age=20␣
↪|
| Create Ospfv3 Srv6 Location Tlv | Lsa=${Lsa} |
```

TesterLibrary.Protocol.ospfv3.**edit_ospfv3**(*Session*, ***kwargs*)

编辑 OSPFv3 协议会话对象参数

参数 **Session** (list(Ospfv3Router)) -- OSPFv3 协议会话对象列表

关键字参数

- **Name** (*str*) -- OSPFv3 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 OSPFv3 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **InstanceId** (*int*) -- 实例 ID, 类型为：number, 取值范围：0-255, 默认值：0
- **AreaId** (*str*) -- 区域 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableExtendedLsa** (*bool*) -- 使能扩展 LSA, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ExtendedLsaMode** (*str*) -- 扩展 LSA 模式, 类型为：string, 默认值：Full, 取值范围：

  NONE

  MixedModeOriginateOnly

  MixedModeOriginateSPF

  Full
- **AreaExtendedLsaMode** (*str*) -- 扩展区域 LSA 模式, 类型为：string, 默认值：InheritGlobal, 取值范围：

  InheritGlobal

  NONE

  MixedModeOriginateOnly

  MixedModeOriginateSPF

  Full
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 取值范围：Broadcast 或 P2P, 默认值：Broadcast

- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **InterfaceId** (*int*) -- 接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：10
- **Cost** (*int*) -- 接口开销, 类型为：number, 取值范围：1-65535, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'V6BIT', 'EBIT', 'RBIT'], 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

  Unused15

  Unused14

  Unused13

  Unused12

  Unused11

  Unused10

  Unused9

  Unused8

  Unused7

  Unused6

  Unused5

  Unused4

  Unused3

  Unused2

  Unused1

  Unused0
- **EnableOspfv3Mtu** (*bool*) -- 使能 OSPFv3 MTU, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GracefulRestartReason** (*str*) -- 平滑重启原因, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  SOFTWARE

  RELOADORUPGRADE

SWITCH

- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **HelloInterval** (*int*) -- Hello 包间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **RouterDeadInterval** (*int*) -- 路由器失效间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：40
- **LsaRetransInterval** (*int*) -- LSA 重传间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：5
- **LsaRefreshTime** (*int*) -- LSA 刷新间隔 (秒), 类型为：number, 取值范围：1-1800, 默认值：1800

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Ospfv3 | Session=${Session} |
```

TesterLibrary.Protocol.ospfv3.**edit_ospfv3_port_config**(*Ports*, ***kwargs*)

修改 Ospfv3 端口统计对象

参数 **Ports** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **TransmitRate** (*int*) -- OSPFv3 Message Tx Rate (messages/second), 取值范围：1-9000, 默认值：100
- **SessionOutstanding** (*int*) -- OSPFv3 Session Outstanding, 取值范围：1-1000, 默认值：20
- **UpdateMsgTransmitRate** (*int*) -- Deprecated. OSPFv3 Update Message Tx Rate (messages/second), 取值范围：1-9000, 默认值：10
- **EnableLoop** (*bool*) -- Enable Loop Back, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Ospfv3 Port Config | Ports=${Ports} | TransmitRate=100 |
```

TesterLibrary.Protocol.ospfv3.**establish_ospfv3**(*Sessions*)

建立 OSPFv3 协议会话

参数 **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Establish Ospfv3 | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ospfv3.**get_ospfv3_statistic**(*Session*, *StaItems=None*)

获取 OSPFv3 协议会话统计结果

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  RouterState

  AdjacencyState

  TxHello

  RxHello

  TxDd

  RxDd

  TxRouterLsa

  RxRouterLsa

  TxNetworkLsa

  RxNetworkLsa

  TxInterAreaPrefixLsa

  RxInterAreaPrefixLsa

  TxInterAreaRouterLsa

  RxInterAreaRouterLsa

  TxAsExternalLsa

  RxAsExternalLsa

  TxNssaLsa

  RxNssaLsa

  TxLinkLsa

  RxLinkLsa

  TxIntraAreaPrefixLsa

  RxIntraAreaPrefixLsa

  TxOpaqueRouterInfoLsa

  RxOpaqueRouterInfoLsa

  TxSrv6LocatorLsa

  RxSrv6LocatorLsa

  TxRequest

  RxRequest

  TxUpdate

  RxUpdate

TxAck

RxAck

返回

eg:

```
{
    'AdjacencyState': 'Full',
    'TxUpdate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv3SessionResultPropertySet |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Ospfv3 Statistic | Session=${Session} | StaItems=@{StaItems}
↪ |
| Clear Result |
```

TesterLibrary.Protocol.ospfv3.**grace_restart_ospfv3**(*Sessions*)

平滑重启 OSPFv3 协议会话

参数 **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Grace Restart Ospfv3 | Sessions=${Sessions} |
```

TesterLibrary.Protocol.ospfv3.**wait_ospfv3_adjacency_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 OSPFv3 协议会话达到指定邻接状态

参数

- **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 OSPFv3 协议会话达到的邻接状态, 类型为：string, 默认值：FULL, 支持下列状态：

  DOWN

  INIT

  TWOWAY

  EXSTART

  EXCHANGE

  LOADING

  FULL

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话邻接状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ospfv3 Adjacency State | Sessions=${Sessions} | State=FULL |␣
↪Interval=2 | TimeOut=120 |
```

TesterLibrary.Protocol.ospfv3.**wait_ospfv3_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 OSPFv3 协议会话达到指定状态

参数

- **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 OSPFv3 协议会话达到的状态, 类型为：string, 默认值：达到 DR 或 BACKUP 或 DROTHER, 支持下列状态：

  NOTSTART

  P2P

  WAITING

  DR

  BACKUP

  DROTHER

  DISABLE

  DOWN

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ospfv3 State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.ospfv3.**withdraw_ospfv3_lsa**(*Sessions=None*, *Type=None*, *Lsa=None*)

撤销 OSPFv3 协议会话 lsa

参数

- **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list

- **Type** (*str*) -- OSPFv3 lsa 类型, 类型为：string, 支持的 lsa 类型：

  router

  network

  InterAreaPrefix

  InterAreaRouter

  AsExternal

- **Lsa** (*list*) -- OSPFv3 lsa 列表, 类型为：list, 当 Type=None 时参数生效

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Ospfv3 Lsa | Sessions=${Sessions} | Type=router |
| Withdraw Ospfv3 Lsa | Sessions=${Sessions} | Lsa=${Lsas} |
```

TesterLibrary.Protocol.pcep module

TesterLibrary.Protocol.pcep.**create_pcep**(*Port*, ***kwargs*)

创建 PCEP 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- PCEP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 PCEP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Role** (*str*) -- PCEP 角色, 类型为：string, 默认值：PCE, 取值范围:

  PCE

  PCC

- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPv4, 取值范围:

  IPv4

  IPv6

- **UseGatewayAsDutIp** (*bool*) -- 使用网关地址作为 DUT 地址, 选中则使用接口上配置的网关 IP 地址作为 DUT 地址；未选中则自定义 DUT IP 地址, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SessionIpAddress** (*str*) -- 会话 IP 地址, 用于 PCEP 连接的 IP 类型, 类型为：string, 默认值：Interface_IP, 取值范围:

  Interface_IP

  Router_ID

- **PeerIpv4Address** (*str*) -- DUT IPv4 地址, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv4 时可见, 指定 DUT 的 IPv4 地址, 类型为：string, 默认值：192.85.1.1, 取值范围：有效的 Ipv4 地址
- **PeerIpv4AddressStep** (*str*) -- DUT IPv4 地址跳变, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv4 时可见, 指定 DUT IPv4 地址的增量步长, 类型为：string, 默认值：0.0.0.1, 取值范围：有效的 Ipv4 地址

- **PeerIpv6Address** (*str*) -- DUT IPv6 地址, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv6 时可见, 指定 DUT 的 IPv6 地址, 类型为：string, 默认值：2000::1, 取值范围：有效的 Ipv6 地址
- **PeerIpv6AddressStep** (*str*) -- DUT IPv6 地址跳变, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv6 时可见, 指定 DUT 的 IPv6 地址的增量步长, 类型为：string, 默认值：::1, 取值范围：有效的 Ipv6 地址
- **SessionInitiator** (*bool*) -- 会话发起者, 选中则主动发起会话建立请求；未选中则监听对端的发起会话建立请求。双方均主动发起会话建立请求时, IP 地址大的一方优先级更高, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Negotiation** (*bool*) -- 使能 Negotiation, 选中则对 Keepalive Timer 和 Dead Timer 的值进行协商, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **KeepAlive** (*str*) -- Keep Alive 间隔 (sec), KEEPALIVE 消息的发送间隔，以秒为单位, 类型为：string, 默认值：30, 0-65535
- **MinKeepAlive** (*int*) -- KEEPALIVE 消息发送间隔的最小值。以秒为单位, 类型为：number, 取值范围：0-255, 默认值：0
- **MaxKeepAlive** (*int*) -- KEEPALIVE 消息发送间隔的最大值，以秒为单位, 类型为：number, 取值范围：0-255, 默认值：255
- **Dead** (*str*) -- Dead 间隔 (sec), 从未收到对端消息到 PCEP 会话断开连接之间的时间间隔。类型为：string, 取值范围：0-65535, 默认值：120
- **MinDeadAlive** (*int*) -- 最小可接受 Dead 间隔 (sec), 类型为：number, 取值范围：0-255, 默认值：0
- **MaxDeadAlive** (*int*) -- 最大可接受 Dead 间隔 (sec), 类型为：number, 取值范围：0-255, 默认值：255
- **EnableStatefulCapability** (*bool*) -- 选中则 OPEN 消息中包含 Stateful PCE Capability TLV, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **StatefulCapability** (*list*) -- 使能 PCE Stateful Capability 选中时可见, 单击单元格并从下拉菜单中选择一个或多个能力, 类型为：list, 默认值：['LSP_UPDATE','LSP_INSTANTIATION'], 取值范围:

  LSP_UPDATE

  INCLUDE_DB_VERSION

  LSP_INSTANTIATION

  TRIGGERED_RESYNC

  DELTA_LSP_SYN

  TRIGGERED_INITIAL_SYNC

- **EnableSegmentRoutingCapability** (*list*) -- 选择段路由扩展, OPEN 消息中将包括该 Capability TLV, 类型为：list, 默认值：['SR'], 取值范围：

  SR

  SRv6

- **PathSetupTypeList** (*list*) -- 添加路径建立类型, 类型为：list, 默认值：[0,1]
- **SrCapabilityFlags** (*list*) -- 选择一个或多个 SR 能力标志, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SR 时可见, 类型为：list, 默认值：['NONTBIT','NFlag','XFlag'],

  NONTBIT

  NFlag

XFlag

- **Srv6CapabilityFlags** (*list*) -- 选择一个或多个 SRv6 能力标志, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SRv6 时可见, 类型为：list, 默认值：['NONTBIT','NFlag','XFlag'],

  NONTBIT

  NFlag

  XFlag

- **MSDs** (*list*) -- 选择一个或多个 MSD 类型, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SRv6 时可见, 类型为：list, 默认值：['NONTBIT']

  NONTBIT

  MaxiSegmentLeft

  MaxiEndPop

  MaxiHEncaps

  MaxiEndD

- **MaximumSidDepth** (*int*) -- 指定 SID 的最大数量, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SR 时可见, 类型为：number, 默认值：0, 取值范围：0-255
- **MaxSegmentsLeft** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum Segments Left 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxEndPop** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum End Pop 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxHencaps** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum H.Encaps 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum End D 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **EnableDbVersionTlv** (*bool*) -- 选中则配置 DB version TLV, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LspStateDbVersion** (*int*) -- 指定 LSP 状态数据库的初始版本号, 选中使能 DB Version TLV 时可见, 类型为：number, 默认值：1, 取值范围：1-18446744073709551614

返回 PCEP 协议会话对象, 类型：object

返回类型 (Pcep)

实际案例

```
| Create Pcep | Port=${Port} |
```

TesterLibrary.Protocol.pcep.**create_pcep_bw_object**(*PcepLsps*, ***kwargs*)

创建 PCEP Bw Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Bw Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Bw Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **Bandwidth** (*str*) -- 类型为: string, 取值范围: 0-4294967295, 默认值: 0

返回 Pcep Bw Object 对象列表, 类型: object / list

返回类型 (PcepBwObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Bw Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_endpoint_object**(*PcepLsps*, ***kwargs*)

创建 PCEP Endpoint Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Endpoint Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Endpoint Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False

返回 Pcep Endpoint Object 对象列表, 类型: object / list

返回类型 (PcepEndPointObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Endpoint Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_lsp_auto_tx_parameters**(*PcepAutoParameters*, ***kwargs*)

创建 PCEP PCE Lsp Auto Tx Parameters 对象

参数 **PcepAutoParameters** (BgpRouter) -- : PCEP LSP Auto Parameters 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Tx Parameters 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Auto Tx Parameters 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **ObjectCategory** (*str*) -- 选择对象类别, 类型为: string, 默认值: LSP, 取值范围:

  BANDWIDTH

  RP

  NO_PATH

  ENDPOINT

  METRIC

  ERO

  RRO

  LSPA

  SRP

  LSP

  XRO
- **SelectObjectHandle** (*str*) -- 选择对象, 类型为: string, 默认值: ""

返回 Pcep Lsp Auto Tx Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoTxParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pce Auto Update Parameters | PcepLsp=${Egress} |
| ${LspAutoTx} | Create Pcep Lsp Auto Tx Parameters | PcepAutoParameters=$
↪{Parameter} |
```

TesterLibrary.Protocol.pcep.**create_pcep_lspa_object**(*PcepLsps*, ***kwargs*)

创建 PCEP Lspa Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Lspa Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Lspa Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SetupPriority** (*int*) -- 指定 TE LSA 抢占资源的优先级。类型为：number, 取值范围：0-7, 默认值：0
- **HoldingPriority** (*int*) -- 指定 TE LSA 持有资源的优先级。类型为：number, 取值范围：0-7, 默认值：0
- **LFlag** (*bool*) -- LSPA L Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Affinities** (*bool*) -- 选中则对 32 位掩码和链接属性进行比较，设置包含链接条件和排除链接条件, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ExcludeAny** (*int*) -- Affinities 选中时启用, 排除与 32 位掩码中任何属性匹配的链接, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **IncludeAny** (*int*) -- Affinities 选中时启用, 包含与 32 位掩码中任何属性匹配的链接, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **IncludeAll** (*int*) -- Affinities 选中时启用, 包含与 32 位掩码中全部属性匹配的链接, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Pcep Lspa Object 对象列表, 类型: object / list

返回类型 (PcepLspaObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Lspa Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_metric_list**(*PcepLsps*, ***kwargs*)

创建 PCEP Metric List 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Metric List 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Metric List 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Metric List 对象列表, 类型: object / list

返回类型 (PcepMetricListConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Metric List | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_metric_object**(*PcepMetricLists*, ***kwargs*)

创建 PCEP Metric Object 对象

参数 **PcepMetricLists** (BgpRouter) -- ：PCEP Metric List 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Metric Object 对象, 类型为：string
- **Enable** (*bool*) -- PCEP Metric Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **BoundFlag** (*bool*) -- PCReq 消息中 METRIC Object 的 B(Bound) 位是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ComputedFlag** (*bool*) -- PCReq 消息中 METRIC Object 的 C(Computed Metric) 位是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MetricType** (*str*) -- 指定度量值类型, 类型为：string, 默认值：MAX_SID_DEPTH, 取值范围:

  IGP_METRIC

  TE_METRIC

  HOP_COUNTS

  MAX_SID_DEPTH

- **MetricValue** (*int*) -- 指定最大度量值, 类型为：number, 取值范围：0-4294967295, 默认值：10

返回 Pcep Metric Object 对象列表, 类型: object / list

返回类型 (PcepMetricObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Metric List | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Metric Object | PcepMetricLists=${Object} |
```

TesterLibrary.Protocol.pcep.**create_pcep_no_path_reason**(*PcepLsps*, ***kwargs*)

创建 PCEP No Path Reason 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP No Path Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP No Path Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **NoPathType** (*str*) -- No-Path 类型, 类型为: string, 默认值: NOT_SATISFYING_CONSTRAINTS, 取值范围:

  NOT_SATISFYING_CONSTRAINTS

  PCE_CHAIN_BROKEN

- **CFlag** (*bool*) -- C Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **NoPathReason** (*str*) -- No-Path 原因, 类型为: string, 默认值: NONTBIT, 取值范围:

  NONTBIT

  PCE_UNAVAILABLE

  UNKNOWN_DESTINATION

  UNKNOWN_SOURCE

返回 Pcep No Path Object 对象列表, 类型: object / list

返回类型 (PcepNoPathObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep No Path Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pcc_auto_delegation_parameters**(*PcepLsps*, ***kwargs*)

创建 PCEP Auto Delegation Parameters 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Delegation Parameters 对象名称, 类型为:string
- **Enable** (*bool*) -- PCEP Auto Delegation Parameters 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True

返回 Pcep Pcc Lsp Auto Delegation Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoDelegationParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pcc Auto Delegation Parameters | PcepLsp=$
↪{Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pcc_auto_request_parameters**(*PcepLsps*, ***kwargs*)

创建 PCEP PCC Lsp Auto Request Parameters 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Request Parameters 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Auto Request Parameters 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Pcep Pce Lsp Auto Request Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoRequestParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pcc Auto Request Parameters | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pcc_auto_sync_parameters**(*PcepLsps*, ***kwargs*)

创建 PCEP PCC Lsp Auto Sync Parameters 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP PCC LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Sync Parameters 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Auto Sync Parameters 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Pcep Pce Lsp Auto Sync Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoSyncParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pcc Auto Sync Parameters | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pcc_lsp**(*Sessions*, ***kwargs*)

创建 PCEP PCC LSP 对象

参数 **Sessions** (Pcep) -- ：PCEP 协议会话对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP PCC LSP 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PCEP PCC LSP, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LspCount** (*int*) -- Lsp 数量, 类型为：number, 默认值：1, 取值范围：1-65535
- **AutoGenSymbolicName** (*bool*) -- 系统自动生成 Symbolic Name, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SymbolicName** (*str*) -- 设置 Symbolic Name, 类型为：string, 默认值：PLSP_@s

- **PathSetupType** (*str*) -- 建立 LSP 的方法, 类型为：string, 默认值：SEGMENT_ROUTING, 取值范围：

  SEGMENT_ROUTING

  SRv6

- **SourceIpv4Address** (*str*) -- 起始源 IP 地址, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **SourceIpv4AddressStep** (*str*) -- 源 IP 地址的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **SourceIpv4AddressSessionOffset** -- 源 IPv4 地址接口跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围: IPv4 地址
- **DestinationIpv4Address** (*str*) -- 起始目的 IP 地址, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **DestinationIpv4AddressStep** (*str*) -- 目的 IP 地址的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **DestinationIpv4AddressSessionOffset** (*str*) -- 目的 IPv4 地址跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围: IPv4 地址
- **SourceIpv6Address** (*str*) -- 源 IPv6 地址的跳变步长, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **SourceIpv6AddressStep** (*str*) -- 源 IPv6 地址的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **SourceIpv6AddressSessionOffset** (*str*) -- 源 IPv6 地址接口跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围: IPv6 地址
- **DestinationIpv6Address** (*str*) -- 起始目的 IPv6 地址, 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **DestinationIpv6AddressStep** (*str*) -- 目的 IPv6 地址的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **DestinationIpv6AddressSessionOffset** (*str*) -- 目的 IPv6 地址跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围: IPv6 地址
- **LspInitiateMethod** (*str*) -- LSP 初始方式, 类型为：string, 默认值：REPORT, 取值范围：

  REPORT

  PCE_INITIATE

  SYNCHRONIZATION

  REQUEST

- **ImmediateDelegation** (*bool*) -- 直接托管, 会话建立后自动将 LSP 托管给 PCE, LSP 初始方式为 Report Method、Synchronization Method 或 Request Method 时可见, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **DelegationInSynchronization** (*bool*) -- 同步中托管, LSP 初始方式为 Synchronization Method 时可见, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Pcc Lsp 对象列表, 类型: object / list

返回类型 (PccLspConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pcc_lsp_info**(*PcepLsps*, ***kwargs*)

创建 PCEP PCC LSP INFO 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP PCC LSP INFO 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP PCC LSP INFO 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **Administrator** (*bool*) -- 使能 Administrative, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **State** (*str*) -- 设置 Initial LSP State, 类型为: string, 默认值: GOING_UP, 取值范围:

  DOWN

  UP

  ACTIVE

  GOING_DOWN

  GOING_UP

  RESERVED_5

  RESERVED_6

  RESERVED_7
- **AutoGeneratedPlspId** (*bool*) -- 自动生成 PLSP-ID, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PlspId** (*int*) -- 指定起始 PLSP-ID, 类型为: number, 默认值: 1, 取值范围: 1-1048575
- **Step** (*int*) -- 指定 PLSP-ID 的跳变步长, 类型为: number, 默认值: 1, 取值范围: 1-1048575
- **LspId** (*int*) -- 指定起始 LSP ID, 类型为: number, 默认值: 1, 取值范围: 1-65535
- **LspIdStep** (*int*) -- 指定同一个会话中 LSP-ID 的跳变步长, 类型为: number, 默认值: 1, 取值范围: 1-65535
- **LspIdSessionOffset** (*int*) -- 指定 PCEP 会话块中 LSP-ID 在会话之间的跳变步长, 类型为: number, 默认值: 1, 取值范围: 1-65535
- **TunnelId** (*int*) -- 指定起始隧道 ID, 类型为: number, 默认值: 1, 取值范围: 1-65535
- **TunnelStep** (*int*) -- 指定同一个会话中隧道 ID 的跳变步长, 类型为: number, 默认值: 1, 取值范围: 1-65535
- **TunnelSessionOffset** (*int*) -- 指定 PCEP 会话块中隧道 ID 在会话之间的跳变步长, 类型为: number, 默认值: 1, 取值范围: 1-65535
- **ExtendedTunnelIPv4Id** (*str*) -- 指定扩展隧道 ID, 类型为: string, 默认值: 10.0.0.1, 取值范围: IPv4 地址

- **ExtendedTunnelIPv4IdStep** (*str*) -- 指定同一个会话中扩展隧道 ID 的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **ExtendedTunnelIPv4IdSessionOffset** (*str*) -- 指定 PCEP 会话块中扩展隧道 ID 在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围: IPv4 地址
- **ExtendedTunnelIPv6Id** (*str*) -- 指定扩展隧道 ID, 类型为：string, 默认值：2000:1::1, 取值范围: IPv6 地址
- **ExtendedTunnelIPv6IdStep** (*str*) -- 指定同一个会话中扩展隧道 ID 的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **ExtendedTunnelIPv6IdSessionOffset** -- 指定 PCEP 会话块中扩展隧道 ID 在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围: IPv6 地址

返回 Pcep Pcc Lsp Info 对象列表, 类型: object / list

返回类型 (PccLspInfoObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${LspInfo} | Create Pcep Pcc Lsp Info | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pce_auto_initiate_parameters**(*PcepLsps*, ***kwargs*)

创建 PCEP PCE Lsp Auto Initiate Parameters 对象

参数 **PcepLsps** (PceLspConfig) -- ：PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Initiate Parameters 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Auto Initiate Parameters 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Pcep Pce Lsp Auto Initiate Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoInitiateParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pce Auto Initiate Parameters | PcepLsp=${Egress}␣
↪|
```

TesterLibrary.Protocol.pcep.**create_pcep_pce_auto_reply_parameters**(*PcepLsps*, ***kwargs*)

创建 PCEP PCE Lsp Auto Reply Parameters 对象

参数 **PcepLsps** (PceLspConfig) -- ：PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Reply Parameters 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Auto Reply Parameters 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Pcep Pce Lsp Auto Reply Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoReplyParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pce Auto Reply Parameters | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pce_auto_update_parameters**(*PcepLsps*, ***kwargs*)

创建 PCEP PCE Lsp Auto Update Parameters 对象

参数 **PcepLsps** (PceLspConfig) -- ：PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Update Parameters 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Auto Update Parameters 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Pcep Pce Lsp Auto Update Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoUpdateParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pce Auto Update Parameters | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pce_lsp**(*Sessions*, ***kwargs*)

创建 PCEP PCE LSP 对象

参数 **Sessions** (Pcep) -- ：PCEP 协议会话对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP PCE LSP 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PCEP PCE LSP, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LspCount** (*int*) -- Lsp 数量, 类型为：number, 默认值：1, 取值范围：1-65535
- **AutoGenSymbolicName** (*bool*) -- 系统自动生成 Symbolic Name, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SymbolicName** (*str*) -- 设置 Symbolic Name, 类型为：string, 默认值：PLSP_@s
- **PathSetupType** (*str*) -- 建立 LSP 的方法, 类型为：string, 默认值：SEGMENT_ROUTING, 取值范围：

  SEGMENT_ROUTING

SRv6

- **SourceIpv4Address** (*str*) -- 起始源 IP 地址, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **SourceIpv4AddressStep** (*str*) -- 源 IP 地址的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **SourceIpv4AddressSessionOffset** -- 源 IPv4 地址接口跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围: IPv4 地址
- **DestinationIpv4Address** (*str*) -- 起始目的 IP 地址, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **DestinationIpv4AddressStep** (*str*) -- 目的 IP 地址的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **DestinationIpv4AddressSessionOffset** -- 目的 IPv4 地址跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围: IPv4 地址
- **SourceIpv6Address** (*str*) -- 源 IPv6 地址的跳变步长, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **SourceIpv6AddressStep** (*str*) -- 源 IPv6 地址的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **SourceIpv6AddressSessionOffset** -- 源 IPv6 地址接口跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围: IPv6 地址
- **DestinationIpv6Address** (*str*) -- 起始目的 IPv6 地址, 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **DestinationIpv6AddressStep** (*str*) -- 目的 IPv6 地址的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **DestinationIpv6AddressSessionOffset** -- 目的 IPv6 地址跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围: IPv6 地址
- **LspInitiateMethod** (*str*) -- LSP 初始方式, 类型为：string, 默认值：UPDATE, 取值范围:

  UPDATE

  PCE_INITIATE

  REPLY

- **ImmediateUpdate** (*bool*) -- 直接更新, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Pcep Pce Lsp 对象列表, 类型: object / list

返回类型 (PceLspConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
```

TesterLibrary.Protocol.pcep.**create_pcep_pce_lsp_info**(*PcepLsps*, ***kwargs*)

创建 PCEP PCE LSP INFO 对象

参数 **PcepLsps** (PceLspConfig) -- ：PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP PCE LSP INFO 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP PCE LSP INFO 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Administrator** (*bool*) -- 使能 Administrative, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **State** (*str*) -- 设置 Initial LSP State, 类型为：string, 默认值：GOING_UP, 取值范围：

  DOWN

  UP

  ACTIVE

  GOING_DOWN

  GOING_UP

  RESERVED_5

  RESERVED_6

  RESERVED_7

返回 Pcep Pce Lsp Info 对象列表, 类型：object / list

返回类型 (PceLspInfoObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Info} | Create Pcep Pce Lsp Info | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_rp_object**(*PcepLsps*, ***kwargs*)

创建 PCEP PCC Rp Object 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP PCC LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP PCC Rp Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP PCC Rp Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AutoGeneratedId** (*bool*) -- 自动生成 RP-ID 类型为：bool, 取值范围：True 或 False, 默认值：True

- **RpId** (*int*) -- 指定起始 RP-ID, Auto-Generated RP-ID 未选中时启用, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **RpIdStep** (*int*) -- 指定 PLSP-ID 的跳变步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **Priority** (*int*) -- 指定请求的优先级。数字越大，优先级越高, 类型为：number, 取值范围：0-7, 默认值：0
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCRep 消息中 I Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **BFlag** (*bool*) -- RP Object 的 B(Bi-directional) 位置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **OFlag** (*bool*) -- RP Object 的 O(strict/loose) 位置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Pcc Rp Object 对象列表, 类型: object / list

返回类型 (PcepRpObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Rp Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_sr_ero_object**(*PcepLsps*, ***kwargs*)

创建 PCEP Sr Ero Object 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Sr Ero Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Sr Ero Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Metric List 对象列表, 类型: object / list

返回类型 (PcepSrEroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Sr Ero Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_sr_ero_sub_object**(*PcepSrEroObjects*, ***kwargs*)

创建 PCEP Sr Ero Sub Object 对象

参数 **PcepSrEroObjects** (PcepSrEroObjectConfig) -- ：PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Sr Ero Sub Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Sr Ero Sub Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- PCReq 消息中 P Flag 是否置位, 类型为：string, 默认值：STRICT, 取值范围:

  STRICT

  LOOSE
- **NaiType** (*str*) -- PCReq 消息中 I Flag 是否置位, 类型为：string, 默认值：IPV4_NODE_ID, 取值范围:

  ABSENT

  IPV4_NODE_ID

  IPV6_NODE_ID

  IPV4_ADJACENCY

  IPV6_ADJACENCY_GLOBAL

  UNNUMBERED_ADJACENCY

  IPV6_ADJACENCY_LINK_LOCAL
- **MFlag** (*bool*) -- M Flag, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **CFlag** (*bool*) -- C Flag, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SFlag** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- F Flag, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SidLabel** (*int*) -- SID Label, 类型为：number, 取值范围：0-1048575, 默认值：16
- **SidLabelStep** (*int*) -- SID Label 跳变, 类型为：number, 取值范围：0-1048575, 默认值：0
- **SidLabelSessionOffset** (*int*) -- SID Label 接口间跳变, 类型为：number, 取值范围：0-1048575, 默认值：0
- **SidIndex** (*int*) -- SID Index（32 Bits）, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **SidIndexStep** (*int*) -- SID Index 跳变（32 Bits）, 类型为：number, 取值范围：0-4294967295, 默认值：0

- **SidIndexSessionOffset** (*int*) -- SID Index 接口间跳变（32 Bits）, 类型为：number, 取值范围：0-1048575, 默认值：0
- **SidTrafficClass** (*int*) -- SID Traffic Class（3 bits）, 类型为：number, 取值范围：0-7, 默认值：0
- **SidTimeToLive** (*int*) -- SID Time To Liv, 类型为：number, 取值范围：0-255, 默认值：255
- **SidBottomOfStack** (*bool*) -- SID Bottom Of Stack Flag（1 Bit）, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NaiIpv4NodeId** (*str*) -- NAI IPv4 Node ID, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiIpv6NodeId** (*str*) -- NAI IPv6 Node ID, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiLocalIpv4Address** (*str*) -- NAI Local IPv4 Address, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiLocalIpv6Address** (*str*) -- NAI Local IPv6 Address, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiRemoteIpv4Address** (*str*) -- NAI Remote IPv4 Address, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **NaiRemoteIpv6Address** (*str*) -- NAI Remote IPv6 Address, 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **NaiLocalNodeId** (*str*) -- NAI Local Node-ID, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiLocalInterfaceId** (*int*) -- NAI Local Interface ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **NaiRemoteNodeId** (*str*) -- NAI Remote Node-ID, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **NaiRemoteInterfaceId** (*int*) -- NAI Remote Interface ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Pcep Sr Ero Sub Object 对象列表, 类型: object / list

返回类型 (PcepSrEroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Sr Ero Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Sr Ero Sub Object | PcepSrEroObjects=${Object} |
```

TesterLibrary.Protocol.pcep.**create_pcep_sr_rro_object**(*PcepLsps*, ***kwargs*)

创建 PCEP Sr Rro Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Sr Ero Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Sr Ero Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Metric List 对象列表, 类型: object / list

返回类型 (PcepSrRroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Sr Rro Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_sr_rro_sub_object**(*PcepSrRroObjects*, ***kwargs*)

创建 PCEP Sr Rro Sub Object 对象

参数 **PcepLsps** (PcepSrRroObjectConfig) -- ：PCEP PCE Sr Rro 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Sr Rro Sub Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Sr Rro Sub Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **NaiType** (*str*) -- 子对象中 NT 字段的值, 类型为：string, 默认值：IPV4_NODE_ID, 取值范围:

  ABSENT

  IPV4_NODE_ID

  IPV6_NODE_ID

  IPV4_ADJACENCY

  IPV6_ADJACENCY_GLOBAL

  UNNUMBERED_ADJACENCY

  IPV6_ADJACENCY_LINK_LOCAL
- **MFlag** (*bool*) -- 子对象中的 M Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **CFlag** (*bool*) -- 子对象中的 C Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SFlag** (*bool*) -- 子对象中的 S Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- 子对象中的 F Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SidIndex** (*int*) -- 指定起始 SID Index, 类型为：number, 取值范围：0-1048575, 默认值：1
- **SidLabel** (*int*) -- 指定起始 SID Label, 类型为：number, 取值范围：0-1048575, 默认值：16
- **SidTrafficClass** (*int*) -- 指定流量类型字段的值, 类型为：number, 取值范围：0-7, 默认值：0

- **SidTimeToLive** (*int*) -- 指定 TTL 字段值, 类型为：number, 取值范围：0-255, 默认值：255
- **SidBottomOfStack** (*bool*) -- 是否指定的 SID Label 为标签栈的栈底标签, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NaiIpv4NodeId** (*str*) -- 指定 NAI IPv4 节点 ID, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiIpv6NodeId** (*str*) -- 指定 NAI IPv6 节点 ID, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiLocalIpv4Address** (*str*) -- 指定 NAI 本地 IPv4 地址, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiLocalIpv6Address** (*str*) -- 指定 NAI 本地 IPv6 地址, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiRemoteIpv4Address** (*str*) -- 指定 NAI 远端 IPv4 地址, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **NaiRemoteIpv6Address** (*str*) -- 指定 NAI 远端 IPv6 地址, 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **NaiLocalNodeId** (*str*) -- 指定 NAI 远端节点 ID, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiLocalInterfaceId** (*int*) -- 指定 NAI 本地接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **NaiRemoteNodeId** (*str*) -- 指定 NAI 远端节点 ID, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **NaiRemoteInterfaceId** (*int*) -- 指定 NAI 远端接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Pcep Sr Ero Sub Object 对象列表, 类型: object / list

返回类型 (PcepSrRroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Sr Rro Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Sr Rro Sub Object | PcepSrRroObjects=${Object} |
```

TesterLibrary.Protocol.pcep.**create_pcep_srp_info**(*PcepLsps*, ***kwargs*)

创建 PCEP Srp Info 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Srp Info 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Srp Info 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AutoGeneratedId** (*bool*) -- 自动生成 SRP-ID, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SrpId** (*int*) -- 指定起始 SRP-ID, 类型为：number, 取值范围：0-4294967295, 默认值：1

- **SrpIdStep** (*int*) -- 指定 SRP-ID 的跳变步长, 类型为：number, 取值范围：0-4294967295, 默认值：1

返回 Pcep Srp Info 对象列表, 类型: object / list

返回类型 (PcepSrpObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srp Info | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_srv6_ero_object**(*PcepLsps*, ***kwargs*)

创建 PCEP Srv6 Ero Object 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Srv6 Ero Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Srv6 Ero Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Srv6 Ero Object 对象列表, 类型: object / list

返回类型 (PcepSrv6EroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srv6 Ero Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_srv6_ero_sub_object**(*PcepSrv6EroObjects*, ***kwargs*)

创建 PCEP Srv6 Ero Sub Object 对象

参数 **PcepSrv6EroObjects** (PcepSrv6EroObjectConfig) -- ：PCEP Srv6 Ero 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Srv6 Ero Sub Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Srv6 Ero Sub Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- 建立 LSP 使用的路由类型, 类型为：string, 默认值：STRICT, 取值范围:

  STRICT

  LOOSE

- **NaiType** (*str*) -- 指定端点行为, 类型为：string, 默认值：IPV6_NODE_ID, 取值范围:

  ABSENT

  IPV4_NODE_ID

  IPV6_NODE_ID

  IPV4_ADJACENCY

  IPV6_ADJACENCY_GLOBAL

  UNNUMBERED_ADJACENCY

  IPV6_ADJACENCY_LINK_LOCAL

- **SFlag** (*bool*) -- 子对象中的 S Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- 子对象中的 F Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EndpointBehavior** (*str*) -- 指定端点行为, 类型为：string, 默认值：Invalid, 取值范围:

  Invalid

  EndNoPspUsp

  EndPsp

  EndUsp

  EndPspUsp

  EndXNoPspUsp

  EndXPsp

  EndXUsp

  EndXPspUsp

  EndTNoPspUsp

  EndTPsp

  EndTUsp

  EndTPspUsp

  EndB6Encaps

  EndBM

  EndDX6

  EndDX4

  EndDT6

  EndDT4

  EndDT46

  EndDX2

  EndDX2V

  EndDT2U

  EndDT2M

  ENDB6EncapsRed

EndUSD

EndPSPUSD

EndUSPUSD

EndPSPUSPUSD

EndXUSD

EndXPSPUSD

EndXUSPUSD

EndXPSPUSPUSD

EndTUSD

EndTPSPUSD

EndTUSPUSD

EndTPSPUSPUSD

- **SRv6Sid** (*str*) -- 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiIpv6NodeId** (*str*) -- 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiLocalIpv6Address** (*str*) -- 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiRemoteIpv6Address** (*str*) -- 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **NaiLocalInterfaceId** (*int*) -- 类型为：number, 取值范围：0-4294967295, 默认值：0
- **NaiRemoteInterfaceId** (*int*) -- 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Pcep Srv6 Ero Sub Object 对象列表, 类型: object / list

返回类型 (PcepSrv6EroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srv6 Ero Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Srv6 Ero Sub Object | PcepSrEroObjects=${Object}
↪|
```

TesterLibrary.Protocol.pcep.**create_pcep_srv6_rro_object**(*PcepLsps*, ***kwargs*)

创建 PCEP Srv6 Rro Object 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP PCC LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Srv6 Rro Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Srv6 Rro Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Srv6 Rro Onject 对象列表, 类型：object / list

返回类型 (PcepSrv6RroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srv6 Rro Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_srv6_rro_sub_object**(*PcepSrv6RroObjects*, ***kwargs*)

创建 PCEP Srv6 Rro Sub Object 对象

参数 **PcepSrv6RroObjects** (PcepSrv6RroObjectConfig) -- ：PCEP Srv6 Rro 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Srv6 Rro Sub Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Srv6 Rro Sub Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **NaiType** (*str*) -- 子对象中 NT 字段的值, 类型为：string, 默认值：IPV6_NODE_ID, 取值范围:

  ABSENT

  IPV4_NODE_ID

  IPV6_NODE_ID

  IPV4_ADJACENCY

  IPV6_ADJACENCY_GLOBAL

  UNNUMBERED_ADJACENCY

  IPV6_ADJACENCY_LINK_LOCAL
- **SFlag** (*bool*) -- 子对象中的 S Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- 子对象中的 F Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EndpointBehavior** (*str*) -- 指定端点行为, 类型为：string, 默认值：Invalid, 取值范围:

  Invalid

  EndNoPspUsp

  EndPsp

  EndUsp

  EndPspUsp

  EndXNoPspUsp

  EndXPsp

  EndXUsp

EndXPspUsp

EndTNoPspUsp

EndTPsp

EndTUsp

EndTPspUsp

EndB6Encaps

EndBM

EndDX6

EndDX4

EndDT6

EndDT4

EndDT46

EndDX2

EndDX2V

EndDT2U

EndDT2M

ENDB6EncapsRed

EndUSD

EndPSPUSD

EndUSPUSD

EndPSPUSPUSD

EndXUSD

EndXPSPUSD

EndXUSPUSD

EndXPSPUSPUSD

EndTUSD

EndTPSPUSD

EndTUSPUSD

EndTPSPUSPUSD

- **SRv6Sid** (*str*) -- 指定 SRv6 SID, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiIpv6NodeId** (*str*) -- 指定 NAI IPv6 节点 ID, 指定 NAI IPv6 节点 ID, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiLocalIpv6Address** (*str*) -- 指定 NAI 本地 IPv6 地址, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiRemoteIpv6Address** (*str*) -- 指定 NAI 远端 IPv6 地址, 类型为: string, 默认值：2001::1, 取值范围: IPv6 地址
- **NaiLocalInterfaceId** (*int*) -- 指定 NAI 本地接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **NaiRemoteInterfaceId** (*int*) -- 指定 NAI 远端接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Pcep Srv6 Rro Sub Object 对象列表, 类型: object / list

返回类型 (PcepSrv6RroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srv6 Rro Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Srv6 Rro Sub Object | PcepSrv6RroObjects=$
↪{Object} |
```

TesterLibrary.Protocol.pcep.**create_pcep_xro_object**(*PcepLsps*, ***kwargs*)

创建 PCEP XRO Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP XRO Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP XRO Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **FFlag** (*bool*) -- PCReq 消息中 F Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False

返回 Pcep XRO Object 对象列表, 类型: object / list

返回类型 (PcepXroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep XRO Object | PcepLsp=${Egress} |
```

TesterLibrary.Protocol.pcep.**create_pcep_xro_sub_object**(*PcepXroObjects*, ***kwargs*)

创建 PCEP XRO Object 对象

参数 **PcepXroObjects** (PcepXroObjectConfig) -- : PCEP XRO Object 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Xro Sub Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Xro Sub Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **XFlag** (*bool*) -- 类型为: bool, 取值范围: True 或 False, 默认值: False

- **Type** (*str*) -- 类型为：string, 默认值：IPv4_PREFIX, 取值范围:

  IPv4_PREFIX

  IPv6_PREFIX

  UNNUMBERED_INTERFACE_ID

  AUTONOMOUS_SYS_NUM

  SRLG
- **PrefixLength** (*int*) -- 类型为：number, 取值范围：0-32, 默认值：24
- **Attribute** (*str*) -- 类型为：string, 默认值：INTERFACE, 取值范围:

  INTERFACE

  NODE

  SRLG
- **Ipv4Address** (*str*) -- 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **Ipv6Address** (*str*) -- 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **TeRouterId** (*str*) -- 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **InterfaceId** (*int*) -- 类型为：number, 默认值：0
- **AsNumber** (*int*) -- 类型为：number, 默认值：0
- **SrlgId** (*int*) -- 类型为：number, 默认值：0

返回 Pcep Xro Sub Object 对象列表, 类型: object / list

返回类型 (PcepXroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Xro Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Xro Sub Object | PcepXroObjects=${Object} |
```

TesterLibrary.Protocol.pcep.**edit_pcep**(*Sessions*, ***kwargs*)

编辑 PCEP 协议会话对象参数

参数 **Sessions** (Pcep) -- ：PCEP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- PCEP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 PCEP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Role** (*str*) -- PCEP 角色, 类型为：string, 默认值：PCE, 取值范围:

  PCE

  PCC

- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPv4, 取值范围:

  IPv4

  IPv6

- **UseGatewayAsDutIp** (*bool*) -- 使用网关地址作为 DUT 地址, 选中则使用接口上配置的网关 IP 地址作为 DUT 地址；未选中则自定义 DUT IP 地址, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **SessionIpAddress** (*str*) -- 会话 IP 地址, 用于 PCEP 连接的 IP 类型, 类型为：string, 默认值：Interface_IP, 取值范围:

  Interface_IP

  Router_ID

- **PeerIpv4Address** (*str*) -- DUT IPv4 地址, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv4 时可见, 指定 DUT 的 IPv4 地址, 类型为：string, 默认值：192.85.1.1, 取值范围：有效的 Ipv4 地址

- **PeerIpv4AddressStep** (*str*) -- DUT IPv4 地址跳变, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv4 时可见, 指定 DUT IPv4 地址的增量步长, 类型为：string, 默认值：0.0.0.1, 取值范围：有效的 Ipv4 地址

- **PeerIpv6Address** (*str*) -- DUT IPv6 地址, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv6 时可见, 指定 DUT 的 IPv6 地址, 类型为：string, 默认值：2000::1, 取值范围：有效的 Ipv6 地址

- **PeerIpv6AddressStep** (*str*) -- DUT IPv6 地址跳变, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv6 时可见, 指定 DUT 的 IPv6 地址的增量步长, 类型为：string, 默认值：::1, 取值范围：有效的 Ipv6 地址

- **SessionInitiator** (*bool*) -- 会话发起者, 选中则主动发起会话建立请求；未选中则监听对端的发起会话建立请求。双方均主动发起会话建立请求时，IP 地址大的一方优先级更高, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **Negotiation** (*bool*) -- 使能 Negotiation, 选中则对 Keepalive Timer 和 Dead Timer 的值进行协商, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **KeepAlive** (*str*) -- Keep Alive 间隔 (sec), KEEPALIVE 消息的发送间隔，以秒为单位, 类型为：string, 默认值：30, 0-65535

- **MinKeepAlive** (*int*) -- KEEPALIVE 消息发送间隔的最小值。以秒为单位, 类型为：number, 取值范围：0-255, 默认值：0

- **MaxKeepAlive** (*int*) -- KEEPALIVE 消息发送间隔的最大值，以秒为单位, 类型为：number, 取值范围：0-255, 默认值：255

- **Dead** (*str*) -- Dead 间隔 (sec), 从未收到对端消息到 PCEP 会话断开连接之间的时间间隔。类型为：string, 取值范围：0-65535, 默认值：120

- **MinDeadAlive** (*int*) -- 最小可接受 Dead 间隔 (sec), 类型为：number, 取值范围：0-255, 默认值：0

- **MaxDeadAlive** (*int*) -- 最大可接受 Dead 间隔 (sec), 类型为：number, 取值范围：0-255, 默认值：255

- **EnableStatefulCapability** (*bool*) -- 选中则 OPEN 消息中包含 Stateful PCE Capability TLV, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **StatefulCapability** (*list*) -- 使能 PCE Stateful Capability 选中时可见, 单击单元格并从下拉菜单中选择一个或多个能力, 类型为：list, 默认值：['LSP_UPDATE','LSP_INSTANTIATION'], 取值范围:

  LSP_UPDATE

  INCLUDE_DB_VERSION

LSP_INSTANTIATION

TRIGGERED_RESYNC

DELTA_LSP_SYN

TRIGGERED_INITIAL_SYNC

- **EnableSegmentRoutingCapability** (*list*) -- 选择段路由扩展, OPEN 消息中将包括该 Capability TLV, 类型为：list, 默认值：['SR'], 取值范围：

  SR

  SRv6

- **PathSetupTypeList** (*list*) -- 添加路径建立类型, 类型为：list, 默认值：[0,1]

- **SrCapabilityFlags** (*list*) -- 选择一个或多个 SR 能力标志, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SR 时可见, 类型为：list, 默认值：['NONTBIT','NFlag','XFlag'],

  NONTBIT

  NFlag

  XFlag

- **Srv6CapabilityFlags** (*list*) -- 选择一个或多个 SRv6 能力标志, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SRv6 时可见, 类型为：list, 默认值：['NONTBIT','NFlag','XFlag'],

  NONTBIT

  NFlag

  XFlag

- **MSDs** (*list*) -- 选择一个或多个 MSD 类型, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SRv6 时可见, 类型为：list, 默认值：['NONTBIT']

  NONTBIT

  MaxiSegmentLeft

  MaxiEndPop

  MaxiHEncaps

  MaxiEndD

- **MaximumSidDepth** (*int*) -- 指定 SID 的最大数量, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SR 时可见, 类型为：number, 默认值：0, 取值范围：0-255

- **MaxSegmentsLeft** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum Segments Left 时可见, 类型为：number, 默认值：8, 取值范围：0-255

- **MaxEndPop** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum End Pop 时可见, 类型为：number, 默认值：8, 取值范围：0-255

- **MaxHencaps** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum H.Encaps 时可见, 类型为：number, 默认值：8, 取值范围：0-255

- **MaxEndD** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum End D 时可见, 类型为：number, 默认值：8, 取值范围：0-255

- **EnableDbVersionTlv** (*bool*) -- 选中则配置 DB version TLV, 类型为: bool, 取值范围：True 或 False, 默认值：False
- **LspStateDbVersion** (*int*) -- 指定 LSP 状态数据库的初始版本号, 选中使能 DB Version TLV 时可见, 类型为：number, 默认值：1, 取值范围：1-18446744073709551614

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | Role=PCC |
```

TesterLibrary.Protocol.pcep.**edit_pcep_port_config**(*Ports*, ***kwargs*)

修改 PCEP 端口配置对象

Args:

Ports (Port): 测试仪表端口对象, 类型为：object

关键字参数

- **MaxOutstanding** (*int*) -- 最大会话负载数量, 取值范围：1-65535, 默认值：100
- **RetryCount** (*int*) -- 会话尝试建立次数, 取值范围：0-65535, 默认值：5
- **RetryInterval** (*int*) -- 会话尝试建立间隔（sec）, 取值范围：0-65535, 默认值：30
- **MaxLspPerMessage** (*int*) -- 消息中 LSP 的最大个数, 取值范围：1-2000, 默认值：100

返回 PCEP Client Custom Options 对象, 类型：object / list

返回类型 (PCEPPortRateConfig)

实际案例

```
| Edit PCEP Client Port Config | Ports=${Port} | TcpServerPort=10 |
```

TesterLibrary.Protocol.pcep.**get_pcep_lsp_block_statistic**(*Session*, *SessionId*, *Lsp*, *StaItems: Optional[list] = None*)

获取 PCEP LSP BLOCK 统计结果

参数

- **Session** (Pcep) -- : PCEP 协议会话对象, 类型为：Object
- **SessionId** (*int*) -- PCEP 协议会话 ID, 类型为：Number
- **Lsp** (PccLspConfig) -- : LSP 对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  LspIdentify

SessionIndex

SessionLocalIP

SessionPeerIP

Role

LspCount

RequestedLsps

RepliedLsps

DelegatedLsps

UpdatedLsps

RevokedLsps

ReturnedLsps

InitiatedLsps

StateDownLsps

StateUpLsps

StateActiveLsps

StateGoingDownLsps

StateGoingUpLsps

StateOtherLsps

返回

eg:

```
{
    'LspCount': 1,
    'RequestedLsps': 0,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepLspBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Lsp Block Statistic | Session=@{Session} | SessionId=1
↪| Lsp=@{Lsp} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pcep.**get_pcep_lsp_statistic**(*Session*, *SessionId*, *Lsp*, *LspId*, *StaItems: Optional[list] = None*)

获取 PCEP LSP 统计结果

参数

- **Session** (Pcep) -- : PCEP 协议会话对象, 类型为: Object
- **SessionId** (*int*) -- PCEP 协议会话 ID, 类型为: Number
- **Lsp** (PccLspConfig) -- : LSP 对象, 类型为: Object

- **LspId** (*int*) -- LSP 对象 ID, 类型为：Number
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  LspIdentify

  SessionIndex

  LspIndex

  SessionLocalIP

  SessionPeerIP

  Role

  SymbolicName

  LspSourceIP

  LspDestinationIP

  LspState

  PLSPId

  LSPId

  SRPId

  RPId

返回

eg:

```
{
    'PLSPId': 1,
    'LSPId': 1,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepLspStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Lsp Statistic | Session=@{Session} | SessionId=1 |␣
↪Lsp=@{Lsp} | LspId=1 | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pcep.**get_pcep_port_statistic**(*Port*, *StaItems: Optional[list] = None*)

获取 PCEP Port 统计结果

参数

- **Port** (Port) -- PCEP 协议会话所在的端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockCount

SessionBlockIdleCount

SessionBlockPendingCount

SessionBlockUpCount

返回

eg:

```
{
    'SessionBlockCount': 1,
    'SessionBlockIdleCount': 0,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepPortStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Port Statistic | Port=@{Port} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pcep.**get_pcep_session_block_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 PCEP session block 统计结果

参数

- **Session** (Pcep) -- ：PCEP 协议会话对象，类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  SessionCount

  IdleCount

  PendingCount

  UpCount

  LspCount

  StateDownLsps

  StateUpLsps

  StateActiveLsps

  StateGoingDownLsps

  StateGoingUpLsps

  StateOtherLsps

  TxOpenCount

  RxOpenCount

  TxKeepaliveCount

RxKeepaliveCount

TxReportCount

RxReportCount

TxUpdateCount

RxUpdateCount

TxRequestCount

RxRequestCount

TxReplyCount

RxReplyCount

TxInitiateCount

RxInitiateCount

TxCloseCount

RxCloseCount

TxErrorCount

RxErrorCount

返回

eg:

```
{
    'TxErrorCount': 1,
    'RxErrorCount': 0,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepSessionBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Session Block Statistic | Session=@{Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pcep.**get_pcep_session_statistic**(*Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 PCEP session 统计结果

参数

- **Session** (Pcep) -- : PCEP 协议会话对象, 类型为: Object
- **SessionId** (*int*) -- PCEP 协议会话 ID, 类型为: Number
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为: list，目前支持的统计项

  Role

  LocalIP

PeerIP

State

LspCount

StateDownLsps

StateUpLsps

StateActiveLsps

StateGoingDownLsps

StateGoingUpLsps

StateOtherLsps

TxOpenCount

RxOpenCount

TxKeepaliveCount

RxKeepaliveCount

TxReportCount

RxReportCount

TxUpdateCount

RxUpdateCount

TxRequestCount

RxRequestCount

TxReplyCount

RxReplyCount

TxInitiateCount

RxInitiateCount

TxCloseCount

RxCloseCount

TxErrorCount

RxErrorCount

返回

eg:

```
{
    'TxErrorCount': 1,
    'RxErrorCount': 0,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepSessionStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Session Statistic | Session=@{Session} | SessionId=@
↪{SessionId} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pcep.**pcep_establish**(*Sessions*)

建立 PCEP 会话

参数 **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Establish | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pcep.**pcep_pcc_delegate_lsp**(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送托管 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Delegate Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pcc_end_sync**(*Sessions*, *Lsps=None*)

停止 PCC 向 PCE 发送初始状态同步报文

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc End Sync | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pcc_initial_sync**(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送初始状态同步报文

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Initial Sync | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pcc_remove_delegate_lsp**(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送删除托管 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Remove Delegate Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pcc_report_lsp**(*Sessions*, *Lsps=None*)

PCC 向 PCE 报告 LSP 状态

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Report Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pcc_request_lsp**(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送路径计算请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Request Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pcc_revoke_lsp**(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送取消托管 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Revoke Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pcc_synchronize_lsp**(*Sessions*, *Lsps=None*)

PCC 向 PCE 报告 LSP 状态

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Synchronize Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pce_initiate_lsp**(*Sessions*, *Lsps=None*)

PCE 向 PCC 发送初始化 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pce Initiate Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pce_remove_initiated_lsp**(*Sessions*, *Lsps=None*)

PCE 向 PCC 发送删除指定 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pce Remove Initiate Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pce_return_lsp**(*Sessions*, *Lsps=None*)

使 Stateful PCE 向 PCC 归还 LSP 托管权限

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pce Return Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_pce_update_lsp**(*Sessions*, *Lsps=None*)

使 Stateful PCE 向 PCC 发送更新 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pce Update Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

TesterLibrary.Protocol.pcep.**pcep_resume_keep_alive**(*Sessions*)

指定 PCEP 会话恢复发送 Keepalive 报文

参数 **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Resume Keep Alive | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pcep.**pcep_stop_keep_alive**(*Sessions*)

指定 PCEP 会话暂停发送 Keepalive 报文

参数 **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Stop Keep Alive | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pcep.**wait_pcep_state**(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 PCEP 协议会话达到指定状态

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **State** (*str*) -- 等待 PCEP 协议会话达到的状态, 类型为: string, 默认值: 达到 UP, 支持下列状态:

  DISABLED

  IDLE

PENDING

UP

CLOSING

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Pcep State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.pim module

TesterLibrary.Protocol.pim.**create_pim**(*Port*, ***kwargs*)

创建 PIM 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- PIM 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SessionMode** (*str*) -- 协议模式, 类型为：string, 默认值：SM, 支持版本：

  SM

  SSM
- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPV4, 支持版本：

  IPV4

  IPV6
- **DrPriority** (*int*) -- DR 优先级, 类型为：number, 取值范围：1-65535, 默认值：1
- **DrAddr** (*str*) -- DR 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.0
- **DrIpv6Addr** (*str*) -- DR IPv6 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：'::'
- **GenIdMode** (*str*) -- GenID 模式, 类型为：string, 默认值：FIXED, 支持参数：

  FIXED

  INCR

  RAND
- **RegisterEnable** (*bool*) -- Register 使能, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **BsrEnable** (*bool*) -- BSR 使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **BsrPriority** (*int*) -- BSR 优先级, 类型为：number, 取值范围：0-255, 默认值：1
- **BsrInterval** (*int*) -- BSR 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-3600, 默认值：60
- **HelloInterval** (*int*) -- Hello 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-3600, 默认值：30
- **HelloHoldTime** (*int*) -- Hello 消息超时时间（秒）, 类型为：number, 取值范围：1-65535, 默认值：105
- **JoinPruneInterval** (*int*) -- Join/Prune 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-65535, 默认值：60
- **JoinPruneHoldTime** (*int*) -- Join/Prune 消息超时时间（秒）, 类型为：number, 取值范围：1-65535, 默认值：210

返回 PIM 协议会话对象, 类型：object

返回类型 (PimRouter)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| ${DrAddr} | Set Variable | ${Session.DrAddr} |
```

TesterLibrary.Protocol.pim.**create_pim_group**(*Session*, ***kwargs*)

创建 PIM Group 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM Group 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM Group 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **GroupCheck** (*bool*) -- 协议模式, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GroupType** (*str*) -- 组类别, 类型为：string, 默认值：ANY_G, 支持版本：

  ANY_G

  S_G

  S_G_RPT

  ANY_RP
- **GroupAddr** (*str*) -- 组地址, 类型为：string, 取值范围：IPv4 地址, 默认值：225.0.0.1
- **GroupCount** (*int*) -- 组数目, 类型为：number, 取值范围：1-65535（BigTao）1-500000（DarYu）, 默认值：1
- **GroupModifierStep** (*int*) -- 组地址增量步进, 类型为：number, 取值范围：0-65535, 默认值：1
- **GroupModifierBit** (*int*) -- 组地址增量位, 类型为：number, 取值范围：1-32, 默认值：32

- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：10.10.10.10
- **JoinSrc** (*str*) -- Join 源地址, 类型为：string, 取值范围：IPv4 地址, 默认值：1.1.1.1
- **JoinMaskLen** (*int*) -- Join 掩码长度, 类型为：number, 取值范围：1-32, 默认值：32
- **PruneSrcAddr** -- Prune 源地址, 类型为：string, 取值范围：IPv4 地址, 默认值：1.1.1.1
- **PruneMaskLen** (*int*) -- Prune 掩码长度, 类型为：number, 取值范围：1-32, 默认值：32

返回 PIM Group 对象, 类型：object

返回类型 (PimGroupConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim Group | Session=${Session} | GroupAddr=255.0.0.2 |
```

TesterLibrary.Protocol.pim.**create_pim_ipv6_group**(*Session*, ***kwargs*)

创建 PIM IPv6 Group 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv6 Group 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM IPv6 Group 对象, 类型为: bool, 取值范围: True 或 False, 默认值：True
- **GroupCheck** (*bool*) -- 协议模式, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GroupType** (*str*) -- 组类别, 类型为：string, 默认值：ANY_G, 支持版本：

  ANY_G

  S_G

  S_G_RPT

  ANY_RP
- **GroupAddr** (*str*) -- 组地址, 类型为：string, 取值范围：IPv6 地址, 默认值：ff1e::1
- **GroupCount** (*int*) -- 组数目, 类型为: number, 取值范围: 1-65535（BigTao）1-500000（DarYu）, 默认值：1
- **GroupModifierStep** (*int*) -- 组地址增量步进, 类型为：number, 取值范围：0-65535, 默认值：1
- **GroupModifierBit** (*int*) -- 组地址增量位, 类型为：number, 取值范围：1-128, 默认值：128
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **JoinSrc** (*str*) -- Join 源地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1

- **JoinMaskLen** (*int*) -- Join 掩码长度, 类型为：number, 取值范围：1-128, 默认值：64
- **PruneSrcAddr** (*str*) -- Prune 源地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PruneMaskLen** (*int*) -- Prune 掩码长度, 类型为：number, 取值范围：1-128, 默认值：64

返回 PIM IPv6 Group 对象, 类型：object

返回类型 (PimIpv6GroupConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim IPv6 Group | Session=${Session} | GroupAddr=ff1e::2 |
```

TesterLibrary.Protocol.pim.**create_pim_ipv6_register_group**(*Session*, ***kwargs*)

创建 PIM IPv6 Register Group 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv6 Register Group 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM IPv6 Register Group 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **MulticastGroupToSourceDistribution** (*str*) -- 组播地址和源地址映射方式, 类型为：string, 默认值：PAIR, 支持参数：

  PAIR

  BACKBONE
- **RegisterTransmitMode** (*str*) -- 注册发送模式, 类型为：string, 默认值：CONTINUOUS, 支持参数：

  FIXED

  CONTINUOUS
- **FixedModeCount** (*int*) -- 固定模式数量, 类型为：number, 取值范围：1-1000, 默认值：5
- **MulticastGroupCount** (*int*) -- 多播组数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartMulticastGroupAddr** (*str*) -- 多播组起始地址, 类型为：string, 取值范围：IPv6 地址, 默认值：ff1e::2
- **MulticastGroupStep** (*int*) -- 多播组步长, 类型为：number, 取值范围：1-255, 默认值：1
- **MulticastGroupPrefixLength** (*int*) -- 多播组前缀长度, 类型为：number, 取值范围：1-128, 默认值：64
- **MulticastSourceCount** (*int*) -- 组播源数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartMulticastSourceAddr** (*str*) -- 多播组起始地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2001::1
- **MulticastSourceStep** (*int*) -- 多播组步长, 类型为：number, 取值范围：1-255, 默认值：1

- **MulticastSourcePrefixLength** (*int*) -- 多播组前缀长度, 类型为：number, 取值范围：1-128, 默认值：64
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::2
- **RegisterTransmitInterval** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：10-180, 默认值：60

返回 Create Pim Ipv6 Register Group 对象, 类型：object

返回类型 (PimIpv6RegisterGroupConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim Ipv6 Register Group | Session=${Session} | RpAddr=3000::2 |
```

TesterLibrary.Protocol.pim.**create_pim_ipv6_rp_map**(*Session*, ***kwargs*)

创建 PIM IPv6 Rp Map 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv6 Rp Map 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM IPv6 Rp Map 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **MulticastGroupAddr** (*str*) -- 组播地址和源地址映射方式, 类型为：string, 取值范围：IPv6 地址, 默认值：ff1e::1
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：0-128, 默认值：128
- **RpPriority** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：0-255, 默认值：0
- **RpHoldTime** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：1-65535, 默认值：150

返回 PIM IPv6 Rp Map 对象, 类型：object

返回类型 (PimIpv6RpMapConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim Ipv6 Rp Map | Session=${Session} | RpAddr=3000::1 |
```

TesterLibrary.Protocol.pim.**create_pim_register_group**(*Session*, ***kwargs*)

创建 PIM IPv4 Register Group 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv4 Register Group 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM IPv4 Register Group 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **MulticastGroupToSourceDistribution** (*str*) -- 组播地址和源地址映射方式, 类型为：string, 默认值：PAIR, 支持参数：

  PAIR

  BACKBONE

- **RegisterTransmitMode** (*str*) -- 注册发送模式, 类型为：string, 默认值：CONTINUOUS, 支持参数：

  FIXED

  CONTINUOUS

- **FixedModeCount** (*int*) -- 固定模式数量, 类型为：number, 取值范围：1-1000, 默认值：5
- **MulticastGroupCount** (*int*) -- 多播组数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartMulticastGroupAddr** (*str*) -- 多播组起始地址, 类型为：string, 取值范围：IPv4 地址, 默认值：225.0.1.1
- **MulticastGroupStep** (*int*) -- 多播组步长, 类型为：number, 取值范围：1-255, 默认值：1
- **MulticastGroupPrefixLength** (*int*) -- 多播组前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **MulticastSourceCount** (*int*) -- 组播源数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartMulticastSourceAddr** (*str*) -- 多播组起始地址, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1
- **MulticastSourceStep** (*int*) -- 多播组步长, 类型为：number, 取值范围：1-255, 默认值：1
- **MulticastSourcePrefixLength** (*int*) -- 多播组前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：10.10.10.20
- **RegisterTransmitInterval** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：10-180, 默认值：60

返回 PIM IPv4 Register Group 对象, 类型：object

返回类型 (PimRegisterGroupConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim Register Group | Session=${Session} | RpAddr=20.10.10.20 |
```

TesterLibrary.Protocol.pim.**create_pim_rp_map**(*Session*, ***kwargs*)

创建 PIM IPv4 Rp Map 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv4 Rp Map 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM IPv4 Rp Map 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **MulticastGroupAddr** (*str*) -- 组播地址和源地址映射方式, 类型为：string, 取值范围：IPv4 地址, 默认值：255.0.0.1
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：10.10.10.10
- **PrefixLength** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：0-32, 默认值：32
- **RpPriority** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：0-255, 默认值：0
- **RpHoldTime** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：1-65535, 默认值：150

返回 PIM IPv6 Rp Map 对象, 类型：object

返回类型 (PimIpv6RpMapConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim IPv4 Rp Map | Session=${Session} | RpAddr=20.10.10.10 |
```

TesterLibrary.Protocol.pim.**edit_pim**(*Session*, ***kwargs*)

编辑 PIM 协议会话对象参数

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SessionMode** (*str*) -- 协议模式, 类型为：string, 默认值：SM, 支持版本：

  SM

  SSM
- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPV4, 支持版本：

  IPV4

  IPV6
- **DrPriority** (*int*) -- DR 优先级, 类型为：number, 取值范围：1-65535, 默认值：1
- **DrAddr** (*str*) -- DR 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.0
- **DrIpv6Addr** (*str*) -- DR IPv6 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：'::'
- **GenIdMode** (*str*) -- GenID 模式, 类型为：string, 默认值：FIXED, 支持参数：

  FIXED

  INCR

  RAND
- **RegisterEnable** (*bool*) -- Register 使能, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **BsrEnable** (*bool*) -- BSR 使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **BsrPriority** (*int*) -- BSR 优先级, 类型为：number, 取值范围：0-255, 默认值：1
- **BsrInterval** (*int*) -- BSR 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-3600, 默认值：60
- **HelloInterval** (*int*) -- Hello 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-3600, 默认值：30
- **HelloHoldTime** (*int*) -- Hello 消息超时时间（秒）, 类型为：number, 取值范围：1-65535, 默认值：105
- **JoinPruneInterval** (*int*) -- Join/Prune 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-65535, 默认值：60
- **JoinPruneHoldTime** (*int*) -- Join/Prune 消息超时时间（秒）, 类型为：number, 取值范围：1-65535, 默认值：210

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Edit Pim | Session=${Session} | HelloInterval=60 |
```

TesterLibrary.Protocol.pim.**edit_pim_port_config**(*Ports*, ***kwargs*)

修改 PIM 协议会话的端口配置

参数 **Ports** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **MsgTransRate** (*int*) -- PIM Message Transmit Rate (messages/sec), 类型为：number, 取值范围：1-10000, 默认值：500
- **TriggerHelloDelay** (*int*) -- Trigger Hello Delay (sec), 类型为：number, 取值范围：0-60, 默认值：5
- **DisableHelloExpireTimer** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **DisableRecvHelloInNeighborState** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **DisableNonHelloRecv** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pim Port Config | Ports=${Ports} | DisableNonHelloRecv=True |
```

TesterLibrary.Protocol.pim.**get_pim_group_stats**(*Session*, *Group*, *StaItems: Optional[list] = None*)

获取 Pim Group Stats 统计结果

参数

- **Session** (PimRouter) -- BGP 协议会话对象, 类型为：Object
- **Group** (PimGroupConfig) -- Pim 协议组对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxAnyG

  TxSG

  TxRP

  TxRpt

返回

eg:

```
{
    'TxRpt': 10,
    'RxRpt': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | TxHello | RxHello |
| Subscribe Result | Types=PimGroupStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pim Group Stats | Session=${Session} | Group=${Group} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pim.**get_pim_session_stats**(*Session*, *StaItems: Optional[list] = None*)

获取 Pim Session Stats 统计结果

参数

- **Session** (PimRouter) -- BGP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  NeighborNum

  TxHello

  RxHello

  TxJoin

  RxJoin

TxAnyG

RxAnyG

TxSG

RxSG

TxRP

RxRP

TxRpt

RxRpt

TxBsr

TxRegister

RxRegisterStop

返回

eg:

```
{
    'TxRpt': 10,
    'RxRpt': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | TxHello | RxHello |
| Subscribe Result | Types=PimSessionStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pim Session Stats | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pim.**pim_change_gen_id**(*Sessions*)

修改 PIM 协议会话的 GenId

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pim Join Group | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pim.**pim_join_group**(*Sessions*)

PIM 协议会话发送加入组数据包

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pim Join Group | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pim.**pim_leave_group**(*Sessions*)

PIM 协议会话发送离开组数据包

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Leave Group | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pim.**pim_start_boot_strap**(*Sessions*)

启动 PIM 协议会话 BootStrap

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Start Boot Strap | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pim.**pim_start_register**(*Sessions*)

PIM 协议会话开始发送 Register 消息

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Start Register | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pim.**pim_stop_boot_strap**(*Sessions*)

停止 PIM 协议会话 BootStrap

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Stop Boot Strap | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pim.**pim_stop_register**(*Sessions*)

PIM 协议会话停止发送 Register 消息

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pim Stop Register | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pim.**wait_pim_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 PIM 协议会话达到指定状态

参数

- **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 PIM 协议会话达到的状态, 类型为：string, 默认值：达到 NEIGHBOR, 支持下列状态：

  DISABLED

  HELLO

  NEIGHBOR

  IDLE

  NOTSTARTED
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Pim State | Sessions=${Sessions} | State=NEIGHBOR | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.pppoe module

TesterLibrary.Protocol.pppoe.**abort_pppoe**(*Sessions*)

中断 PPPoE 协议会话

参数 **Sessions** (list (PppoeClent)) -- PPPoE 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Abort Pppoe | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pppoe.**connect_pppoe**(*Sessions*)

连接 PPPoE 协议会话

参数 **Sessions** (list (PppoeClent)) -- PPPoE 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Connect Pppoe | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pppoe.**create_pppoe**(*Port*, *EmulationMode='CLIENT'*, ***kwargs*)

创建 PPPoE 协议会话对象

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：object
- **EmulationMode** (*str*) -- PPPoE 角色, 默认值：CLIENT, 取值范围：

  CLIENT SERVER

关键字参数

- **Name** (*str*) -- PPPoE 协会话名称
- **Enable** (*bool*) -- 使能 PPPoE 协议会话, 默认值：True
- **AuthenticationType** (*str*) -- 认证方式, 默认值：NO_AUTHENTICATION, 取值范围：

  NO_AUTHENTICATION NEGOTIATION CHAP_MD5 PAP
- **Username** (*str*) -- 用户名, 默认值：xinertel, 取值范围：string length in [1,126]
- **Password** (*str*) -- 密码, 默认值：xinertel, 取值范围：string length in [1,126]
- **ServiceName** (*str*) -- 服务名, 默认值：””, 取值范围：string length in [0,255]
- **EnableMaxPayloadTag** (*bool*) -- 使能最大净荷标签, 默认值：False
- **MaxPayloadBytes** (*int*) -- 最大净荷 (字节), 取值范围：1-65535, 默认值：1500

- **LcpConfigReqTimeout** (*int*) -- LCP Configure-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **LcpConfigReqMaxAttempts** (*int*) -- LCP Configure-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **LcpTermReqTimeout** (*int*) -- LCP Terminate-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **LcpTermReqMaxAttempts** (*int*) -- LCP Terminate-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **NcpConfigReqTimeout** (*int*) -- NCP Configure-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **NcpConfigReqMaxAttempts** (*int*) -- NCP Configure-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **LcpNcpMaxNak** (*int*) -- LCP/NCP 最大 Nak 数量, 取值范围：1-65535, 默认值：5
- **EnableMruNegotiation** (*bool*) -- 使能 MRU 协商, 默认值：True
- **MruSize** (*int*) -- MRU(字节), 取值范围：128-65535, 默认值：1492
- **EnableEchoRequest** (*bool*) -- 使能 Echo-Request 报文, 默认值：False
- **EchoRequestInterval** (*int*) -- Echo-Request 间隔 (sec), 取值范围：1-65535, 默认值：10
- **EchoRequestMaxAttempts** (*int*) -- Echo-Request 最大尝试次数, 取值范围：1-65535, 默认值：3
- **EnableMagicNumber** (*bool*) -- 使能 Magic Number, 默认值：True
- **PadiTimeout** (*int*) -- Client 参数, PADI 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PadiMaxAttempts** (*int*) -- Client 参数, PADI 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PadrTimeout** (*int*) -- Client 参数, PADR 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PadrMaxAttempts** (*int*) -- Client 参数, PADR 最大尝试次数, 取值范围：1-65535, 默认值：10
- **EnableRelayAgent** (*bool*) -- Client 参数, 启用中继代理, 默认值：False
- **RelayAgentDestMac** (*str*) -- Client 参数, 中继代理 MAC 地址, 取值范围：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **RelayAgentDestMacStep** (*str*) -- Client 参数, 中继代理 MAC 地址跳变, 取值范围：有效的 mac 地址, 默认值：00:00:00:00:00:01
- **UseRelayAgentPadi** (*bool*) -- Client 参数, PADI 中包含中继代理信息, 默认值：True
- **UseRelayAgentPadr** (*bool*) -- Client 参数, PADR 中包含中继代理信息, 默认值：True
- **RelayAgentType** (*str*) -- Client 参数, 中继代理类型, 默认值：RFC2516, 取值范围：

  RFC2516 DSL_FORUM
- **RelaySessionId** (*str*) -- Client 参数, 中继会话 ID, 取值范围：string length in [0,12], 默认值：""
- **CircuitId** (*str*) -- Client 参数, 环路 ID, 取值范围：string length in [0,63], 默认值：@s

- **RemoteId** (*str*) -- Client 参数, 远程 ID, 取值范围：string length in [0,63], 默认值：@m-@p
- **ChapChalReqTimeout** (*int*) -- Client 参数, CHAP Challenge Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapAckTimeout** (*int*) -- Client 参数, CHAP Ack 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapMaxReplyAttempts** (*int*) -- Client 参数, CHAP Reply 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PapReqTimeout** (*int*) -- Client 参数, PAP Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PapReqMaxAttempts** (*int*) -- Client 参数, PAP Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **EnableAutoRetry** (*bool*) -- Client 参数, 使能 PPPoE 协议会话, 默认值：False
- **AutoRetryCount** (*int*) -- Client 参数, 重连次数, 取值范围：1-65535, 默认值：65535
- **LcpDelay** (*int*) -- Client 参数, LCP 推迟时间 (ms), 取值范围：1-65535, 默认值：0
- **EnableAutoFillIpv6** (*bool*) -- Client 参数, 启用获取 Global IPv6 地址, 默认值：True
- **AcName** (*str*) -- Server 参数, 访问集中器名称, 默认值：Xinertel
- **ChapReplyTimeout** (*int*) -- Server 参数, CHAP Reply 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapMaxChalAttempts** (*int*) -- Server 参数, CHAP Challenge 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PapPeerReqTimeout** (*int*) -- Server 参数, 等待 PAP Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **Ipv4Start** (*str*) -- Server 参数, IPv4 起始地址, 默认值：192.0.1.0
- **Ipv4Step** (*str*) -- Server 参数, IPv4 地址步长, 默认值：0.0.0。1
- **Ipv4Count** (*int*) -- Server 参数, IPv4 地址数量, 取值范围：1-65535, 默认值：3
- **Ipv6InterfaceId** (*str*) -- Server 参数, 起始 Interface ID, 默认值：”::2”
- **Ipv6InterfaceIdStep** (*str*) -- Server 参数, Interface ID 跳变步长, 默认值：”::1”
- **Ipv6PrefixStart** (*str*) -- Server 参数, IPv6 起始前缀, 默认值：”2002::”
- **Ipv6PrefixStep** (*str*) -- Server 参数, IPv6 前缀跳变步长, 默认值：”0:0:0:1::”
- **Ipv6Count** (*int*) -- Server 参数, IPv6 前缀数量, 取值范围：1-65535, 默认值：1
- **EnableForceConnectMode** (*bool*) -- 强制重连模式, 默认值：False
- **UnconnectedSessionThreshold** (*int*) -- 未连接会话门限值, 取值范围：1-65535, 默认值：1
- **MAndOFlag** (*str*) -- Server 参数, M 与 O 标志位, 默认值：M0_O0, 支持 M0_O0 M0_O1 M1

返回 PPPoE 协议会话对象, 类型：object

返回类型 (PppoeClent)

实际案例

```
| Create Pppoe | Port=${Port} |
| Create Pppoe | Port=${Port} | EmulationMode=Server |
```

TesterLibrary.Protocol.pppoe.**create_pppoe_custom_option**(*Session*, ***kwargs*)

编辑 PPPoE 自定义选项

参数 **Session** (PppoeClent) or (PppoeServer) -- PPPoE 协议会话对象

关键字参数

- **OptionValue** (*int*) -- 选项标识符, 默认值: 0, 取值范围: 0-65535
- **SubPortocolType** (*str*) -- 包含选项的消息类型, 默认值：1 sec, 支持类型:

  LinkControlProtocol

  IPControlProtocol

  IPv6ControlProtocol

  PPPoEPADIandPADR
- **UseWildcards** (*bool*) -- 使用通配符, 默认值：False
- **StringIsHexadecimal** (*int*) -- 使能十六进制字符, 默认值：False
- **OptionData** (*str*) -- 十进制选项载荷
- **OptionHexData** (*int*) -- 十六进制选项载荷

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Create Pppoe Custom Option | Sessions=${Sessions} | OptionValue=1 |␣
↪SubPortocolType=LinkControlProtocol | OptionData=55 |
```

TesterLibrary.Protocol.pppoe.**disconnect_pppoe**(*Sessions*)

断开 PPPoE 协议会话

参数 **Sessions** (list (PppoeClent)) -- PPPoE 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Disconnect Pppoe | Sessions=${Sessions} |
```

TesterLibrary.Protocol.pppoe.**edit_pppoe_clinet**(*Session*, ***kwargs*)

创建 PPPoE 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- PPPoE 协会话名称

- **Enable** (*bool*) -- 使能 PPPoE 协议会话, 默认值：True
- **EmulationMode** (*str*) -- PPPoE 角色, 默认值：CLIENT, 取值范围：

  CLIENT SERVER PPPoL2TP
- **AuthenticationType** (*str*) -- 认证方式, 默认值：NO_AUTHENTICATION, 取值范围：

  NO_AUTHENTICATION NEGOTIATION CHAP_MD5 PAP
- **Username** (*str*) -- 用户名, 默认值：xinertel, 取值范围：string length in [1,126]
- **Password** (*str*) -- 密码, 默认值：xinertel, 取值范围：string length in [1,126]
- **ServiceName** (*str*) -- 服务名, 默认值：””, 取值范围：string length in [0,255]
- **EnableMaxPayloadTag** (*bool*) -- 使能最大净荷标签, 默认值：False
- **MaxPayloadBytes** (*int*) -- 最大净荷 (字节), 取值范围：1-65535, 默认值：1500
- **LcpConfigReqTimeout** (*int*) -- LCP Configure-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **LcpConfigReqMaxAttempts** (*int*) -- LCP Configure-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **LcpTermReqTimeout** (*int*) -- LCP Terminate-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **LcpTermReqMaxAttempts** (*int*) -- LCP Terminate-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **NcpConfigReqTimeout** (*int*) -- NCP Configure-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **NcpConfigReqMaxAttempts** (*int*) -- NCP Configure-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **LcpNcpMaxNak** (*int*) -- LCP/NCP 最大 Nak 数量, 取值范围：1-65535, 默认值：5
- **EnableMruNegotiation** (*bool*) -- 使能 MRU 协商, 默认值：True
- **MruSize** (*int*) -- MRU(字节), 取值范围：128-65535, 默认值：1492
- **EnableEchoRequest** (*bool*) -- 使能 Echo-Request 报文, 默认值：False
- **EchoRequestInterval** (*int*) -- Echo-Request 间隔 (sec), 取值范围：1-65535, 默认值：10
- **EchoRequestMaxAttempts** (*int*) -- Echo-Request 最大尝试次数, 取值范围：1-65535, 默认值：3
- **EnableMagicNumber** (*bool*) -- 使能 Magic Number, 默认值：True
- **PadiTimeout** (*int*) -- Client 参数, PADI 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PadiMaxAttempts** (*int*) -- Client 参数, PADI 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PadrTimeout** (*int*) -- Client 参数, PADR 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PadrMaxAttempts** (*int*) -- Client 参数, PADR 最大尝试次数, 取值范围：1-65535, 默认值：10

- **EnableRelayAgent** (*bool*) -- Client 参数, 启用中继代理, 默认值：False
- **RelayAgentDestMac** (*str*) -- Client 参数, 中继代理 MAC 地址, 取值范围：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **RelayAgentDestMacStep** (*str*) -- Client 参数, 中继代理 MAC 地址跳变, 取值范围：有效的 mac 地址, 默认值：00:00:00:00:00:01
- **UseRelayAgentPadi** (*bool*) -- Client 参数, PADI 中包含中继代理信息, 默认值：True
- **UseRelayAgentPadr** (*bool*) -- Client 参数, PADR 中包含中继代理信息, 默认值：True
- **RelayAgentType** (*str*) -- Client 参数, 中继代理类型, 默认值：RFC2516, 取值范围：

  RFC2516 DSL_FORUM
- **RelaySessionId** (*str*) -- Client 参数, 中继会话 ID, 取值范围：string length in [0,12], 默认值：""
- **CircuitId** (*str*) -- Client 参数, 环路 ID, 取值范围：string length in [0,63], 默认值：@s
- **RemoteId** (*str*) -- Client 参数, 远程 ID, 取值范围：string length in [0,63], 默认值：@m-@p
- **ChapChalReqTimeout** (*int*) -- Client 参数, CHAP Challenge Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapAckTimeout** (*int*) -- Client 参数, CHAP Ack 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapMaxReplyAttempts** (*int*) -- Client 参数, CHAP Reply 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PapReqTimeout** (*int*) -- Client 参数, PAP Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PapReqMaxAttempts** (*int*) -- Client 参数, PAP Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **EnableAutoRetry** (*bool*) -- Client 参数, 使能 PPPoE 协议会话, 默认值：False
- **AutoRetryCount** (*int*) -- Client 参数, 重连次数, 取值范围：1-65535, 默认值：65535
- **LcpDelay** (*int*) -- Client 参数, LCP 推迟时间 (ms), 取值范围：1-65535, 默认值：0
- **EnableAutoFillIpv6** (*bool*) -- Client 参数, 启用获取 Global IPv6 地址, 默认值：True
- **AcName** (*str*) -- Server 参数, 访问集中器名称, 默认值：Xinertel
- **ChapReplyTimeout** (*int*) -- Server 参数, CHAP Reply 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapMaxChalAttempts** (*int*) -- Server 参数, CHAP Challenge 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PapPeerReqTimeout** (*int*) -- Server 参数, 等待 PAP Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **Ipv4Start** (*str*) -- Server 参数, IPv4 起始地址, 默认值：192.0.1.0
- **Ipv4Step** (*str*) -- Server 参数, IPv4 地址步长, 默认值：0.0.0。1

- **Ipv4Count** (*int*) -- Server 参数, IPv4 地址数量, 取值范围：1-65535, 默认值：3
- **Ipv6InterfaceId** (*str*) -- Server 参数, 起始 Interface ID, 默认值：”::2”
- **Ipv6InterfaceIdStep** (*str*) -- Server 参数, Interface ID 跳变步长, 默认值：”::1”
- **Ipv6PrefixStart** (*str*) -- Server 参数, IPv6 起始前缀, 默认值：”2002::”
- **Ipv6PrefixStep** (*str*) -- Server 参数, IPv6 前缀跳变步长, 默认值：”0:0:0:1::”
- **Ipv6Count** (*int*) -- Server 参数, IPv6 前缀数量, 取值范围：1-65535, 默认值：1
- **EnableForceConnectMode** (*bool*) -- 强制重连模式, 默认值：False
- **UnconnectedSessionThreshold** (*int*) -- 未连接会话门限值, 取值范围：1-65535, 默认值：1
- **MAndOFlag** (*str*) -- Server 参数, M 与 O 标志位, 默认值：M0_O0, 支持

  M0_O0 M0_O1 M1

返回 PPPoE 协议会话对象, 类型：object

返回类型 (PppoeClent)

实际案例

```
| Create Pppoe | Port=${Port} |
```

TesterLibrary.Protocol.pppoe.**get_pppoe_client_block_statistic**(*Session*, *StaItems=None*)

获取 PPPoE Server Block Statistic 统计结果

参数

- **Session** (PppoeClient) -- 测试仪表端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取 PPPoE Client Block Statistic 统计项目，类型为：list，目前支持的统计项

  IpcpState

  Ipv6cpState

  SessionCount

  SessionsUp

  SessionsRetried

  AttemptedConnects

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

  MaxSetupTime

  MinSetupTime

  AverageSetupTime

SuccessfulSetupRate
TxPadi
RxPado
TxPadr
RxPads
TxPadt
RxPadt
TxLcpConfigRequest
RxLcpConfigRequest
TxLcpConfigAck
RxLcpConfigAck
TxLcpConfigNak
RxLcpConfigNak
TxLcpConfigReject
RxLcpConfigReject
TxLcpEchoRequest
RxLcpEchoRequest
TxLcpEchoReply
RxLcpEchoReply
TxLcpTerminateRequest
RxLcpTerminateRequest
TxLcpTerminateAck
RxLcpTerminateAck
TxChap
RxChap
TxPap
RxPap
TxIpcp
RxIpcp
TxIpv6cp
RxIpv6cp
TxIpv4
RxIpv4
TxIpv6
RxIpv6

返回

eg:

```
{
    'SessionCount': 10,
    'SessionsUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionCount | SessionsUp |
| Subscribe Result | Types=PppoeClientBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Client Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pppoe.**get_pppoe_client_statistic**(*Session*, *Index=1*, *StaItems=None*)

获取 PPPoE Client Statistic 统计结果

参数

- **Session** (PppoeClient) -- 测试仪表端口对象, 类型为：Object
- **Index** (*int*) -- PppoeClient Block 里会话的 index, 默认值为: 1
- **StaItems** (*list*) -- 需要获取 PPPoE Client Block Statistic 统计项目，类型为：list，目前支持的统计项

  IpcpState

  Ipv6cpState

  MacAddress

  PeerMacAddress

  PppoeSessionId

  VlanId

  InnerVlanId

  Ipv4Address

  PeerIpv4Address

  Ipv6LinklocalAddress

  PeerIpv6LinklocalAddress

  Ipv6GlobalAddress

  SessionsRetried

  AttemptedConnects

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

  SetupTime

  TxPadi

RxPado

TxPadr

RxPads

TxPadt

RxPadt

TxLcpConfigRequest

RxLcpConfigRequest

TxLcpConfigAck

RxLcpConfigAck

TxLcpConfigNak

RxLcpConfigNak

TxLcpConfigReject

RxLcpConfigReject

TxLcpEchoRequest

RxLcpEchoRequest

TxLcpEchoReply

RxLcpEchoReply

TxLcpTerminateRequest

RxLcpTerminateRequest

TxLcpTerminateAck

RxLcpTerminateAck

TxChap

RxChap

TxPap

RxPap

TxIpcp

RxIpcp

TxIpv6cp

RxIpv6cp

TxIpv4

RxIpv4

TxIpv6

RxIpv6

返回

eg:

```
{
    'SessionCount': 10,
    'SessionsUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionCount | SessionsUp |
| Subscribe Result | Types=PppoeClientStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Client Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pppoe.**get_pppoe_port_statistic**(*Port*, *StaItems=None*)

获取 PPPoE Port Statistic 统计结果

参数

- **Port** (Port) -- 测试仪表端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取 PPPoE Port Statistic 统计项目, 类型为：list, 目前支持的统计项

  SessionBlockCount

  SessionCount

  SessionsUp

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

返回

eg:

```
{
    'SessionBlockCount': 10,
    'SessionCount': 100,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionBlockCount | SessionCount |
| Subscribe Result | Types=PppoePortStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pppoe.**get_pppoe_server_block_statistic**(*Session*, *StaItems=None*)

获取 PPPoE Server Block Statistic 统计结果

参数

- **Session** (PppoeServer) -- 测试仪表端口对象, 类型为：Object

- **StaItems** (*list*) -- 需要获取 PPPoE Server Block Statistic 统计项目，类型为：list，目前支持的统计项

  IpcpState

  Ipv6cpState

  SessionCount

  SessionsUp

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

  MaxSetupTime

  MinSetupTime

  AverageSetupTime

  SuccessfulSetupRate

  RxPadi

  TxPado

  RxPadr

  TxPads

  TxPadt

  RxPadt

  TxLcpConfigRequest

  RxLcpConfigRequest

  TxLcpConfigAck

  RxLcpConfigAck

  TxLcpConfigNak

  RxLcpConfigNak

  TxLcpConfigReject

  RxLcpConfigReject

  TxLcpEchoRequest

  RxLcpEchoRequest

  TxLcpEchoReply

  RxLcpEchoReply

  TxLcpTerminateRequest

  RxLcpTerminateRequest

  TxLcpTerminateAck

  RxLcpTerminateAck

  TxChap

  RxChap

  TxPap

RxPap

TxIpcp

RxIpcp

TxIpv6cp

RxIpv6cp

TxIpv4

RxIpv4

TxIpv6

RxIpv6

返回

eg:

```
{
    'SessionCount': 10,
    'SessionsUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionCount | SessionsUp |
| Subscribe Result | Types=PppoeServerBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Server Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pppoe.**get_pppoe_server_statistic**(*Session*, *Index=1*, *StaItems=None*)

获取 PPPoE Server Statistic 统计结果

参数

- **Session** (PppoeServer) -- 测试仪表端口对象, 类型为：Object
- **Index** (*int*) -- PppoeServer Block 里会话的 index, 默认值为: 1
- **StaItems** (*list*) -- 需要获取 PPPoE Server Statistic 统计项目，类型为: list，目前支持的统计项

  IpcpState

  Ipv6cpState

  MacAddress

  PeerMacAddress

  PppoeSessionId

  VlanId

  InnerVlanId

  Ipv4Address

  PeerIpv4Address

Ipv6LinklocalAddress
PeerIpv6LinklocalAddress
SuccessfulConnects
FailedConnects
SucessfulDisconnects
FailedDisconnects
SetupTime
RxPadi
TxPado
RxPadr
TxPads
TxPadt
RxPadt
TxLcpConfigRequest
RxLcpConfigRequest
TxLcpConfigAck
RxLcpConfigAck
TxLcpConfigNak
RxLcpConfigNak
TxLcpConfigReject
RxLcpConfigReject
TxLcpEchoRequest
RxLcpEchoRequest
TxLcpEchoReply
RxLcpEchoReply
TxLcpTerminateRequest
RxLcpTerminateRequest
TxLcpTerminateAck
RxLcpTerminateAck
TxChap
RxChap
TxPap
RxPap
TxIpcp
RxIpcp
TxIpv6cp
RxIpv6cp
TxIpv4
RxIpv4

TxIpv6

RxIpv6

返回

eg:

```
{
    'SessionCount': 10,
    'SessionsUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionCount | SessionsUp |
| Subscribe Result | Types=PppoeServerStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Server Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.pppoe.**wait_pppoe_ipcp_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 PPPoE IPCP 达到指定状态

参数

- **Sessions** (list (PppoeClent)) -- PPPoE 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 PPPoE IPCP 达到的状态, 类型为：string, 默认值：达到 CONNECTED, 支持下列状态：

  NONE

  IDLE

  CONNECTED

  CONNECTING

  DISCONNECTING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Pppoe State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.pppoe.**wait_pppoe_ipv6cp_state**(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 PPPoE IPv6CP 达到指定状态

参数

- **Sessions** (list (PppoeClent)) or (list (Pppoev3Router)) -- PPPoE 或 Pppoev3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 PPPoE IPv6CP 达到的状态, 类型为：string, 默认值：达到 CONNECTED, 支持下列状态：

  NONE

  IDLE

  CONNECTED

  CONNECTING

  DISCONNECTING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Pppoe State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.rip module

TesterLibrary.Protocol.rip.**advertise_rip**(*Sessions*)

通告 RIP 协议路由

参数 **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Rip | Sessions=${Sessions} |
```

TesterLibrary.Protocol.rip.**create_rip**(*Port*, ***kwargs*)

创建 RIP 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- RIP 协议会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 RIP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- RIP 版本, 类型为：string, 默认值：RIPV2, 支持版本：

  RIPV1

  RIPV2

  RIPNG
- **UpdateType** (*str*) -- 仿真路由器指定发送 RIP 消息的通信方式, 类型为：string, 默认值：MULTICAST, 支持方式：

  BROADCAST

  MULTICAST

  UNICAST
- **DutIpv4Address** (*str*) -- 指定接收 RIP 消息的路由器的 IP 地址, 当 RIP 版本为 RIPv1 或者 RIPv2 时，该选项可配。类型为：string, 默认值：224.0.0.9
- **DutIpv6Address** (*str*) -- 指定接收 RIP 消息的路由器的 IP 地址, 当 RIP 版本为 RIPng 并且更新类型指定为 Unicast 时，该选项可配。类型为：string, 默认值：ff02::9
- **AuthMethod** (*str*) -- 认证方式, 当 RIP 版本为 RIPv2 时配置该选项。类型为：string, 默认值：NONE, 支持方式：

  NONE

  SIMPLE

  MD5
- **Password** (*str*) -- 当认证方式为 Simple/MD5 时，输入的认证密码, 类型为：string, 默认值：Xinetel
- **Md5KeyId** (*int*) -- 当认证方式为 MD5 时，输入的 MD5 密钥, 类型为：number, 取值范围：0-255, 默认值：1
- **UpdateInterval** (*int*) -- 发送 RIP 更新消息的时间间隔，单位为秒, 类型为：number, 取值范围：1-65535, 默认值：30
- **UpdateJitter** (*int*) -- 发送 RIP 更新消息的时间抖动, 类型为：number, 取值范围：0-5, 默认值：0
- **MaxRoutePerUpdate** (*int*) -- 更新消息中可携带的最大路由数, 类型为：number, 取值范围：1-70, 默认值：25
- **SplitHorizon** (*bool*) -- 是否开启水平分割功能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableViewRoutes** (*bool*) -- 是否需要查看学到的路由信息, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **EnableIpAddrValidation** (*bool*) -- 验证收到的 IP 地址是否和本地地址在同一网段, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 RIP 协议会话对象, 类型：object

返回类型 (RipRouter)

实际案例

```
| Create Rip | Port=${Port} | EnableIpAddrValidation=True |
```

TesterLibrary.Protocol.rip.**create_rip_ipv4_route**(*Session*, ***kwargs*)

创建 RIP IPv4 路由对象

参数 **Session** (RipRouter) -- Rip 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- RIP IPv4 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 RIP IPv4 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteCount** (*str*) -- 路由池中包含的路由的个数, 类型为：string, 取值范围：1-16777215, 默认值：1
- **StartIpv4Prefix** (*str*) -- 指定起始 IPv4 地址, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：192.168.1.0
- **PrefixLength** (*int*) -- 地址前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **Increment** (*str*) -- 增量步长, 类型为：string, 取值范围：1-255, 默认值：1
- **NextHop** (*str*) -- 指定路由下一跳, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：0.0.0.0
- **Metric** (*int*) -- 路由度量, 16 表示不可达。类型为：number, 取值范围：1-16, 默认值：1
- **RouteTag** (*int*) -- 路由标签域的值, 0 表示没有 tag. 类型为：number, 取值范围：0-65535, 默认值：0

返回 RIP IPv4 路由对象, 类型：object

返回类型 (RipIpv4RouteConfig)

实际案例

```
| ${Session} | Create Rip | Port=${Port} |
| Create Rip Ipv4 Route | Session=${Session} | Metric=10 |
```

TesterLibrary.Protocol.rip.**create_rip_ipv6_route**(*Session*, ***kwargs*)

创建 RIP IPv6 路由对象

参数 **Session** (RipRouter) -- Rip 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- RIP IPv6 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 RIP IPv6 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **RouteCount** (*int*) -- 路由池中包含的路由的个数, 类型为：number, 取值范围：1-2147483647, 默认值：1
- **StartIpv6Prefix** (*str*) -- 起始 IPv6 地址, 类型为：string, 取值范围：有效的 IPv6 地址, 默认值：'2000::'
- **RouteStep** (*str*) -- IP 地址的增加步长, 类型为：string, 取值范围：有效的 IPv6 地址, 默认值：'0:0:0:1::'
- **PrefixLength** (*int*) -- 地址的前缀长度, 类型为：number, 取值范围：1-128, 默认值：64
- **NextHop** (*str*) -- 路由下一跳, 类型为：string, 取值范围：有效的 IPv6 地址, 默认值：'::'
- **Metric** (*int*) -- 路由度量, 类型为：number, 取值范围：1-16, 默认值：1
- **RouteTag** (*int*) -- 路由标签域的值, 0 表示没有 tag, 类型为：number, 取值范围：0-65535, 默认值：0

返回 RIP IPv6 路由对象, 类型：object

返回类型 (RipIpv6RouteConfig)

实际案例

```
| ${Session} | Create Rip | Port=${Port} |
| Create Rip Ipv6 Route | Session=${Session} |
```

TesterLibrary.Protocol.rip.**edit_rip**(*Session*, ***kwargs*)

编辑 Rip 协议会话对象参数

参数 **Session** (RipRouter) -- Rip 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- RIP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 RIP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- RIP 版本, 类型为：string, 默认值：RIPV2, 支持版本：

  RIPV1

  RIPV2

  RIPNG
- **UpdateType** (*str*) -- 仿真路由器指定发送 RIP 消息的通信方式, 类型为：string, 默认值：MULTICAST, 支持方式：

  BROADCAST

  MULTICAST

  UNICAST
- **DutIpv4Address** (*str*) -- 指定接收 RIP 消息的路由器的 IP 地址, 当 RIP 版本为 RIPv1 或者 RIPv2 时，该选项可配。类型为：string, 默认值：224.0.0.9
- **DutIpv6Address** (*str*) -- 指定接收 RIP 消息的路由器的 IP 地址, 当 RIP 版本为 RIPng 并且更新类型指定为 Unicast 时，该选项可配。类型为：string, 默认值：ff02::9
- **AuthMethod** (*str*) -- 认证方式, 当 RIP 版本为 RIPv2 时配置该选项。类型为：string, 默认值：NONE, 支持方式：

  NONE

SIMPLE

MD5

- **Password** (*str*) -- 当认证方式为 Simple/MD5 时，输入的认证密码, 类型为：string, 默认值：Xinetel
- **Md5KeyId** (*int*) -- 当认证方式为 MD5 时，输入的 MD5 密钥, 类型为：number, 取值范围：0-255, 默认值：1
- **UpdateInterval** (*int*) -- 发送 RIP 更新消息的时间间隔，单位为秒, 类型为：number, 取值范围：1-65535, 默认值：30
- **UpdateJitter** (*int*) -- 发送 RIP 更新消息的时间抖动, 类型为：number, 取值范围：0-5, 默认值：0
- **MaxRoutePerUpdate** (*int*) -- 更新消息中可携带的最大路由数, 类型为：number, 取值范围：1-70, 默认值：25
- **SplitHorizon** (*bool*) -- 是否开启水平分割功能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableViewRoutes** (*bool*) -- 是否需要查看学到的路由信息, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableIpAddrValidation** (*bool*) -- 验证收到的 IP 地址是否和本地地址在同一网段, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Rip | Session=${Session} | EnableViewRoutes=True |
```

TesterLibrary.Protocol.rip.**edit_rip_port_config**(*Ports*, ***kwargs*)

修改 RIP 端口统计对象

参数 **Ports** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数 **UpdateRoutesTransmitRate** (*int*) -- RIP Tx Rate (messages/sec) , 取值范围：1-1000000000, 默认值：1000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Rip Port Config | Ports=${Ports} | UpdateRoutesTransmitRate=100 |
```

TesterLibrary.Protocol.rip.**get_rip_router_from_route**(*Route*)

获取 OSPF LSA 对应的绑定流源或目的端点对象

参数 **Route** (Port) -- Rip Ipv4 / Ipv6 Route 对象, 类型为：object

返回 Rip Route 对应的绑定流源或目的端点对象, 类型：object

实际案例

```
| ${Session} | Create Rip | Port=${Port} |
| ${RouterLsa} | Create Rip Ipv4 Router | Session=${Session} | Age=20 |
| ${Point} | Get Rip Router From Route | Route=${RouterLsa} |
```

TesterLibrary.Protocol.rip.**get_rip_session_block_statistic**(*Session*, *StaItems: Optional[list] = None*)

获取 RIP 协议会话统计结果

参数

- **Session** (RipRouter) -- RIP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  SessionCount

  TxAdvertised

  RxAdvertised

  TxWithdrawn

  RxWithdrawn

返回

eg:

```
{
    'TxAdvertised': 10,
    'RxAdvertised': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv2SessionResultPropertySet |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Rip Session Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.rip.**get_rip_session_statistic**(*Session*, *SessionId*, *StaItems=None*)

获取 RIP 协议会话统计结果

参数

- **Session** (RipRouter) -- RIP 协议会话对象, 类型为：Object
- **SessionId** (*int*) -- RIP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

SessionId

SessionState

TxAdvertised

RxAdvertised

TxWithdrawn

RxWithdrawn

返回

eg:

```
{
    'TxAdvertised': 10,
    'RxAdvertised': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv2SessionResultPropertySet |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Rip Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Protocol.rip.**resume_rip**(*Sessions*)

恢复 RIP 协议

参数 **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Rip | Sessions=${Sessions} |
```

TesterLibrary.Protocol.rip.**suspend_rip**(*Sessions*)

暂停 RIP 协议

参数 **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Suspend Rip | Sessions=${Sessions} |
```

TesterLibrary.Protocol.rip.**wait_rip_state**(*Sessions*, *State='OPEN'*, *Interval=1*, *TimeOut=60*)

等待 RIP 协议会话达到指定状态

参数

- **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 RIP 协议会话达到的状态, 类型为：string, 默认值：达到 OPEN, 支持下列状态：

  DISABLED

  NOTSTART

  CLOSED

  OPEN

  SUSPENDED
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Rip State | Sessions=${Sessions} | State=OPEN | Interval=2 |␣
↪TimeOut=120 |
```

TesterLibrary.Protocol.rip.**withdraw_rip**(*Sessions*)

撤销 RIP 协议通告路由

参数 **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Rip | Sessions=${Sessions} |
```

TesterLibrary.Protocol.vxlan module

TesterLibrary.Protocol.vxlan.**binding_vxlan_multicast_group**(*Segments*, *MulticastGroups*)

创建 Vxlan Multicast Group 对象

参数

- **Segments** (VxlanSegmentConfig) -- Vxlan Segment 对象, 类型：object
- **MulticastGroups** (Ipv4MulticastGroup) -- Vxlan Multicast Group 对象, 类型：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Binding Vxlan Multicast Group | Segments=${Segments} | MulticastGroups=$
↪{MulticastGroups} |
```

TesterLibrary.Protocol.vxlan.**binding_vxlan_vm**(*Segments*, *Vms*)

绑定 Vxlan Vm 对象

参数

- **Segments** (VxlanSegmentConfig) -- Vxlan Segment 对象, 类型：object
- **Vms** ('VxlanVmProperty') -- Vxlan Vm 对象, 类型：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Binding Vxlan Vm | Segments=${Segments} | Vms=${Vms} |
```

TesterLibrary.Protocol.vxlan.**binding_vxlan_vtep**(*Vteps*, *Vms*)

绑定 Vxlan Vtep 对象

参数

- **Vteps** (Vxlan) -- Vxlan 协议会话对象, 类型：object
- **Vms** ('VxlanVmProperty') -- Vxlan Vm 对象, 类型：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Binding Vxlan Vtep | Vteps=${Vxlan} | Vms=${Vms} |
```

TesterLibrary.Protocol.vxlan.**create_vxlan**(*Port*, ***kwargs*)

创建 Vxlan 协议会话对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- Vxlan 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 Vxlan 协议会话, 默认值：True
- **AutoUdpSourcePort** (*bool*) -- 自动计算 UDP 源端口, 默认值：True
- **UdpSourcePort** (*int*) -- 配置 UDP 源端口, 取值范围：3-4095, 默认值：1025
- **EnableUdpChecksum** (*bool*) -- 使能计算 UDP 校验和, 默认值：False
- **EvpnLearning** (*bool*) -- 使能 EVPN 学习, 默认值：False
- **OvsdbLearning** (*bool*) -- 使能 OVSDB 学习, 默认值：False
- **MulticastType** (*str*) -- 组播类型, 默认值：IGMP, 取值范围：

  IGMP

  PIM

  MLD
- **VtepTunnelIp** (*str*) -- VTEP 隧道 IP 地址, 默认值：INTERFACEIP, 取值范围：

  INTERFACEIP

  ROUTERID
- **EnableIrb** (*bool*) -- 默认值：False
- **RPAddress** (*str*) -- 选择 PIM 的 RP 地址, 取值范围：IPv4 地址, 默认值：192.0.0.1
- **RPIpv6Address** (*str*) -- 选择 PIM 的 RP 地址, 取值范围：IPv6 地址, 默认值：2000::1
- **IrbMode** (*str*) -- 默认值：Symmetric, 取值范围：

  Symmetric

返回 Vxlan 协议会话对象, 类型：object

返回类型 (Vxlan)

实际案例

```
| Create Vxlan | Port=${Port} |
```

TesterLibrary.Protocol.vxlan.**create_vxlan_segment**(***kwargs*)

创建 Vxlan Segment 对象

参数 **Port** (Port) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- Vxlan 协会话名称, 类型为：string

- **Enable** (*bool*) -- 使能 Vxlan 协议会话, 默认值：True
- **StartVni** (*int*) -- 起始 VNI, 取值范围：0-16777215, 默认值：0
- **VniCount** (*int*) -- VNI 个数, 取值范围：1-65535, 默认值：1
- **VniStep** (*int*) -- VNI 跳变步长, 取值范围：1-65535, 默认值：1
- **CommunicationType** (*str*) -- 学习方式, 默认值：UNICAST, 取值范围：

  UNICAST

  MULTICAST

  VxlanEVPN
- **VniDistributionType** (*str*) -- VNI 在 VPN 之间的分配方式, 默认值：ROUNDROBIN, 取值范围：

  ROUNDROBIN

  LINEAR
- **EnableL3Vni** (*bool*) -- 使能 L3VNI, 默认值：False
- **StartL3Vni** (*int*) -- 起始 L3VNI, 取值范围：1-16777215, 默认值：1
- **L3VniStep** (*int*) -- L3VNI 跳变步长, 取值范围：1-16777215, 默认值：1
- **L3VniCount** (*int*) -- L3 VNI 数量, 取值范围：1-65535, 默认值：1
- **VniTrafficType** (*str*) -- 流端点模式, 默认值：ROUNDROBIN, 取值范围：

  L2VNI

  L3VNI

  L2AndL3VNI
- **EnableVmArp** (*bool*) -- 使能 VM ARP, 默认值：False

返回 Vxlan Segment 对象, 类型：object

返回类型 (VxlanSegmentConfig)

实际案例

```
| Create Vxlan Segment | Port=${Port} |
```

TesterLibrary.Protocol.vxlan.**edit_vxlan**(*Session*, ***kwargs*)

创建 Vxlan 协议会话对象

参数 **Session** (Vxlan) -- Vxlan 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- Vxlan 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 Vxlan 协议会话, 默认值：True
- **AutoUdpSourcePort** (*bool*) -- 自动计算 UDP 源端口, 默认值：True
- **UdpSourcePort** (*int*) -- 配置 UDP 源端口, 取值范围：3-4095, 默认值：1025
- **EnableUdpChecksum** (*bool*) -- 使能计算 UDP 校验和, 默认值：False
- **EvpnLearning** (*bool*) -- 使能 EVPN 学习, 默认值：False
- **OvsdbLearning** (*bool*) -- 使能 OVSDB 学习, 默认值：False

- **MulticastType** (*str*) -- 组播类型, 默认值：IGMP, 取值范围：

  IGMP

  PIM

  MLD

- **VtepTunnelIp** (*str*) -- VTEP 隧道 IP 地址, 默认值：INTERFACEIP, 取值范围：

  INTERFACEIP

  ROUTERID

- **EnableIrb** (*bool*) -- 默认值：False
- **RPAddress** (*str*) -- 选择 PIM 的 RP 地址, 取值范围：IPv4 地址, 默认值：192.0.0.1
- **RPIpv6Address** (*str*) -- 选择 PIM 的 RP 地址, 取值范围：IPv6 地址, 默认值：2000::1
- **IrbMode** (*str*) -- 默认值：Symmetric, 取值范围：

  Symmetric

返回 Vxlan 协议会话对, 类型：object

返回类型 (Vxlan)

实际案例

```
| Create Vxlan | Port=${Port} |
```

TesterLibrary.Protocol.vxlan.**get_vxlan_statistic**(*Session*, *StaItems=None*)

获取测试仪表 vxlan 统计

参数

- **Session** (Vxlan) -- Vxlan 协议会话对象, 类型为：object
- **StaItems** (*list*) -- 需要获取 Vxlan 统计项目, 类型为：list，目前支持的统计项

  VtepId: VXLAN 会话的名称

  VtepState: VXLAN 会话的状态

  TotalVmCount：VM 总数

  ResolvedVmCount: 已解析 VM

  UnresolvedVmCount: 未解析 VM

返回

```
{"TotalVmCount": 100, "ResolvedVmCount": 100}
```

返回类型 dict

实际案例

```
| Subscribe Result |
| Start Protocol |
| Sleep | 10 |
| Stop Protocol |
| Sleep | 3 |
| ${Port} | Get Ports |
| ${Session} | Get Session | Ports=@{Port} | Protocols=vxlan |
| ${StaItems} | Create List | TotalVmCount | ResolvedVmCount |
| &{Result} | Get Vxlan Statistic | Session=${Session} | StaItems=@{StaItems}
↪|
```

TesterLibrary.Protocol.vxlan.**get_vxlan_vm_property**(*interface*)

获取 Vxlan Vm Property 对象

参数 **Interface** (Interface) -- Interface 对象, 类型为：object

返回 Vxlan Vm Property 对象, 类型：object

返回类型 (VxlanVmProperty)

实际案例

```
| Get Vxlan Vm Property | Interface=${Interface} |
```

TesterLibrary.Protocol.vxlan.**start_vxlan_ping**(*Interfaces*, ***kwargs*)

启动 Vxlan Ping

参数 **Interfaces** (Interface) -- Interface 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Start Vxlan Ping | Interfaces=${Interface} |
```

TesterLibrary.Protocol.vxlan.**stop_vxlan_ping**(*Interfaces*, ***kwargs*)

停止 Vxlan Ping

参数 **Interfaces** (Interface) -- Interface 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Vxlan Ping | Interfaces=${Interface} |
```

Module contents

1.1.4 TesterLibrary.Statistic package

Submodules

TesterLibrary.Statistic.common module

TesterLibrary.Statistic.common.**clear_result**(*All=True*, *Objects=None*)

清除测试仪表统计

参数

- **All** (*bool*) -- 是否清除所有已经订阅的统计视图的数据, 默认位: True
- **Objects** (*list*) -- 指定需要清空视图的对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| Clear Result |
```

TesterLibrary.Statistic.common.**get_port_latency_statistic**(*Port*, *StaItems=None*, *Mode=True*)

获取测试仪表端口时延统计结果

参数

- **Port** (Port) -- 指定需要获取结果的端口对象 object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取端口统计项目, 目前支持的统计项

  PortID：端口名称

  MinLatency：最小时延

  MaxLatency：最大时延

  AvaLatency：平均时延

返回

eg:

```
{
    'MinLatency': 1.2311,
    'MaxLatency': 5.123,
}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | MaxLatency |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Port} | Get Ports |
| &{Result} | Get Port Latency Statistic | Port=${Port} | StaItems=@{StaItems}
↪ |
| Clear Result |
```

TesterLibrary.Statistic.common.**get_port_statistic**(*Port*, *StaItems=None*, *Mode=True*)

获取测试仪表端口统计结果

参数

- **Port** (Port) -- 指定需要获取结果的端口对象 object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取端口统计项目, 目前支持的统计项

  TxTotalFrames：发送报文总数

  RxTotalFrames：接收报文总数

  TxStreamFrames：发送流报文总数

  TxSignatureStreamFrames：发送带标签流报文总数

  RxSignatureStreamFrames：接收带标签流报文总数

  TxFrameRate：发送报文速率 (fps)

  RxFrameRate：接收报文速率 (fps)

  TxL1Rate：发送线速 (bps)

  RxL1Rate：接收线速 (bps)

  TxUtil：发送百分比 (%)

  RxUtil：接收百分比 (%)

  TxByteRate：发送字节速率 (Bps)

  RxByteRate：接收字节速率 (Bps)

  TxBitRate：发送比特速率 (bps)

  RxBitRate：接收比特速率 (bps)

  TxTotalBytes：发送字节总数

  RxTotalBytes：接收字节总数

  RxFCSErr：接收 FCS 错误数

  RxIpv4ChecksumError：接收 IPv4 Checksum Error 报文数

  RxTcpChecksumError：接收 TCP Checksum Error 报文数

  RxUdpChecksumError：接收 UDP Checksum Error 报文数

  RxPrbsFillBytes：接收 PRBS 填充字节数

  RxPrbsErrorBits：接收 PRBS 错误位数

RxPrbsErrorFrames：接收 PRBS 错误报文数

RxIpv4Frames：接收 IPv4 帧数

RxIpv6Frames：接收 IPv6 帧数

RxTcpFrames：接收 TCP 帧数

RxUdpFrames：接收 UDP 帧数

RxMplsCount：接收 MPLS 帧数

RxIcmpFrames：接收 ICMP 帧数

RxVlanFrames：接收 VLAN 帧数

RxFCoEFrames：接收 FCoE 帧数

RxPauseFrames：接收 Pause 帧数

RxUndersizeFrames：接收超短帧数

RxOversizeFrames：接收超长帧数

RxJumboFrames：接收巨型帧数

RxOutofSequenceCount：接收乱序帧数

RxFilter0Count：接收过滤帧数 _0

RxFilter1Count：接收过滤帧数 _1

RxFilter2Count：接收过滤帧数 _2

RxFilter3Count：接收过滤帧数 _3

RxFilter4Count：接收过滤帧数 _4

RxFilter5Count：接收过滤帧数 _5

RxFilter6Count：接收过滤帧数 _6

RxFilter7Count：接收过滤帧数 _7

RxPktLossCount：接收丢包帧数

RxInOrderCount：接收有序帧数

RxReorderCount：接收重排序帧数

RxRepeatFrameCount：接收重复帧数

RxPortLateCount：接收端口延迟帧数

RxCorrectedRSFECErrorsFramesCodewords：接收 Corrected RS FEC Error 帧数（codewords）

RxUncorrectedRSFECErrorsFramesCodewords：接收 Uncorrected RS FEC Error 帧数（codewords）

RxCorrectedBaseRFECErrorsFramesCodewords：接收 Corrected BaseR FEC Error 帧数（codewords）

RxUncorrectedBaseRFECErrorsFramesCodewords：接收 Uncorrected BaseR FEC Error 帧数（codewords）

TxCrcFrameCount：发送 CRC 帧数

TxErr3CheckFrameCount：发送 IP Checksum Error 报文数

TxErr4CheckFrameCount：发送 L4 Checksum Error 报文数

TxIpv4Count：发送 IPv4 帧数

TxIpv6Count：发送 IPv6 帧数

TxMplsCount：发送 MPLS 帧数
TxIpv4FrameCount：发送 IPv4 流帧数
TxIpv6FrameCount：发送 IPv6 流帧数
TxVlanFrameCount：发送 VLAN 流帧数
TxMplsFrameCount：发送 MPLS 流帧数
TxOversizeFrames：发送超长帧数
TxUndersizeFrames：发送超短帧数
TxJumboFrames：发送巨型帧数
RxPFCFrames：接收 PFC 帧数
RxPFCRate：接收 PFC 速率
RxPFCPriority0Frames：接收 PFC 优先级是 0 的帧数
RxPFCPriority1Frames：接收 PFC 优先级是 1 的帧数
RxPFCPriority2Frames：接收 PFC 优先级是 2 的帧数
RxPFCPriority3Frames：接收 PFC 优先级是 3 的帧数
RxPFCPriority4Frames：接收 PFC 优先级是 4 的帧数
RxPFCPriority5Frames：接收 PFC 优先级是 5 的帧数
RxPFCPriority6Frames：接收 PFC 优先级是 6 的帧数
RxPFCPriority7Frames：接收 PFC 优先级是 7 的帧数
TxPFCFrames：发送 PFC 帧数
TxPFCRate：发送 PFC 速率
TxPFCPriority0Frames：发送 PFC 优先级是 0 的帧数
TxPFCPriority1Frames：发送 PFC 优先级是 1 的帧数
TxPFCPriority2Frames：发送 PFC 优先级是 2 的帧数
TxPFCPriority3Frames：发送 PFC 优先级是 3 的帧数
TxPFCPriority4Frames：发送 PFC 优先级是 4 的帧数
TxPFCPriority5Frames：发送 PFC 优先级是 5 的帧数
TxPFCPriority6Frames：发送 PFC 优先级是 6 的帧数
TxPFCPriority7Frames：发送 PFC 优先级是 7 的帧数
RxARPFrames：接收 ARP 报文数
TxARPFrames：发送 ARP 报文数
RxBroadcastFrames：接收广播报文数
TxBroadcastFrames：发送广播报文数
RxIpv4LengthErrorFrames：接收 IPv4 长度错误帧数
RxUserDefinedCapture0Frames：接收自定义统计 0 报文数
RxUserDefinedCapture0Rate：接收自定义统计 0 报文速率 (fps)
RxUserDefinedCapture1Frames：接收自定义统计 1 报文数
RxUserDefinedCapture1Rate：接收自定义统计 1 报文速率 (fps)
RxUserDefinedCapture2Frames：接收自定义统计 2 报文数
RxUserDefinedCapture2Rate：接收自定义统计 2 报文速率 (fps)

RxUserDefinedCapture3Frames：接收自定义统计 3 报文数

RxUserDefinedCapture3Rate：接收自定义统计 3 报文速率 (fps)

RxUserDefinedCapture4Frames：接收自定义统计 4 报文数

RxUserDefinedCapture4Rate：接收自定义统计 4 报文速率 (fps)

RxUserDefinedCapture5Frames：接收自定义统计 5 报文数

RxUserDefinedCapture5Rate：接收自定义统计 5 报文速率 (fps)

RxFirstFrameArrivalTime：接收第一个帧的时间

RxLastFrameArrivalTime：接收最后一个帧的时间

返回

eg:

```
{
    'RxFirstFrameArrivalTime': 1000,
    'RxLastFrameArrivalTime': 1000,
}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | TxTotalFrames | RxTotalFrames |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Port} | Get Ports |
| &{Result} | Get Port Statistic | Port=${Port} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Statistic.common.**get_stream_rx_statistic**(*Stream*, *Port*, *StreamID=1*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板统计结果

参数

- **Stream** (StreamTemplate) --
- **Port** (Port) -- 接收端口对象 object, 类型为：object
- **StreamID** (*int*) -- 指定需要获取结果的流模板中流的 Id, 默认值：1
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项：需要获取流模板统计项目

  StreamID

  StreamBlockID

  ChannelId

  PortID

LoadBalance
RxStreamFrames：接收报文数
RxFrameRate：接收报文速率 (fps)
RxByteRate：接收字节速率 (Bps)
RxSeqErr
RxPayloadErr
MinLatency：最小时延 (us)
MaxLatency：最大时延 (us)
AvaLatency：平均时延 (us)
ShortTermAvgLatency：短期平均时延 (us)
RxBitRate：接收比特速率 (bps)
RxUtil：接收百分比 (%)
MinJitter：最小延迟抖动 (us)
MaxJitter：最大延迟抖动 (us)
AvaJitter：平均延迟抖动 (us)
ShortTermAvgJitter：短期平均延迟抖动 (us)
RxLossStreamFrames：实时丢包数
RxIpLengthErrorCount
RxIpv4ChecksumErrorFrames：接收 IPv4 Checksum Error 报文数
PrbsFillBytes：接收 PRBS 填充字节
DuplicateFrames：接收重复帧
InOrderFrames：接收有序帧
ReOrderFrames：接收重排序帧
PrbsErrorBits：接收 PRBS 错误位数
PrbsErrorFrames：接收 PRBS 错误帧数
RxFcsErrorFrames：接收 FCS 错误帧
RxFcsErrorFrameRate：接收 FCS 错误帧速率 (fps)
TcpChecksumErrorFrames：接收 TCP/UDP 校验错误帧
RxL1Rate：接收线速 (bps)
RxTotalBytes：接收总字节数
RxLateCount：接收延迟计数
RxInSequenceCount：接收按顺序计数
RxOutofSequenceCount：接收未按顺序计数
RxMinInterArrivalTime：接收最小到达时间 (us)
RxMaxInterArrivalTime：接收最大到达时间 (us)
RxAvgInterArrivalTime：接收平均到达时间 (us)
RxShortTermAvgInterArrivalTime：接收短期平均到达时间 (us)

返回

eg:

```
{"RxAvgInterArrivalTime": 1000, "RxShortTermAvgInterArrivalTime":␣
↪1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | RxAvgInterArrivalTime |␣
↪RxShortTermAvgInterArrivalTime |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Stream Rx Statistic | Stream=${Stream} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Statistic.common.**get_stream_statistic**(*Stream*, *StreamID=1*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板统计结果

参数

- **Stream** (StreamTemplate) --
- **StreamID** (*int*) -- 指定需要获取结果的流模板中流的 Id, 默认值：1
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项：需要获取流模板统计项目

  StreamBlockID：流量模板名称

  StreamID：流量 ID

  TxPortID：发送端口

  RxPortID：接收端口

  LoadBalance：接收端口

  TxStreamFrames：发送报文数

  RxStreamFrames：接收报文数

  TxFrameRate：发送报文速率 (fps)

  RxFrameRate：接收报文速率 (fps)

  TxL1Rate：发送线速 (bps)

  RxL1Rate：接收线速 (bps)

  RxLossStreamFrames：实时丢包数

  RealtimeLossRate：实时丢包率 (%)

  ResumeTime：恢复时间 (s)

  StartTime：流启动时间

  TxUtil：发送百分比 (%)

RxUtil：接收百分比 (%)

RxPayloadErr：接收 Payload Error 报文数

RxSeqErr：接收 Sequence Error 报文数

RxIpLengthErrorCount：接收 IP 长度错误计数

TxByteRate：发送字节速率 (Bps)

RxByteRate：接收字节速率 (Bps)

TxBitRate：发送比特速率 (bps)

RxBitRate：接收比特速率 (bps)

MinLatency：最小延迟 (us)

MaxLatency：最大延迟 (us)

AvaLatency：平均延迟抖动 (us)

MinJitter：最小延迟抖动 (us)

MaxJitter：最大延迟抖动 (us)

AvaJitter：平均延迟抖动 (us)

RxIpv4ChecksumErrorFrames：接收 Ipv4 Checksum 错误

PrbsFillBytes：接收端口

DuplicateFrames：接收端口

InOrderFrames：接收端口

ReOrderFrames：接收端口

PrbsErrorBits：接收端口

PrbsErrorFrames：接收端口

RxFcsErrorFrames：接收 FCS Checksum 错误

RxFcsErrorFrameRate：接收 FCS Checksum 错误速率

TcpChecksumErrorFrames：接收 TCP Checksum 错误

LostStreamFrames：丢包数

返回

eg:

```
{"RxFcsErrorFrames": 1000, "LostStreamFrames": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | RxIpv4ChecksumError |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Stream Statistic | Stream=${Stream} | StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Statistic.common.**get_stream_tx_statistic**(*Stream*, *Port=None*, *StreamID=1*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板统计结果

参数

- **Stream** (StreamTemplate) --
- **Port** (Port) -- 发送端口对象 object, 类型为：object
- **StreamID** (*int*) -- 指定需要获取结果的流模板中流的 Id, 默认值：1
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项：需要获取流模板统计项目

  StreamID

  StreamBlockID

  ChannelId

  PortID

  TxStreamFrames：发送报文数

  TxFrameRate：发送报文速率 (fps)

  TxByteRate：发送字节速率 (Bps)

  TxBitRate：发送比特速率 (bps)

  TxL1Rate：发送线速 (bps)

  TxUtil：发送百分比 (%)

  TxTotalBytes：发送总字节数

返回

eg:

```
{"TxL1Rate": 1000, "TxTotalBytes": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | TxUtil | TxTotalBytes |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Stream Tx Statistic | Stream=${Stream} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Statistic.common.**get_streamblock_rx_statistic**(*Stream*, *Port*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板块统计结果

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Port** (Port) -- 接收端口对象 object, 类型为：object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项：需要获取流模板统计项目

  StreamBlockID

  PortID

  LoadBalance

  RxStreamFrames：接收报文数

  RxFrameRate：接收报文速率 (fps)

  RxByteRate：接收字节速率 (Bps)

  RxSeqErr

  RxPayloadErr

  MinLatency：最小时延 (us)

  MaxLatency：最大时延 (us)

  AvaLatency：平均时延 (us)

  ShortTermAvgLatency：短期平均时延 (us)

  RxBitRate：接收比特速率 (bps)

  RxUtil：接收线速 (bps)

  MinJitter：最小延迟抖动 (us)

  MaxJitter：最大延迟抖动 (us)

  AvaJitter：平均延迟抖动 (us)

  ShortTermAvgJitter：短期平均延迟抖动 (us)

  RxLossStreamFrames

  RxIpLengthErrorCount

  RxL1Rate

  RxIpv4ChecksumErrorFrames：接收 IPv4 Checksum Error 报文数

  PrbsFillBytes：接收 PRBS 填充字节

  DuplicateFrames：接收重复帧

  InOrderFrames：接收有序帧

  ReOrderFrames：接收重排序帧

  PrbsErrorBits：接收 PRBS 错误位数

  PrbsErrorFrames：接收 PRBS 错误帧数

  RxFcsErrorFrames：接收 FCS 错误帧

  RxFcsErrorFrameRate：接收 FCS 错误帧速率 (fps)

  TcpChecksumErrorFrames：接收 TCP/UDP 校验错误帧

  RxAvgRate

RxAvgFps

RxMaxRate

RxMaxFps

RxTotalBytes：接收总字节数

RxLateCount：接收延迟计数

RxInSequenceCount：接收按顺序计数

RxOutofSequenceCount：接收未按顺序计数

RxMinInterArrivalTime：接收最小到达时间 (us)

RxMaxInterArrivalTime：接收最大到达时间 (us)

RxAvgInterArrivalTime：接收平均到达时间 (us)

RxShortTermAvgInterArrivalTime：接收短期平均到达时间 (us)

返回

eg:

```
{"TxUtil": 1000, "TxTotalBytes": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | RxIpv4ChecksumError |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Streamblock Rx Statistic | Stream=${Stream} | Port=${Port}
↪| StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Statistic.common.**get_streamblock_statistic**(*Stream*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板块统计结果

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项：需要获取流模板统计项目

  TxPortID：发送端口

  RxPortID：接收端口

  TxStreamFrames：发送带标签流报文总数

  RxStreamFrames：接收带标签流报文总数

  TxFrameRate：发送报文速率 (fps)

RxFrameRate：接收报文速率 (fps)

TxL1Rate：发送线速 (bps)

RxL1Rate：接收线速 (bps)

TxUtil：发送百分比 (%)

RxUtil：接收百分比 (%)

RxLossStreamFrames：实时丢包数

RealtimeLossRate：实时丢包率 (%)

ResumeTime：恢复时间 (s)

StartTime：流启动时间

MinLatency：最小延迟 (us)

MaxLatency：最大延迟 (us)

AvaLatency：平均延迟 (us)

MinJitter：最小延迟抖动 (us)

MaxJitter：最大延迟抖动 (us)

AvaJitter：平均延迟抖动 (us)

TxByteRate：发送字节速率 (Bps)

RxByteRate：接收字节速率 (Bps)

TxBitRate：发送比特速率 (bps)

RxBitRate：接收比特速率 (bps)

TxTotalBytes：发送字节总数

RxTotalBytes：接收字节总数

RxPayloadErr：接收 Payload Error 报文数

RxInSequenceCount：接收 Sequence Error 报文数

RxFCSErr：接收 FCS 错误数

RxIpv4ChecksumError：接收 IPv4 Checksum Error 报文数

RxTcpChecksumError：接收 TCP Checksum Error 报文数

返回

eg:

```
{"TxTotalFrames": 1000, "RxTotalFrames": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | RxIpv4ChecksumError |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
```

(下页继续)

(续上页)

```
| &{Result} | Get Streamblock Statistic | Stream=${Stream} | StaItems=@
↪{StaItems} |
| Clear Result |
```

TesterLibrary.Statistic.common.**get_streamblock_tx_statistic**(*Stream*, *Port=None*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板块统计结果

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Port** (Port) -- 发送端口对象 object, 类型为：object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项：需要获取流模板统计项目

  StreamBlockID

  PortID

  ChannelCount

  TxStreamFrames：发送报文数

  TxFrameRate：发送报文速率 (fps)

  TxByteRate：发送字节速率 (Bps)

  TxBitRate：发送比特速率 (bps)

  TxL1Rate：发送线速 (bps)

  TxUtil：发送百分比 (%)

  TxTotalBytes：发送字节数

返回

eg:

```
{"TxUtil": 1000, "TxTotalBytes": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | RxIpv4ChecksumError |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Streamblock Tx Statistic | Stream=${Stream} | Port=${Port}␣
↪| StaItems=@{StaItems} |
| Clear Result |
```

TesterLibrary.Statistic.common.**reset_statistic**()

重置测试仪表已缓存统计

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Reset Statistic |
```

TesterLibrary.Statistic.common.**save_result**(*Path*, *FileName*)

保存测试仪表统计结果到 DB 文件

参数

- **Path** (*str*) -- 保存文件的路径
- **FileName** (*str*) -- 保存文件名称

返回 返回保存的 DB 文件的绝对路径字符串

返回类型 str

实际案例

robotframework:

```
| @{Subscribe} | Create List | PortStats | StreamBlockStats |
| Subscribe Result | Types=@{Subscribe} |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${DB} | Save Result | Path=D: est | FileName=test |
```

TesterLibrary.Statistic.common.**subscribe_result**(*Types=None*)

订阅测试仪表统计视图

参数 **Types** (*list*) -- 需要订阅测试仪表统计视图列表，当传入为 None 时，订阅当前配置的所有视图，目前支持的统计视图：

PortStats

PortAvgLatencyStats

StreamStats

StreamTxStats

StreamRxStats

StreamBlockStats

StreamBlockTxStats

StreamBlockRxStats

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| @{Types} | Create List | PortStats | StreamBlockStats |
| Subscribe Result | Types=${Types} |
```

Module contents

1.1.5 TesterLibrary.Stream package

Subpackages

TesterLibrary.Stream.Header package

Subpackages

TesterLibrary.Stream.Header.Access package

Submodules

TesterLibrary.Stream.Header.Access.common module

TesterLibrary.Stream.Header.Access.common.**edit_header_ppp**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 PPP 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 PPP 头部在流量模板中所有 PPP 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Addresses** (*str*) --
- **Controls** (*int*) --
- **Protocol** (*int*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Ppp | Stream=${Stream} | Level=0 | Addresses=192.168.0.1 |
```

TesterLibrary.Stream.Header.Access.common.**edit_header_pppoe**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 PPPoE 报文头部内容

参数

- **Stream** (StreamTemplate) --

- **Level** (*int*) -- 要修改的 PPPoE 头部在流量模板中所有 PPPoE 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** (*int*) --
- **Type** (*int*) --
- **Code** (*str*) --
- **SessionId** (*int*) --
- **PayloadLen** (*int*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Pppoe | Stream=${Stream} | Level=0 | Code=11 |
```

TesterLibrary.Stream.Header.Access.l2tpv2 module

TesterLibrary.Stream.Header.Access.l2tpv2.**edit_header_l2tpv2_data**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 L2tpv2 Data 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 L2tpv2 Data 头部在流量模板中所有 L2tpv2 Data 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **TunnelId** (*int*) --
- **SessionId** (*int*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header L2tpv2 Data | Stream=${Stream} | Level=0 | TunnelId=1000 |
```

Module contents

TesterLibrary.Stream.Header.Basic package

Submodules

TesterLibrary.Stream.Header.Basic.common module

TesterLibrary.Stream.Header.Basic.common.**edit_header_custom**(*Stream*, *Level=0*, *Index=None*, ***kwargs*)

修改测试仪表流量模板中 Custom 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 Custom 头部在流量模板中所有 Custom 头部的序列号, 默认值: 0, 范围: 0-65535
- **Index** (*int*) -- Custom 头部在中需要修改 pattern 位于所有 PatternByte 和 Checksum 的序号

关键字参数

- **Pattern** --
- **Checksum** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Custom |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_custom | Stream=${Stream} | Pattern=1212121212 |
| edit_header_custom | Stream=${Stream} | Checksum=Auto |
| edit_header_custom | Stream=${Stream} | Pattern=343434343 |Checksum=Auto |
| edit_header_custom | Stream=${Stream} | Index=0 |Pattern=565656565 |
| edit_header_custom | Stream=${Stream} | Index=2 |Pattern=787878787 |
```

Module contents

TesterLibrary.Stream.Header.Gre package

Submodules

TesterLibrary.Stream.Header.Gre.common module

TesterLibrary.Stream.Header.Gre.common.**edit_header_gre**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Ethernet 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 vlan 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **ChecksumPresent** (*int*) --
- **Routing** (*int*) --
- **KeyPresent** (*int*) --
- **SequenceNumberPresent** (*int*) --
- **Reserved** (*int*) --
- **Version** (*int*) --
- **Protocol** (*str*) --
- **EnableKeepAlive** (*int*) --
- **KeepAlivePeriod** (*int*) --
- **KeepAliveRetries** (*int*) --
- **Checksum** (*dict*) -- e.g: {'checksum': 123, 'reserved': 123}
- **Key** (*int*) --
- **SequenceNumber** (*int*) --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | GRE |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Gre | Stream=${Stream} | Level=0 | KeepAlivePeriod=100 |␣
↪KeepAliveRetries=200 |
```

Module contents

TesterLibrary.Stream.Header.L2 package

Submodules

TesterLibrary.Stream.Header.L2.common module

TesterLibrary.Stream.Header.L2.common.**edit_header_arp**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 ARP 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 vlan 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **HardwareType** (*int*) --
- **ProtocolType** (*str*) --
- **HardwareSize** (*int*) --
- **ProtocolSize** (*int*) --
- **Opcode** (*int*) --
- **SendMac** (*str*) --
- **SendIpv4** (*str*) --
- **TargetMac** (*str*) --
- **TargetIpv4** (*str*) --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Arp | Stream=${Stream} | Level=0 | SendMac=00:00:01:01:01:01 |
```

TesterLibrary.Stream.Header.L2.common.**edit_header_ethernet**(*Stream*, *Level=0*, *DestMacAdd=None*, *SourceMacAdd=None*, *ProtocolType=None*)

修改测试仪表流量模板中 Ethernet 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 vlan 头部的序列号, 默认值: 0, 范围: 0-65535
- **DestMacAdd** (*str*) -- 目的 mac 地址
- **SourceMacAdd** (*str*) -- 源 mac 地址
- **ProtocolType** (*str*) -- 上层协议类型

Returns：

bool: 布尔值 Bool (范围：True / False)

实际案例

```
| Edit Header Ethernet | Stream=${Stream} | Level=0 |␣
↪DestMacAdd=00:01:01:01:01:02 |
```

TesterLibrary.Stream.Header.L2.common.**edit_header_mpls**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 MPLS 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 MPLS 头部在流量模板中所有 MPLS 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Label** (*int*) --
- **Exp** (*str*) --
- **Bottom** (*int*) --
- **TTL** (*int*) --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | MPLS | IPv4 |
| Create Stream header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header MPLS | Stream=${Stream} | Level=0 | TTL=192.168.1.1 | TTL=100 |
```

TesterLibrary.Stream.Header.L2.common.**edit_header_vlan**(*Stream*, *Level=0*, *ID=None*, *Priority=None*, *CFI=None*)

修改测试仪表流量模板中 vlan 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 vlan 头部的序列号, 默认值: 0, 范围: 0-65535
- **ID** (*int*) -- vlan id
- **Priority** (*int*) -- vlan 优先级
- **CFI** (*str*) -- vlan cfi 值

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Vlan | Stream=${Stream} | Level=0 | ID=4094 |
```

TesterLibrary.Stream.Header.L2.isis module

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_area_address_entry**(*Stream*, *Level=0*, *TlvIndex=0*, *EntryIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Hello/Lsp 报文中 Tlv 头部 Area Address Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Hello/Lsp 头部在流量模板中所有 ISIS L1/L2 Hello/Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis Area Address Entry 节点在流量模板中所有 Isis Area Address Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **tlvLength** (*hex*) -- Length, 默认值：1, 取值范围：1-255
- **AreaAddress** (*hex*) -- Area Address, 默认值：00, 长度：0-255byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1hello | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} | Index=0␣
↪| lspEntries=1 |
| Edit Header Isis Area Address Entry | Stream=${Stream} | TlvIndex=0 |␣
↪EntryIndex=0 | remainTime=10 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_csnp**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Isis L1/L2 Scnp 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Isis L1/L2 Scnp 头部在流量模板中所有 Isis L1/L2 Scnp 头部的序列号

关键字参数

- **InterRoutingProtocolDiscriminator** (*int*) -- Intradomain Routeing Protocol Discriminator, 默认值：<AUTO>83
- **lengthIndicator** (*int*) -- Length Indicator, 默认值：<AUTO>33
- **versionIdExtend** (*int*) -- Version/Protocol ID Extension, 默认值：<AUTO>1
- **idLength** (*int*) -- ID Length, 默认值：<AUTO>6
- **reserved1** (*int*) -- 默认值：3, 取值范围：0-7
- **pDUType** (*int*) -- PDU Type, 默认值：<AUTO>24, 取值范围：0-31
- **version** (*int*) -- 默认值：1, 取值范围：0-255
- **reserved2** (*int*) -- 默认值：3, 取值范围：0-255
- **maxAreaAddress** (*int*) -- Maximum Area Addresses, 默认值：3, 取值范围：0-3
- **pDULength** (*int*) -- PDU Length, 默认值：<AUTO>33

- **lspId** (*hex int*) -- Source ID, 默认值：00000000000000, 取值范围：00000000000000-FFFFFFFFFFFFFF
- **startLspId** (*hex int*) -- Start LSP-ID, 默认值：0000000000000000, 取值范围：0000000000000000-FFFFFFFFFFFFFFFF
- **endLspId** (*hex int*) -- End LSP-ID, 默认值：0000000000000000, 取值范围：0000000000000000-FFFFFFFFFFFFFFFF
- **CsnpDataTlvOptionHeader** (*list*) -- 可插入的选项，默认无选项，可选值：

  isIsLspEntries

  authentionInfo

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis csnp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_external_metric_entry**(*Stream*, *Level=0*, *TlvIndex=0*, *EntryIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1 Lsp 报文中 Tlv 头部 Metric Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Lsp 头部在流量模板中所有 ISIS L1/L2 Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis External Metric Entry 节点在流量模板中所有 Isis External Metric Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **reserved** (*hex*) -- Distribution, 默认值：0, 取值范围：0-1
- **defaultMetricIEbit** (*int*) -- Default Metric I/E Bit, 默认值：0, 取值范围：0-1
- **defaultMetric** (*int*) -- Default Metric, 默认值：0, 取值范围：0-63
- **delayMetricsbit** (*int*) -- Delay Metric S bit, 默认值：0, 取值范围：0-1
- **delayMetricSBit** (*int*) -- Delay Metric R bit, 默认值：0, 取值范围：0-1
- **delayMetric** (*int*) -- Delay Metric, 默认值：0, 取值范围：0-63
- **expenseMetricsBit** (*int*) -- Expense Metric S Bit, 默认值：0, 取值范围：0-1

- **expenseMetricIEbit** (*int*) -- Expense Metric R Bit, 默认值：0, 取值范围：0-1
- **expenseMetric** (*int*) -- Expense Metric, 默认值：0, 取值范围：0-63
- **errorMetricsBit** (*int*) -- Error Metric S Bit, 默认值：0, 取值范围：0-1
- **errorMetricIEbit** (*int*) -- Error Metric R Bit, 默认值：0, 取值范围：0-1
- **errorMetric** (*int*) -- Error Metric, 默认值：0, 取值范围：0-63
- **ipAddress** (*str*) -- IP Address, 默认值：192.168.0.2, 取值范围：有效的 ipv4 地址
- **subMask** (*hex*) -- Subnet Mask, 默认值：00000000, 长度：4byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} | Index=0␣
↪| lspEntries=1 |
| Edit Header Isis External Metric Entry | Stream=${Stream} | TlvIndex=0 |␣
↪EntryIndex=0 | errorMetricIEbit=1 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_hello**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Isis L1/L2 Hello 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Isis L1/L2 Hello 头部在流量模板中所有 Isis L1/L2 Hello 头部的序列号

关键字参数

- **InterRoutingProtocolDiscriminator** (*int*) -- Intradomain Routeing Protocol Discriminator, 默认值：<AUTO>83
- **lengthIndicator** (*int*) -- Length Indicator, 默认值：<AUTO>33
- **versionIdExtend** (*int*) -- Version/Protocol ID Extension, 默认值：<AUTO>1
- **idLength** (*int*) -- ID Length, 默认值：<AUTO>6
- **commonReserved1** (*int*) -- 默认值：3, 取值范围：0-7
- **pDUType** (*int*) -- PDU Type, 默认值：<AUTO>24, 取值范围：0-31
- **version** (*int*) -- 默认值：1, 取值范围：0-255
- **commonReserved2** (*int*) -- 默认值：3, 取值范围：0-255
- **maxAreaAddress** (*int*) -- Maximum Area Addresses, 默认值：3, 取值范围：0-3
- **fixedReserve1** (*int*) -- Reserved, 默认值：0, 取值范围：0-63
- **circuitType** (*int*) -- Circuit Type, 默认值：1, 取值范围：1-3

- **senderSystemID** (*hex*) -- Source ID, 默认值：000000000001, 长度: 6byte
- **holderTimer** (*int|hex*) -- Holding Timer, l1 默认值：51, 取值范围：0-65535; l2 默认值：0033, 长度：2byte
- **pDULength** (*int*) -- PDU Length, 默认值:<AUTO>27, 取值范围:0-65535
- **fixedReserve2** (*int*) -- Reserved, 默认值：0, 取值范围：0-1
- **priority** (*int*) -- Priority, 默认值：0, 取值范围：0-127
- **designatedSystemID** (*hex*) -- LAN ID, 默认值：00000000010001, 长度为 7byte
- **isIsTlv** (*list*) -- TLV Header, 默认值：”, 取值范围：

  isIsAreaAddress (l1) / areaAddress (l2)

  padding

  authentionInfo

  protocolSupport

  ipInterfaceAddress

  neighbor

  restartSignal

  Ipv6InterfaceAddress

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1helloHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis hello | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_internal_metric_entry**(*Stream*, *Level=0*, *TlvIndex=0*, *EntryIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Lsp 报文中 Tlv 头部 Metric Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Lsp 头部在流量模板中所有 ISIS L1/L2 Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis Internal Metric Entry 节点在流量模板中所有 Isis Internal Metric Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **reserved** (*hex*) -- Distribution, 默认值：0, 取值范围：0-1
- **defaultMetricIEbit** (*int*) -- Default Metric I/E Bit, 默认值：0, 取值范围：0-1
- **defaultMetric** (*int*) -- Default Metric, 默认值：0, 取值范围：0-63
- **delayMetricsbit** (*int*) -- Delay Metric S bit, 默认值：0, 取值范围：0-1
- **delayMetricSBit** (*int*) -- Delay Metric R bit, 默认值：0, 取值范围：0-1
- **delayMetric** (*int*) -- Delay Metric, 默认值：0, 取值范围：0-63
- **expenseMetricsBit** (*int*) -- Expense Metric S Bit, 默认值：0, 取值范围：0-1
- **expenseMetricIEbit** (*int*) -- Expense Metric R Bit, 默认值：0, 取值范围：0-1
- **expenseMetric** (*int*) -- Expense Metric, 默认值：0, 取值范围：0-63
- **errorMetricsBit** (*int*) -- Error Metric S Bit, 默认值：0, 取值范围：0-1
- **errorMetricIEbit** (*int*) -- Error Metric R Bit, 默认值：0, 取值范围：0-1
- **errorMetric** (*int*) -- Error Metric, 默认值：0, 取值范围：0-63
- **ipAddress** (*str*) -- IP Address, 默认值：192.168.0.2, 取值范围：有效的 ipv4 地址
- **subMask** (*hex*) -- Subnet Mask, 默认值：00000000, 长度：4byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} | Index=0␣
↪| lspEntries=1 |
| Edit Header Isis Internal Metric Entry | Stream=${Stream} | TlvIndex=0 |␣
↪EntryIndex=0 | errorMetricIEbit=1 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_lsp**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Isis L1/L2 Lsp 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Isis L1/L2 Hello 头部在流量模板中所有 Isis L1/L2 Hello 头部的序列号

关键字参数

- **InterRoutingProtocolDiscriminator** (*int*) -- Intradomain Routeing Protocol Discriminator, 默认值：<AUTO>83, 取值范围：00-FF
- **lengthIndicator** (*int*) -- Length Indicator, 默认值：<AUTO>27, 取值范围：0-255

- **versionIdExtend** (*int*) -- Version/Protocol ID Extension, 默认值: <AUTO>1, 取值范围: 0-255
- **idLength** (*int*) -- ID Length, 默认值: <AUTO>6, 取值范围: 0-255
- **reserved1** (*int*) -- Reserved, 默认值: 0, 取值范围: 0-7
- **pDUType** (*int*) -- PDU Type, 默认值: <AUTO>18, 取值范围: 0-31
- **version** (*int*) -- Version, 默认值: 1, 取值范围: 0-255
- **reserved2** (*int*) -- Reserved, 默认值: 0, 取值范围: 0-255
- **maxAreaAddress** (*int*) -- Maximum Area Addresses, 默认值: 3, 取值范围: 0-3
- **pDULength** (*int*) -- Reserved, 默认值: 0, 取值范围: 0-63
- **remainTime** (*int*) -- Circuit Type, 默认值: 1, 取值范围: 1-3
- **lspId** (*hex*) -- Source ID, 默认值: 0000000000000000, 长度: 8byte
- **seqcenceNum** (*hex*) -- Seqcence Number, 默认值: 00000000, 长度: 4byte
- **checksum** (*hex*) -- Checksum, 默认值: <AUTO>0000, 长度: 2byte
- **partitionRepair** (*int*) -- Partition Repair Bit, 默认值: 0, 取值范围: 0-1
- **attchment** (*int*) -- Attchment, 默认值: 0, 取值范围: 0-15
- **OverloadBit** (*int*) -- Overload Bit, 默认值: 0, 取值范围: 0-1
- **TypeOfIntermediateSystem** (*int*) -- Type of Intermediate System, 默认值: 0, 取值范围: 0-3
- **LspisIsTlvOptionSet** (*list*) -- TLV Header, 默认值: ”, 取值范围:

  isIsAreaAddress

  isIsReachability

  extendedReachability

  isIsIpInterReachability

  isIsProtocolsSupported

  isIsIPExternalReachability

  ipInterfaceAddress

  Ipv6InterfaceAddress

  isIsIpv6Reachability

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1helloHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_lsp_entry**(*Stream*, *Level=0*, *TlvIndex=0*, *LspIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Csnp 报文中 Tlv 头部 Lsp Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Csnp 头部在流量模板中所有 ISIS L1/L2 Csnp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **LspIndex** (*int*) -- 要修改的 Isis Lsp Entry 节点在流量模板中所有 Isis Lsp Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **remainTime** (*int*) -- 默认值：1, 取值范围：0-65535
- **lspId** (*hex*) -- 默认值：0000000000000001, 取值范围：0000000000000001-FFFFFFFFFFFFFFFF
- **lspSeqcenceNum** (*hex*) -- 默认值：00000001, 取值范围：00000001-FFFFFFFF
- **checksum** (*hex*) -- 默认值：<AUTO>0000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1csnp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} | Index=0␣
↪| lspEntries=1 |
| Edit Header IsisLsp Entry | Stream=${Stream} | TlvIndex=0 | LspIndex=0 |␣
↪remainTime=10 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_metric_entry**(*Stream*, *Level=0*, *TlvIndex=0*, *EntryIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Lsp 报文中 Tlv 头部 Metric Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Lsp 头部在流量模板中所有 ISIS L1/L2 Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis Metric Entry 节点在流量模板中所有 Isis Metric Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **reserved** (*hex*) -- Distribution, 默认值：0, 取值范围：0-1
- **defaultMetricIEbit** (*int*) -- Default Metric I/E Bit, 默认值：0, 取值范围：0-1
- **defaultMetric** (*int*) -- Default Metric, 默认值：0, 取值范围：0-63
- **delayMetricsbit** (*int*) -- Delay Metric S bit, 默认值：0, 取值范围：0-1
- **delayMetricSBit** (*int*) -- Delay Metric R bit, 默认值：0, 取值范围：0-1
- **delayMetric** (*int*) -- Delay Metric, 默认值：0, 取值范围：0-63
- **expenseMetricsBit** (*int*) -- Expense Metric S Bit, 默认值：0, 取值范围：0-1
- **expenseMetricIEbit** (*int*) -- Expense Metric R Bit, 默认值：0, 取值范围：0-1
- **expenseMetric** (*int*) -- Expense Metric, 默认值：0, 取值范围：0-63
- **errorMetricsBit** (*int*) -- Error Metric S Bit, 默认值：0, 取值范围：0-1
- **errorMetricIEbit** (*int*) -- Error Metric R Bit, 默认值：0, 取值范围：0-1
- **errorMetric** (*int*) -- Error Metric, 默认值：0, 取值范围：0-63
- **isNeighbor** (*hex*) -- IS Neighbor, 默认值：00000000000000, 长度：7byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} | Index=0␣
↪| lspEntries=1 |
| Edit Header Isis Metric Entry | Stream=${Stream} | TlvIndex=0 |␣
↪EntryIndex=0 | errorMetricIEbit=1 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_nlpid_entry**(*Stream*, *Level=0*, *TlvIndex=0*, *NlpidIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Hello/Lsp 报文中 Tlv 头部 NLPID Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Hello/Lsp 头部在流量模板中所有 ISIS L1/L2 Hello/Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **NlpidIndex** (*int*) -- 要修改的 Isis NLPID Entry 节点在流量模板中所有 Isis NLPID Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **tlvLength** (*hex*) -- Length, 默认值：1, 取值范围：1-255
- **entryId** (*hex*) -- Area Address, 默认值：01, 长度：0-255byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1hello | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} | Index=0␣
↪| lspEntries=1 |
| Edit Header Isis Nlpid Entry | Stream=${Stream} | TlvIndex=0 | NlpidIndex=0␣
↪| remainTime=10 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_psnp**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Isis L1/L2 Pcnp 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Isis L1/L2 Pcnp 头部在流量模板中所有 Isis L1/L2 Pcnp 头部的序列号

关键字参数

- **InterRoutingProtocolDiscriminator** (*int*) -- Intradomain Routeing Protocol Discriminator, 默认值：<AUTO>83
- **lengthIndicator** (*int*) -- Length Indicator, 默认值：<AUTO>33
- **versionIdExtend** (*int*) -- Version/Protocol ID Extension, 默认值：<AUTO>1
- **idLength** (*int*) -- ID Length, 默认值：<AUTO>6

- **reserved1** (*int*) -- 默认值：3, 取值范围：0-7
- **pDUType** (*int*) -- PDU Type, 默认值：<AUTO>24, 取值范围：0-31
- **version** (*int*) -- 默认值：1, 取值范围：0-255
- **reserved2** (*int*) -- 默认值：3, 取值范围：0-255
- **maxAreaAddress** (*int*) -- Maximum Area Addresses, 默认值：3, 取值范围：0-3
- **pDULength** (*int*) -- PDU Length, 默认值：<AUTO>33
- **sourceId** (*hex*) -- Source ID, 默认值：000000000000, 长度: 6byte
- **reserved** (*hex*) -- Reserved, 默认值：00, 长度: 1byte
- **CsnpDataTlvOptionHeader** (*list*) -- 可插入的选项，默认无选项，可选值：

  isIsLspEntries

  authentionInfo

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis psnp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_sub_tlv**(*Stream*, *SubTlv*, *Level=0*, *TlvIndex=0*, *SubTlvIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Lsp 报文中 Tlv 头部 Sub Tlv 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **SubTlv** (*str*) -- Isis Sub Tlv 节点类型，支持：

  adGroupSubtlv

  ipv4InterfaceAddressSubtlv

  ipv4NeighborAddressSubtlv

  maxLinkBandwidthSubtlv

  ReservableLinkBandwidthSubtlv

  unReservedBandwidthSubtlv

  interfaceIpv6Subtlv

  neigbhorIpv6Subtlv

- **Level** (*int*) -- 要修改的 ISIS L1/L2 Lsp 头部在流量模板中所有 ISIS L1/L2 Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535

- **EntryIndex** (*int*) -- 要修改的 Isis Sub Tlv 节点在流量模板中所有 Isis Sub Tlv 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **adGroupSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：3, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  adminGroupValue (int): Length, 默认值：0, 取值范围：0-4294967245

- **ipv4InterfaceAddressSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：7, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  ipv4InterfaceAddressValue (str): IP Interface Address, 默认值：192.168.0.2, 取值范围：有效的 ipv4 地址

- **ipv4NeighborAddressSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：8, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  ipv4NeighborAddressValue (str): IP Neighbor Address, 默认值：192.168.0.2, 取值范围：有效的 ipv4 地址

- **maxLinkBandwidthSubtlv** 选项支持：-- tlvCode (int): Type, 默认值：9, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  maxBandwidthValue (int): Maximum Link Bandwidth, 默认值：0, 取值范围：0-4294967245

- **ReservableLinkBandwidthSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：10, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  ReservableLinkBandwidthValue (int): Reservable Link Bandwidth, 默认值：0, 取值范围：0-4294967245

- **unReservedBandwidthSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：11, 取值范围：0-255

  tlvLength (int): Length, 默认值：32, 取值范围：0-255

  resBandwidth0Value (int): Unreserved ResBandwidth Priority0, 默认值：0, 取值范围：0-4294967245

  resBandwidth1Value (int): Unreserved ResBandwidth Priority1, 默认值：0, 取值范围：0-4294967245

  resBandwidth2Value (int): Unreserved ResBandwidth Priority2, 默认值：0, 取值范围：0-4294967245

  resBandwidth3Value (int): Unreserved ResBandwidth Priority3, 默认值：0, 取值范围：0-4294967245

  resBandwidth4Value (int): Unreserved ResBandwidth Priority4, 默认值：0, 取值范围：0-4294967245

  resBandwidth5Value (int): Unreserved ResBandwidth Priority5, 默认值：0, 取值范围：0-4294967245

  resBandwidth6Value (int): Unreserved ResBandwidth Priority6, 默认值：0, 取值范围：0-4294967245

resBandwidth7Value (int): Unreserved ResBandwidth Priority7, 默认值：0, 取值范围：0-4294967245

- **interfaceIpv6Subtlv** 选项支持：-- tlvCode (int): Type, 默认值：12, 取值范围：0-255

  tlvLength (int): Length, 默认值：16, 取值范围：0-255

  interfaceIpv6Value (str): Interface IPv6 Value, 默认值：2001::2, 取值范围：有效的 ipv6 地址

- **neigbhorIpv6Subtlv** 选项支持：-- tlvCode (int): Type, 默认值：13, 取值范围：0-255

  tlvLength (int): Length, 默认值：16, 取值范围：0-255

  neighboripv6Value (str): Neighbor IPv6 Value, 默认值：2001::2, 取值范围：有效的 ipv6 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} | Index=0␣
↪| lspEntries=1 |
| Edit Header Isis Metric Entry | Stream=${Stream} | TlvIndex=0 |␣
↪EntryIndex=0 | errorMetricIEbit=1 |
```

TesterLibrary.Stream.Header.L2.isis.**edit_header_isis_tlv_header**(*Stream*, *Option*, *Level=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Csnp/Hello/Lsp/Psnp 报文中 Tlv 头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Option** (*str*) -- ISIS L1/L2 Csnp 支持：

  isIsLspEntries

  authentionInfo

  ISIS L1/L2 Hello 支持：

  isIsAreaAddress

  padding

  authentionInfo

  protocolSupport

  ipInterfaceAddress

  neighbor

  restartSignal

Ipv6InterfaceAddress

ISIS L1/L2 Lsp 支持:

isIsAreaAddress

isIsReachability

extendedReachability

isIsIpInterReachability

isIsProtocolsSupported

isIsIPExternalReachability

ipInterfaceAddress

Ipv6InterfaceAddress

isIsIpv6Reachability

- **Level** (*int*) -- 要修改的 ISIS L1/L2 Csnp/Hello 头部在流量模板中所有 ISIS L1/L2 Csnp/Hello 头部的序列号, 默认值: 0, 范围: 0-65535
- **Index** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **isIsLspEntries** 选项支持: (*ISIS L1/L2 Csnp/Psnp*) -- tlvCode (int): 默认值: <AUTO>9

  length (int): 默认值: <AUTO>0

  lspEntries (int): lsp entry 个数，默认值: 0

- **authentionInfo** 选项支持: (*ISIS L1/L2 Hello*) -- tlvCode (int): 默认值: <AUTO>10

  length (int): 默认值: <AUTO>0

  authenticationType (int): 默认值: 1, 取值范围: 0-255

  authenticationLength (int): 默认值: 1, 取值范围: 0-255

  authentication (hex int): 默认值: 01, 取值范围: 长度 0-255

- **isIsAreaAddress** 选项支持: (*ISIS L1/L2 Hello/Lsp*) -- tlvCode (int): 默认值: <AUTO>9, 取值范围: 0-255

  tlvLength (int): 默认值: <AUTO>0, 取值范围: 0-255

  AreaAddressEntries (int): area address entry 个数，默认值: 0

- **padding** 选项支持: (*ISIS L1/L2 Hello*) -- tlvCode (int): 默认值: <AUTO>9, 取值范围: 0-255

  tlvLength (int): 默认值: <AUTO>0, 取值范围: 0-255

  padding (hex): 默认值: 00, 取值范围: 00-FF

- **authentionInfo** 选项支持: -- tlvCode (int): 默认值: <AUTO>10

  length (int): 默认值: <AUTO>0

  authenticationType (int): Authentication Type, 默认值: 1, 取值范围: 0-255

  authenticationLength (int): Authentication Length, 默认值: 1, 取值范围: 0-255

  authentication (hex): Authentication, 默认值: 00, 最大长度 255byte

- **protocolSupport** 选项支持：(*ISIS L1/L2 Hello/Lsp*) -- tlvCode (int): 默认值：<AUTO>129

  length (int): 默认值：<AUTO>2

  NlPIDEntriesField (int): NLPID Entries 个数

- **ipInterfaceAddress** 选项支持：(*ISIS L1/L2 Hello/Lsp*) -- tlvCode (int): 默认值：<AUTO>9, 取值范围：0-255

  tlvLength (int): 默认值：<AUTO>0, 取值范围：0-255

  ipv4InterfaceAddress (list): IPv4 Interface Address, 列表长度最大 1024, 元素默认值：192.168.0.2, 取值范围：有效的 ipv4 地址

- **neighbor** 选项支持：(*ISIS L1/L2 Hello*) -- tlvCode (int): 默认值：<AUTO>6, 取值范围：0-255

  tlvLength (int): 默认值：<AUTO>2, 取值范围：0-255

  MacAdd (list): IS Neighbors, 列表长度最大 10, 元素默认值：00:00:00:13:40:20, 取值范围：有效的 mac 地址

- **restartSignal** 选项支持：(*ISIS L1/L2 Hello*) -- tlvCode (int): Type, 默认值：<AUTO>211, 取值范围：0-255

  tlvLength (int): Length, 默认值：<AUTO>3, 取值范围：0-255

  reserved1 (int): Reserved, 默认值：0, 取值范围：0-31

  suppressAdjacency (int): Suppress Adjacency Advertisement, 默认值：0, 取值范围：0-1

  restartAck (int): Restart Acknowledgement, 默认值：0, 取值范围：0-1

  restartReq (int): Restart Request, 默认值：0, 取值范围：0-1

  remainTime (int): Remaining Time, 默认值：0, 取值范围：0-65535

  restartNeighborIdField (str): Restarting Neighbor ID, 默认：000000000000, 长度：6byte

- **Ipv6InterfaceAddress** 选项支持：(*ISIS L1/L2 Hello/Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>232, 取值范围：0-255

  tlvLength (int): Length, 默认值：<AUTO>16, 取值范围：0-255

  ipv6InterfaceAddress (list): IPv6 Interface Address, 列表最大长度 1024, 元素默认值：2001::2, 取值范围：有效的 ipv6 地址

- **isIsReachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>2, 取值范围：0-255

  tlvLength (int): Length, 默认值：<AUTO>12, 取值范围：0-255

  virtualFlag (int): Virtual Flag, 默认值：<AUTO>12, 取值范围：0-255

  metricEntries (int): Metric Entry 个数, 默认值：0, 最大：1024

- **extendedReachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>2, 取值范围：0-255

  length (int): Length, 默认值：<AUTO>12, 取值范围：0-255

  neighborID (hex): Neighbor ID, 默认值: 00000000000000, 长度: 7 byte

  metric (hex): metric, 默认值：000000, 长度：3 byte

  tlvLength (int): Sub-TLV Length, 默认值：1, 取值范围：0-255

  iisNeighborSubTlv (list): Sub-TLV 类型, 支持类型：

adGroupSubtlv

ipv4InterfaceAddressSubtlv

ipv4NeighborAddressSubtlv

maxLinkBandwidthSubtlv

ReservableLinkBandwidthSubtlv

unReservedBandwidthSubtlv

interfaceIpv6Subtlv

neigbhorIpv6Subtlv

- **isIsIpInterReachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>128, 取值范围：0-255

  length (int): Length, 默认值：<AUTO>12, 取值范围：0-255

  internalmetricEntries (int): Internal Metric Entry 个数, 默认值：0, 最大：1024

- **isIsIPExternalReachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>130, 取值范围：0-255

  length (int): Length, 默认值：<AUTO>12, 取值范围：0-255

  externalmetricEntries (int): External Metric Entry 个数, 默认值：0, 最大：1024

- **isIsIpv6Reachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>236, 取值范围：0-255

  tlvLength (int): Length, 默认值：<AUTO>16, 取值范围：0-255

  metric (int): metric, 默认值：0, 取值范围：0-4294967295

  ubit (int): Up/Down Bit, 默认值：0, 取值范围：0-1

  xbit (int): External Origin Bit, 默认值：0, 取值范围：0-1

  sbit (int): Sub-TLV Bit, 默认值：0, 取值范围：0-1

  reserved (int): Reserved Bit, 默认值：0, 取值范围：0-31

  prefixLength (int): Prefix Length, 默认值：0, 取值范围：0-255

  prefix (hex): Prefix, 默认值：00, 长度：0-255byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1csnp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} | Index=0␣
↪| lspEntries=1 |
```

Module contents

TesterLibrary.Stream.Header.L3 package

Submodules

TesterLibrary.Stream.Header.L3.common module

TesterLibrary.Stream.Header.L3.common.**edit_header_ipv4**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IPv4 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 IPv4 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **HeadLen** --
- **Tos** --
- **TotalLength** --
- **Flags** --
- **ID** --
- **Offset** --
- **TTL** --
- **Protocol** --
- **Checksum** --
- **Source** --
- **Destination** --
- **Padding** --
- **Gateway** --
- **TosPrecedence** --
- **HeaderOption** -- 插入 HeaderOption 字段，支持传入列表，支持的参数有：

  EndOfOption

  Nop

  Security

  LooseSourceRoute

  StrictSourceRoute

  RouterAlert

  RecordRoute

  TimeStamp

  StreamIdentifier

General

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| ${HeaderOption} | Create List | EndOfOption | LooseSourceRoute |
| Edit Header IPv4 | Stream=${Stream} | Level=0 | Source=192.168.1.1 |␣
↪HeaderOption=${HeaderOption} |
```

TesterLibrary.Stream.Header.L3.common.**edit_header_ipv4_option**(*Stream*, *Type*, *Level=0*, *Index=0*, *Header='ipv4'*, ***kwargs*)

修改测试仪表流量模板中 IPv4 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 IPv4 头部在流量模板中所有 IPv4 头部的序列号, 默认值: 0, 范围: 0-65535
- **Index** (*int*) -- 要修改的 IPv4 Option 头部在流量模板中所有 IPv4 Option 的序列号, 默认值: 0, 范围: 0-65535
- **Header** -- 要修改的流量头部，默认修改 ipv4 头部对象，支持头部对象包括：

  ipv4 destunreach parameterproblem redirect sourcequench timeexceeded
- **Type** (*list*) -- 插入 HeaderOption 字段，支持传入列表，支持的参数有：

  EndOfOption

  Nop

  Security

  LooseSourceRoute

  StrictSourceRoute

  RouterAlert

  RecordRoute

  TimeStamp

  StreamIdentifier

  General
- **Args** (*Keyword*) -- EndOfOption 类型支持：

  optiontype

  Nop 类型支持：

  optiontype

  Security 类型支持：

optiontype
length
security
compartments
handlingRestrictions
txControlCode

LooseSourceRoute 类型支持:

optiontype
length
pointer
addressList

StrictSourceRoute 类型支持:

optiontype
length
pointer
addressList

RouterAlert 类型支持:

optiontype
length
routerAlertValue

RecordRoute 类型支持:

optiontype
length
pointer
addressList

TimeStamp 类型支持:

optiontype
length
pointer
overflow
flag
timeStamp
timeStampSet

StreamIdentifier 类型支持:

optiontype
length
systemId

General 类型支持:

optiontype

length

value

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header IPv4 | Stream=${Stream} | Level=0 | Source=192.168.1.1 |␣
↪HeaderOption=RouterAlert |
| Edit Header IPv4 Option | Stream=${Stream} | Type=RouterAlert |␣
↪routerAlertValue=1 |
```

TesterLibrary.Stream.Header.L3.common.**edit_header_ipv6**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IPv6 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 IPv6 头部在流量模板中所有 IPv6 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **TrafficClass** --
- **FlowLabel** --
- **PayloadLength** --
- **NextHeader** --
- **HopLimit** --
- **Source** --
- **Destination** --
- **Gateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header IPv6 | Stream=${Stream} | Level=0 | Source=2000::1 |
```

TesterLibrary.Stream.Header.L3.icmpv4 module

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_dest_unreach**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Dest Unreach 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Dest Unreach 头部在流量模板中所有 Icmp Dest Unreach 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Unused** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --

- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Dest Unreach | Stream=${Stream} | Level=0 | Identifier=100␣
↪|
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_echo_reply**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Echo Reply 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围：0-65535：要修改的 Icmp Echo Reply 头部在流量模板中所有 Icmp Echo Reply 头部的序列号

关键字参数

- **Type** (*int*) -- 范围：0-255
- **Code** (*int*) -- 范围：0-255
- **Checksum** (*str*) -- 范围：0000-ffff、AUTO，其中 ffff 表示产生错误，AUTO 表示自动计算 Checksum
- **Identifier** (*int*) -- 范围：0-65535
- **SequenceNumber** (*int*) -- 范围：0-65535

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoReply |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Echo Reply | Stream=${Stream} | Level=0 | Identifier=100 |␣
↪SequenceNumber=200 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_echo_request**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Echo Request 报文头部内容

参数

- **Stream** (StreamTemplate) --

- **Level** (*int*) -- 范围: 0-65535: 要修改的 Icmp Echo Request 头部在流量模板中所有 Icmp Echo Request 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-255
- **Code** (*int*) -- 范围: 0-255
- **Checksum** (*str*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **Identifier** (*int*) -- 范围: 0-65535
- **SequenceNumber** (*int*) -- 范围: 0-65535

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoRequest |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_icmp_echorequest | Stream=${Stream} | Level=0 | Identifier=100
↪| SequenceNumber=200 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_information_reply**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Information Reply 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Information Reply 头部在流量模板中所有 Icmp Information Reply 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Information Reply | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_information_request**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Information Request 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Information Request 头部在流量模板中所有 Icmp Information Request 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Information Request | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_mask_reply**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Mask Reply 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Mask Reply 头部在流量模板中所有 Icmp Mask Reply 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

- **AddrMask** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4MaskReply |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Mask Reply | Stream=${Stream} | Level=0 | Identifier=100 |␣
↪SequenceNumber=200 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_mask_request**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Mask Request 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Mask Request 头部在流量模板中所有 Icmp Mask Request 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --
- **AddrMask** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Mask Request | Stream=${Stream} | Level=0 | Identifier=100␣
↪| SequenceNumber=200 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_parameter_problem**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Parameter Problem 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Parameter Problem 头部在流量模板中所有 Icmp Parameter Problem 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Pointer** --
- **Reserve** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Parameter Problem | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_redirect**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Redirect 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Redirect 头部在流量模板中所有 Icmp Redirect 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **GatewayAddress** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --

- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Redirect | Stream=${Stream} | Level=0 | Identifier=100 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_source_quench**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Source Quench 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Source Quench 头部在流量模板中所有 Icmp Source Quench 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --

- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Source Quench | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_time_exceeded**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Time Exceeded 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Time Exceeded 头部在流量模板中所有 Icmp Time Exceeded 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --

- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Time Exceeded | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_time_stamp_reply**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Time Stamp Reply 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Time Stamp Reply 头部在流量模板中所有 Icmp Time Stamp Reply 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --
- **originateTimestamp** --
- **receiveTimestamp** --
- **transmitTimestamp** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Time Stamp Reply | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

TesterLibrary.Stream.Header.L3.icmpv4.**edit_header_icmp_time_stamp_request**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Time Stamp Request 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Time Stamp Request 头部在流量模板中所有 Icmp Time Stamp Request 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --
- **OriginateTimestamp** --
- **ReceiveTimestamp** --
- **TransmitTimestamp** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Time Stamp Request | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

TesterLibrary.Stream.Header.L3.icmpv6 module

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_destination_unreachable**(*Stream*
Level=
***kwarg*

修改测试仪表流量模板中 Icmpv6 Destination Unreachable 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Destination Unreachable 头部在流量模板中所有 Icmpv6 Destination Unreachable 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **HeaderData** --
- **Version** --
- **TrafficClass** --
- **FlowLable** --
- **PayloadLength** --
- **NextHeader** --
- **HopLimit** --
- **Source** --
- **Destination** --
- **Gateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | DestinationUnreachable |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Destination Unreachable | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_echo_reply**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Echo Reply 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Echo Reply 头部在流量模板中所有 Icmpv6 Echo Reply 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Icmpv6EchoReply |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Echo Reply | Stream=${Stream} | Level=0 | Identifier=100␣
↪|
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_echo_request**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Echo Request 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Echo Request 头部在流量模板中所有 Icmpv6 Echo Request 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Icmpv6EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Echo Request | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_group_records**(*Stream*, *Level=0*, *Index=0*, *Header='mldv2rep* ***kwargs*)

修改测试仪表流量模板中 ICMPv6 Mldv2 Report 报文头部 Group Records 内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 ICMPv6 头部在流量模板中所有 IGMPv3 头部的序列号
- **Index** -- 要修改的 ICMPv6 Group Records 头部在流量模板中所有 ICMPv6 Group Records 的序列号
- **Header** -- 要修改的流量头部，默认修改 mldv2report 头部对象，其他对象包括：

  mldv2report

- **Args** (*Keyword*) -- recordType: 类型为 int，默认值：1

  auxDataLen: 类型为 int，默认值：0

  numberOfSources: 类型为 int，默认值：1

  multicastAddress: 类型为 list, 组播 ipv4 地址，默认值：225.0.0.1

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Report | Stream=${Stream} | Level=0 | Type=1 |
| Edit Header Icmpv6 Group Records | Stream=${Stream} | recordType=10 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_header_option**(*Stream*, *Option*, *Level=0*, *Index=0*, *Header='routersoli* ***kwargs*)

修改测试仪表流量模板中 ICMPv6 报文头部 Header Option 内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Option** -- ICMPv6 报文头部 Header Option 类型, 类型为：string, 支持的类型包括：

  optionSourceLinkLayerAddress

  optionTargetLinkLayerAddress

  optionPrefixInformation

  optionMTU

  generalTLV

  generalWildcardTLV
- **Level** -- 要修改的 ICMPv6 头部在流量模板中所有 IGMPv3 头部的序列号
- **Index** -- 要修改的 ICMPv6 Header Option 头部在流量模板中所有 ICMPv6 Header Option 的序列号
- **Header** -- 要修改的流量头部，默认修改 mldv2report 头部对象，其他对象包括：

  routersolicit

  routeradvertise

  icmpv6redirect

  neighborsolicit

  neighboradvertise
- **Args** (*Keyword*) -- optionSourceLinkLayerAddress:

  type

  length

  address

  optionTargetLinkLayerAddress:

  type

  length

  address

  optionPrefixInformation:

  type

  length

  prefixLength

  onLinkFlag

  autonomousFlag

  reserved

  validLifetime

  preferredLifetime

  reserved2

prefixAddress

optionMTU:

type

length

reserved3

mtu

generalTLV:

type

length

value

generalWildcardTLV:

type

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Report | Stream=${Stream} | Level=0 | Type=1 |
| Edit Header Icmpv6 Group Records | Stream=${Stream} | recordType=10 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_mldv1_done**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv1 Done 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv1 Done 头部在流量模板中所有 Icmpv6 Mldv1 Done 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **MaxRespDelay** --
- **Reserved** --
- **MulticastAddress** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv1Done |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv1 Done | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_mldv1_query**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv1 Query 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv1 Query 头部在流量模板中所有 Icmpv6 Mldv1 Query 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **MaxRespDelay** --
- **Reserved** --
- **MulticastAddress** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv1Query |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv1 Query | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_mldv1_report**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv1 Report 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv1 Report 头部在流量模板中所有 Icmpv6 Mldv1 Report 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **MaxRespDelay** --
- **Reserved** --

- **MulticastAddress** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv1Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv1 Report | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_mldv2_query**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv2 Query 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv2 Query 头部在流量模板中所有 Icmpv6 Mldv2 Query 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **MaxRespCode** --
- **Reserved** --
- **GroupAddress** --
- **Resv** --
- **Sflag** --
- **Qrv** --
- **Qqic** --
- **NumberOfSources** --
- **SourceAddressList** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Query |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Query | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_mldv2_report**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv2 Report 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv2 Report 头部在流量模板中所有 Icmpv6 Mldv2 Report 头部的序列号

关键字参数

- **Type** --
- **Unused** --
- **Checksum** --
- **Reserved** --
- **NumberOfGroupRecords** --
- **GroupRecords** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Report | Stream=${Stream} | Level=0 | Type=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_neighbor_advertise**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Neighbor Advertise 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Neighbor Advertise 头部在流量模板中所有 Icmpv6 Neighbor Advertise 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Rflag** --
- **Sflag** --
- **Oflag** --
- **Reserve** --
- **TargetAddress** --
- **HeaderOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | NeighborAdvertise |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Neighbor Advertise | Stream=${Stream} | Level=0 | Code=1␣
↪|
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_neighbor_solicitation**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Neighbor Solicitation 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Neighbor Solicitation 头部在流量模板中所有 Icmpv6 Neighbor Solicitation 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **TargetAddress** --
- **HeaderOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | NeighborSolicitation |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Neighbor Solicitation | Stream=${Stream} | Level=0 |␣
↪Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_packet_too_big**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Packet Too Big 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Packet Too Big 头部在流量模板中所有 Icmpv6 Packet Too Big 头部的序列号

关键字参数

- **Type** --

- **Code** --
- **Checksum** --
- **Mtu** --
- **HeaderData** --
- **Version** --
- **TrafficClass** --
- **FlowLable** --
- **PayloadLength** --
- **NextHeader** --
- **HopLimit** --
- **Source** --
- **Destination** --
- **Gateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | PacketTooBig |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Packet Too Big | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_parameter_problem**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Parameter Problem 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Parameter Problem 头部在流量模板中所有 Icmpv6 Parameter Problem 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Pointer** --
- **HeaderData** --
- **Version** --
- **TrafficClass** --
- **FlowLable** --
- **PayloadLength** --
- **NextHeader** --

- **HopLimit** --
- **Source** --
- **Destination** --
- **Gateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Icmpv6ParameterProblem |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Parameter Problem | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_redirect**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Redirect 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Redirect 头部在流量模板中所有 Icmpv6 Redirect 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **TargetAddress** --
- **DestAddress** --
- **HeaderOption** --
- **RedirectedHdrOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Icmpv6Redirect |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Redirect | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_redirected_header**(*Stream*, *Level=0*, *Index=0*, *Header='icmp*
***kwargs*)

修改测试仪表流量模板中 ICMPv6 Redirected 报文头部 Header 内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 ICMPv6 头部在流量模板中所有 IGMPv3 头部的序列号
- **Index** -- 要修改的 ICMPv6 Redirected Header 头部在流量模板中所有 ICMPv6 Redirected Header 的序列号
- **Header** -- 要修改的流量头部，默认修改 icmpv6redirect 头部对象，支持对象包括：

  icmpv6redirect

- **Args** (*Keyword*) -- type: 类型为 int，默认值：4, 取值范围：

  Source Link-Layer Address：1

  Target Link-Layer Address：2

  Prefix Information：3

  Redirected Header：4

  MTU：5

  length: 类型为 int，默认值：4

  reserved1: 类型为 int，默认值：0

  reserved2: 类型为 int，默认值：0

  Version

  TrafficClass

  FlowLable

  PayloadLength

  NextHeader

  HopLimit

  Source

  Destination

  Gateway

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Report | Stream=${Stream} | Level=0 | Type=1 |
| Edit Header Icmpv6 Redirected Header | Stream=${Stream} | type=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_router_advertise**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Router Advertise 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Router Advertise 头部在流量模板中所有 Icmpv6 Router Advertise 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **CurHopLimit** --
- **ManagedAddrFlag** --
- **OtherConfigFlag** --
- **Reserved** --
- **RouterLifetime** --
- **ReachableTime** --
- **RetransTime** --
- **HeaderOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | routeradvertise |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Router Advertise | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_router_solicitation**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Router Solicitation 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Router Solicitation 头部在流量模板中所有 Icmpv6 Router Solicitation 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **HeaderOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | routersolicit |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Router Solicitation | Stream=${Stream} | Level=0 |␣
↪Code=1 |
```

TesterLibrary.Stream.Header.L3.icmpv6.**edit_header_icmpv6_time_exceed**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Time Exceed 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Time Exceed 头部在流量模板中所有 Icmpv6 Time Exceed 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | TimeExceed |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Time Exceed | Stream=${Stream} | Level=0 | Code=1 |
```

TesterLibrary.Stream.Header.L3.igmp module

TesterLibrary.Stream.Header.L3.igmp.**edit_header_igmpv1_query**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv1 Query 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv1 Query 头部在流量模板中所有 IGMPv1 Query 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 11
- **Unused** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum

- **GroupAddress** (*int*) -- 范围: ipv4 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv1Query |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv1_query | Stream=${Stream} | Level=0 | GroupAddress=225.0.
↪1.1 |
```

TesterLibrary.Stream.Header.L3.igmp.**edit_header_igmpv1_report**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv1 Report 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv1 Report 头部在流量模板中所有 IGMPv1 Report 头部的序列号, 范围 0-65535

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 12
- **Unused** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv1 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv1_report | Stream=${Stream} | Level=0 | GroupAddress=225.0.
↪1.1 |
```

TesterLibrary.Stream.Header.L3.igmp.**edit_header_igmpv2_query**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv2 Query 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv2 Query 头部在流量模板中所有 IGMPv2 Query 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 12

- **MaxResponseTime** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv2Query |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv2_query | Stream=${Stream} | Level=0 | MaxResponseTime=15␣
↪| GroupAddress=225.0.1.1 |
```

TesterLibrary.Stream.Header.L3.igmp.**edit_header_igmpv2_report**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv2 Report 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv2 Report 头部在流量模板中所有 IGMPv2 Report 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 11
- **MaxResponseTime** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv2 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv2_report | Stream=${Stream} | Level=0 | MaxResponseTime=15␣
↪| GroupAddress=225.0.1.1 |
```

TesterLibrary.Stream.Header.L3.igmp.**edit_header_igmpv3_group_records**(*Stream*, *Level=0*, *Index=0*, *Header='igmpv3repo* *SourceAddressList=[]*, *ExceedauxDataList=[]*, ***kwargs*)

修改测试仪表流量模板中 IGMPv3 Report 报文头部 Group Records 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv3 头部在流量模板中所有 IGMPv3 头部的序列号
- **Index** (*int*) -- 要修改的 IGMPv3 Group Records 头部在流量模板中所有 IGMPv3 Group Records 的序列号
- **Header** (*str*) -- 要修改的流量头部，默认修改 igmpv3report 头部对象，其他对象包括：

  igmpv3report
- **SourceAddressList** (*list*) -- Source Address List, 传入 ipv4 地址列表, 类型为 list
- **ExceedauxDataList** (*list*) -- Exceed Aux Data List, 传入 data 列表, 类型为 list
- **Args** (*Keyword*) -- RecordType (int): 类型为 int，默认值：1

  AuxDataLen (int): 类型为 int，默认值：0

  NumberOfSources (int): 类型为 int，默认值：1

  MulticastAddress (str): 类型为组播 ipv4 地址，默认值：225.0.0.1

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv3 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv3_report | Stream=${Stream} | Level=0 | NumGroupRecords=15␣
↪|
| Edit Header IPv4 Option | Stream=${Stream} | recordType=10 |
```

TesterLibrary.Stream.Header.L3.igmp.**edit_header_igmpv3_query**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv3 Query 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object

- **Level** (*int*) -- 要修改的 IGMPv3 Query 头部在流量模板中所有 IGMPv3 Query 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 11
- **MaxResponseTime** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址
- **Reserved** (*str*) -- 范围: 00-ff, 默认值: 00
- **SuppressFlag** (*int*) -- 范围: 0 or 1
- **Qrv** (*str*) -- 范围: 000 or 111
- **Qqic** (*int*) -- 范围: 0 or 255
- **NumberOfSources** (*int*) -- 范围: 0-65535
- **SourceAddressList** (*list*) -- 范围: ipv4 address list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv3Query |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv3_query | Stream=${Stream} | Level=0 | MaxResponseTime=15
↪| GroupAddress=225.0.1.1 |
```

TesterLibrary.Stream.Header.L3.igmp.**edit_header_igmpv3_report**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv2 Report 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv3 Report 头部在流量模板中所有 IGMPv3 Report 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 22
- **Reserved1** (*str*) -- 范围: 范围: 00-ff, 默认值: 00
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **Reserved2** (*str*) -- 范围: 范围: 00-ff, 默认值: 00
- **NumGroupRecords** (*int*) -- 范围: 0-65535、AUTO, AUTO 表示自动计算
- **GroupRecords** (*int*) -- 范围: 0-65535

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv3 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv3_report | Stream=${Stream} | Level=0 | NumGroupRecords=15␣
↪|
```

Module contents

TesterLibrary.Stream.Header.L4 package

Submodules

TesterLibrary.Stream.Header.L4.common module

TesterLibrary.Stream.Header.L4.common.**edit_header_tcp**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 TCP 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 tcp 头部在流量模板中所有 tcp 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **SourcePort** (*str*) --
- **DestPort** (*str*) --
- **SeqNum** (*int*) --
- **AckNum** (*int*) --
- **DataOffset** (*int*) --
- **Reserved** (*str*) --
- **Flags** (*int*) --
- **WindowSize** (*int*) --
- **Checksum** (*str*) --
- **UrgentPointer** (*str*) --
- **Option** (*str*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Tcp | Stream=${Stream} | Level=0 | SourcePort=1024 |
```

TesterLibrary.Stream.Header.L4.common.**edit_header_udp**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 UDP 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 udp 头部在流量模板中所有 udp 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **SourcePort** (*str*) --
- **DestPort** (*str*) --
- **Length** (*int*) --
- **Checksum** (*str*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Udp | Stream=${Stream} | Level=0 | SourcePort=1024 |
```

Module contents

TesterLibrary.Stream.Header.Routing package

Submodules

TesterLibrary.Stream.Header.Routing.ospfv2 module

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_ack**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Link State Acknowledge 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Link State Acknowledge 头部在流量模板中所有 OSPFv2 Link State Acknowledge 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --

- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --
- **PacketOptionsReserved7** --
- **PacketOptionsReserved6** --
- **PacketOptionsDcBit** --
- **PacketOptionsEaBit** --
- **PacketOptionsNpBit** --
- **PacketOptionsMcBit** --
- **PacketOptionsEBit** --
- **PacketOptionsReserved0** --
- **LsaHeaderCount** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Ack | Stream=${Stream} | Level=0 | InterfaceMtu=9000 |␣
↪LsaHeaderCount=1 |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_dd**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Database Description 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Database Description 头部在流量模板中所有 OSPFv2 Database Description 头部的序列号, 默认值：0, 范围：0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --

- **AuthType** (*str*) -- SimplePassword

  MD5

  UserDefined

  NoAuth
- **AuthValue1** --
- **AuthValue2** --
- **PacketOptionsReserved7** --
- **PacketOptionsReserved6** --
- **PacketOptionsDcBit** --
- **PacketOptionsEaBit** --
- **PacketOptionsNpBit** --
- **PacketOptionsMcBit** --
- **PacketOptionsEBit** --
- **PacketOptionsReserved0** --
- **InterfaceMtu** --
- **SequenceNumber** --
- **DdOptionsReserved7** --
- **DdOptionsReserved6** --
- **DdOptionsReserved5** --
- **DdOptionsReserved4** --
- **DdOptionsReserved3** --
- **DdOptionsIBit** --
- **DdOptionsMBit** --
- **DdOptionsMsBit** --
- **LsaHeaderCount** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪OSPFv2DatabaseDescription |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Dd | Stream=${Stream} | Level=0 | InterfaceMtu=9000 |␣
↪LsaHeaderCount=1 |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_hello**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Hello 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object

- **Level** (*int*) -- 要修改的 OSPFv2 Hello 头部在流量模板中所有 OSPFv2 Hello 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --
- **PacketOptionsReserved7** --
- **PacketOptionsReserved6** --
- **PacketOptionsDcBit** --
- **PacketOptionsEaBit** --
- **PacketOptionsNpBit** --
- **PacketOptionsMcBit** --
- **PacketOptionsEBit** --
- **PacketOptionsReserved0** --
- **NetworkMask** --
- **HelloInterval** --
- **RouterPriority** --
- **RouterDeadInterval** --
- **DesignatedRouter** --
- **BackupDesignatedRouter** --
- **Neighbors** --

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | OSPFv2Hello |
| ${Neighbors} | Create List | 2.2.2.2 | 3.3.3.3 | 4.4.4.4 |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Hello | Stream=${Stream} | Level=0 | AuthType=2 |␣
↪Neighbors=${Neighbors} |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_lsa**(*Stream*, *HeaderType*, *Level=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 报文中 Lsa 头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **HeaderType** (*str*) -- Ospfv2LinkStateUpdateQueryHeader

  Ospfv2LinkStateRequestHeader

  Ospfv2DatabaseDescriptionHeader

  Ospfv2LinkStateAcknowledgeHeader
- **Level** (*int*) -- 要修改的 OSPFv2 HeaderType 头部在流量模板中所有 OSPFv2 HeaderType 头部的序列号, 默认值: 0, 范围: 0-65535
- **Index** (*int*) -- 要修改的 OSPFv2 HeaderType Lsa 头部在流量模板中所有 OSPFv2 HeaderType Lsa 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **LsaAge** --
- **Reserved7** --
- **Reserved6** --
- **DcBit** --
- **EaBit** --
- **NpBit** --
- **McBit** --
- **EBit** --
- **Reserved0** --
- **LsType** --
- **LinkStateId** --
- **AdvertisingRouter** --
- **LsSequenceNumber** --
- **LsChecksum** --
- **LsaLength** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪OSPFv2DatabaseDescription |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Dd | Stream=${Stream} | Level=0 | InterfaceMtu=9000 |␣
↪LsaHeaderCount=2 |
| Edit Header Ospfv2 Lsa | Stream=${Stream} | Index=0 | LsaAge=10 |␣
↪LinkStateId=4.4.4.4 |
| Edit Header Ospfv2 Lsa | Stream=${Stream} | Index=1 | LsaAge=20 |␣
↪LinkStateId=5.5.5.5 |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_request**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Link State Request 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Link State Request 头部在流量模板中所有 OSPFv2 Link State Request 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --
- **LsaHeaderCount** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | OSPFv2Unknown |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Request | Stream=${Stream} | Level=0 | AuthType=2 |␣
↪LsaHeaderCount=2 |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_unknown**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Unknown 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Unknown 头部在流量模板中所有 OSPFv2 Unknown 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | OSPFv2Unknown |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Unknown | Stream=${Stream} | Level=0 | AuthType=2 |␣
↪AuthValue1=1 |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_update**(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Link State Update 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Link State Update 头部在流量模板中所有 OSPFv2 Link State Update 头部的序列号

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --
- **NumberOfLsas** --

- **LsaHeaders** -- 支持的参数有：

  Router Network Summary SummaryAsbr AsExternal

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | ospfv2linkstateupdate |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr |␣
↪AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2 |␣
↪LsaHeaders=${LsaHeaders} |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_update_lsa**(*Stream*, *Type*, *Level=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Update 报文中 Lsa 头部内容.

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Type** (*str*) -- OSPFv2 报文 lsa 类型支持：

  Router

  Network

  Summary

  SummaryAsbr

  External

- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **Index** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号

关键字参数

- **LsaAge** (*str, optional*) --
- **Reserved7** (*int, optional*) --
- **Reserved6** --
- **DcBit** --
- **EaBit** --
- **NpBit** --
- **McBit** --
- **EBit** --
- **Reserved0** --
- **LsType** --

- **LinkStateId** --
- **AdvertisingRouter** --
- **LsSequenceNumber** --
- **LsChecksum** --
- **LsaLength** --
- **LSA** 的参数 (*External*) -- RouterLsaReserved1
  NumberOfLinks
  RouterLsaLinkCount
  reserved7Router
  reserved6Router
  dcBitRouter
  eaBitRouter
  npBitRouter
  mcBitRouter
  eBitRouter
  reserved0Router
- **LSA** 的参数 -- NetworkMask:
  AttachedRoute1:
  AttachedRouteCount:
- **LSA** 的参数 -- NetworkMask:
  LsaReserved1:
  LsaMetric:
  TosMetricsCount:
  NetworkMask:
  LsaReserved1:
  LsaMetric:
  TosMetricsCount:
- **LSA** 的参数 -- NetworkMask: External
  ExternalOptionsEBit: External
  ExternalOptionsReserved: External
  ExternalRouteMetric: External
  ForwardingAddress: External
  ExternalRouteTag: External
  TosMetricsCount: External

返回 (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr |␣
↪AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2 |␣
↪LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=0 |␣
↪Type=Router | RouterLsaReserved1=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=1 |␣
↪Type=Network | NetworkMask=255.255.0.0 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=2 |␣
↪Type=Summary | LsaReserved1=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=3 |␣
↪Type=SummaryAsbr | LsaMetric=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=4 |␣
↪Type=AsExternal | ExternalOptionsEBit=1 |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_update_nework_attached_ro**

修改测试仪表流量模板中 OSPFv2 Update 报文中 Nework Lsa 头部 Attached Route 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **LsaIndex** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号
- **Index** (*int*) -- 要修改的 OSPFv2 update Nework Lsa Attached Route 头部在流量模板中所有 Nework Lsa Attached Route 头部的序列号

关键字参数 **RouterID** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr |␣
↪AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2 |␣
↪LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=1 |␣
↪Type=Network | AttachedRouteCount=2 |
| Edit Header Ospfv2 Update Nework Attached Route | Stream=${Stream} |␣
↪Level=0 | LsaIndex=1 | Index=0 | RouterID=2.2.2.2 |
```

(下页继续)

(续上页)

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_update_route_link_tos_met**

修改测试仪表流量模板中 OSPFv2 Update 报文中 Route Lsa 头部 Link 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **LsaIndex** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号
- **MetricIndex** (*int*) -- 要修改的 OSPFv2 update Route Lsa Link 头部在流量模板中所有 OSPFv2 update Route Lsa Link 头部的序列号
- **Index** (*int*) -- 要修改的 OSPFv2 update Route Lsa Link Tos Metric 头部在流量模板中所有 OSPFv2 update Route Lsa Link Tos Metric 头部的序列号

关键字参数

- **RouterLsaLinkType** --
- **RouterLsaMetricReserved** --
- **RouterTosLinkMetrics** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr |␣
↪AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2 |␣
↪LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=0 |␣
↪Type=Router | RouterLsaLinkCount=2 |
| Edit Header Ospfv2 Update Route Lsa Link | Stream=${Stream} | Level=0 |␣
↪LsaIndex=0 | Index=0 | RouterLsaTosMetricsCount=2 |
| Edit Header Ospfv2 Update Route Link Tos Metric | Stream=${Stream} |␣
↪Level=0 | LsaIndex=0 | MetricIndex=0 | Index=0 | RouterLsaMetricReserved=1 |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_update_route_lsa_link**(*Stre*
Leve
Lsai
dex=
In-
dex=
***kw*

修改测试仪表流量模板中 OSPFv2 Update 报文中 Route Lsa 头部 Link 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **LsaIndex** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号
- **Index** (*int*) -- 要修改的 OSPFv2 update Route Lsa Link 头部在流量模板中所有 OSPFv2 update Route Lsa Link 头部的序列号

关键字参数

- **LinkId** (*int*) --
- **LinkData** (*int*) --
- **RouterLsaLinkType** (*int*) -- 1：Point-to-Point 2：Transit 3：Stub 4：Virtual
- **NumRouterLsaTosMetrics** (*int*) --
- **RouterLinkMetrics** (*int*) --
- **RouterLsaTosMetricsCount** (*int*) --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr |␣
↪AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2 |␣
↪LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=0 |␣
↪Type=Router | RouterLsaLinkCount=2 |
| Edit Header Ospfv2 Update Route Lsa Link | Stream=${Stream} | Level=0 |␣
↪LsaIndex=0 | Index=0 | LinkId=2.2.2.2 |
```

TesterLibrary.Stream.Header.Routing.ospfv2.**edit_header_ospfv2_update_tos_metric**(*Stream*,
Type,
Level=0,
LsaIn-
dex=0,
In-
dex=0,
***kwargs*

修改测试仪表流量模板中 OSPFv2 Update 报文中 Summary、SummaryAsbr 或 AsExternal Lsa 头部 Tos Metric 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Type** (*str*) -- OSPFv2 Update 报文中 Lsa 类型，支持:

  Summary SummaryAsbr AsExternal
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **LsaIndex** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号
- **Index** (*int*) -- 要修改的 Summary、SummaryAsbr 或 AsExternal Lsa 头部在流量模板中所有 Summary、SummaryAsbr 或 AsExternal Lsa 头部的序列号

关键字参数

- **Summary 或 SummaryAsbr,** 支持 **Args** -- MetricReserved LinkMetrics
- **AsExternal,** 支持 **Args** -- EBit RouteTos RouteMetrics ForwardingAddress RouteTag

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr |␣
↪AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2 |␣
↪LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=2 |␣
↪Type=Summary | TosMetricsCount=2 |
| Edit Header Ospfv2 Update Tos Metric | Stream=${Stream} | Type=Summary |␣
↪Level=0 | LsaIndex=2 | Index=0 | MetricReserved=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=3 |␣
↪Type=SummaryAsbr | TosMetricsCount=2 |
| Edit Header Ospfv2 Update Tos Metric | Stream=${Stream} | Type=SummaryAsbr␣
↪| Level=0 | LsaIndex=3 | Index=0 | MetricReserved=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=4 |␣
↪Type=AsExternal | TosMetricsCount=2 |
| Edit Header Ospfv2 Update Tos Metric | Stream=${Stream} | Type=AsExternal |␣
↪Level=0 | LsaIndex=4 | Index=0 | EBit=1 |
```

Module contents

Module contents

Submodules

TesterLibrary.Stream.common module

TesterLibrary.Stream.common.**add_stream**(*Type='raw', Ports=None, Names=None, FilePath=None, IncludeCrc=True, SrcPoints=None, DstPoints=None, SrcInterface=None, DstInterface=None, Bidirection=None, Direction=None, Layer=None, TrafficMeshMode=None, EndpointMapping=None, AutoCreateTunnel=False, StreamOnly=None, **kwargs*)

测试仪表创建流量

参数

- **Type** (*str*) -- 创建绑定流类型，支持 raw 流、binding 流以及 pcap 文件导入流:

  raw

  binding

  pcap

- **Ports** (list (Port)) -- raw 流和 pcap 流参数，端口对象, 测试仪表端口对象 object 列表
- **Names** (*list*) -- raw 流参数，流量名称, 流量名称列表
- **FilePath** (*str*) -- pcap 流参数，当 Type 为 pcap 需要指定导入 pcap 文件的路径
- **IncludeCrc** (*bool*) -- pcap 流参数，当 Type 为 pcap 指定导入方式是否携带 CRC, 布尔值 Bool (范围：True / False)
- **SrcInterface** (*list*) -- binding 流参数，指定源接口
- **DstInterface** (*list*) -- binding 流参数，指定目的接口
- **SrcPoints** (*list*) -- binding 流参数，指定源端点
- **DstPoints** (*list*) -- binding 流参数，指定目的端点
- **Bidirection** (*bool*) -- 是否是能双向流量, 布尔值 Bool (范围: True / False)
- **Direction** (*str*) -- binding 流参数
- **Layer** (*str*) -- binding 流参数，指定接口网络层，默认值：IPV4，支持值:

  ETHERNETII

  VLAN

  GRE

  IPV4

  IPV6

- **TrafficMeshMode** (*str*) -- binding 流参数, 默认值：MANY_TO_MANY，支持值:

ONE_TO_ONE

MANY_TO_MANY

FULL_MESH

CONGESTION

LEARNING

BACK_BONE

PAIR

- **EndpointMapping** (*str*) -- binding 流参数, 默认值: ROUND_ROBIN, 支持值:

  ROUND_ROBIN

  MANY_TO_MANY

- **AutoCreateTunnel** (*bool*) -- binding 流参数, 自动绑定隧道, 默认值: False
- **StreamOnly** (*bool*) -- True 为每个 flow 创建一个 stream, False 为多个 flow 创建一个 stream, 默认值: False

返回 创建的流量对象 object 列表

返回类型 list (StreamTemplate)

实际案例

```
# raw流
| ${Streams} | add_stream | Port=${Port} |
# pcap流
| ${Streams} | add_stream | Type=pcap | Port=${Port} | FilePath=${Pcap_File_
↪Path} | IncludeCrc=True |
# binding流
| ${Streams} | add_stream | Type=binding | SrcPoints=${Points_1} | DstPoints=
↪${Points_2} | Bidirection=True |
```

TesterLibrary.Stream.common.**create_stream_header**(*Stream*, *HeaderTypes*, *Index=None*)

创建流量报文头部

参数

- **Stream** (StreamTemplate) --
- **Index** (*int*) -- 报文头部创建在当前流量头部的序号, 类型为: number, 取值范围 None 或 0-16383, 当 Index 为 None 表示重新创建流量报文类型
- **HeaderTypes** (*list*) -- 报文头部类型列表, 类型为: list, 支持的报文头部 (不区分大小写):

  ethernetii

  vlan

  vxlan

  arp

  ipv4

  ipv6

  tcp

  udp

l2tpv2data
ppp
pppoe
icmpv4echorequest
destunreach
icmpv4echoreply
informationreply
informationrequest
icmpv4parameterproblem
icmpv4redirect
sourcequench
timeexceeded
timestampreply
timestamprequest
icmpmaskrequest
icmpmaskreply
destinationunreachable
icmpv6echoreply
icmpv6echorequest
packettoobig
icmpv6parameterproblem
timeexceed
routersolicit
routeradvertise
icmpv6redirect
neighborsolicit
neighboradvertise
mldv1query
mldv1report
mldv1done
mldv2query
mldv2report
igmpv1
igmpv1query
igmpv2
igmpv2query
igmpv3report
igmpv3query
custom

ospfv2linkstateupdate

ospfv2linkstaterequest

ospfv2databasedescription

ospfv2linkstateacknowledge

ospfv2unknown

ospfv2hello

mpls

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${HeaderTypes} | Create List | EthernetII | IPv4 | TCP |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
```

TesterLibrary.Stream.common.**del_stream**(*Ports=None*, *Streams=None*)

删除测试仪流量

参数

- **Ports** (list (Port)) -- 测试仪表端口对象列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量对象列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream_1} | Add Stream | Ports=${Port_1}
| ${Stream_2} | Add Stream | Ports=${Port_2}
| ${Stream_2} | Add Stream | Ports=${Port_3}
| Del Stream | Streams=${Stream_1} |
| Del Stream | Ports=${Port_2} |
| Del Stream |
```

TesterLibrary.Stream.common.**edit_modifier**(*Stream*, *Attribute*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中指定报文字段的跳变域

参数

- **Stream** (StreamTemplate) --
- **Attribute** (*str*) -- 要修改的报文字段参数名称
- **Level** (*int*) -- 当 HeaderType=None 表示要修改的报文字段在流量模板中所有报文头部的序列号，当 HeaderType!=None 表示要修改的报文字段在流量模板中所有相同类型报文头部的序列号，默认值：0

Keyword Args: 跳变域参数参数，支持以下参数:

Type (str): 跳变类型:

Increment

Decrement

Random

List

Start (str): 跳变起始数据

Count (int): 跳变数量

Step (int): 跳变步长

Range (int): 随机跳变范围

Seed (int): 随机跳变种子

StreamType (str): 流跳变类型

Offset (int): 跳变偏移位

Mask (str): 跳变掩码

List (list): list 跳变时, 跳变列表

HeaderType (str): 要跳变的报文头部名称, 默认是 None, 如果 HeaderType 为 None, 修改的报文头 Level 决定 (要修改的报文字段在流量模板中所有报文头部的序列号)，支持:

ethernetii

vlan

vxlan

arp

ipv4

ipv6

tcp

udp

l2tpv2data

ppp

icmpv4echorequest

destunreach

icmpv4echoreply

informationreply

informationrequest

icmpv4parameterproblem

icmpv4redirect

sourcequench

timeexceeded

timestampreply

timestamprequest

icmpmaskrequest

icmpmaskreply

destinationunreachable

icmpv6echoreply

icmpv6echorequest

packettoobig

icmpv6parameterproblem

timeexceed

routersolicit

routeradvertise

icmpv6redirect

neighborsolicit

neighboradvertise

mldv1query

mldv1report

mldv1done

mldv2query

mldv2report

igmpv1

igmpv1query

igmpv2

igmpv2query

igmpv3report

igmpv3query

custom

ospfv2linkstateupdate

ospfv2linkstaterequest

ospfv2databasedescription

ospfv2linkstateacknowledge

ospfv2unknown

ospfv2hello

mpls

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Streams} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | TCP |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
# 不指定HeaderType, Level=1选中IPv4头部
| Edit Modifier | Stream=${Stream} | Level=1 | Attribute=source | Start=192.
↪168.1.1 | Count=10 | Step=1 |
# 指定HeaderType=IPv4, Level=0选中IPv4头部
| Edit Modifier | Stream=${Stream} | Level=0 | HeaderType=IPv4 |␣
↪Attribute=destination | Start=192.168.1.1 | Count=10 | Step=1 |
```

TesterLibrary.Stream.common.**edit_stream**(*Stream*, ***kwargs*)

修改测试仪表流量模板参数

参数 **Stream** (StreamTemplate) --

关键字参数

- **RepeatCount** (*int*) -- 流模板发送重复次数，默认值：1，支持值：1-4294967295
- **EnableSignature** (*bool*) -- 启用签名，默认值：True
- **FrameLengthType** (*str*) -- 流模板帧长度类型类型为：string，默认值：FIXED，支持值：

  FIXED INCREMENT RANDOM AUTO DECREMENT IMIX
- **RandomLengthSeed** (*int*) -- 随机种子，类型为：number，默认值：10900842，支持值：0-4294967295
- **FixedLength** (*int*) -- 固定帧长, 默认值：128，支持值：12-16383
- **MinLength** (*int*) -- 最小帧长, 默认值：128，支持值：12-16383
- **MaxLength** (*int*) -- 最大帧长, 默认值：256，支持值：12-16383
- **StepLength** (*int*) -- 帧长跳变步长, 默认值：1，支持值：1-8192
- **PayloadType** (*str*) -- 净荷类型, 默认值：CYCLE，支持值：

  CYCLE INCREMENT RANDOM
- **PayloadValue** (*str*) -- 帧长跳变步长, 默认值：0x0
- **PayloadValueType** (*str*) -- 净荷类型, 默认值：CYCLE，支持值：

  SINGLE_BYTE DOUBLE_BYTE
- **EnableNDResponse** (*bool*) -- 使用 ARP ND 自动回复, 默认值：False
- **TopLayerType** (*str*) -- 流模板报文模板类型, 默认值：IPV4，支持值：

  ETHERNETII VLAN GRE IPV4 IPV6
- **RxPorts** (list (Port)) -- 指定流量收端口
- **TrafficMeshMode** (*str*) -- binding 流参数, 默认值：MANY_TO_MANY，支持值：

  ONE_TO_ONE MANY_TO_MANY FULL_MESH CONGESTION LEARNING BACK_BONE PAIR
- **HostsMesh** (*str*) -- binding 流参数, 默认值：ROUND_ROBIN，支持值：

  ROUND_ROBIN MANY_TO_MANY

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Stream | Stream=${Stream} | TopLayerType=ETHERNETII |␣
↪TrafficMeshMode=FULL_MESH |
```

TesterLibrary.Stream.common.**get_streams**(*Ports=None*)

获取测试仪表流量对象

参数 **Ports** (list (Port)) -- 测试仪表端口对象列表

返回 测试仪表流量对象列表

返回类型 Streams (list (StreamTemplate))

实际案例

```
| Get Streams | Ports=${Ports} |
```

TesterLibrary.Stream.common.**select_rx_port**(*Streams*, *RxPorts*, *Mode=1*, *ExcludeTxPort=True*)

选择流量的收端口

参数

- **Streams** (list (StreamTemplate)) -- 测试仪表流量对象 object 列表
- **RxPorts** (list (Port)) -- 指定流量收端口对象列表
- **Mode** -- 模式, 默认值：ONE_TO_ONE，支持类型

  ONE_TO_ONE

  ONE_TO_MANY

  MANY_TO_ONE

  PAIR
- **ExcludeTxPort** (*bool*) -- 是否包括流量发送端口

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Select Rx Port | Streams=${Streams} | RxPorts=${Ports} | Mode=ONE_TO_MANY |␣
↪ExcludeTxPort=True |
```

TesterLibrary.Stream.common.**start_l2_learning**(*Type=None*, *Ports=None*, *Streams=None*, *WaitLearningDone=True*, *WaitTime=30*)

启动测试仪表流量二层学习

参数

- **Type** (*str*) -- 启动流二层学习的的类型，类型为：string，支持：Tx 或 Rx
- **Ports** (list (Port)) -- 测试仪表端口对象 object 列表，类型为：list，端口对象 object 列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量模板对象 object 列表，类型为：list，目流量模板对象 object 列表

- **WaitLearningDone** (*bool*) -- 是否等待二层学习完成，类型为：Bool (范围：True / False)
- **WaitTime** (*int*) -- 等待二层学习完成时间，类型为：number，默认值：30 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start L2 Learning |
| Sleep | 10 |
| Stop L2 Learning |
| Sleep | 3 |
```

TesterLibrary.Stream.common.**start_l3_learning**(*Ports=None*, *Streams=None*)

启动测试仪表流量三层 ARP ND 学习

参数

- **Ports** (list (Port)) -- 测试仪表端口对象 object 列表，类型为：list，端口对象 object 列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量模板对象 object 列表，类型为：list，目流量模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start L3 Learning |
| Sleep | 10 |
| Stop L3 Learning |
| Sleep | 3 |
```

TesterLibrary.Stream.common.**start_stream**(*Type=None*, *Objects=None*)

测试仪表开始发送数据流

参数

- **Type** (*str*) -- 当值为 None 时发送所有流量，按指定端口发送数据流或者按指定流模板发送数据流，类型 string(”port” 或”stream”),
- **Objects** (*list*) -- 按指定端口发送数据流或者按指定流模板发送数据流时需要指定端口对象或流模板对象列表

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| @{Objects} | Get Ports |
| Start Stream | Type=port | Objects=@{Objects} |
| Sleep | 10 |
| Stop Stream | Type=port | Objects=@{Objects} |
```

TesterLibrary.Stream.common.**stop_l2_learning**(*Ports=None*, *Streams=None*)

停止测试仪表流量二层学习

参数

- **Ports** (list (Port)) -- 测试仪表端口对象 object 列表，类型为：list，端口对象 object 列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量模板对象 object 列表，类型为：list，目流量模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start L2 Learning |
| Sleep | 10 |
| Stop L2 Learning |
| Sleep | 3 |
```

TesterLibrary.Stream.common.**stop_l3_learning**(*self*, *Ports=None*, *Streams=None*)

停止测试仪表流量三层 ARP ND 学习

参数

- **Ports** (list (Port)) -- 测试仪表端口对象 object 列表，类型为：list，端口对象 object 列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量模板对象 object 列表，类型为：list，目流量模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start L3 Learning |
| Sleep | 10 |
| Stop L3 Learning |
| Sleep | 3 |
```

TesterLibrary.Stream.common.**stop_stream**(*Type=None*, *Objects=None*)

测试仪表停止发送数据流

参数

- **Type** (*str*) -- 当值为 None 时发送所有流量，按指定端口发送数据流或者按指定流模板停止数据流, 类型 string(”port” 或”stream”),
- **Objects** (*list*) -- 按指定端口停止数据流或者按指定流模板发送数据流时需要指定端口对象或流模板对象列表

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| @{Objects} | Get Ports |
| Start Stream | Types=port | Objects=@{Objects} |
| Sleep | 10 |
| Stop Stream | Types=port | Objects=@{Objects} |
```

TesterLibrary.Stream.common.**wait_stream_state**(*Stream=None*, *State=['READY']*, *TimeOut=60*)

测试仪表停止发送数据流

参数

- **Stream** (list (StreamTemplate)) -- 测试仪表流模板对象列表
- **State** (*list*) -- 流模板状态, 默认值 READY, 支持:

  DISABLED NOTREADY READY RUNNING STOPPED PAUSED
- **TimeOut** (*int*) -- 超时时间, 默认值:60

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Start Stream |
| Sleep | 10 |
| Wait Stream State |
| Sleep | 3 |
```

TesterLibrary.Stream.imix module

TesterLibrary.Stream.imix.**add_imix_distribution_frame**(*IMix*, *Type='fixed'*, *Length=None*, *Min=None*, *Max=None*, *Weight=None*)

在 Imix 模板添加自定义帧长

参数

- **IMix** (Imix) -- 测试仪表 Imix 模板对象

- **Type** (*str*) -- 测试仪表 Imix 模板自定义帧长类型 fixed random
- **Length** (*int*) -- 测试仪表 Imix 模板自定义帧长长度，random 时有效
- **Min** (*int*) -- 测试仪表 Imix 模板自定义帧长最小帧长，random 时有效
- **Max** (*int*) -- 测试仪表 Imix 模板自定义帧长最大帧长，random 时有效
- **Weight** (*int*) -- 测试仪表 Imix 模板自定义帧长权重

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Imix} | Create Imix | Name=Imix_1 | Seed=10121112 |
| Add IMix Distribution Frame | IMix=${Imix} | Type=random | Min=64 | Max=128␣
↪| Weight=50 |
| Add IMix Distribution Frame | IMix=${Imix} | Type=random | Min=128 |␣
↪Max=256 | Weight=50 |
| Bind Stream Imix | Stream=${Stream} | Imix=${Imix} |
```

TesterLibrary.Stream.imix.**bind_stream_imix**(*Stream*, *IMix*)

将 Imix 模板和流量模板绑定

参数

- **Stream** (StreamTemplate) -- 测试仪表流量模板 StreamTemplate 对象
- **IMix** (Imix) -- 测试仪表 Imix 模板对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Imix} | Create Imix | Name=Imix_1 | Seed=10121112 |
| Bind Stream Imix | Stream=${Stream} | Imix=${Imix} |
```

TesterLibrary.Stream.imix.**create_imix**(*Name*, *Seed=None*)

创建流量 Imix 模板

参数

- **Name** (*str*) -- 创建的 Imix 模板名称
- **Seed** (*int*) -- Imix 模板随机种子

返回 Imix 模板对象

返回类型 (Imix)

实际案例

```
| Create Imix | Name=Imix_1 | Seed=10121112 |
```

TesterLibrary.Stream.imix.**del_imix_distribution_frame**(*IMix*, *Index=None*)

在 Imix 模板删除指定自定义帧长

参数

- **IMix** (Imix) -- 测试仪表 Imix 模板对象
- **Index** (*int*) -- 测试仪表 Imix 模板自定义帧长序号

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Imix} | Create Imix | Name=Imix_1 | Seed=10121112 |
| Add IMix Distribution Frame | IMix=${Imix} | Type=random | Min=64 | Max=128␣
↪| Weight=50 |
| Add IMix Distribution Frame | IMix=${Imix} | Type=random | Min=128 |␣
↪Max=256 | Weight=50 |
| Bind Stream Imix | Stream=${Stream} | Imix=${Imix} |
| Del IMix Distribution Frame | IMix=${Imix} | Index=1
```

TesterLibrary.Stream.imix.**get_imix_from_name**(*Name*)

通过 Imix 模板名称获取流量 Imix 模板对象

参数 **Name** (*str*) -- 创建的 Imix 模板名称

返回 Imix 模板对象

返回类型 (Imix)

实际案例

```
| ${Imix_TCPv4} | Get Imix From Name | Name=TCPv4 |
```

Module contents

1.1.6 TesterLibrary.Wizard package

Submodules

TesterLibrary.Wizard.benchmark module

TesterLibrary.Wizard.benchmark.**benchmark_stream_use_exist**(*Config*, *Streams*)

编辑测试套件使用已存在流量

参数

- **Config** (wizard) -- 仪表测试测试套件对象 object
- **Streams** (list (StreamTemplate)) -- 仪表测试流模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| @{Items} | Create List | throughput | frameloss |
| @{FrameSize} | Create List | 256 | 1024 | 16383 |
| ${Streams} | Add Stream | Type=binding | | SrcPoints=@{SrcPoints} |␣
↪DstPoints=@{SrcPoints} |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items} |
| Edit Benchmark Path | Configs=${Config} | Path=C:/test |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Benchmark Stream Use Exist | Config=${Wizard} | Streams=${Streams} |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=1000 |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{FrameSize}␣
↪|
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

TesterLibrary.Wizard.benchmark.**create_benchmark**(*Type='rfc2544'*, *Items=None*)

创建测试仪表测试套件

参数

- **Type** (*str*) -- 测试仪表测试套件类型, 支持的类型:

  rfc2544

  rfc2889

  rfc3918

  Asymmetric

- **Items** (*list*) -- 测试仪表测试套件中的测试项, 测试套件中的测试项列表, 支持测试项目如下:

  throughput: rfc2544 吞吐量测试

  backtoback: rfc2544 背靠背测试

  frameloss: rfc2544 丢包率测试

  latency: rfc2544 时延测试

  addressCachingCapacity: rfc2889 地址缓存容量测试

  addressLearningRate: rfc2889 地址学习速率测试

  broadcastLatency: rfc2889 广播帧转发测试

  broadcastForwarding: rfc2889 广播帧时延测试

  congestion: rfc2889 拥塞控制测试

  erroredFrameFilter: rfc2889 错误帧过滤测试

  forwarding: rfc2889 转发测试

  forwardPressure:

  maxForwarding:

  mixedThroughput: rfc3918 混合吞吐量测试

  scaledGroupForwarding: rfc3918 组转发矩阵测试

  multicastThroughput: rfc3918 聚合组播吞吐量测试

  multicastGroupCapacity: rfc3918 组播组容量测试

  multicastLatency: rfc3918 组播转发时延测试

multicastJoinLeaveLatency: rfc3918 加入离开时延测试

返回

仪表测试测试套件对象 object

(list (test_config)): 仪表测试测试套件测试项对象 object 列表

返回类型 (wizard_config)

实际案例

robotframework:

```
| ${Items} | Create List | throughput | frameloss |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items} |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} | Mode=meshed |␣
↪Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=${L2_TestTime} |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{L2_
↪FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

TesterLibrary.Wizard.benchmark.**create_benchmark_streams**(*Config*, *Items*, *Type*, *SrcPoints*, *DstPoints*, *Bidirectional=False*, *Mode='1v1'*, *Mapping='roundrobin'*, *Monitors=()*)

创建测试仪表测试套件流量

参数

- **Config** (wizard_config) -- 仪表测试测试套件对象 object
- **Items** (*list*) -- 测试仪表测试套件中的测试项, 测试套件中的测试项列表, 支持测试项目如下:

  throughput

  backtoback

  frameloss

  latency

  addressCachingCapacity

  addressCachingRate

  broadcastLatency

  broadcastForwarding

  congestion

  erroredFrameFilter

  forwarding

  forwardPressure

  maxForwarding

mixedThroughput

scaledGroupForwarding

multicastThroughput

multicastGroupCapacity

multicastLatency

multicastJoinLeaveLatency

- **Type** (*str*) -- 创建绑定流类型, 支持类型有:

  eth

  ipv4

  ipv6

- **SrcPoints** (*list*) -- 创建绑定流类型源端点对象 object 列表
- **DstPoints** (*list*) -- 创建绑定流类型目的端点, 目的端点对象 object 列表
- **Bidirectional** (*bool*) -- 是否使能双向流量, 默认值: False
- **Mode** (*str*) -- 创建绑定流 topo 类型

  1v1

  m2m

  meshed

- **Mapping** (*str*) -- 创建绑定流端点模式, 支持类型

  roundrobin

  manytomany

- **Monitors** (*list*) -- 作为镜像端口的测试仪表端口对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Items} | Create List | throughput | frameless |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items} |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} | Mode=meshed |␣
↪Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=${L2_TestTime} |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{L2_
↪FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

TesterLibrary.Wizard.benchmark.**del_benchmark**()

TesterLibrary.Wizard.benchmark.**edit_benchmark_address_learning_capacity**(*Config*, *MinAddressCount=1*, *MaxAddressCount=65536*, *InitAddressCount=20480*, *Resolution=2*, *AgingTime=15*, *LearningRate=10000*)

编辑 RFC2889 测试套件地址容量测试项参数

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **MinAddressCount** (*int*) -- 学习地址最小值, 默认值: 1, 范围: 1-16777216
- **MaxAddressCount** (*int*) -- 习地址最小值默认值: 65536, 范围: 1-16777216
- **InitAddressCount** (*int*) -- 20480, 范围: 1-16777216
- **Resolution** (*int*) -- 精度 (%), 默认值: 2, 范围: 1-100
- **AgingTime** (*int*) -- 老化时间 (秒), 默认值: 50, 范围: 1-3600
- **LearningRate** (*int*) -- 地址学习速率 (帧/秒), 默认值: 10000, 范围: 1-148809523

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_address_learning_rate**(*Config*, *MinRateCount=1488*, *MaxRateCount=1488*, *InitRateCount=1488*, *Resolution=2*, *AgingTime=15*, *AddressCount=1000*)

编辑 RFC2889 测试套件地址容量测试项参数

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **MinRateCount** (*int*) -- 学习地址最小值, 默认值: 1488, 范围: 1-148809523
- **MaxRateCount** (*int*) -- 习地址最小值默认值: 1488, 范围: 1-148809523
- **InitRateCount** (*int*) -- 1488, 范围: 1-148809523
- **Resolution** (*int*) -- 精度 (%), 默认值: 2, 范围: 1-100
- **AgingTime** (*int*) -- 老化时间 (秒), 默认值: 50, 范围: 1-3600
- **AddressCount** (*int*) -- 地址数量, 默认值: 1000, 范围: 1-4294967295

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_burst_count_loop**(*Config*, *Mode='step'*, *Start=1*, *End=1*, *Step=1*, *Custom=[1, 2]*)

编辑 RFC2889 测试套件突发帧数，设置测试项: 广播帧转发测试、拥塞控制测试、错误帧过滤测试、转发测试

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Mode** (*str*) -- 负载单位, 默认值: step, 支持类型:

  step

custom

- **Start** (*int*) -- 开始帧数, 默认值: 1, 范围: 1-65535
- **End** (*int*) -- 结束帧数, 默认值: 1, 范围: 1-65535
- **Step** (*int*) -- 1, 范围: 1-65535
- **Custom** (*list[int]*) -- 自定义帧数, 默认值: [1, 2]

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_duration**(*Config*, *Trial=1*, *Mode='second'*, *Count=100*)

编辑测试套件测试时长设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Trial** (*int*) -- 测试验次数, 默认值: 1
- **Mode** (*str*) -- 模式, 默认值: second, 支持 second 和 burst
- **Count** (*int*) -- 突发包个数 (帧) 或时长 (秒), 默认值: 100, 范围: 1-80000000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_errored_frame_filtering**(*Config*, *CrcTested=True*, *CrcFrameLength=64*, *UndersizedTested=True*, *UndersizedFrameLength=60*, *OversizedTested=True*, *OversizedFrameLength=1519*, *MaxLegalFrameLength=1518*, *BurstSize=1*)

编辑 RFC2889 测试套件错误帧过滤测试项参数

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **CrcTested** (*bool*) -- 使能错误类型 CRC, 默认值: True
- **CrcFrameLength** (*int*) -- CRC 帧长度, 默认值: 64, 范围: 64-10000
- **UndersizedTested** (*bool*) -- 使能超短帧长度, 默认值: True
- **UndersizedFrameLength** (*int*) -- 60, 范围: 58-63
- **OversizedTested** (*int*) -- 使能超长帧长度, 默认值: True
- **OversizedFrameLength** (*int*) -- 超长帧长度, 默认值: 1519, 范围: 1519-16383
- **MaxLegalFrameLength** (*int*) -- 最大合法帧长, 默认值: 1518, 范围: 1-4294967295
- **BurstSize** (*int*) -- 地址数量, 默认值: 1000, 范围: 1-4294967295

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_frame**(*Config*, *Type='custom'*, *Length=128*, *Min=128*, *Max=256*, *Start=128*, *End=256*, *Step=128*, *Custom=None*, *ImixTemplates=None*)

编辑测试套件帧长度设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Type** (*str*) -- 帧长度类型, 默认值: custom, 支持类型:

  fixed

  random

  step

  custom

  fixed

  imix
- **Length** (*int*) -- 固定帧长值, 默认值: 128, 范围: 58-16383
- **Min** (*int*) -- 最小帧长值, 默认值: 128, 范围: 58-16383
- **Max** (*int*) -- 最大帧长值, 默认值: 128, 范围: 58-16383
- **Start** (*int*) -- 开始帧长值, 默认值: 128, 范围: 58-16383
- **End** (*int*) -- 结束帧长值, 默认值: 128, 范围: 58-16383
- **Step** (*int*) -- 跳变帧步长值, 默认值: 128, 范围: 58-16383
- **Custom** (*int*) -- 自定义帧长列表, 默认值: [64, 128, 256, 512, 1024, 1280, 1518], 范围: 58-16383
- **ImixTemplates** (*list*) -- IMix 模板列表, 默认值: None

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

robotframework:

```
| @{Items} | Create List | throughput | frameloss |
| @{FrameSize} | Create List | 256 | 1024 | 16383 |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items} |
| Edit Benchmark Path | Configs=${Config} | Path=C:/test |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} | Mode=meshed |␣
↪Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=1000 |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{FrameSize}␣
↪|
```

(下页继续)

(续上页)

```
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

TesterLibrary.Wizard.benchmark.**edit_benchmark_latency**(*Configs*, *, *Type='FIFO'*, *DelayBefore=2*, *DelayAfter=10*)

编辑测试套件时间参数

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Type** (*str*) -- 时延类型, 支持以下类型:

  LIFO: (Store and forward)

  FIFO: (Bit forwarding);

  LILO

  FILO
- **DelayBefore** (*int*) -- 启动流前延迟时间，单位：秒，，默认值：2，范围：1-3600
- **DelayAfter** (*int*) -- 停止流后延迟时间，单位：秒，，默认值：10，范围：1-3600

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_learning**(*Configs*, *Frequency*, *, *EnableLearning=True*, *LearningRate=1000*, *LearningRepeat=5*, *DelayBefore=2*, *EnableArp=False*, *ArpRate=1000*, *ArpRepeat=5*)

编辑测试套件地址学习设置

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Frequency** (*str*) -- 学习频率, 支持以下类型:

  once

  trial

  frame

  iter
- **EnableLearning** (*bool*) -- 使能地址学习, 默认值: True
- **LearningRate** (*int*) -- 地址学习速率，单位：帧/秒, 默认值：1000, 范围：1-14880952
- **LearningRepeat** (*int*) -- 学习重复次数, 默认值: 5, 范围: 1-65536

- **DelayBefore** (*int*) -- 学习延迟时间，单位: 秒, 默认值: 2, 范围: 1-65536
- **EnableArp** (*bool*) -- 使能三层 ARP 学习, 默认值: True
- **ArpRate** (*int*) -- ARP 学习速率,单位: 秒, 默认值: 1000, 范围: 1-14880952
- **ArpRepeat** (*int*) -- ARP 学习重复次数, 默认值: 5, 范围: 1-65536

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_multicast_base_parameters**(*Configs*, *Version='igmpv2'*, *Ipv4GroupAddressStart=225.0.0.1*, *Ipv4GroupAddressStep=0.1.0.0*, *Ipv4PrefixLength=32*, *Ipv6GroupAddressStart=ff1e:*, *Ipv6GroupAddressStep=0:0:0:0:0:0:0:1*, *Ipv6PrefixLength=128*, *GroupIncrement=1*, *JoinGroupDelay=15*, *LeaveGroupDelay=15*, *JoinLeaveSendRate=1*, *GroupDistributeMode='even'*)

编辑 RFC3918 测试套件-组播参数

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Version** (*str*) -- 负载单位, 默认值: percent, 支持类型:

  igmpv1

  igmpv2

  igmpv3

  mldv1

  mldv2
- **Ipv4GroupAddressStart** (*str*) -- 起始 IP 地址, 默认值: 225.0.0.1
- **Ipv4GroupAddressStep** (*str*) -- 起始 IP 步长, 默认值: 0.1.0.0
- **Ipv4PrefixLength** (*int*) -- IP 前缀长度, 默认值: 32, 范围: 1-32
- **Ipv6GroupAddressStart** (*str*) -- 起始 IPv6 地址, 默认值: ffle:

- **Ipv6GroupAddressStep** (*str*) -- 起始 IPv6 步长, 默认值: 0:0:0:1:
- **Ipv6PrefixLength** (*int*) -- IPv6 前缀长度, 默认值: 128, 范围: 1-128
- **GroupIncrement** (*int*) -- 组跳变步长, 默认值: 1, 范围: 1-4294967295
- **JoinGroupDelay** (*int*) -- 加入组延迟(秒), 默认值: 15, 范围: 0-4294967295
- **LeaveGroupDelay** (*int*) -- 离开组延迟(秒), 默认值: 15, 范围: 0-4294967295
- **JoinLeaveSendRate** (*int*) -- 组播发消息速率(包/秒), 默认值: 1000, 范围: 0-1000000000
- **GroupDistributeMode** (*str*) -- 组播组分布模式, 默认值: even, 范围:

  even: 平均

  weigh: 权重

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_multicast_group_count_loop**(*Config*, *LoopMode='step'*, *FixedGroup=10*, *MinGroup=10*, *MaxGroup=50*, *StartGroup=10*, *EndGroup=50*, *StepGroup=10*, *CustomGroup=(10, 20, 100)*)

RFC3918 测试套件-组播组

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **LoopMode** (*str*) -- 模式, 默认值: step, 支持:

  fixed random step custom
- **FixedGroup** (*str*) -- 固定, 默认值: 10
- **MinGroup** (*str*) -- 最小, 默认值: 10
- **MaxGroup** (*int*) -- 最大, 默认值: 50
- **StartGroup** (*int*) -- 开始, 默认值: 10

- **EndGroup** (*int*) -- 结束, 默认值: 50
- **StepGroup** (*int*) -- 步长, 默认值: 10
- **CustomGroup** (*list[int]*) -- 自定义比例, 默认值: (10, 20, 100)

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_multicast_join_leave_delay**(*Config*, *DelayBetweenJoinAndStartStream=10*, *DelayBetweenJoinAndLeave=10*)

RFC3918 测试套件-配置加入离开组时延

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **DelayBetweenJoinAndStartStream** (*int*) -- 从启动流量开始到发送加入组报文之间的时间间隔。单位是秒, 默认值: 10, 范围: 0-3600
- **DelayBetweenJoinAndLeave** (*int*) -- 发送加入组报文和发送离开组报文之间的时间间隔。单位是秒, 默认值: 10, 范围: 0-3600

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_multicast_mixed_throughput_unicast_streams**

RFC3918 测试套件组播混合吞吐量-配置单播流量

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Streams** (list[(StreamTemplate)]) -- 仪表测试流模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_multicast_other**(*Configs*, *StopTestWhenFailed=True*, *VerifyFreq='topo_changed'*, *DurationMode='second'*, *TimeDurationCount=1*, *BurstDurationCount=100*, *TxFrameRate=1000*)

编辑 RFC3918 测试套件-组播参数-其他

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **StopTestWhenFailed** (*bool*) -- 如果流验证失败停止测试, 默认值: True
- **VerifyFreq** (*str*) -- 验证频率, 默认值: topo_changed, 支持: none topo_changed frame iteration
- **DurationMode** (*str*) -- 时长模式, 默认值: second, 支持: second burst
- **TimeDurationCount** (*int*) -- 发送秒速, 默认值: 1000
- **BurstDurationCount** (*int*) -- 帧发送数量, 默认值: 100
- **TxFrameRate** (*int*) -- 帧发送速率 (帧/秒), 默认值: 1000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_multicast_stream_tos**(*Configs*, *Tos=0*, *FlowLabel=0*, *TTL=7*, *Priority=0*)

编辑 RFC3918 测试套件-组播参数-流配置

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Tos** (*int*) -- IPv4 TOS 值, 默认值: 0
- **FlowLabel** (*int*) -- IPv6 Flow Label 值, 默认值: 0
- **TTL** (*int*) -- IPv4 TTL 值, 默认值: 10
- **Priority** (*int*) -- VLAN 优先级, 默认值: 0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_multicast_traffic_ratio_loop**(*Config*, *LoopMode='step'*, *FixedRatio=10*, *MinRatio=10*, *MaxRatio=50*, *StartRatio=10*, *EndRatio=50*, *StepRatio=10*, *CustomRatio=(10, 20, 100)*)

RFC3918 测试套件配置组播混合吞吐量-组播百分比设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **LoopMode** (*str*) -- 模式, 默认值: step, 支持:
  fixed random step custom
- **FixedRatio** (*str*) -- 固定比例, 默认值: 10
- **MinRatio** (*str*) -- 最小比例, 默认值: 10
- **MaxRatio** (*int*) -- 最大比例, 默认值: 50
- **StartRatio** (*int*) -- 开始比例, 默认值: 10
- **EndRatio** (*int*) -- 结束比例, 默认值: 50
- **StepRatio** (*int*) -- 步长, 默认值: 10
- **CustomRatio** (*list[int]*) -- 自定义比例, 默认值: (10, 20, 100)

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_path**(*Configs*, *Path*)

编辑测试套件地址学习设置

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Path** (*str*) -- 测试结果 DB 文件保存的绝对路径

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_result_file_name**(*Config*, *EnableCustomResult=False*, *ResultFileName=None*, *AddTimeStamp=True*)

配置自定义测试结果名称

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object 列表
- **EnableCustomResult** (*bool*) -- 使用自定义测试结果名称, 默认值: False
- **ResultFileName** (*str*) -- 结果名称
- **AddTimeStamp** (*bool*) -- 添加时间戳, 默认值: False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_search**(*Config*, *Mode='binary'*, *Lower=1*, *Upper=100*, *Init=10*, *Step=10*, *Resolution=1*, *Ratio=50*, *Acceptance=0*, *Ignore=False*, *EnableLatency=False*, *Maxlatency=30*)

编辑测试套件测试负载设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object

- **Mode** (*str*) -- 负载类型, 默认值: binary, 支持类型:

  binary

  step

  combo
- **Lower** (*int*) -- 速率下限 (%), 默认值: 1, 范围: 0.001-100
- **Upper** (*int*) -- 速率上限 (%), 默认值: 100, 范围: 0.001-100
- **Init** (*int*) 初始速率 (%) -- 10, 范围: 0.001-100
- **Step** (*int*) -- 步长 (%), 默认值: 10, 范围: 0.001-100
- **Resolution** (*int*) -- 精度 (%), 默认值: 1, 范围: 0.001-100
- **Ratio** (*int*) -- 二分法查找百分比 (%), 默认值: 50, 范围: 0.001-99.9999
- **Acceptance** (*int*) -- 可接受的丢包率, 默认值: 0, 范围: 0-100
- **Ignore** (*bool*) -- 忽略上下限, 默认值: False
- **EnableLatency** (*bool*) -- 使能时延吞吐量, 默认值: False
- **Maxlatency** (*int*) -- 最大允许时延, 默认值: 30

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_traffic_load_loop**(*Config*, *LoadUnit='percent'*, *LoadMode='custom'*, *FixedLoad=10*, *LoadMin=10*, *LoadMax=50*, *LoadStart=10*, *LoadEnd=50*, *LoadStep=10*, *LoadCustom=(10, 20, 50)*)

编辑测试套件负载设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **LoadUnit** (*str*) -- 负载单位, 默认值: percent, 支持类型:

  percent

  fps

  mbps

kbps

bps

Bps

ifg

- **LoadMode** (*str*) -- 负载类型, 默认值: custom, 支持类型:

  fixed

  random

  step

  custom

- **FixedLoad** (*int*) -- 固定负载, 默认值: 10, 范围: 0.001-100
- **LoadMin** (*int*) -- 10, 范围: 0.001-100
- **LoadMax** (*int*) -- 最大负载, 默认值: 50, 范围: 0.001-100
- **LoadStart** (*int*) -- 开始负载, 默认值: 10, 范围: 0.001-100
- **LoadEnd** (*int*) -- 结束负载, 默认值: 50, 范围: 0.001-100
- **LoadStep** (*int*) -- 负载步长, 默认值: 10, 范围: 0.001-100
- **LoadCustom** (*list[int]*) -- 自定义负载, 默认值: [10, 20, 50]

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

TesterLibrary.Wizard.benchmark.**edit_benchmark_transport_layer**(*Configs*, *HeaderType=None*, *EnableRandomPort=True*, *SrcPortBase=7*, *SrcPortStep=1*, *SrcPortCount=0*, *DstPortBase=7*, *DstPortStep=1*, *DstPortCount=0*)

编辑测试套件使用已存在流量

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **HeaderType** (*str*) -- 报文头类型, 默认值: none, 支持: none tcp udp
- **EnableRandomPort** (*bool*) -- 使能随机端口, 默认值: True
- **SrcPortBase** (*int*) -- 源端口起始值, 默认值: 7, 范围: 0-65535
- **SrcPortStep** (*int*) -- 源端口步长, 默认值: 1, 范围: 0-65535
- **SrcPortCount** (*int*) -- 源端口数量, 默认值: 0, 范围: 0-65535

- **DstPortBase** (*int*) -- 目的端口起始值, 默认值: 7, 范围: 0-65535
- **DstPortStep** (*int*) -- 1, 范围: 0-65535
- **DstPortCount** (*int*) -- 目的端口数量, 默认值: 0, 范围: 0-65535

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Items} | Create List | throughput | frameless |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items} |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} | Mode=meshed |␣
↪Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=${L2_TestTime} |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{L2_
↪FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

TesterLibrary.Wizard.benchmark.**export_benchmark_result**(*Result*, *Path*, *Sheet*)

将测试套件的结果导出到 excel 表格

参数

- **Result** (*list*) -- 测试套件直接结果 DB 文件中获取指定测试结果数据列表
- **Path** (*str*) -- 导出文件的路径, (例如：”C:/Report.xls”)
- **Sheet** -- (str) 导入的 excel 文件的 sheet 页名称, (例如：”ipv4 natp”)

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${result} | Export Benchmark Result | ${Result} | ${Path} | ${Sheet} |
```

TesterLibrary.Wizard.benchmark.**format_benchmark_result**(*Result*)

格式化列表为二维表格形式

参数 **Result** (*list*) -- 测试套件直接结果 DB 文件中获取指定测试结果数据列表

返回 PrettyTable 对象

返回类型 (PrettyTable)

实际案例

robotframework:

```
| ${result} | Format Benchmark Result | ${Result} |
```

TesterLibrary.Wizard.benchmark.**get_benchmark_result**(*DB*, *Type*, *Item*, *FrameSize=None*, *Mbps=None*, *All=False*)

从测试套件执行结果 DB 文件中获取指定测试结果数据

参数

- **DB** (*str*) -- 测试结果 DB 文件的绝对路径, (例如: "C:/TestSuite/Benchmark/2021_07_29_21_10_36/Asymmetric_throughput_summary2021-07-29_21-11-08/Asymmetric_throughput_summary_2021-07-29_21-11-08.db")
- **Type** (*str*) -- 测试套件类型, (取值范围: RFC2544 / Asymmetric / RFC2889 / RFC3918)
- **Item** (*str*) -- 测试套件中的测试项目, (取值范围: Throughput / Latency / FrameLoss)
- **FrameSize** (*list*) -- 测试套件测试帧长, (取值范围: [64, 128, 256, 512, 1024. 1280, 1518])

返回 测试套件直接结果 DB 文件中获取指定测试结果数据列表

返回类型 list

实际案例

robotframework:

```
| ${DB}  == "C:/TestSuite/Benchmark/2021_07_29_21_10_36/Asymmetric_throughput_
↪summary2021-07-29_21-11-08/Asymmetric_throughput_summary_2021-07-29_21-11-
↪08.db") |
| ${Type} == "Asymmetric" |
| ${Item} == "Throughput" |
| ${FrameSize} == [64, 128, 256, 512, 1024. 1280, 1518] |
| ${result} | Get Benchmark Result | ${DB} | ${Type} | ${Item} | ${FrameSize}␣
↪|
```

TesterLibrary.Wizard.benchmark.**relate_benchmark_ports**(*Config*, *Ports*)

指定测试仪表测试套件使用的端口

参数

- **Config** (wizard_config) -- 仪表测试测试套件对象 object
- **Ports** (list (Port)) -- 测试仪表端口对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Items} | Create List | throughput | frameless |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items} |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} | Mode=meshed |␣
↪Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=${L2_TestTime} |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{L2_
↪FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

TesterLibrary.Wizard.benchmark.**run_benchmark**(*Mode=0*, *Timer=1800*, *Analyzer=False*)

执行测试仪表配置中的测试套件

参数

- **Mode** (*int*) -- 智能脚本的执行模式, (取值, 0 / 1) 0: 连续执行, 1: 单步执行
- **Timer** (*int*) -- 测试套件执行超时时间, (单位, 秒), 默认值: 1800 秒
- **Analyzer** (*bool*) -- 结果分析器开关, 布尔值 Bool (范围：True / False), 默认值: False

返回 测试结果 DB 文件的绝对路径

返回类型 str

实际案例

robotframework:

```
| ${result} | Run Benchmark |
| ${result} | Run Benchmark | Mode==1 |
| ${result} | Run Benchmark | Timer==3600 |
| ${result} | Run Benchmark | Analyzer==True |
| ${result} | Run Benchmark | Mode==1 | Timer==3600 | Analyzer==True |
```

TesterLibrary.Wizard.mpls module

TesterLibrary.Wizard.mpls.**create_mpls_wizard**(*Type*)

测试仪表创建 MPLS 向导

参数 **Type** (*str*) -- mpls 向导类型, 支持: mpls_ip_vpn mpls_6vpe ldp_vpls

返回 bool：布尔值 (范围：True / False)

实际案例

```
| Create Mpls Wizard | Type=mpls_ip_vpn |
```

TesterLibrary.Wizard.mpls.**edit_lsp_ping**(*Wizard*, ***kwargs*)

配置 MPLS 流量 LSP Ping 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **EnableLspPing** (*bool*) -- False,
- **DestinationIpv4Address** (*str*) -- ' 127.0.0.1',
- **PingInterval** (*int*) -- 4,
- **PingTimeout** (*int*) -- 2,
- **TimeToLive** (*int*) -- 255,
- **LspExpValue** (*int*) -- 0,
- **ValidateFecStack** (*bool*) -- False,
- **PadMode** (*str*) -- 支持: TransmitWithoutPadTlv RequestPeerToDropPadTlv RequestPeerToCopyPadTlv
- **PadData** (*list*) -- [],

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_customer_port**(*Wizard*, *Port*, ***kwargs*)

配置 MPLS 向导客户侧端口

参数

- **Wizard** (WizardConfig) -- wizard config
- **Port** (list(*:obj:'Port*))) -- 测试仪表端口对象列表

关键字参数

- **PortIndex** (*int*) -- 默认值: 0,
- **EnableSubInterface** (*bool*) -- 默认值: False,
- **SubInterfaceCount** (*int*) -- 默认值: 1,
- **DutIpv4Address** (*str*) -- 默认值: '192.85.1.1',
- **DutIpv4AddressStep** (*str*) -- 默认值: '0.0.1.0',
- **Ipv4PrefixLength** (*int*) -- 默认值: 24,
- **DutIpv6Address** (*str*) -- 默认值: '::',

- **DutIpv6AddressStep** (*str*) -- 默认值: '0:0:0:1::',
- **Ipv6PrefixLength** (*int*) -- 默认值: 64,
- **VlanId** (*int*) -- 默认值: 1,
- **VlanIdStep** (*int*) -- 默认值: 1

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_fec128**(*Wizard*, ***kwargs*)

配置 MPLS fec128 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **StartVcId** (*int*) -- Start vc id
- **StepVcId** (*int*) -- Step vc id

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_fec129**(*Wizard*, ***kwargs*)

配置 MPLS fec129 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **Agi** (*str*) -- AGI
- **AgiIncrement** (*str*) -- AGI increment
- **Saii** (*str*) -- SAII
- **SaiiIncrement** (*str*) -- SAII increment
- **Taii** (*str*) -- TAII
- **TaiiIncrement** (*str*) -- TAII increment
- **EnableBgpAutoDiscovery** (*bool*) -- Enable BGP auto discovery
- **DutAsNumber** (*int*) -- DUT AS number
- **RdAssignment** (*str*) -- RD assignment: UseRT Manual
- **AgiAssignment** (*str*) -- AGI assignment UseRT Manual
- **Rt** (*str*) -- RT
- **RtIncrement** (*str*) -- RT increment
- **Rd** (*str*) -- RD
- **RdIncrement** (*str*) -- RD increment

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_host**(*Wizard*, ***kwargs*)

配置 MPLS 向导 Host 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **HostMacStart** (*str*) -- Host MAC start
- **HostMacStep** (*str*) -- Host MAC step
- **EnableOverlapHosts** (*bool*) -- Enable overlap hosts
- **EnableHostVlan** (*bool*) -- Enable host VLAN
- **NumberOfCustomerSideVlanHeaders** (*int*) -- Number of customer side VLAN headers
- **NumberOfProviderSideVlanHeaders** (*int*) -- Number of provider side VLAN headers
- **VlanIdStart** (*int*) -- VLAN ID start
- **VlanIdStepPerVpls** (*int*) -- VLAN ID step per VPLS
- **VlanIdStepPerHost** (*int*) -- VLAN ID step per host
- **HostAssignment** (*int*) -- Host assignment: HostsOrMacsPerCe HostsOrMacsPerVpls TotalHostsOrMacs
- **HostsPerCustomerCe** (*int*) -- Host per customer CE
- **HostsPerProviderCe** (*int*) -- Host per provider CE
- **HostsPerVpls** (*int*) -- Host per VPLS
- **CustomerHostPercent** (*int*) -- Customer host percent
- **ProviderHostPercent** (*int*) -- Providier host percent

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_provider_port**(*Wizard*, *Port*, ***kwargs*)

配置 MPLS 向导提供商侧端口

参数

- **Wizard** (WizardConfig) -- wizard config
- **Port** (list(*:obj:'Port*))) -- 测试仪表端口对象列表

关键字参数

- **PortIndex** (*int*) -- 默认值: 0,
- **EnableSubInterface** (*bool*) -- 默认值: False,
- **SubInterfaceCount** (*int*) -- 默认值: 1,
- **DutIpv4Address** (*str*) -- 默认值: '192.85.1.1',
- **DutIpv4AddressStep** (*str*) -- 默认值: '0.0.1.0',
- **Ipv4PrefixLength** (*int*) -- 默认值: 24,
- **VlanId** (*int*) -- 默认值: 1,
- **VlanIdStep** (*int*) -- 默认值: 1

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_provider_route_reflector**(*Wizard*, ***kwargs*)

TesterLibrary.Wizard.mpls.**edit_mpls_provider_router_basic_parameters**(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **DutRouterId='10.0.0.1',** --
- **DutAsNumber=1,** --
- **Enable4ByteDutAs=False,** --
- **FourByteDutAsNumber='1** -- 1',
- **IgpProtocol=EnumMplsIgpProtocols.OSPF,** --
- **MplsProtocol=EnumMplsMplsProtocols.LDP,** --
- **EnablePRouter=True,** --
- **PRoutersPerInterface=1,** --
- **TopologyType=EnumMplsTopologyType.Tree,** --
- **PRouterStartIp='1.0.0.1',** --
- **PRouterPrefixLength=24,** --
- **PRouterIdStart='192.0.1.1',** --
- **PRouterIdStep='0.0.1.0',** --
- **PeRoutersPerInterface=1,** --
- **PeRouterIdStart='10.0.0.2',** --
- **PeRouterIdStep='0.0.0.1',** --
- **EnableRouteReflectors=None** --
- **Enable6Vpe=None** --

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_provider_router_isis**(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由路由器 OSPF 协议

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **UseSrcMacAsSystemId** (*bool*) -- Use source MAC as system id
- **SystemId** (*str*) -- System ID
- **SystemIdStep** (*str*) -- System ID step
- **Level** (*str*) -- Level: L1 L2 L1L2
- **NetworkType** (*str*) -- Network type BROADCAST P2P
- **RouterPriority** (*int*) -- Router priority

- **MetricMode** (*str*) -- Metric mode NARROW WIDE NARROWWIDE
- **AuthenticationMode** (*str*) -- Authentication mode NONE SIMPLE MD5
- **Password** (*str*) -- Password
- **AreaId** (*list*) -- Area ID, hex int
- **EnableGracefulRestart** (*bool*) -- Enable graceful restart
- **MultiTopologyId** (*str*) -- Multi-topology ID NOSHOW IPV4 IPV6
- **EnableBfd** (*bool*) -- Enable BFD
- **HelloPadding** (*bool*) -- Enable hello padding
- **Algorithm** (*int*) -- SR algorithm
- **SidLabelBase** (*int*) -- SR SID/Label base
- **SidLabelRange** (*int*) -- SR SID/Label range
- **NodeSidIndex** (*int*) -- SR node SID index
- **NodeSidIdnexStep** (*int*) -- SR node SID index step

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_provider_router_ldp**(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由路由器 LDP 协议

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **HelloType** (*str*) -- Hello type: DIRECT TARGETED DIRECT_TARGETED
- **TransportAddressTlvMode** (*str*) -- Transport address TLV mode TESTER_IP ROUTER_ID NONE
- **LabelAdvertisementMode** (*str*) -- Label advertisement mode: DU DOD
- **EgressLabelMode** (*str*) -- Egress label mode: NEXT_AVAILABLE IMPLICIT EXPLICIT
- **MinLabel** (*int*) -- Min Label
- **AuthenticationMode** (*str*) -- Authentication mode: NONE MD5
- **Password** (*str*) -- Password

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_provider_router_ospf**(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由路由器 OSPF 协议

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **AreaId** (*str*) -- Area ID
- **NetworkType** (*str*) -- Network type: BROADCAST or P2P

- **RouterPriority** (*int*) -- Router priority
- **AuthenticationType** (*str*) -- Authentication type NONE SIMPLE MD5
- **Password** (*str*) -- Password
- **Md5Key** (*int*) -- MD5 key
- **Options** (*list*) -- Options: NONTBIT TOSBIT EBIT MCBIT NPBIT EABIT DCBIT OBIT DNBIT
- **EnableGracefulRestart** (*bool*) -- Enable graceful restart
- **GracefulRestartReason** (*str*) -- Gracefull restart reason: UNKNOWN SOFTWARE RELOADORUPGRADE SWITCH
- **EnableBfd** (*bool*) -- Enable BFD
- **Algorithm** (*int*) -- SR algorithm
- **SidLabelBase** (*int*) -- SR SID/Label base
- **SidLabelRange** (*int*) -- SR SID/Label range
- **NodeSidIndex** (*int*) -- SR node SID index
- **NodeSidIdnexStep** (*int*) -- SR node SID index step

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_provider_router_rip**(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由路由器 Rip 协议

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **RipVersion** (*str*) -- RIP version: RIPV1 RIPV2 RIPNG
- **UpdateType** (*str*) -- Update type: BROADCAST MULTICAST UNICAST
- **UpdateInterval** (*int*) -- Update interval(sec)
- **UpdateJitter** (*int*) -- Update jitter
- **MaxRouteNumPerUpdate** (*int*) -- Max route per update
- **AuthenticationMode** (*str*) -- Authentication mode: NONE SIMPLE MD5
- **Password** (*str*) -- Password
- **Md5KeyId** (*int*) -- MD5 key ID
- **SplitHorizon** (*bool*) -- Enable split horizon

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_pwe_basic_parameters**(*Wizard*, ***kwargs*)

配置 MPLS pwe 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **NumberOfPseudoWire** (*int*) -- Number of pseudowire
- **Mtu** (*int*) -- MTU
- **GroupId** (*int*) -- Group ID
- **EnableCBit** (*bool*) -- Enable C-Bit
- **IncludeStatusTlv** (*bool*) -- Include status TLV
- **StatusCode** (*list*) -- Status code: PseudowireNotForwarding LocalAttachmentCircuitReceiveFault LocalAttachmentCircuitTransmitFault LocalPsnFacingPwIngressReceiveFault LocalPsnFacingPwEgressTransmitFault
- **EnableOverrideEncapsulation** (*bool*) -- Enable override encapsulation
- **Encapsulation** (*str*) -- Encapsulation EthernetVlan Ethernet EthernetVpls
- **EnableOverlapVcidsOnDifferentPes** (*bool*) -- Enable overlap VC IDs on different PEs
- **EnableCreateProviderHostsForUnusedVpls** (*bool*) -- Enable Create provider hosts for unsued VPLSs
- **FecType** (*str*) -- FEC type: FEC128 FEC129

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_vpls_basic_parameters**(*Wizard*, ***kwargs*)

配置 MPLS VPLS 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **NumberOfVpls** (*int*) -- Number of VPLSs
- **Mtu** (*int*) -- MTU
- **GroupId** (*int*) -- Group ID
- **EnableCBit** (*bool*) -- Enable C-Bit
- **IncludeStatusTlv** (*bool*) -- Include status TLV
- **StatusCode** (*list*) -- Status code: PseudowireNotForwarding LocalAttachmentCircuitReceiveFault LocalAttachmentCircuitTransmitFault LocalPsnFacingPwIngressReceiveFault LocalPsnFacingPwEgressTransmitFault
- **EnableOverrideEncapsulation** (*bool*) -- Enable override encapsulation
- **Encapsulation** (*str*) -- Encapsulation EthernetVlan Ethernet EthernetVpls
- **VplsAssignment** (*str*) -- Vpls assginment: RoundRobin Sequential
- **CreateProviderHostsForUnusedVplss** (*bool*) -- Enable Create provider hosts for unsued VPLSs
- **FecType** (*str*) -- FEC type: FEC128 FEC129

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_vpn_as_number**(*Wizard*, ***kwargs*)

TesterLibrary.Wizard.mpls.**edit_mpls_vpn_customer_parameters**(*Wizard*, ***kwargs*)

配置 MPLS VPN 客户侧参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **VpnAssignment** (*int*) -- number of vpns
- **CeProtocol** (*str*) -- CE protocol type, params: BGP RIP ISIS OSPF Static Mixed
- **CeProAssignment** (*str*) -- CE protocol assignment (e.g. 'BGP=100%')
- **CustomerRdStart** (*str*) -- customer RD start (e.g. '1:0')
- **CustomerRdStepPerVpnEnabled** (*bool*) -- customer Rd step PerVpn enabled
- **CustomerRdStepPerVpn** (*str*) -- customer Rd step PerVpn (e.g. '1:0')
- **CustomerRdStepPerCeEnabled** (*bool*) -- customer Rd step PerCe enabled
- **CustomerRdStepPerCe** (*str*) -- customer Rd step PerCe (e.g. '0:0')

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_vpn_ipv4_route_customer_parameters**(*Wizard*, ***kwargs*)

配置 MPLS VPN 路由客户端侧参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **CustomerStartRoute** (*str*) -- customer start route Ipv4Address
- **CustomerRouteStep** (*str*) -- customer route step Ipv4Address
- **CustomerPrefixLength** (*int*) -- customer prefix length
- **CustomerRoutesPerCe** -- (int) customer route PerCe
- **CustomerOverlapRoutes** (*bool*) -- customer overlap routes
- **CustomerRouteType** (*str*) -- customer route type, Internal or External

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_vpn_ipv4_route_provider_parameters**(*Wizard*, ***kwargs*)

配置 MPLS VPN 路由提供商端侧参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **ProviderStartRoute** (*str*) -- provider start route, Ipv4Address
- **ProviderRouteStep** (*str*) -- provider route step, Ipv4Address
- **ProviderPrefixLength** (*int*) -- prefix length
- **ProviderRoutesPerCe** (*int*) -- provider route PerCe
- **ProviderOverlapRoutes** (*bool*) -- provider overlap routes
- **ProviderLabelType** (*str*) -- provider route type, LabelPerSite or LabelPerRoute
- **ProviderStartLabel** (*int*) -- start label

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_vpn_ipv6_route_customer_parameters**(*Wizard*, ***kwargs*)

配置 MPLS VPN 路由客户端侧 IPv6 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **CustomerStartRoute** (*str*) -- customer start route, Ipv6Address
- **CustomerRouteStep** (*str*) -- customer route step, Ipv6Address
- **CustomerPrefixLength** (*int*) -- prefix length
- **CustomerRoutesPerCe** (*int*) -- customer route PerCe
- **CustomerOverlapRoutes** (*bool*) -- customer overlap routes
- **CustomerRouteType** (*str*) -- customer route type, Internal or External

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_vpn_ipv6_route_provider_parameters**(*Wizard*, ***kwargs*)

配置 MPLS VPN 路由提供商端侧 IPv6 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **ProviderStartRoute** (*str*) -- provider start route Ipv6Address
- **ProviderRouteStep** (*str*) -- provider route step Ipv6Address
- **ProviderPrefixLength** (*int*) -- prefix length
- **ProviderRoutesPerCe** (*int*) -- provider route PerCe
- **ProviderOverlapRoutes** (*bool*) -- provider overlap routes
- **ProviderLabelType** (*str*) -- provider route type, LabelPerSite or LabelPerRoute
- **ProviderStartLabel** (*int*) -- start label

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_vpn_parameters**(*Wizard*, ***kwargs*)

配置 MPLS VPN 基本参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **NumberOfVpns** (*int*) -- number of vpns
- **RdAssignment** (*str*) -- Route Target Assignment, support: UseRT、Manual
- **RouteTargetStart** (*str*) -- route target start (e.g. '1:0')
- **RouteTargetStep** (*str*) -- route target step (e.g. '1:0')

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_mpls_vpn_provider_parameters**(*Wizard*, ***kwargs*)

配置 MPLS VPN 提供商侧参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **ProviderDisSel** (*str*) -- provider distribution selector, support: VPNsPerPE、PEsPerVPN
- **ProviderDisSelCount** (*int*) -- provider distribution selector count
- **ProviderMeshed** (*bool*) -- provider meshed
- **ProviderRdStart** (*str*) -- provider RD start (e.g. '1:0')
- **ProviderRdStepPerVpnEnabled** (*bool*) -- provider Rd step PerVpn enabled
- **ProviderRdStepPerVpn** (*str*) -- route target start (e.g. '1:0')
- **ProviderRdStepPerCeEnabled** (*bool*) -- provider Rd step PerCe enabled
- **ProviderRdStepPerCe** (*str*) -- provider Rd step PerCe (e.g. '0:0')

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**edit_traffic_parameters**(*Wizard*, ***kwargs*)

配置 MPLS 流量

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **TrafficFlow** (*str*) -- traffic flow type, support: None FullyMeshedInVpn FullyMeshedInVpls Customer2Provider Provider2Customer CustomerProviderBoth
- **StreamBlockGrouping** (*str*) -- streamblock grouping type, support: Aggregate VPNAggregate NotAggregate
- **UseSingleStreamNumber** (*bool*) -- use single stream number

- **TrafficLoadPercentProvider** (*int*) -- traffic load percent provider
- **TrafficLoadPercentCustomer** (*bool*) -- traffic load percent customer

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

TesterLibrary.Wizard.mpls.**expand_mpls_wizard**(*Wizard*)

生成测试仪表 MPLS 向导配置

参数 **Wizard** (WizardConfig) -- wizard config

返回 bool：布尔值 (范围：True / False)

实际案例

```
| Expand Mpls Wizard | Wizard=${Wizard} |
```

Module contents

1.2 Submodules

1.3 TesterLibrary.base module

class TesterLibrary.base.**Meta**(*name*, *bases*, *attrs*)

基类：type

class TesterLibrary.base.**TesterLibrary**(*logLevel=20*, *logHandle=LogHandle.LOG_FILE*)

基类：object

property API

ROBOT_AUTO_KEYWORDS = False

ROBOT_LIBRARY_SCOPE = 'GLOBAL'

static abort_dot1x(*Sessions*)

中断 802.1x 会话

参数 **Sessions** (Dot1x) -- 802.1x 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Abort Dot1x | Sessions=${Sessions} |
```

static abort_l2tp(*Sessions*)

中断 L2tp 协议会话

参数 **Sessions** (L2tp) -- L2tp 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Abort L2tp | Sessions=${Sessions} |
```

static abort_pppoe(*Sessions*)

中断 PPPoE 协议会话

参数 **Sessions** (list (PppoeClent)) -- PPPoE 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Abort Pppoe | Sessions=${Sessions} |
```

static abort_request_ldp_label(*Configs*)

中止 LDP 协议会话 LSP 请求标签

参数 **Configs** (*list*) -- LDP LSP 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Abort Request Ldp Label | Configs=${Configs} |
```

static add_imix_distribution_frame(*IMix*, *Type='fixed'*, *Length=None*, *Min=None*, *Max=None*, *Weight=None*)

在 Imix 模板添加自定义帧长

参数

- **IMix** (Imix) -- 测试仪表 Imix 模板对象
- **Type** (*str*) -- 测试仪表 Imix 模板自定义帧长类型 fixed random
- **Length** (*int*) -- 测试仪表 Imix 模板自定义帧长长度，random 时有效
- **Min** (*int*) -- 测试仪表 Imix 模板自定义帧长最小帧长，random 时有效
- **Max** (*int*) -- 测试仪表 Imix 模板自定义帧长最大帧长，random 时有效
- **Weight** (*int*) -- 测试仪表 Imix 模板自定义帧长权重

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Imix} | Create Imix | Name=Imix_1 | Seed=10121112 |
| Add IMix Distribution Frame | IMix=${Imix} | Type=random | Min=64 |␣
↪Max=128 | Weight=50 |
| Add IMix Distribution Frame | IMix=${Imix} | Type=random | Min=128 |␣
↪Max=256 | Weight=50 |
| Bind Stream Imix | Stream=${Stream} | Imix=${Imix} |
```

static add_stream(*Type='raw', Ports=None, Names=None, FilePath=None, IncludeCrc=True, SrcPoints=None, DstPoints=None, SrcInterface=None, DstInterface=None, Bidirection=None, Direction=None, Layer=None, TrafficMeshMode=None, EndpointMapping=None, AutoCreateTunnel=False, StreamOnly=None, **kwargs*)

测试仪表创建流量

参数

- **Type** (*str*) -- 创建绑定流类型，支持 raw 流、binding 流以及 pcap 文件导入流:

  raw

  binding

  pcap

- **Ports** (list (*Port*)) -- raw 流和 pcap 流参数，端口对象, 测试仪表端口对象 object 列表
- **Names** (*list*) -- raw 流参数，流量名称, 流量名称列表
- **FilePath** (*str*) -- pcap 流参数，当 Type 为 pcap 需要指定导入 pcap 文件的路径
- **IncludeCrc** (*bool*) -- pcap 流参数，当 Type 为 pcap 指定导入方式是否携带 CRC, 布尔值 Bool (范围：True / False)
- **SrcInterface** (*list*) -- binding 流参数，指定源接口
- **DstInterface** (*list*) -- binding 流参数，指定目的接口
- **SrcPoints** (*list*) -- binding 流参数，指定源端点
- **DstPoints** (*list*) -- binding 流参数，指定目的端点
- **Bidirection** (*bool*) -- 是否是能双向流量, 布尔值 Bool (范围：True / False)
- **Direction** (*str*) -- binding 流参数
- **Layer** (*str*) -- binding 流参数，指定接口网络层，默认值：IPV4，支持值:

  ETHERNETII

  VLAN

  GRE

  IPV4

  IPV6

- **TrafficMeshMode** (*str*) -- binding 流参数, 默认值：MANY_TO_MANY, 支持值：

  ONE_TO_ONE

  MANY_TO_MANY

  FULL_MESH

  CONGESTION

  LEARNING

  BACK_BONE

  PAIR

- **EndpointMapping** (*str*) -- binding 流参数, 默认值：ROUND_ROBIN, 支持值：

  ROUND_ROBIN

  MANY_TO_MANY

- **AutoCreateTunnel** (*bool*) -- binding 流参数, 自动绑定隧道, 默认值：False
- **StreamOnly** (*bool*) -- True 为每个 flow 创建一个 stream, False 为多个 flow 创建一个 stream, 默认值：False

返回 创建的流量对象 object 列表

返回类型 list (StreamTemplate)

实际案例

```
# raw流
| ${Streams} | add_stream | Port=${Port} |
# pcap流
| ${Streams} | add_stream | Type=pcap | Port=${Port} | FilePath=${Pcap_
↪File_Path} | IncludeCrc=True |
# binding流
| ${Streams} | add_stream | Type=binding | SrcPoints=${Points_1} |␣
↪DstPoints=${Points_2} | Bidirection=True |
```

static advertise_bgp(*Sessions*)

通告 BGP 协议会话 lsa

参数 **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Bgp | Sessions=${Sessions} |
```

static advertise_bgp_route(*Routes*)

通告 BGP 协议指定 lsa

参数 **Routes** (*list*) -- BGP 协议路由对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Bgp Route | Routes=${Routes} |
```

static advertise_isis(*Lsps*)

通告 Isis 协议会话 lsp

参数 **Lsps** (IsisLspConfig) -- ISIS LSP 对象, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Isis | Lsp=${Lsp} |
```

static advertise_ldp_label(*Configs*)

通告 LDP 协议会话 LSP 标签

参数 **Configs** (*list*) -- LDP LSP 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Ldp Label | Configs=${Configs} |
```

static advertise_ospf_lsa(*Sessions=None*, *Type=None*, *Lsa=None*)

通告 OSPFv2 协议会话 lsa

参数

- **Sessions** (list(OspfRouter)) -- OSPFv2 协议会话对象列表, 类型为：list
- **Type** (*str*) -- OSPFv2 lsa 类型, 类型为：string, 支持的 lsa 类型：

  Router

  Network

  Summary

  AsbrSummary

  External
- **Lsa** (*list*) -- OSPFv2 lsa 列表, 类型为：list, 当 Type=None 时参数生效

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Ospf Lsa | Sessions=${Sessions} | Type=router |
| Advertise Ospf Lsa | Sessions=${Sessions} | Lsa=${Lsas} |
```

static advertise_ospfv3_lsa(*Sessions=None*, *Type=None*, *Lsa=None*)

通告 OSPFv3 协议会话 lsa

参数

- **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list
- **Type** (*str*) -- OSPFv3 lsa 类型, 类型为：string, 支持的 lsa 类型：

  Router

  Network

  InterAreaPrefix

  InterAreaRouter

  AsExternal

  Link
- **Lsa** (*int*) -- OSPFv3 lsa 列表, 类型为：list, 当 Type=None 时参数生效

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Ospfv3 Lsa | Sessions=${Sessions} | Type=router |
| Advertise Ospfv3 Lsa | Sessions=${Sessions} | Lsa=${Lsas} |
```

static advertise_rip(*Sessions*)

通告 RIP 协议路由

参数 **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Advertise Rip | Sessions=${Sessions} |
```

static apply_igmp_querier(*Sessions*)

IGMP Querier 增量配置下发到后台

参数 **Sessions** (list (IgmpQuerier)) -- IGMP Querier 协会话对象, 类型为：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Apply Igmp Querier |
```

static apply_mld_querier(*Sessions*)

MLD Querier 增量配置下发到后台

参数 **Sessions** (list (MldQuerier)) -- MLD Querier 协会话对象, 类型为：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Apply Mld Querier |
```

static benchmark_stream_use_exist(*Config*, *Streams*)

编辑测试套件使用已存在流量

参数

- **Config** (wizard) -- 仪表测试测试套件对象 object
- **Streams** (list (StreamTemplate)) -- 仪表测试流模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| @{Items} | Create List | throughput | frameloss |
| @{FrameSize} | Create List | 256 | 1024 | 16383 |
| ${Streams} | Add Stream | Type=binding | | SrcPoints=@{SrcPoints} |␣
↪DstPoints=@{SrcPoints} |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items}
↪ |
| Edit Benchmark Path | Configs=${Config} | Path=C:/test |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Benchmark Stream Use Exist | Config=${Wizard} | Streams=${Streams} |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=1000 |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@
↪{FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

static bfd_admin_down(*Sessions*)

设置 BFD 会话状态 AdminDown

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Admin Down | Sessions=${Sessions} |
```

static bfd_admin_up(*Sessions*)

设置 BFD 会话状态 AdminUp

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Admin Up | Sessions=${Sessions} |
```

static bfd_disable_demand_mode(*Sessions*)

关闭 BFD demand 模式

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Disable Demand Mode | Sessions=${Sessions} |
```

static bfd_enable_demand_mode(*Sessions*)

开启 BFD Demand 模式

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Enable Demand Mode | Sessions=${Sessions} |
```

static bfd_initiate_poll(*Sessions*)

发送 BFD poll Sequence

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Initiate Poll | Sessions=${Sessions} |
```

static bfd_resume_pdus(*Sessions*)

恢复发送 BFD PDU

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Resume Pdus | Sessions=${Sessions} |
```

static bfd_set_diagnostic_state(*Sessions*, *State*)

设置 BFD 状态诊断码

参数

- **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list
- **State** (*str*) -- 设置状态诊断码, 类型为：string, 默认值：NO_DIAGNOSTIC, 支持的状态诊断码：

  NO_DIAGNOSTIC

  CONTROL_DETECTION_TIME_EXPIRED

  ECHO_FUNCTION_FAILED

  NEIGHBOR_SIGNAL_SESSION_DOWN

  FORWARDING_PLANE_RESET

  PATH_DOWN

  CONCATENATED_PATH_DOWN

  ADMINISTRATIVELY_DOWN

  REVERSE_CONCATENATED_PATH_DOWN

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Set Diagnostic State | Sessions=${Sessions} |
```

static bfd_stop_pdus(*Sessions*)

停止发送 BFD PDU

参数 **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Bfd Stop Pdus | Sessions=${Sessions} |
```

static bind_stream_imix(*Stream*, *IMix*)

将 Imix 模板和流量模板绑定

参数

- **Stream** (StreamTemplate) -- 测试仪表流量模板 StreamTemplate 对象
- **IMix** (Imix) -- 测试仪表 Imix 模板对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Imix} | Create Imix | Name=Imix_1 | Seed=10121112 |
| Bind Stream Imix | Stream=${Stream} | Imix=${Imix} |
```

static binding_multicast_group(*Session*, *Memberships*, *MulticastGroup*)

将全局组播组绑定到组播协议会话上

参数

- **Session** (Mld, *Igmp*) -- IGMP/MLD 协会话对象, 类型为：object
- **Memberships** (MldMembershipsConfig) -- 组播协议和组播组绑定关系对象, 类型为：object
- **MulticastGroup** (MldSelectMulticastGroupCommand) -- 全局组播组对象, 类型为：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Group} | Create Multicast Group | Version=IPV4 | Start=225.0.1.1 |␣
↪Number=20 |
| ${Session} | Create Igmp | Port=${Port} | Version=IGMPV3 |
| ${Memberships} | Create Memberships | Session=${Session} | Start=225.0.
↪1.1 | DeviceGroupMapping=ONETOONE |
| binding_multicast_group | Session=${Session} | Memberships=$
↪{Memberships} | MulticastGroup=${Group} |
```

static binding_vxlan_multicast_group(*Segments*, *MulticastGroups*)

创建 Vxlan Multicast Group 对象

参数

- **Segments** (VxlanSegmentConfig) -- Vxlan Segment 对象, 类型：object
- **MulticastGroups** (Ipv4MulticastGroup) -- Vxlan Multicast Group 对象, 类型：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Binding Vxlan Multicast Group | Segments=${Segments} | MulticastGroups=$
↪{MulticastGroups} |
```

static binding_vxlan_vm(*Segments*, *Vms*)

绑定 Vxlan Vm 对象

参数

- **Segments** (VxlanSegmentConfig) -- Vxlan Segment 对象, 类型: object
- **Vms** ('VxlanVmProperty') -- Vxlan Vm 对象, 类型: object

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Binding Vxlan Vm | Segments=${Segments} | Vms=${Vms} |
```

static binding_vxlan_vtep(*Vteps*, *Vms*)

绑定 Vxlan Vtep 对象

参数

- **Vteps** (Vxlan) -- Vxlan 协议会话对象, 类型: object
- **Vms** ('VxlanVmProperty') -- Vxlan Vm 对象, 类型: object

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Binding Vxlan Vtep | Vteps=${Vxlan} | Vms=${Vms} |
```

static clear_result(*All=True*, *Objects=None*)

清除测试仪表统计

参数

- **All** (*bool*) -- 是否清除所有已经订阅的统计视图的数据, 默认位: True
- **Objects** (*list*) -- 指定需要清空视图的对象

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

robotframework:

```
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| Clear Result |
```

static connect_bgp(*Sessions*)

连接 BGP 协议会话

参数 **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Connect Bgp | Sessions=${Sessions} |
```

static connect_chassis(*Chassis*)

连接测试仪表机箱后台.

参数 **Chassis** (*str*) -- 机箱主机 IP 地址列表

返回 Chassis 对象列表

返回类型 list

实际案例

robotframework:

```
| ${Hosts}= | Create List | 192.168.0.10 | 192.168.0.10 |
| ${Chassis} | Connect Chassis | Chassis=${Hosts} |
```

static connect_l2tp(*Sessions*)

建立 L2tp 协议会话

参数 **Sessions** (L2tp) -- L2tp 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Connect L2tp | Sessions=${Sessions} |
```

static connect_pppoe(*Sessions*)

连接 PPPoE 协议会话

参数 **Sessions** (list (PppoeClent)) -- PPPoE 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Connect Pppoe | Sessions=${Sessions} |
```

static create_benchmark(*Type='rfc2544', Items=None*)

创建测试仪表测试套件

参数

- **Type** (*str*) -- 测试仪表测试套件类型, 支持的类型:

  rfc2544

  rfc2889

  rfc3918

  Asymmetric

- **Items** (*list*) -- 测试仪表测试套件中的测试项, 测试套件中的测试项列表, 支持测试项目如下:

  throughput: rfc2544 吞吐量测试

  backtoback: rfc2544 背靠背测试

  frameloss: rfc2544 丢包率测试

  latency: rfc2544 时延测试

  addressCachingCapacity: rfc2889 地址缓存容量测试

  addressLearningRate: rfc2889 地址学习速率测试

  broadcastLatency: rfc2889 广播帧转发测试

  broadcastForwarding: rfc2889 广播帧时延测试

  congestion: rfc2889 拥塞控制测试

  erroredFrameFilter: rfc2889 错误帧过滤测试

  forwarding: rfc2889 转发测试

  forwardPressure:

  maxForwarding:

  mixedThroughput: rfc3918 混合吞吐量测试

  scaledGroupForwarding: rfc3918 组转发矩阵测试

  multicastThroughput: rfc3918 聚合组播吞吐量测试

  multicastGroupCapacity: rfc3918 组播组容量测试

  multicastLatency: rfc3918 组播转发时延测试

  multicastJoinLeaveLatency: rfc3918 加入离开时延测试

返回

仪表测试测试套件对象 object

(list (`test_config`)): 仪表测试测试套件测试项对象 object 列表

返回类型 (`wizard_config`)

实际案例

robotframework:

```
| ${Items} | Create List | throughput | frameloss |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items}
↪ |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} |␣
↪Mode=meshed | Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=${L2_TestTime} |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{L2_
↪FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

static create_benchmark_streams(*Config*, *Items*, *Type*, *SrcPoints*, *DstPoints*, *Bidirectional=False*, *Mode='1v1'*, *Mapping='roundrobin'*, *Monitors=()*)

创建测试仪表测试套件流量

参数

- **Config** (wizard_config) -- 仪表测试测试套件对象 object
- **Items** (*list*) -- 测试仪表测试套件中的测试项, 测试套件中的测试项列表, 支持测试项目如下:

  throughput

  backtoback

  frameloss

  latency

  addressCachingCapacity

  addressCachingRate

  broadcastLatency

  broadcastForwarding

  congestion

  erroredFrameFilter

  forwarding

  forwardPressure

  maxForwarding

  mixedThroughput

  scaledGroupForwarding

  multicastThroughput

  multicastGroupCapacity

  multicastLatency

  multicastJoinLeaveLatency

- **Type** (*str*) -- 创建绑定流类型, 支持类型有:

  eth

  ipv4

  ipv6
- **SrcPoints** (*list*) -- 创建绑定流类型源端点对象 object 列表
- **DstPoints** (*list*) -- 创建绑定流类型目的端点, 目的端点对象 object 列表
- **Bidirectional** (*bool*) -- 是否使能双向流量, 默认值: False
- **Mode** (*str*) -- 创建绑定流 topo 类型

  1v1

  m2m

  meshed
- **Mapping** (*str*) -- 创建绑定流端点模式, 支持类型

  roundrobin

  manytomany
- **Monitors** (*list*) -- 作为镜像端口的测试仪表端口对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Items} | Create List | throughput | frameless |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items}
↪ |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} |␣
↪Mode=meshed | Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=${L2_TestTime} |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{L2_
↪FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

static create_bfd(*Port*, ***kwargs*)

创建 BFD 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- BFD 协议会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 BFD 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterRole** (*str*) -- BFD 会话的角色, 类型为：string, 默认值：Active, 支持角色：

  Active

  Passive

- **TimeIntervalUnit** (*str*) -- 时间间隔的单位。类型为：string, 默认值：milliseconds, 支持单位：

  milliseconds

  microseconds

- **DesiredMinTXInterval** (*int*) -- 期望的最小发送时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：50
- **RequiredMinRXInterval** (*int*) -- 需要的最小接收时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：50
- **RequiredMinEchoRXInterval** (*int*) -- 需要的最小 Echo 报文接收时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：0
- **DetectMultiple** (*int*) -- 用于检测超时的时间因子, 类型为：number, 取值范围：2-100, 默认值：3
- **AuthenticationType** (*str*) -- 认证方式, 类型为：string, 默认值：None, 支持的方式：

  NONE

  SIMPLE_PASSWORD

  KEYED_MD5

  METICULOUS_KEYED_MD5

  KEYED_SHA1

  METICULOUS_KEYED_SHA1

- **Password** (*str*) -- 当认证方式不为 NONE 时，在该单元格输入认证密码。密码可以是数字、字母或者数字和字母的组合，最长为 16 位。类型为：string, 默认值：Xinertel
- **KeyID** (*int*) -- 当认证方式不为 NONE 时，在该单元格输入 Key ID, 类型为：number, 取值范围：0-255, 默认值：1

返回 BFD 协议会话对象, 类型：object

返回类型 (BfdRouter)

实际案例

```
| Create bfd | Port=${Port} | TimeIntervalUnit=microseconds |
```

static create_bfd_ipv4_session(*Session*, ***kwargs*)

创建 BFD IPv4 会话对象

参数 **Session** (BfdRouter) -- BFD 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BFD IPv4 会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 BFD IPv4 会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **NumberOfSessions** (*str*) -- BFD IPv4 会话的数目, 类型为：string, 取值范围：1-4294967295, 默认值：1

- **StartDestinationAddress** (*str*) -- 指定第一个目的 IPv4 地址, 类型为: string, 取值范围: 有效的 ipv4 地址, 默认值: 192.0.1.0
- **DestinationAddressIncrement** (*str*) -- 指定下一个目的 IPv4 地址的增量, 类型为: string, 取值范围: 有效的 ipv4 地址, 默认值: 0.0.0.1
- **EnableMyDiscriminator** (*bool*) -- 是否指定本地标识符, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **MyDiscriminator** (*int*) -- 指定本地标识符的初始值。只有使能本地标识符被选中才可编辑, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **MyDiscriminatorIncrement** (*int*) -- 指定下一个本地标识符的增量。只有使能本地标识符被选中才可编辑。类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **EnableYourDiscriminator** (*bool*) -- 是否指定对端标识符, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **YourDiscriminator** (*int*) -- 指定对端标识符的初始值。只有使能本地标识符被选中才可编辑, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **YourDiscriminatorIncrement** (*int*) -- 指定下一个对端标识符的增量。只有使能本地标识符被选中才可编辑。类型为: number, 取值范围: 1-4294967295, 默认值: 1

返回 BFD IPv4 会话对象, 类型: object

返回类型 (BfdIpv4SessionConfig)

实际案例

```
| ${Session} | Create Bfd | Port=${Port} |
| Create Bfd Ipv4 Session | Session=${Session} | NumberOfSessions=10 |
```

static create_bfd_ipv6_session(*Session*, ***kwargs*)

创建 BFD IPv6 路由对象

参数 **Session** (BfdRouter) -- BFD 协议会话对象列表, 类型为: object

关键字参数

- **Name** (*str*) -- BFD IPv6 路由名称, 类型为: string
- **Enable** (*bool*) -- 使能 BFD IPv6 路由, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **NumberOfSessions** (*str*) -- BFD IPv6 会话的数目, 类型为: string, 取值范围: 1-4294967295, 默认值: 1
- **StartDestinationAddress** (*str*) -- 指定第一个目的 IPv6 地址, 类型为: string, 取值范围: 有效的 ipv6 地址, 默认值: 2000::1
- **DestinationAddressIncrement** (*str*) -- 指定下一个目的 IPv4 地址的增量, 类型为: string, 取值范围: 有效的 ipv4 地址, 默认值: ::1
- **EnableMyDiscriminator** (*bool*) -- 是否指定本地标识符, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **MyDiscriminator** (*int*) -- 指定本地标识符的初始值。只有使能本地标识符被选中才可编辑, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **MyDiscriminatorIncrement** (*int*) -- 指定下一个本地标识符的增量。只有使能本地标识符被选中才可编辑。类型为: number, 取值范围: 1-4294967295, 默认值: 1

- **EnableYourDiscriminator** (*bool*) -- 是否指定对端标识符, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **YourDiscriminator** (*int*) -- 指定对端标识符的初始值。只有使能本地标识符被选中才可编辑, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **YourDiscriminatorIncrement** (*int*) -- 指定下一个对端标识符的增量。只有使能本地标识符被选中才可编辑。类型为: number, 取值范围: 1-4294967295, 默认值: 1

返回 BFD IPv6 会话对象, 类型: object

返回类型 (BfdIpv6SessionConfig)

实际案例

```
| ${Session} | Create Bfd | Port=${Port} |
| Create Bfd Ipv6 Session | Session=${Session} |
```

static create_bgp(*Port*, ***kwargs*)

创建 BGP 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为: object

关键字参数

- **Name** (*str*) -- BGP 协会话名称, 类型为: string
- **Enable** (*bool*) -- 使能 BGP 协议会话, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IpVersion** (*str*) -- IP 版本, 类型为: string, 默认值: BOTH, 支持版本:

  BOTH

  IPV4

  IPV6
- **BgpInitiator** (*bool*) -- BGP 会话发起者, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **AsNumber** (*int*) -- 自治域, 类型为: number, 取值范围: 1-65535, 默认值: 1
- **AsNumberStep** (*int*) -- 自治域跳变, 类型为: number, 取值范围: 0-65535, 默认值: 1
- **Enable4ByteAs** (*bool*) -- 使能 4 字节自治域, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **AsNumber4Byte** (*int*) -- 4 字节自治域, 类型为: number, 取值范围: 0.1-65535.65535, 默认值: 1.1
- **AsNumber4ByteStep** (*int*) -- 4 字节自治域跳变, 类型为: number, 取值范围: 0.1-65535.65535, 默认值: 0.1
- **DutAsNumber** (*int*) -- DUT 自治域, 类型为: number, 取值范围: 1-65535, 默认值: 1
- **DutAsNumberStep** (*int*) -- DUT 自治域跳变, 类型为: number, 取值范围: 1-65535, 默认值: 1
- **Enable4ByteDutAs** (*bool*) -- 使能 DUT4 字节自治域, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **Dut4ByteAsNumber** (*int*) -- DUT4 字节自治域, 类型为: number, 取值范围: 0.1-65535.65535, 默认值: 1.1

- **Dut4ByteAsNumberStep** (*int*) -- DUT4 字节自治域跳变, 类型为：number, 取值范围：0.1-65535.65535, 默认值：0.1
- **BgpType** (*str*) -- BGP 类型, 类型为：string, 取值范围：EBGP, IBGP, 默认值：IBGP
- **UseGatewayAsDutIp** (*bool*) -- 使用网关地址作为 DUT 地址, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **BgpSessionIpAddressType** (*str*) -- 会话 IP 类型, 类型为：string, 取值范围：INTERFACE_IP, ROUTE_ID, 默认值：INTERFACE_IP
- **DutIpv4Address** (*str*) -- DUT IPv4 地址, 当 IP 版本为 IPv4，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID, 类型为：string, 取值范围：IPv4 地址, 默认值：2.1.1.1
- **DutIpv4AddressStep** (*str*) -- DUT IPv4 地址跳变, 当 IP 版本为 IPv4，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID 增量步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **DutIpv6Address** (*str*) -- DUT IPv4 地址, 当 IP 版本为 IPv6，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **DutIpv6AddressStep** (*str*) -- DUT IPv4 地址跳变, 当 IP 版本为 IPv6，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID 增量步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **HoldTime** (*int*) -- Hold Time 间隔 (sec), 类型为：number, 取值范围：3-65535, 默认值：90
- **KeepaliveTime** (*int*) -- Keep Alive 间隔 (sec), 类型为：number, 取值范围：1-65535, 默认值：30
- **ConnectRetryCount** (*int*) -- 重连次数, 取值范围：0-65535, 默认值：0
- **ConnectRetryInterval** (*int*) -- 重连间隔 (sec), 取值范围：10-300, 默认值：30
- **MaxRoutesPerUpdateMessage** (*int*) -- Update 报文中最大路由数量, 取值范围：10-300, 默认值：2000
- **RouteRefreshMode** (*str*) -- Route Refresh 模式, 类型为：string, 取值范围：None; Route Refresh, 默认值：None
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RestartTime** (*int*) -- 平滑重启时间（秒）, 取值范围：3-4095, 默认值：90
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Authentication** (*str*) -- 使用的认证类型, 类型为：string, 取值范围：None 或 MD5, 默认值：None
- **Password** (*str*) -- 认证密码, 类型为：类型为：string, 取值范围：字符串，由 1-255 个数字、字母或特殊字符组成, 默认值：xinertel
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableSr** (*bool*) -- 使能 SR, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 BGP 协议会话对, 类型：object

返回类型 (BgpRouter)

实际案例

```
| Create Bgp | Port=${Port} |
```

static create_bgp_capability(*Session*, ***kwargs*)

创建 BGP Capability 对象

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP Capability 名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP Capability, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **CapabilityCode** (*int*) -- Capability Code, 类型为：number, 默认值：1, 取值范围：1-255
- **CapabilityValue** (*str*) -- Capability 值类型为：list

返回 BGP Capability 对象, 类型：object

返回类型 (BgpCapabilityConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${CapabilityValue} | Create List | 1 | 2 | 3 | 4 | 5 |
| Create Bgp Capability | Session=${Session} | CapabilityCode=5 |␣
↪CapabilityValue=${CapabilityValue} |
```

static create_bgp_evpn_ethernet_segment_routes(*Session*, ***kwargs*)

创建 Bgp Evpn Ethernet Segment Routes 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **NextHopIpv6** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **EnableOriginatorId** (*bool*) -- 是否启用 Originator ID, 当仿真路由器作为 BGP 路由反射器时使用该属性, 类型为：bool, 默认值：False

- **OriginatorId** (*str*) -- 指定 originator ID 的值。该值用于标识路由发起者的 router id, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **EthernetSegmentType** (*str*) -- 类型为：string, 默认值：OPERATOR, 取值范围：

  OPERATOR

  IEEE802

  BRIDGEDLAN

  MACBASED

  ROUTEID

  AS
- **EthernetSegmentIdentifier** (*str*) -- 类型为：string, 默认值：00:00:00:00:00:00:00:00:00:00
- **EviCount** (*int*) -- 创建的 EVI （EVPN instance，EVPN 实例）数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **DataPlanEncapsulation** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

  SRv6
- **EsImportRoute** (*str*) -- 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:00

返回 Bgp Evpn Ethernet Segment Routes 对象, 类型：object / list

返回类型 (EvpnRouteEthernetSegmentConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Ethernet Segment Routes | Session=${Session} |
```

static create_bgp_evpn_inclusive_multicast_routes(*Session*, ***kwargs*)

创建 Bgp Evpn Inclusive Multicast Routes 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **NextHopIpv6** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **EnableOriginatorId** (*bool*) -- 是否启用 Originator ID, 当仿真路由器作为 BGP 路由反射器时使用该属性, 类型为：bool, 默认值：False
- **OriginatorId** (*str*) -- 指定 originator ID 的值。该值用于标识路由发起者的 router id, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **EviCount** (*int*) -- 创建的 EVI （EVPN instance，EVPN 实例）数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **EthernetTagId** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：1
- **PmsiTunnelType** (*str*) -- 指定多播报文传输所使用隧道的类型。只支持 INGRESS_REPLICATION（头端复制隧道。隧道标识携带本地 PE 的单播隧道目的端 IP 地址）。类型为：string, 默认值：INGRESS_REPLICATION
- **DataPlanEncapsulation** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

  SRv6

- **Label1** (*int*) -- 封装标签（VNI/VSID）, 数据平面封装为 VXLAN 时可见, 指定转发二层业务流量所使用封装标签的起始值, 类型为：number, 取值范围：1-16777215, 默认值：0
- **Label1Step** (*int*) -- 指定转发二层业务流量所使用封装标签的跳变步长, 类型为：number, 取值范围：1-16777215, 默认值：1

- **EnableCustomMplsLabel** (*bool*) -- 使能自定义 MPLS 标签, 类型为:bool, 默认值：False
- **MplsLabel** (*int*) -- MPLS 标签, 类型为：number, 取值范围：1-1048575, 默认值：0
- **MplsLabelStep** (*int*) -- MPLS 标签跳变, 类型为：number, 取值范围：1-1048575, 默认值：0

返回 Bgp Evpn Inclusive Multicast Routes 对象, 类型：object / list

返回类型 (EvpnRouteInclusiveMulticastConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Inclusive Multicast Routes | Session=${Session} |
```

static create_bgp_evpn_ip_prefix_routes(*Session*, ***kwargs*)

创建 Bgp Evpn Ip Prefix Routes 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **NextHopIpv6** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **EnableOriginatorId** (*bool*) -- 是否启用 Originator ID, 当仿真路由器作为 BGP 路由反射器时使用该属性, 类型为：bool, 默认值：False
- **OriginatorId** (*str*) -- 指定 originator ID 的值。该值用于标识路由发起者的 router id, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1

- **EthernetSegmentType** (*str*) -- 指定以太网段标识的类型，用于确定以太网段标识的格式, 类型为：string, 默认值：OPERATOR, 取值范围：

  OPERATOR

  IEEE802

  BRIDGEDLAN

  MACBASED

  ROUTEID

  AS

- **EthernetSegmentIdentifier** (*str*) -- 指定 CE 和 PE 之间连接的标识, 类型为：string, 默认值：00:00:00:00:00:00:00:00:00:00
- **EthernetTagId** (*int*) -- 指定广播域(例如 VLAN)的标识, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **NetWorkCount** (*int*) -- 指定要创建的网络数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **EviCount** (*int*) -- 创建的 EVI （EVPN instance，EVPN 实例）数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **VpnDistributionType** (*str*) -- RT、RD 以及 VNI 值在 VPN 之间的分配方式, 类型为：string, 默认值：ROUNDROBIN, 取值范围：

  ROUNDROBIN

  LINEAR

- **DataPlanEncapsulation** (*str*) -- 封装有效负载所使用的头部类型, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

  SRv6

- **EnableCustomMplsLabel** (*bool*) -- 使能自定义 MPLS 标签, 类型为：bool, 默认值：False
- **Label1** (*int*) -- 封装标签（VNI/VSID）, 数据平面封装为 VXLAN 时可见, 指定转发二层业务流量所使用封装标签的起始值, 类型为：number, 取值范围：1-16777215, 默认值：0
- **Label1Step** (*int*) -- 指定转发二层业务流量所使用封装标签的跳变步长, 类型为：number, 取值范围：1-16777215, 默认值：1
- **IpType** (*str*) -- 类型为：string, 默认值：IPV4, 取值范围：

  IPV4

  IPV6

- **StartIpv4Address** (*str*) -- 起始 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：100.0.0.2
- **Ipv4Increment** (*str*) -- IPv4 地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **Ipv4PrefixLength** (*int*) -- IPv4 地址前缀, 类型为：number, 取值范围：1-32, 默认值：24
- **GatewayIp** (*str*) -- 类型为：有效的 ipv4 地址, 默认值：0.0.0.0

- **StartIpv6Address** (*str*) -- 起始 IPv6 地址, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **Ipv6Increment** (*str*) -- IPv6 地址跳变, 类型为：有效的 ipv6 地址, 默认值：::1
- **Ipv6PrefixLength** (*int*) -- IPv6 地址前缀, 类型为：number, 取值范围：1-128, 默认值：64
- **GatewayIpv6** (*str*) -- 类型为：有效的 ipv6 地址, 默认值：'::'
- **EnableIncludeRouterMac** (*bool*) -- 是否包含路由 MAC, 类型为：bool, 默认值：False
- **RouterMac** (*str*) -- 路由 MAC 地址, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **MplsLabel** (*int*) -- MPLS 标签, 类型为：number, 取值范围：1-1048575, 默认值：0
- **MplsLabelStep** (*int*) -- MPLS 标签跳变, 类型为：number, 取值范围：1-1048575, 默认值：0

返回 Bgp Evpn Ip Prefix Routes 对象, 类型：object / list

返回类型 (EvpnRouteIpPrefixConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Ip Prefix Routes | Session=${Session} |
```

static create_bgp_evpn_mac_ip_routes(*Session*, ***kwargs*)

创建 Bgp Evpn Mac Ip Routes 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 指定路由属性中的 ORIGIN 值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **NextHopIpv6** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **EnableOriginatorId** (*bool*) -- 是否启用 Originator ID, 当仿真路由器作为 BGP 路由反射器时使用该属性, 类型为：bool, 默认值：False

- **OriginatorId** (*str*) -- 指定 originator ID 的值。该值用于标识路由发起者的 router id, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **EthernetSegmentType** (*str*) -- 指定以太网段标识的类型，用于确定以太网段标识的格式, 类型为：string, 默认值：OPERATOR, 取值范围：

  OPERATOR

  IEEE802

  BRIDGEDLAN

  MACBASED

  ROUTEID

  AS
- **EthernetSegmentIdentifier** (*str*) -- 指定 CE 和 PE 之间连接的标识, 类型为：string, 默认值：00:00:00:00:00:00:00:00:00:00
- **EthernetTagId** (*int*) -- 指定广播域(例如 VLAN)的标识, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **NetWorkCount** (*int*) -- 指定要创建的网络数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartMacAddress** (*str*) -- 指定路由块中的起始 MAC 地址, 类型为：有效的 mac 地址, 默认值：00:10:01:00:00:01
- **MacIncrement** (*str*) -- 指定路由块中 MAC 地址的跳变步长, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:01
- **EviCount** (*int*) -- 创建的 EVI （EVPN instance，EVPN 实例）数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **VpnDistributionType** (*str*) -- RT、RD 以及 VNI 值在 VPN 之间的分配方式, 类型为：string, 默认值：ROUNDROBIN, 取值范围：

  ROUNDROBIN

  LINEAR
- **AssociatedIpType** (*str*) -- 指定通告路由中所携带主机 IP 路由的版本, 类型为：string, 默认值：IPV4, 取值范围：

  NONE

  IPV4

  IPV6
- **DataPlanEncapsulation** (*str*) -- 封装有效负载所使用的头部类型, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  VXLAN

  MPLS

SRv6

- **EnableMacMobility** (*bool*) -- 使能 MAC 地址迁移, 类型为：bool, 默认值：False
- **StickyStatic** (*bool*) -- MAC 地址是静态 MAC 地址, 类型为：bool, 默认值：False
- **SequenceNumber** (*int*) -- 指定 MAC 迁移扩展团体属性 TLV 中的序列号起始值, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **Label1** (*int*) -- 封装标签（VNI/VSID）, 数据平面封装为 VXLAN 时可见, 指定转发二层业务流量所使用封装标签的起始值, 类型为：number, 取值范围：1-16777215, 默认值：0
- **Label1Step** (*int*) -- 指定转发二层业务流量所使用封装标签的跳变步长, 类型为：number, 取值范围：1-16777215, 默认值：1
- **StartIpv4Address** (*str*) -- 起始 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：100.0.0.2
- **Ipv4Increment** (*str*) -- IPv4 地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **Ipv4PrefixLength** (*int*) -- IPv4 地址前缀, 类型为：number, 取值范围：1-32, 默认值：24
- **StartIpv6Address** (*str*) -- 起始 IPv6 地址, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **Ipv6Increment** (*str*) -- IPv6 地址跳变, 类型为：有效的 ipv6 地址, 默认值：::1
- **Ipv6PrefixLength** (*int*) -- IPv6 地址前缀, 类型为：number, 取值范围：1-128, 默认值：64
- **EnableLabel2** (*bool*) -- 使能 MPLS Label2, 类型为: bool, 默认值: False
- **Label2** (*int*) -- 封装 2 标签 (L3 VNI), 类型为：number, 取值范围：1-16777215, 默认值：2000
- **Label2Step** (*int*) -- 封装 2 标签跳变 (L3 VNI Step), 类型为：number, 取值范围：1-16777215, 默认值：1
- **EnableIncludeRouterMac** (*bool*) -- 是否包含路由 MAC, 类型为：bool, 默认值：False
- **RouterMac** (*str*) -- 路由 MAC 地址, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **EnableIncludeDefaultGateway** (*bool*) -- 指定默认网关, 类型为：bool, 默认值：False
- **EnableCustomMplsLabel** (*bool*) -- 使能自定义 MPLS 标签, 类型为:bool, 默认值：False
- **MplsLabel** (*int*) -- MPLS 标签, 类型为：number, 取值范围：1-1048575, 默认值：0
- **MplsLabelStep** (*int*) -- MPLS 标签跳变, 类型为：number, 取值范围：1-1048575, 默认值：0
- **EnableCustomMplsLabel2** (*bool*) -- 使能自定义 MPLS Label2, 类型为: bool, 默认值：False
- **MplsLabel2** (*int*) -- MPLS 标签 2, 类型为：number, 取值范围：1-1048575, 默认值：0
- **MplsLabel2Step** (*int*) -- MPLS 标签 2 跳变, 类型为：number, 取值范围：1-1048575, 默认值：0

返回 Bgp Evpn Mac Ip Routes 对象, 类型：object / list

返回类型 (EvpnRouteMacIpConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Mac Ip Routes | Session=${Session} |
```

static create_bgp_evpn_route_ad(*Session*, ***kwargs*)

创建 BGP EVPN 以太自动发现路由对象

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP EVPN 以太自动发现路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP EVPN 以太自动发现路由, 类型为：bool, 默认值：True, 取值范围：True 或 False,
- **Origin** (*str*) -- ORIGIN, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- AS 路径的值, 类型为：list
- **AdRouteType** (*str*) -- 以太自动发现路由类型, 类型为：string, 默认值：ESI, 取值范围：

  EVI

  ESI

  VPWS
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True, 取值范围：True 或 False
- **NextHop** (*str*) -- 下一跳, 类型为：string, 默认值：100.0.0.1, 取值范围：IPv4 地址
- **NextHopIpv6** (*str*) -- IPv6 下一跳, 类型为：string, 默认值：2001::1, 取值范围：IPv6 地址
- **EnableLinkLocalNextHop** (*bool*) -- 使能 IPv6 Link Local 下一跳, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳, 类型为：string, 默认值：fe80::1, 取值范围：IPv6 地址
- **EnableOriginatorId** (*bool*) -- 使能 Originator ID, 类型为：bool, 默认值：False, 取值范围：True 或 False
- **OriginatorId** (*str*) -- Originator ID, 类型为：string, 默认值：192.0.0.1, 取值范围：IPv4 地址
- **VrfRouteTarget** (*str*) -- VRF 路由目标, 类型为：string, 默认值：100:1, 取值范围：AS:Number
- **VrfRouteTargetStep** (*str*) -- VRF 路由目标跳变, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1, 取值范围：AS:Number 或 IPv4:Number

- **VrfRouteDistinguisherStep** (*str*) -- VRF 路由标识跳变, 类型为: string, 默认值: 0:1
- **EthernetSegmentType** (*str*) -- 以太网段类型, 类型为: string, 默认值: OPERATOR, 取值范围:

  OPERATOR

  IEEE802

  BRIDGEDLAN

  MACBASED

  ROUTEID

  AS
- **EthernetSegmentIdentifier** (*str*) -- 以太网段标识, 类型为: string, 默认值: OPERATOR
- **EthernetTagId** (*str*) -- 以太网标签 ID, 类型为: string, 默认值: 00:00:00:00:00:00:00:00:00:00
- **EthernetTagIdStep** (*int*) -- 太网标签 ID 跳变, 类型为: number, 默认值: 1, 取值范围: 0-4294967295
- **EthernetTagCountPerEvi** (*int*) -- 每个 EVI 下以太网标签数, 类型为: number, 默认值: 1, 取值范围: 1-4294967295
- **ActiveStandbyMode** (*str*) -- 主备方式, 类型为: string, 默认值: SINGLE, 取值范围:

  ALL

  SINGLE
- **EviCount** (*int*) -- EVI 数, 类型为: number, 默认值: 1, 取值范围: 1-4294967295
- **DataPlanEncapsulation** (*str*) -- 数据平面封装, 类型为: string, 默认值: NONE, 取值范围:

  NONE

  VXLAN

  MPLS

  SRv6
- **Label1** (*int*) -- 封装标签（VNI/VSID）, 类型为: number, 默认值: 100, 取值范围: 0-16777215
- **Label1Step** (*int*) -- 封装标签跳变（VNI/VSID Step）, 类型为: number, 默认值: 1, 取值范围: 0-16777215
- **IncludeLayer2AttributeExtendedCommunity** (*bool*) -- 以太自动发现路由类型为 VPWS 时可见, 类型为: bool, 默认值: False, 取值范围: True 或 False
- **PBit** (*bool*) -- P Bit 多归单活场景中，该标志位置 1 表示该 PE 为主 PE, 类型为: bool, 默认值: False, 取值范围: True 或 False
- **BBit** (*bool*) -- B Bit 多归单活场景中，该标志位置 1 表示该 PE 为备 PE, 类型为: bool, 默认值: False, 取值范围: True 或 False
- **CBit** (*bool*) -- C Bit 置 1 时发送给该 PE 的 EVPN 报文必须携带控制字, 类型为: bool, 默认值: False, 取值范围: True 或 False
- **L2Mtu** (*bool*) -- 指定最大传输单元，单位是字节, 类型为: bool, 默认值: False, 取值范围: True 或 False

- **EnableCustomMplsLabel** (*bool*) -- 使能自定义 MPLS 标签, 类型为:bool, 默认值: False, 取值范围: True 或 False
- **MplsLabel** (*int*) -- MPLS 标签值, 类型为: number, 默认值: 0, 取值范围: 0-1048575
- **MplsLabelStep** (*int*) -- MPLS 标签步长, 类型为: number, 默认值: 0, 取值范围: 0-1048575

返回 BGP EVPN 以太自动发现路由对象, 类型: object

返回类型 (EvpnRouteAdConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Evpn Route Ad | Session=${Session} |␣
↪EnableCustomMplsLabel=True | MplsLabel=1000 | MplsLabelStep=2 |
```

static create_bgp_flow_spec_conponent_type(*FlowSpec*, *Types*, ***kwargs*)

创建 Bgp Flow Specs Conponent Type 对象, 类型为: object / list

参数

- **FlowSpec** (BgpIpv4FlowSpecConfig) -- BGP Flow Spec 对象, 类型为: object / list
- **Types** (*int*) -- BGP IPv4 FLowSpec Type 序号, 类型为: number, 取值范围: 1-12

关键字参数

- **IpValue** (*str*) -- 指定起始地址, 类型为有效的 ipv4 地址, 默认值: "192.0.1.0"
- **PrefixLength** (*int*) -- 指定前缀长度, 类型为: number, 取值范围: 1-32, 默认值: 24
- **AddressList** (*str*) -- 类型为列表时, 指定地址列表, 类型为有效的 ipv4 地址, 默认值: ""
- **InputType** (*str*) -- 指定类型, 类型为: string, 默认值: RANGE, 取值范围:

  RANGE

  LIST

  RFC_4814
- **Count** (*int*) -- 地址数量, 类型为: number, 取值范围: 1-99, 默认值: 1
- **Step** (*int*) -- 地址跳变步长, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **EqualBit** (*bool*) -- 数据与指定值相等表示匹配, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **LessThanBit** (*bool*) -- 数据小于指定值表示匹配, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **MoreThanBit** (*bool*) -- 数据大于指定值表示匹配, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **AndBit** (*bool*) -- 如果选中 And Bit, 则该 {option/value} 组与前一个 {option/value} 组之间的关系是逻辑与 (AND); 如果未选中 And Bit, 则该 {option/value} 组与前一个 {option/value} 组之间的关系是逻辑或 (OR)。类型为: bool, 取值范围: True 或 False, 默认值: False

- **ValueField** (*int*) -- 类型为：number, 取值范围：1-15, 默认值：1
- **Count** -- 类型为：number, 取值范围：0-99, 默认值：1
- **ValueIncrement** (*int*) -- 类型为：number, 取值范围：0-65535, 默认值：1
- **ValueList** (*int*) -- 类型为：number, 取值范围：0-65535, 默认值：""
- **ValueType** (*str*) -- 指定值的类型, 类型为：string, 默认值：Increment, 取值范围：

  Increment

  List
- **NotBit** (*bool*) -- 选中 Not Bit 时，对计算结果按位取反。类型为：bool, 取值范围：True 或 False, 默认值：False
- **MatchBit** (*bool*) -- 选中 Match Bit 时，(data & value) == value 表示按位匹配；未选中 Match Bit 时，如果数据中包含值掩码，则 (data & value) == True 表示匹配。其中，data 是发送的数据，value 是给定的值掩码。类型为：bool, 取值范围：True 或 False, 默认值：False
- **DSCPValue** (*int*) -- 类型为：number, 取值范围：0-63, 默认值：0

返回 Bgp Flow Spec Conponent Type 对象, 类型：object / list

返回类型 (BgpIpv4FlowSpecType1Component)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpec} | Create Bgp Ipv4 Flow Specs | Session=${Session} |
| Create Bgp Flow Spec Conponent Type | FlowSpec=${FlowSpec} | Type=1 |
```

static create_bgp_flow_spec_custom_path_attribute(*FlowSpec*, ***kwargs*)

创建 BGP Flow Spec Custom Path Attribute 对象, 类型为：object / list

参数 **FlowSpec** (BgpIpv4FlowSpecConfig) -- 所属的 Bgp Flow Spec 对象, 类型为：object / list

关键字参数

- **PathAttributeType** (*int*) -- 路径属性的类型, 类型为：number, 取值范围：1-255, 默认值：1
- **OptionalFlag** (*str*) -- 指定 Optional Flag 的值, 类型为：string, 默认值：OPTION, 取值范围：

  WELL_KNOWN

  OPTION
- **TransitiveFlag** (*str*) -- 指定 Transitive Flag 的值, 类型为：string, 默认值：NONTRANSITIVE, 取值范围：

  NONTRANSITIVE

  TRANSITIVE
- **PartialFlag** (*str*) -- 指定 Partial Flag 的值, 类型为：string, 默认值：PARTIAL, 取值范围：

  COMPLETE

  PARTIAL

- **ExtendedLengthFlag** (*bool*) -- 是否启用 Extended Length Flag, 类型为：bool, 默认值：False
- **AttributeValue** (*str*) -- 指定路径属性的值, 类型为：string, 默认值：""

返回 Bgp Route Pool Custom Path Attribute 对象, 类型：object / list

返回类型 (BgpFlowSpecPathAttributeConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpecs} | Create Bgp Ipv4 Flow Specs | Session=${Session} |
| Create Bgp Flow Spec Custom Path Attribute | FlowSpec=${FlowSpecs} |
```

static create_bgp_flow_specs_actions(*FlowSpec*, ***kwargs*)

创建 Bgp Ipv4 Flow Specs Actions 对象, 类型为：object / list

参数 **FlowSpec** (BgpIpv4FlowSpecConfig) -- BGP Flow Spec 对象, 类型为：object / list

关键字参数

- **EnableTrafficRate** (*bool*) -- 启用流量限速动作。类型为：bool, 取值范围：True 或 False, 默认值：True
- **TrafficRate** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：0
- **AsNum** (*int*) -- 指定 AS 号。类型为：number, 取值范围：1-4294967295, 默认值：1
- **EnableTrafficAction** (*bool*) -- 启用 Traffic action。类型为：bool, 取值范围：True 或 False, 默认值：True
- **SampleBit** (*bool*) -- 启用流量抽样记录。类型为：bool, 取值范围：True 或 False, 默认值：True
- **TerminateBit** (*bool*) -- 撤销已生效的匹配规则。类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableRedirect** -- 启用流量重定向到指定的路由目标动作。类型为：bool, 取值范围：True 或 False, 默认值：False
- **RouteTarget** (*str*) -- 指定重定向的路由目标。类型为：string, 默认值："100:1"
- **EnableTrafficMarking** (*bool*) -- 启用重新标记报文 DSCP 值的动作。类型为：bool, 取值范围：True 或 False, 默认值：True
- **DSCPValue** (*int*) -- 以十六进制形式指定重新标记报文所使用的 DSCP 值。类型为：number, 取值范围：1-4294967295, 默认值：0
- **EnableRedirectToIpNextHop** (*bool*) -- 启用重定向到下一跳动作, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NextHop** (*str*) -- 指定下一跳 IP, 类型为有效的 ipv4 地址, 默认值："0.0.1.0"
- **CopyBit** (*bool*) -- 复制一份规则匹配的流量，并执行重定向到下一跳动作。类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Bgp Ipv4 Flow Specs Actions 对象, 类型：object / list

返回类型 (BgpIpv4FlowSpecAction)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpec} | Create Bgp Ipv4 Flow Specs | Session=${Session} |
| Create Bgp Ipv4 Flow Specs Action | FlowSpec=${FlowSpec} |
```

static create_bgp_ipv4_flow_specs(*Session*, ***kwargs*)

创建 Bgp Ipv4 Flow Specs 对象, 类型为：object / list

参数 Session (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **RouteCount** (*int*) -- 类型为：number, 取值范围：1-8000000, 默认值：1
- **FlowSpecSubAfi** (*str*) -- 指定 SubAFI 的值, 类型为：string, 默认值：FlowSpec, 取值范围：

  FlowSpec

  FlowSpecVpn
- **Origin** (*str*) -- 指定 ORIGIN 的值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE
- **EnableLocalPref** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalPref** (*int*) -- 类型为：number, 默认值：10
- **EnableMed** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MultExitDisc** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：0
- **EnableClusterIdList** (*bool*) -- 是否启用 Cluster ID List, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ClusterIdList** (*str*) -- 指定 cluster ID list 的值, 类型为：有效的 ipv4 地址, 默认值：""
- **EnableCommunity** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CommunityType** (*str*) -- 团体类型, 类型为：string, 默认值：AA_NN, 取值范围：

  AA_NN

  NO_EXPORT

  NO_ADVERTISE

  LOCAL_AS

- **CommunityAsNumber** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 AS 号, 类型为：number, 取值范围：1-65535, 默认值：1
- **CommunityId** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 ID 值, 类型为：number, 取值范围：1-65535, 默认值：1
- **Community** (*str*) -- 当 Type 为 NO_EXPORT 时，团体值为 0xffffff01；当 Type 为 NO_ADVERTISE 时，团体值为 0xffffff02；当 Type 为 LOCAL_AS 时，团体值为 0xffffff03, 类型为：string, 默认值：""
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：string, 默认值：""
- **ComponentType** (*list*) -- 过滤规则, 类型为：list, 默认值：Type1, 取值范围：

  Type1

  Type2

  Type3

  Type4

  Type5

  Type6

  Type7

  Type8

  Type9

  Type10

  Type11

  Type12
- **VrfNum** (*int*) -- VRF 数量, 类型为：number, 默认值：1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：1:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1

返回 Bgp Ipv4 Flow Specs 对象, 类型：object / list

返回类型 (BgpIpv4FlowSpecConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Ipv4 Flow Specs | Session=${Session} |
```

static create_bgp_ipv4_flowspec_performance(*Session*, *MaxRouteCount*, *SourcePrefix*, *DestPrefix*)

创建 bgpflowspec 性能条目

参数

- **Session** -- bgp session

- **MaxRouteCount** -- 支持最大 bgpls 数量
- **SourcePrefix** -- 接口列表
- **DestPrefix** -- BGP ipv4 路由列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

Examples:

static create_bgp_ipv4_route_pool(*Session*, ***kwargs*)

创建 BGP IPv4 路由对象

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP IPv4 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP IPv4 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SubAfi** (*str*) -- Sub-AFI, 类型为：string, 默认值：UNICAST, 支持类型：

  UNICAST

  MULTICAST

  VPN

  LABELED
- **Origin** (*str*) -- 路由属性中的 ORIGIN 值, 类型为：string, 取值范围：Incomplete; IGP; EGP, 默认值：IGP
- **AsPath** (*int*) -- AS Path, 类型为：number, 取值范围：1-255,
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 取值范围：Incomplete; IGP; EGP, 默认值：IGP
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableLocalPref** (*bool*) -- 使能 Local Preference, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalPref** (*int*) -- LocalPref, 当使能 Local Preference 为选中状态时配置该选项, 类型为：number, 取值范围：0-4294967295, 默认值：10
- **EnableMed** (*bool*) -- 使能 Multi Exit Discriminator, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MultExitDisc** (*int*) -- Multi Exit Discriminator, 当使能 Multi Exit Discriminator 为选中状态时配置该选项, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **AtomicAggregate** (*bool*) -- 使能 Atomic Aggregate, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableAggregator** (*bool*) -- 使能 Aggregator, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AggregatorAsNumber** (*int*) -- Aggregator 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **AggregatorIp** (*str*) -- Aggregator IP, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **EnableOriginatorId** (*bool*) -- 使能 Originator ID, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **OriginatorId** (*str*) -- Originator ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **EnableClusterIdList** (*bool*) -- 使能 Cluster ID List, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ClusterIdList** (*str*) -- Cluster ID List, 类型为：string, 取值范围：IPv4 地址, 默认值：空
- **EnableCommunity** (*bool*) -- 使能团体, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CommunityType** (*str*) -- 团体类型, 类型为：string 默认值：AA:NN, 取值范围：

  AA_NN

  NO_EXPORT

  NO_ADVERTISE

  LOCAL_AS
- **CommunityAsNumber** (*int*) -- 团体自治域, 当 Type 为 AA:NN 时，指定团体的 AS 号, 类型为：number, 取值范围：1-65535, 默认值：1
- **CommunityId** (*int*) -- 团体 ID, 当 Type 为 AA:NN 时，指定团体的 ID 值, 类型为：number, 取值范围：1-65535, 默认值：1
- **Community** (*str*) -- 团体, 类型为：list, 默认值：[]
- **CommunityIncrement** (*str*) -- 团体跳变, 类型为：list, 默认值：[]
- **CommunityPerBlockCount** (*int*) -- 每个路由组团体数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：list, 默认值：[], 例如：['0x00:0x02:1:1', '0x01:0x02:1:2', '0x02:0x02:1:3']
- **VrfRd** (*str*) -- VRF 路由标识, 类型为：string, 取值范围：AS:Number 或 IPv4:Number, 默认值：1:1
- **VrfRdStep** (*str*) -- VRF 路由标识跳变, 类型为：string, 取值范围：AS:Number 或 IPv4:Number, 默认值：0:1
- **VrfRt** (*str*) -- VRF 路由目标, 类型为：string, 取值范围：AS:Number, 默认值：100:1
- **VrfRtStep** (*str*) -- VRF 路由目标跳变, 类型为：string, 取值范围：AS:Number, 默认值：0:1
- **VrfCount** (*int) (int*) -- VRF 数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartingLabel** (*int*) -- 起始标签, 类型为：number, 取值范围：0-1048575, 默认值：16
- **LabelType** (*str*) -- 路由标签类型, 类型为：类型为：string, 取值范围：Fixed; Incrementa; Explicit Nul; Implicit Null, 默认值：Fixed
- **FirstRoute** (*str*) -- IPv4 路由起始值, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **RandomRoute** (*bool*) -- 随机路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RouteCount** (*int*) -- 每个会话路由数量, 类型为：number, 取值范围：1-8000000, 默认值：1
- **RouteStep** (*str*) -- IPv4 路由跳变步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.1.0

- **Ipv4RouteStep** (*int*) -- IPv4 路由跳变步长, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **PrefixLength** (*int*) -- IPv4 路由前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **NextHopAddrType** (*str*) -- 下一跳地址类型, 类型为：string, 取值范围：IPv4; IPv6, 默认值：IPv4
- **NextHop** (*str*) -- 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **NextHopStep** (*str*) -- 下一跳步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **IPv6NextHop** (*str*) -- IPv6 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：2000::1
- **IPv6NextHopStep** (*str*) -- IPv6 下一跳步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 Link Local 地址作为下一跳, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Ipv6LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：2000::1
- **Ipv6LinkLocalNextHopStep** (*str*) -- IPv6 Link Local 下一跳步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **EncodeSrTlvs** (*list*) -- 编码 SR TLV, 类型为：list, 默认值：0, 取值范围：

  LABEL_INDEX

  SRGB

  SRV6_VPN_SID

  SRV6_SERVICES
- **OverrideGlobalSrgb** (*bool*) -- 覆盖全局 SRGB, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SrgbBase** (*int*) -- SRGB 起始值, 类型为：number, 取值范围：0-16777215, 默认值：16000
- **SrgbRange** (*int*) -- SRGB 范围, 类型为：number, 取值范围：0-16777215, 默认值：1000
- **LabelIndex** (*int*) -- 标签序号, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **LabelStep** (*int*) -- 标签步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **Srv6SidInfoSubTlvType** (*int*) -- SRv6 SID Information Sub TLV 类型, 类型为：number, 取值范围：1-255, 默认值：1
- **Srv6LocatorBlockLength** (*int*) -- SRv6 Locator Block 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **Srv6LocatorNodeLength** (*int*) -- SRv6 Locator Node 长度, 类型为：number, 取值范围：1-128, 默认值：32
- **Srv6FuncLength** (*int*) -- SRv6 Function 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **Srv6FuncOpcode** (*str*) -- SRv6 Function Opcode, 类型为：string, 取值范围：格式：ff:ff:ff, 默认值：0

- **Srv6ArguLength** (*int*) -- SRv6 Argument 长度, 类型为：number, 取值范围：1-128, 默认值：32
- **Srv6Argument** (*str*) -- SRv6 Argument, 类型为：string, 取值范围：格式：ff:ff:ff, 默认值：0
- **EncodedSrv6ServiceDataSubTlvs** (*list*) -- 编码 SRv6 Service Data Sub TLVs, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  SRV6_ID_STRUCTURE
- **Srv6TranspositionLength** (*int*) -- SRv6 Transposition 长度, 类型为：number, 取值范围：0-24, 默认值：0
- **Srv6TranspositionOffset** (*int*) -- SRv6 Transposition 偏移, 类型为：number, 取值范围：0-15, 默认值：0
- **Srv6Locator** (*str*) -- 使用的认证类型, 类型为：string, 取值范围：None 或 MD5, 默认值：None
- **Srv6LocatorStep** (*str*) -- 认证密码, 类型为：类型为：string, 取值范围：字符串，由 1-255 个数字、字母或特殊字符组成, 默认值：xinertel
- **Srv6EndpointBehavior** (*str*) -- SRv6 Endpoint Behavior, 类型为：string, 默认值：CUSTOM, 支持的值：

  END_DX6

  END_DX4

  END_DT6

  END_DT4

  END_DT46

  END_DX2

  END_DX2V

  END_DT2U

  END_DT2M

  CUSTOM
- **Srv6CustomEndpointBehavior** (*int*) -- 自定义 SRv6 Endpoint Behavior, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0xFFFF

返回 BGP IPv4 路由对象, 类型：object

返回类型 (BgpIpv4RoutepoolConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${Community} | Create List | AA_NN | NO_EXPORT | NO_ADVERTISE | LOCAL_
↪AS |
| ${CommunityIncrement} | Create List | 1:1 | 1:2 | 1:3 | 1:4 |
| ${ExtendedCommunity} | Create List | 0x00:0x02:1:1 | 0x01:0x02:1:2 |␣
↪0x02:0x02:1:3 |
| ${RoutePool} | Create Bgp Ipv4 Route Pool | Session=${Session} |␣
↪Community=${Community} | CommunityIncrement=${CommunityIncrement} |␣
↪ExtendedCommunity=${ExtendedCommunity} |
| ${Community} | Create List | AA_NN | NO_EXPORT |
```

(下页继续)

(续上页)

```
| ${CommunityIncrement} | Create List | 2:1 | 2:2 |
| Edit Configs | Configs=${RoutePool} | Community=${Community} |␣
↪CommunityIncrement=${CommunityIncrement} | CommunityPerBlockCount=2 |
```

static create_bgp_ipv4_vpls(*Session*, ***kwargs*)

创建 Bgp Ipv4 Vpls 对象, 类型为：object / list

参数 Session (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE

  CONFED_SEQUENCE

  CONFED_SET
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：100.0.0.1
- **MultExitDisc** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LocalPreference** (*int*) -- Local 优先级, 类型为：number, 取值范围：1-4294967295, 默认值：10
- **VeId** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **VeIdStep** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：1
- **BlockOffset** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockOffsetStep** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockSize** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：5
- **MtuSize** (*int*) -- 类型为：number, 取值范围：64-9000, 默认值：1500
- **EncapType** (*str*) -- 封装类型, 类型为：string, 默认值：VLAN, 取值范围：

  VLAN

  ETHERNET

  VPLS
- **ControlFlag** (*int*) -- 控制标识。以十进制表示。类型为：number, 取值范围：1-255, 默认值：0
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1

- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **VrfCount** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：1

返回 Bgp Ipv4 Vpls 对象, 类型：object / list

返回类型 (BgpIpv4VplsConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Ipv4 Vpls | Session=${Session} |
```

static create_bgp_ipv6_flow_spec(*Session*, ***kwargs*)

创建 BGP Ipv6 Flow Spec 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：object / list

关键字参数

- **RouteCount** (*int*) -- 类型为：number, 取值范围：1-8000000, 默认值：1
- **FlowSpecSubAfi** (*str*) -- 指定 SubAFI 的值, 类型为：string, 默认值：FlowSpec, 取值范围：

  FlowSpec

  FlowSpecVpn
- **FlowSpecActionType** (*str*) -- 指定 Optional Flag 的值, 类型为：string, 默认值：RedirectRT, 取值范围：

  RedirectRT
- **ComponentType** (*list*) -- 指定 Transitive Flag 的值, 类型为：list, 默认值：Type1, 取值范围：

  Type1

  Type2
- **DestinationPrefix** (*str*) -- 指定 Partial Flag 的值, 类型为: 有效的 ipv6 地址, 默认值：2000::1
- **DestinationPrefixLength** (*bool*) -- 是否启用 Extended Length Flag, 类型为：bool, 默认值：False
- **DestinationPrefixIncrement** (*int*) -- 指定路径属性的长度, 类型为：number, 取值范围：1-128, 默认值：64
- **DestinationPrefixCount** (*int*) -- 指定路径属性的值, 类型为：number, 取值范围：1-65535, 默认值：1
- **DestinationPrefixOffset** (*int*) -- 目的 IP 前缀偏移, 类型为：number, 默认值：0
- **SourcePrefix** (*str*) -- 源 IP 前缀, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **SourcePrefixLength** (*int*) -- 源 IP 前缀长度, 类型为：number, 取值范围：1-128, 默认值：64
- **SourcePrefixIncrement** (*int*) -- 源 IP 前缀偏移, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **SourcePrefixCount** (*int*) -- 源 IP 前缀个数, 类型为：number, 取值范围：1-65535, 默认值：1

- **SourcePrefixOffset** (*int*) -- 源 IP 前缀偏移, 类型为：number, 取值范围：0-255, 默认值：0
- **Origin** (*str*) -- 指定 ORIGIN 的值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE
- **EnableLocalPref** (*bool*) -- 是否启用 Local_PREF 属性。类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalPref** (*int*) -- 指定 Local_PREF 的值。类型为：number, 取值范围：1-4294967295, 默认值：10
- **EnableMed** (*bool*) -- 是否启用 MULTI_EXIT_DISC 属性。类型为：bool, 取值范围：True 或 False, 默认值：False
- **MultExitDisc** (*int*) -- 指定 Multi Exit Discriminator 的值。类型为：number, 取值范围：1-4294967295, 默认值：0
- **EnableClusterIdList** (*bool*) -- 是否启用 Cluster ID List, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ClusterIdList** (*str*) -- 指定 cluster ID list 的值, 类型为：有效的 ipv4 地址, 默认值：""
- **EnableCommunity** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CommunityType** (*str*) -- 团体类型, 类型为：string, 默认值：AA_NN, 取值范围：

  AA_NN

  NO_EXPORT

  NO_ADVERTISE

  LOCAL_AS
- **CommunityAsNumber** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 AS 号, 类型为：number, 取值范围：1-65535, 默认值：1
- **CommunityId** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 ID 值, 类型为：number, 取值范围：1-65535, 默认值：1
- **Community** (*str*) -- 当 Type 为 NO_EXPORT 时，团体值为 0xffffff01；当 Type 为 NO_ADVERTISE 时，团体值为 0xffffff02；当 Type 为 LOCAL_AS 时，团体值为 0xffffff03, 类型为：string, 默认值：""
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：string, 默认值：""
- **VrfNum** (*int*) -- VRF 数量, 类型为：number, 默认值：1
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1

- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为: string, 默认值: 1:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为: string, 默认值: 0:1

返回 Bgp Ipv6 Flow Spec 对象, 类型: object / list

返回类型 (BgpIpv6FlowSpecConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpecs} | Create Bgp Ipv6 Flow Specs | Session=${Session} |
```

static create_bgp_ipv6_flow_spec_action(*FlowSpec*, ***kwargs*)

创建 Bgp Ipv6 Flow Specs Actions 对象, 类型为: object / list

参数 **FlowSpec** (BgpIpv6FlowSpecConfig) -- BGP Flow Spec 对象, 类型为: object / list

关键字参数

- **EnableTrafficRate** (*bool*) -- 启用流量限速动作。类型为: bool, 取值范围: True 或 False, 默认值: True
- **TrafficRate** (*int*) -- 指定流量的最大传输速率, 类型为: number, 取值范围: 1-4294967295, 默认值: 0
- **AsNum** (*int*) -- 指定 AS 号。类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **EnableTrafficAction** (*bool*) -- 启用 Traffic action。类型为: bool, 取值范围: True 或 False, 默认值: True
- **SampleBit** (*bool*) -- 启用流量抽样记录。类型为: bool, 取值范围: True 或 False, 默认值: True
- **TerminateBit** (*bool*) -- 撤销已生效的匹配规则。类型为: bool, 取值范围: True 或 False, 默认值: True
- **EnableRedirect** (*bool*) -- 启用流量重定向到指定的路由目标动作。类型为: bool, 取值范围: True 或 False, 默认值: False
- **RedirectIpv6RouteTarget** (*str*) -- 指定重定向的路由目标。类型为: string, 默认值: "100:1"
- **EnableRedirectToIpv6NextHop** (*bool*) -- 启用重定向到下一跳动作, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **Type** (*str*) -- 类型为: string, 默认值: "TYPE_0x000c", 取值范围:

  TYPE_0x0800

  TYPE_0x000c
- **NextHop** (*str*) -- 指定下一跳 IP, 类型为有效的 ipv6 地址, 默认值: "2000::1"
- **CopyBit** (*bool*) -- 复制一份规则匹配的流量，并执行重定向到下一跳动作。类型为: bool, 取值范围: True 或 False, 默认值: True

返回 Bgp Ipv6 Flow Specs Actions 对象, 类型: object / list

返回类型 (BgpIpv6FlowSpecAction)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${FlowSpec} | Create Bgp Ipv6 Flow Specs | Session=${Session} |
| Create Bgp Ipv6 Flow Specs Action | FlowSpec=${FlowSpec} |
```

static create_bgp_ipv6_route_pool(*Session*, ***kwargs*)

创建 BGP IPv6 路由对象

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP IPv6 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP IPv6 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SubAfi** (*str*) -- Sub-AFI, 类型为：string, 默认值：UNICAST, 支持类型：

  UNICAST

  MULTICAST

  VPN

  LABELED
- **Origin** (*str*) -- 路由属性中的 ORIGIN 值, 类型为：string, 取值范围：Incomplete; IGP; EGP, 默认值：IGP
- **AsPath** (*int*) -- AS Path, 类型为：number, 取值范围：1-255,
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 取值范围：Incomplete; IGP; EGP, 默认值：IGP
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableLocalPref** (*bool*) -- 使能 Local Preference, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalPref** (*int*) -- LocalPref, 当使能 Local Preference 为选中状态时配置该选项, 类型为：number, 取值范围：0-4294967295, 默认值：10
- **EnableMed** (*bool*) -- 使能 Multi Exit Discriminator, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MultExitDisc** (*int*) -- Multi Exit Discriminator, 当使能 Multi Exit Discriminator 为选中状态时配置该选项, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **AtomicAggregate** (*bool*) -- 使能 Atomic Aggregate, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableAggregator** (*bool*) -- 使能 Aggregator, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AggregatorAsNumber** (*int*) -- Aggregator 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **AggregatorIp** (*str*) -- Aggregator IP, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **EnableOriginatorId** (*bool*) -- 使能 Originator ID, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **OriginatorId** (*str*) -- Originator ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0

- **EnableClusterIdList** (*bool*) -- 使能 Cluster ID List, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ClusterIdList** (*str*) -- Cluster ID List, 类型为：string, 取值范围：IPv4 地址, 默认值：空
- **EnableCommunity** (*bool*) -- 使能团体, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CommunityType** (*str*) -- 团体类型, 类型为：string, 取值范围：AA:NN; NO_EXPORT; NO_ADVERTISE; LOCAL_AS, 默认值：AA:NN
- **CommunityAsNumber** (*int*) -- 当 Type 为 AA:NN 时，指定团体的 AS 号, 类型为：number, 取值范围：1-65535, 默认值：1
- **Community** (*str*) -- 团体, 类型为：list, 默认值：[], 取值范围：

  AA_NN

  NO_EXPORT

  NO_ADVERTISE

  LOCAL_AS
- **CommunityIncrement** (*str*) -- 团体跳变, 类型为：list, 默认值：[], 取值范围：AA:NN
- **CommunityPerBlockCount** (*int*) -- 每个路由组团体数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：list, 默认值：[], 例如：['0x00:0x02:1:1', '0x01:0x02:1:2', '0x02:0x02:1:3']
- **VrfRd** (*str*) -- VRF 路由标识, 类型为：string, 取值范围：AS:Number 或 IPv4:Number, 默认值：1:1
- **VrfRdStep** (*str*) -- VRF 路由标识跳变, 类型为：string, 取值范围：AS:Number 或 IPv4:Number, 默认值：0:1
- **VrfRt** (*str*) -- VRF 路由目标, 类型为：string, 取值范围：AS:Number, 默认值：100:1
- **VrfRtStep** (*str*) -- VRF 路由目标跳变, 类型为：string, 取值范围：AS:Number, 默认值：0:1
- **VrfCount** (*int*) -- VRF 数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartingLabel** (*int*) -- 起始标签, 类型为：number, 取值范围：0-1048575, 默认值：16
- **LabelType** (*str*) -- 路由标签类型, 类型为：类型为：string, 取值范围：Fixed; Incrementa; Explicit Nul; Implicit Null, 默认值：Fixed
- **FirstIpv6Route** (*str*) -- IPv6 路由起始值, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **RouteCount** (*int*) -- 每个会话路由数量, 类型为：number, 取值范围：1-8000000, 默认值：1
- **RouteStep** (*str*) -- IPv6 路由跳变步长, 类型为：string, 取值范围：IPv6 地址, 默认值：'0:0:0:1::'
- **Ipv6RouteStep** (*int*) -- IPv6 路由跳变步长, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **PrefixLength** (*int*) -- IPv4 路由前缀长度, 类型为：number, 取值范围：1-32, 默认值：24

- **NextHopAddrType** (*str*) -- 下一跳地址类型, 类型为：string, 取值范围：IPv4; IPv6, 默认值：IPv4
- **NextHop** (*str*) -- 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.1.0
- **NextHopStep** (*str*) -- 下一跳步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **IPv6NextHop** (*str*) -- IPv6 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：2000::1
- **IPv6NextHopStep** (*str*) -- IPv6 下一跳步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能 Link Local 地址作为下一跳, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Ipv6LinkLocalNextHop** (*str*) -- IPv6 Link Local 下一跳地址, 类型为：string, 取值范围：IPv4 地址, 默认值：2000::1
- **Ipv6LinkLocalNextHopStep** (*str*) -- IPv6 Link Local 下一跳步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **EncodeSrTlvs** (*list*) -- 编码 SR TLV, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  SRV6_VPN_SID

  SRV6_SERVICES
- **OverrideGlobalSrgb** (*bool*) -- 覆盖全局 SRGB, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SrgbBase** (*int*) -- SRGB 起始值, 类型为：number, 取值范围：0-16777215, 默认值：16000
- **SrgbRange** (*int*) -- SRGB 范围, 类型为：number, 取值范围：0-16777215, 默认值：1000
- **LabelIndex** (*int*) -- 标签序号, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **LabelStep** (*int*) -- 标签步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **Srv6SidInfoSubTlvType** (*int*) -- SRv6 SID Information Sub TLV 类型, 类型为：number, 取值范围：1-255, 默认值：1
- **Srv6LocatorBlockLength** (*int*) -- SRv6 Locator Block 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **Srv6LocatorNodeLength** (*int*) -- SRv6 Locator Node 长度, 类型为：number, 取值范围：1-128, 默认值：32
- **Srv6FuncLength** (*int*) -- SRv6 Function 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **Srv6FuncOpcode** (*str*) -- SRv6 Function Opcode, 类型为：string, 取值范围：格式：ff:ff:ff, 默认值：0
- **Srv6ArguLength** (*int*) -- SRv6 Argument 长度, 类型为：number, 取值范围：1-128, 默认值：32
- **Srv6Argument** (*str*) -- SRv6 Argument, 类型为：string, 取值范围：格式：ff:ff:ff, 默认值：0

- **EncodedSrv6ServiceDataSubTlvs** (*list*) -- 编码 SRv6 Service Data Sub TLVs, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  SRV6_ID_STRUCTURE

- **Srv6TranspositionLength** (*int*) -- SRv6 Transposition 长度, 类型为：number, 取值范围：0-24, 默认值：0
- **Srv6TranspositionOffset** (*int*) -- SRv6 Transposition 偏移, 类型为：number, 取值范围：0-15, 默认值：0
- **Srv6Locator** (*str*) -- 使用的认证类型, 类型为：string, 取值范围：None 或 MD5, 默认值：None
- **Srv6LocatorStep** (*str*) -- 认证密码, 类型为：类型为：string, 取值范围：字符串，由 1-255 个数字、字母或特殊字符组成, 默认值：xinertel
- **Srv6EndpointBehavior** (*str*) -- SRv6 Endpoint Behavior, 类型为：string, 默认值：CUSTOM, 支持的值：

  END_DX6

  END_DX4

  END_DT6

  END_DT4

  END_DT46

  END_DX2

  END_DX2V

  END_DT2U

  END_DT2M

  CUSTOM

- **Srv6CustomEndpointBehavior** (*int*) -- 自定义 SRv6 Endpoint Behavior, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0xFFFF

返回 BGP IPv6 路由对象, 类型：object

返回类型 (BgpIpv6RoutepoolConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${Community} | Create List | AA_NN | NO_EXPORT | NO_ADVERTISE | LOCAL_
↪AS |
| ${CommunityIncrement} | Create List | 1:1 | 1:2 | 1:3 | 1:4 |
| ${ExtendedCommunity} | Create List | 0x00:0x02:1:1 | 0x01:0x02:1:2 |␣
↪0x02:0x02:1:3 |
| ${RoutePool} | Create Bgp Ipv6 Route Pool | Session=${Session} |␣
↪Community=${Community} | CommunityIncrement=${CommunityIncrement} |␣
↪ExtendedCommunity=${ExtendedCommunity} |
| ${Community} | Create List | AA_NN | NO_EXPORT |
| ${CommunityIncrement} | Create List | 2:1 | 2:2 |
| Edit Configs | Configs=${RoutePool} | Community=${Community} |␣
↪CommunityIncrement=${CommunityIncrement} | CommunityPerBlockCount=2 |
```

static create_bgp_ipv6_vpls(*Session*, ***kwargs*)

创建 Bgp Ipv6 Vpls 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：""
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **LinkLocalNextHop** (*str*) -- 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **MultExitDisc** (*int*) -- 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LocalPreference** (*int*) -- Local 优先级, 类型为：number, 取值范围：1-4294967295, 默认值：10
- **VeId** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：1
- **VeIdStep** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockOffset** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockOffsetStep** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：0
- **BlockSize** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：5
- **MtuSize** (*int*) -- 类型为：number, 取值范围：64-65535, 默认值：1500
- **EncapType** (*str*) -- 封装类型, 类型为：string, 默认值：VLAN, 取值范围：

  VLAN

  ETHERNET

  VPLS
- **EnableRfc4761** (*bool*) -- 类型为：bool, 默认值：True
- **ControlFlag** (*int*) -- 控制标识。以十进制表示。类型为：number, 默认值：0
- **StripVlan** (*bool*) -- 类型为：bool, 默认值：False
- **VeFlooding** (*bool*) -- 类型为：bool, 默认值：False
- **VrfRouteTarget** (*str*) -- 指定 VRF 路由目标起始值, 类型为：string, 默认值：100:1
- **VrfRouteTargetStep** (*str*) -- 指定 VRF 路由目标的跳变步长, 类型为：string, 默认值：0:1
- **VrfRouteDistinguisher** (*str*) -- 指定 VRF 路由标识起始值, 类型为：string, 默认值：10.0.0.2:1
- **VrfRouteDistinguisherStep** (*str*) -- 指定 VRF 路由标识的跳变步长, 类型为：string, 默认值：0:1
- **VrfCount** (*int*) -- 类型为：number, 取值范围：1-65535, 默认值：1

返回 Bgp Ipv6 Vpls 对象, 类型：object / list

返回类型 (BgpIpv6VplsConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Ipv6 Vpls | Session=${Session} |
```

static create_bgp_link_states(*Session*, ***kwargs*)

创建 Bgp Link States 对象, 类型为：object / list

参数 Session (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list

关键字参数

- **Origin** (*str*) -- 路由产生源, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE

- **AsPath** (*str*) -- 自治域路径, 类型为：string, 默认值：""
- **AsPathType** (*str*) -- 自治域路径类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE

  CONFED_SEQUENCE

  CONFED_SET

- **NextHop** (*str*) -- 下一跳, 类型为：有效的 ipv4 地址, 默认值：10.0.0.1
- **IPv6NextHop** (*str*) -- IPv6 下一跳, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **EnableLinkLocalNextHop** (*bool*) -- 使能下一跳本地链路地址, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LinkLocalNextHop** (*str*) -- 下一跳本地链路地址, 类型为：有效的 ipv6 地址, 默认值：fe80::1
- **LocalPreference** (*int*) -- 本地优先级, 类型为：number, 默认值：10
- **Community** (*str*) -- 团体，当 Type 为 NO_EXPORT 时，团体值为 0xffffff01；当 Type 为 NO_ADVERTISE 时，团体值为 0xffffff02；当 Type 为 LOCAL_AS 时，团体值为 0xffffff03, 类型为：string, 默认值：""
- **ExtendedCommunity** (*str*) -- 扩展团体, 类型为：string, 默认值：""
- **ProtocolID** (*str*) -- 协议 ID, 类型为：string, 默认值：DIRECT, 取值范围：

  ISIS_LEVEL_1

  ISIS_LEVEL_2

  OSPFV2

  DIRECT

  STATIC

  OSPFV3

  BGP

- **IdentifierType** (*str*) -- 标识符类型, 类型为：string, 默认值：CUSTOMIZED, 取值范围：

  DEFAULT_LAYER3

  CUSTOMIZED

- **Identifier** (*int*) -- 标识符, 类型为：number, 取值范围：1-1048575, 默认值：0
- **EnableNodeNLRI** (*bool*) -- 使能节点 NLRI, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LocalNodeDescriptorFlag** (*list*) -- 本地节点描述符标志, 类型为：list, 默认值：IGP_ROUTER_ID, 取值范围：

  AS_NUMBER

  BGPLS_IDENTIFIER

  OSPF_AREA_ID

  IGP_ROUTER_ID

  BGP_ROUTER_ID

  MEMBER_ASN

- **AsNumber** (*int*) -- 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **BgpLsIdentifier** (*str*) -- BGP-LS 标识符, 类型为：有效的 ipv4 地址, 默认值：1.0.0.1
- **OspfAreaId** (*str*) -- OSPF 区域 ID, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **IGPRouterIdType** (*str*) -- IGP 路由 ID 类型, 类型为：string, 默认值：OSPF_NONPSEUDONODE, 取值范围：

  ISIS_NONPSEUDONODE

  ISIS_PSEUDONODE

  OSPF_NONPSEUDONODE

  OSPF_PSEUDONODE

- **IsisNonPseud** (*str*) -- ISIS 非伪节点路由 ID, 类型为：string, 默认值：”00:10:12:00:00:01”
- **IsisPseud** (*str*) -- ISIS 伪节点路由 ID, 类型为：string, 默认值：”00:10:12:00:00:01.02”
- **OspfNonPseud** (*str*) -- OSPF 非伪节点路由 ID, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **OspfPseud** (*str*) -- OSPF 伪节点路由 ID, 类型为：string, 默认值：”192.0.0.1:192.0.0.1”
- **BgpRouterId** -- BGP 路由 ID, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **MemberAsn** (*int*) -- 自治域成员, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LocalNodeAttributeFlag** (*str*) -- 本地节点描述符标志类型为：string, 默认值：NODE_FLAG_BITS, 取值范围：

  MULTI_TOPO_ID

  NODE_FLAG_BITS

  NODE_NAME

ISIS_AREA_ID

IPV4_LOCAL_NODE_ROUTERID

IPV6_LOCAL_NODE_ROUTERID

SR_CAPABILITIES

SR_ALGORITHM

SR_LOCAL_BLOCK

SR_SRMA_PREF

SRV6_CAPABILITIES

SRV6_NODE_MSD

- **MultiTopoId** (*str*) -- 多拓扑 ID, 类型为：string, 取值范围：1-4095, 默认值：”“
- **NodeFlagBitIsis** (*str*) -- 节点标志（ISIS）, 类型为：string, 默认值：ATTACHED, 取值范围：

  OVERLOAD

  ATTACHED
- **NodeFlagBitOspfv2** (*list*) -- 节点标志（OSPFv2）, 类型为：list, 默认值：ABR, 取值范围：

  EXTERNAL

  ABR
- **NodeFlagBitOspfv3** (*list*) -- 节点标志（OSPFv3）, 类型为：list, 默认值：ABR, 取值范围：

  EXTERNAL ABR ROUTER V6
- **NodeName** (*str*) -- 节点名称, 类型为：string, 默认值：”“
- **IsisAreaId** (*int*) -- IS-IS 区域 ID, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LocalIpv4RouterIds** (*str*) -- 本端 IPv4 路由 IDs, 类型为：有效的 ipv4 地址, 默认值：”“
- **LocalIpv6RouterIds** (*str*) -- 本端 IPv6 路由 IDs, 类型为：有效的 ipv6 地址, 默认值：”“
- **SidLabelType** (*str*) -- SID/Label 类型, 类型为：string, 默认值：BIT20_LABEL, 取值范围：

  BIT20_LABEL

  BIT32_SID
- **SrCapabilitiesFlags** (*list*) -- SR 能力标志, 类型为：list, 默认值：MPLS_IPv4, 取值范围：

  MPLS_IPv4

  MPLS_IPv6
- **SrCapabilities** (*str*) -- SR 能力, 类型为：string, 默认值：”16:100”
- **SrAlgorithm** (*int*) -- SR 算法, 类型为：number, 取值范围：1-255, 默认值：0
- **SrLocalBlock** (*str*) -- SRLB 范围, 类型为：string, 默认值：”16:100”
- **SrmsPref** (*int*) -- SRMS 优先级, 类型为：number, 取值范围：1-255, 默认值：0

- **Srv6CapabilitiesFlags** (*list*) -- SRv6 能力标志, 类型为：list, 默认值：O_FLAG, 取值范围：

  UNUSED0

  O_FLAG

  UNUSED2

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

  UNUSED7

  UNUSED8

  UNUSED9

  UNUSED10

  UNUSED11

  UNUSED12

  UNUSED13

  UNUSED14

  UNUSED15

- **Srv6MsdFlags** (*list*) -- SRv6 节点 MSD 类型, 类型为：list, 默认值：NONE, 取值范围：

  NONE

  MAX_SEGMENTS_LELT

  MAX_END_POP

  MAX_T_INSERT_SRH

  MAX_T_ENCAPS_SRH

  MAX_END_D_SRH

- **Srv6MsdMaxSegmentLeft** (*int*) -- 最大 Segments Left, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEndPop** (*int*) -- 最大 End Pop, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxInsert** (*int*) -- 最大 T.Insert SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEncap** (*int*) -- 最大 T.Encaps SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEndD** (*int*) -- 最大 End D SRH, 类型为：number, 取值范围：1-255, 默认值：8

返回 Bgp Link States 对象, 类型：object / list

返回类型 (BgpLsNodeConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Link States | Session=${Session} |
```

static create_bgp_link_states_link(*Session*, *LinkState*, ***kwargs*)

创建 Bgp Link States Link 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **LinkState** (BgpLsNodeConfig) -- Bgp Link States 对象列表, 类型为：object / list

关键字参数

- **LinkLocalRemoteIdFlag** (*str*) -- 链路本端/远端 ID 位置, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  LINK_DESCRIPTOR

  LINK_ATTRIBUTE

- **LinkLocalId** (*int*) -- 链路本端 ID, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **LinkRemoteId** (*int*) -- 链路远端 ID, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **RemoteNodeDescriptorFlag** (*str*) -- 远端节点描述符标志, 类型为：string, 默认值：IGP_ROUTER_ID, 取值范围：

  AS_NUMBER

  BGPLS_IDENTIFIER

  OSPF_AREA_ID

  IGP_ROUTER_ID

  BGP_ROUTER_ID

  MEMBER_ASN

- **AsNumber** (*int*) -- 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **BgpLsIdentifier** (*str*) -- BGP-LS 标识符, 类型为：有效的 ipv4 地址, 默认值：1.0.0.1
- **OspfAreaId** (*str*) -- OSPF 区域 ID, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **IGPRouterIdType** (*str*) -- IGP 路由 ID 类型, 类型为：string, 默认值：OSPF_NONPSEUDONODE, 取值范围：

  ISIS_NONPSEUDONODE

  ISIS_PSEUDONODE

  OSPF_NONPSEUDONODE

  OSPF_PSEUDONODE

- **IsisNonPseud** (*str*) -- ISIS 非伪节点路由 ID, 类型为：string, 默认值：”00:10:12:00:00:01”

- **IsisPseud** (*str*) -- OSPF 伪节点路由 ID, 类型为: string, 默认值: ”00:10:12:00:00:01.02”
- **OspfNonPseud** (*str*) -- OSPF 非伪节点路由 ID, 类型为: 有效的 ipv4 地址, 默认值: 192.0.0.1
- **OspfPseud** (*str*) -- OSPF 伪节点路由 ID, 类型为: string, 默认值: ”192.0.0.1:192.0.0.1”
- **BgpRouterId** (*str*) -- BGP 路由 ID, 类型为: 有效的 ipv4 地址, 默认值: 0.0.0.0
- **MemberAsn** (*int*) -- 自治域成员, 类型为: number, 取值范围: 1-1048575, 默认值: 0
- **LinkDescriptorFlag** (*list*) -- 链路描述符标志, 类型为: list, 默认值: UNKNOWN, 取值范围:

  UNKNOWN

  LINK_IDENTIFIES

  IPV4_INTERFACE

  IPV4_NEIGHBOR

  IPV6_INTERFACE

  IPV6_NEIGHBOR

  MULTI_TOPOLOGY
- **Ipv4InterfaceAddr** (*str*) -- IPv4 接口地址, 类型为: 有效的 ipv4 地址, 默认值: 192.168.0.1
- **Ipv4NeighborAddr** (*str*) -- IPv4 邻接地址, 类型为: 有效的 ipv4 地址, 默认值: 192.168.0.2
- **Ipv6InterfaceAddr** (*str*) -- IPv6 接口地址, 类型为: 有效的 ipv6 地址, 默认值: 2001::1
- **Ipv6NeighborAddr** (*str*) -- IPv6 邻接地址, 类型为: 有效的 ipv6 地址, 默认值: 2001::2
- **MultiTopologyId** (*int*) -- 多拓扑 ID, 类型为: number, 取值范围: 1-4095, 默认值: “”
- **LinkAttributeFlag** (*list*) -- 链路属性标志, 类型为: list, 默认值: UNKNOWN, 取值范围:

  UNKNOWN

  IPV4_LOCAL_NODE

  IPV4_REMOTE_NODE

  IPV6_LOCAL_NODE

  IPV6_REMOTE_NODE

  LINK_IDENTIFIES

  LINK_PROTECTION_TYPE

  IGP_METRIC

  SHARED_RISE

  ADI_SID

  LAN_ADJ_SID

  PEERNODE_SID

PEERADJ_SID

PEERSET_SID

SRV6_LINK_MSD

SRV6_END_X_SID

SRV6_LAN_END_X_SID

- **LocalIpv4RouterIds** (*str*) -- 本端 IPv4 路由 IDs, 类型为：有效的 ipv4 地址, 默认值：""
- **RemoteIpv4RouterIds** (*str*) -- 远端 IPv4 路由 IDs, 类型为：有效的 ipv4 地址, 默认值：""
- **LocalIpv6RouterIds** (*str*) -- 本端 IPv6 路由 IDs, 类型为：有效的 ipv6 地址, 默认值：""
- **RemoteIpv6RouterIds** (*str*) -- 远端 IPv6 路由 IDs, 类型为：有效的 ipv6 地址, 默认值：""
- **IgpMetricType** (*str*) -- IGP 度量类型, 类型为：string, 默认值：UNKNOWN, 取值范围：

  ISIS_NARROW

  OSPFv2_LINK

  ISIS_WIDE

- **IgpMetricValue** (*int*) -- IGP 度量值, 类型为：number, 取值范围：IS-IS Narrow Metrics：0-255, OSPFv2 Link Metrics：0-65535, IS-IS Wide Metrics：0-16777215, 默认值：0
- **LinkProteType** (*list*) -- 链路保护类型, 类型为：list, 默认值：UNKNOWN, 取值范围：

  EXTRA_TRAFFIC

  UNPROTECTED

  SHARED

  DEDICATED1

  DEDICATED2

  ENHANCED

  RESERVED4

  RESERVED8

- **ShareRiskGroup** (*int*) -- SRLG 信息, 类型为：number, 取值范围：1-4294967295, 默认值：""
- **OspfAdjSidFlag** (*list*) -- OSPF SR 邻接 SID 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  BACKUP

  VALUE

  LOCAL

  GROUP

  PERSISTENT

- **IsisAdjSidFlag** (*list*) -- ISIS SR 邻接 SID 标志, 类型为：list, 默认值：NOSHOW, 取值范围：

  NOSHOW

  ADDRESS

  BACKUP

  VALUE

  LOCAL

  SET

  PERSISTENT

- **AdjSidWeight** (*int*) -- 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **AdjSidLabel** (*int*) -- SID/Label, 类型为：number, 取值范围：1-255, 默认值：16
- **OspfNeighborId** (*str*) -- OSPF 邻居 ID, 类型为：有效的 ipv4 地址, 默认值：192.0.1.0
- **IsisSystemId** (*str*) -- ISIS 系统 ID, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **PeerNodeFlag** (*list*) -- Peer Node 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  VALUE

  LOCAL

  BACKUP

  PERSISTENT

- **PeerNodeWeight** (*int*) -- Peer Node 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **PeerNodeSidLabel** (*int*) -- Peer Node SID/Index/Label, 类型为：number, 取值范围：1-255, 默认值：16
- **PeerAdjFlag** (*list*) -- Peer Adj 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  VALUE

  LOCAL

  BACKUP

  PERSISTENT

- **PeerAdjWeight** (*int*) -- Peer Adj 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **PeerAdjSidLabel** (*int*) -- Peer Adj 权重, 类型为：number, 取值范围：1-255, 默认值：16
- **PeerSetFlag** (*list*) -- Peer Set 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  VALUE

LOCAL

BACKUP

PERSISTENT

- **PeerSetWeight** (*int*) -- Peer Set 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **PeerSetSidLabel** (*int*) -- Peer Set SID/Index/Label, 类型为：number, 取值范围：1-255, 默认值：16
- **Srv6MsdFlags** (*list*) -- SRv6 链路 MSD 类型, 类型为：list, 默认值：NONE, 取值范围：

  NONE

  MAX_SEGMENTS_LELT

  MAX_END_POP

  MAX_T_INSERT_SRH

  MAX_T_ENCAPS_SRH

  MAX_END_D_SRH
- **Srv6MsdMaxSegmentLeft** (*int*) -- 最大 Segments Left, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEndPop** (*int*) -- 最大 End Pop, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxInsert** (*int*) -- 最大 T.Insert SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEncap** (*int*) -- 最大 T.Encaps SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6MsdMaxEndD** (*int*) -- 最大 End D SRH, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6EndXSidEndpointBehavior** (*str*) -- SRv6 Endpoint 功能指令类型, 类型为：string, 默认值：CUSTOM, 取值范围：

  END

  END_WITH_PSP

  END_WITH_USP

  END_WITH_PSP_USP

  END_X

  END_X_WITH_PSP

  END_X_WITH_USP

  END_X_WITH_PSP_USP

  END_T

  END_T_WITH_PSP

  END_T_WITH_USP

  END_T_WITH_PSP_USP

  END_B6_ENCAPS

  END_BM

  END_DX6

END_DX4

END_DT6

END_DT4

END_DT46

END_DX2

END_DX2V

END_DT2U

END_DT2M

END_B6_ENCAPS_RED

END_WITH_USD

END_WITH_PSP_USD

END_WITH_USP_USD

END_WITH_PSP_USP_USD

END_X_WITH_USD

END_X_WITH_PSP_USD

END_X_WITH_USP_USD

END_X_WITH_PSP_USP_USD

END_T_WITH_USD

END_T_WITH_PSP_USD

END_T_WITH_USP_USD

END_T_WITH_PSP_USP_USD

CUSTOM

- **Srv6EndXSidFlag** (*list*) -- SRv6 End.X SID 标志, 类型为：list, 默认值：NONE, 取值范围：

  NONE

  BACKUP_FLAG

  SET_FLAG

  PERSISTENT_FLAG

- **Srv6EndXSidAlgorithm** (*int*) -- SRv6 End.X SID 算法, 类型为：number, 取值范围：1-255, 默认值：8
- **Srv6EndXSidWeight** (*int*) -- SRv6 End.X SID 权重, 类型为：number, 取值范围：1-255, 默认值：1
- **Srv6EndXSidSid** (*str*) -- SRv6 End.X SID, 类型为：有效的 ipv6 地址, 默认值：::1
- **Srv6LanEndXSidSystemId** (*str*) -- ISIS 邻居 ID, 类型为：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **Srv6LanEndXSidRouterId** (*str*) -- OSPFv3 邻居 ID, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **EnableInterfaceIp** (*bool*) -- 启用本端 IPv4 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InterfaceIp** (*str*) -- 本端 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0

- **EnableNeighborIp** (*bool*) -- 启用远端 IPv4 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NeighborIp** (*str*) -- 远端 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：0.0.0.0
- **EnableInterfaceIpv6** (*bool*) -- 启用本端 IPv6 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InterfaceIpv6** (*str*) -- 本端 IPv6 地址, 类型为：有效的 ipv4 地址, 默认值：2000::1
- **EnableNeighborIpv6** (*bool*) -- 启用远端 IPv6 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NeighborIpv6** (*str*) -- 启用远端 IPv6 地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **EnableGroup** (*bool*) -- 启用组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Group** (*int*) -- 组, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **EnableUniLinkLoss** (*bool*) -- 启用单向链路损耗, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LinkLoss** (*int*) -- 链路损耗, 类型为：number, 取值范围：1-100, 默认值：3
- **LinkLossAflag** (*bool*) -- 链路损耗的 A-flag, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableUniDelay** (*bool*) -- 启用单向延迟, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniDelay** (*int*) -- 单向延迟, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UniAflag** (*bool*) -- 单向延迟的 A-flag, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableUniMinMaxDelay** (*bool*) -- 启用单向延迟最小/最大值, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniMinMaxAflag** (*bool*) -- 启用单向延迟最小/最大值的 A-flag 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniMinDelay** (*int*) -- 单向最小延迟, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UniMaxDelay** (*int*) -- 单向最大延迟, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUniDelayVariation** (*bool*) -- 启用单向延迟变化, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniVarDelay** (*int*) -- 单向延迟变化, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUniResidual** (*bool*) -- 启用单向剩余带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniResBandwidth** (*int*) -- 单向剩余带宽, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUniAva** (*bool*) -- 启用单向可用带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniAvaBandwidth** (*int*) -- 单向可用带宽, 类型为：number, 取值范围：1-4294967295, 默认值：100000

- **EnableUniUtilized** (*bool*) -- 启用单向已用带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UniUtilized** (*int*) -- 单向已用带宽, 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableMaximum** (*bool*) -- 启用最大带宽 (字节/秒), 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Maximum** (*int*) -- 最大带宽 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableReservable** (*bool*) -- 启用预留带宽 (字节/秒), 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Reservable** (*int*) -- 预留带宽 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableUnreserved** (*bool*) -- 启用未预留带宽优先级 (字节/秒), 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UnreservedBandwidth0** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth1** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth2** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth3** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth4** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth5** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth6** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **UnreservedBandwidth7** (*int*) -- 未预留带宽优先级 (字节/秒), 类型为：number, 取值范围：1-4294967295, 默认值：100000
- **EnableTeDefaultMetric** (*bool*) -- 启用 TE 默认度量, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **TeDefaultValue** (*int*) -- TE 默认度量, 类型为：number, 取值范围：1-4294967295, 默认值：0

返回 Bgp Link States Link 对象, 类型：object / list

返回类型 (BgpLsLinkConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${LinkState} | Create Bgp Link States | Sessions=${Session} |
| Create Bgp Link States Link | Sessions=${Session} | LinkState=$
↪{LinkState} |
```

static create_bgp_link_states_prefix(*Session*, *LinkState*, ***kwargs*)

创建 Bgp Link States Prefix 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **LinkState** (BgpLsNodeConfig) -- Bgp Link States 对象列表, 类型为：object / list

关键字参数

- **PrefixDescriptorFlag** (*list*) -- LS 前缀描述符标志, 类型为：list, 默认值：IP_REACH_INFO, 取值范围：

  MULTI_TOPOLOGY

  OSPF_ROUTE_TYPE

  IP_REACH_INFO

- **OspfRouteType** (*str*) -- OSPF 路由类型, 类型为：string, 默认值：INTRA_AREA, 取值范围：

  INTRA_AREA

  INTER_AREA

  EXTERNAL1

  EXTERNAL2

  NSSA1

  NSSA2

- **PrefixCount** (*int*) -- 地址前缀个数, 类型为：number, 取值范围：1-65535, 默认值：1
- **PrefixType** (*str*) -- 地址前缀类型, 类型为：string, 默认值：IPV4, 取值范围：

  IPV4

  IPV6

- **Ipv4Prefix** (*str*) -- IPv4 地址前缀, 类型为：有效的 ipv4 地址, 默认值：1.0.0.0
- **Ipv4PrefixLength** (*int*) -- IPv4 地址前缀步长, 类型为：number, 取值范围：1-32, 默认值：24
- **Ipv4PrefixStep** (*int*) -- IPv4 地址前缀步长, 类型为：number, 取值范围：1-65535, 默认值：1
- **Ipv6Prefix** (*str*) -- IPv6 地址前缀, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Ipv6PrefixLength** (*int*) -- IPv6 地址前缀步长, 类型为：number, 取值范围：1-128, 默认值：64
- **Ipv6PrefixStep** (*int*) -- IPv6 地址前缀长度, 类型为：number, 取值范围：1-65535, 默认值：1
- **MultiTopologyId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-4095, 默认值：“”
- **OspfSrPrefixSidFlag** (*list*) -- OSPF SR 前缀 SID 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  NO_PHP

  MAPPING_SERVER

  EXPLICIT_NULL

VALUE

LOCAL

- **IsisSrPrefixSidFlag** (*list*) -- ISIS SR 前缀 SID 标志, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  RE_ADVER

  NODESID

  NO_PHP

  EXPLICIT_NULL

  VALUE

  LOCAL

- **PrefixAttributeFlag** (*str*) -- LS 前缀描述符标志, 类型为：string, 默认值：UNKNOWN|IGP_FLAGS, 取值范围：

  UNKNOWN

  IGP_FLAGS

  PREFIX_METRIC

  OSPF_FORWARDING

  SR_PREFIX_SID

  SR_RANGE

  SR_ATTRIBUTE_FLAG

  SR_SOURCE

  SRV6_LOCATOR_TLV

- **IgpFlag** (*str*) -- IGP 标志, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  ISIS_UP_DOWN

  OSPF_NO_UNICAST

  OSPF_LOCAL_ADDRESS

  OSPF_PROPAGATE_NSSA

- **PrefixMetric** (*int*) -- 前缀度量, 类型为：number, 取值范围：1-4294967295, 默认值：0

- **OspfForwardtype** (*str*) -- OSPF 转发地址类型, 类型为：string, 默认值：OSPFV2, 取值范围：

  OSPFV2

  OSPFV3

- **Ospfv2ForwardAddr** (*str*) -- OSPFv2 转发地址, 类型为：有效的 ipv4 地址, 默认值：192.168.1.1

- **Ospfv3ForwardAddr** (*str*) -- OSPFv3 转发地址, 类型为：有效的 ipv6 地址, 默认值：2000::1

- **OspfSrPrefixFlag** (*str*) -- OSPF SR 前缀 SID 标志, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  NO_PHP

  MAPPING_SERVER

  EXPLICIT_NULL

  VALUE

  LOCAL

- **IsisSrPrefixFlag** (*str*) -- ISIS SR 前缀 SID 标志, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  RE_ADVER

  NODESID

  NO_PHP

  EXPLICIT_NULL

  VALUE

  LOCAL

- **Algorithm** (*int*) -- 算法, 类型为：number, 取值范围：1-255, 默认值：0

- **SidLabelIndex** (*int*) -- SID/Label/Index, 类型为：number, 取值范围：0-1048575 (20 比特值), 0-4294967295 (32 比特值), 默认值：0

- **Ospfv2SrPrefixAttributeFlag** (*list*) -- OSPFv2 SR 前缀属性标志, 类型为：list, 默认值：NODE, 取值范围：

  ATTACH

  NODE

- **Ospfv3SrPrefixAttributeFlag** (*list*) -- OSPFv3 SR 前缀属性标志, 类型为：list, 默认值：NO_UNICAST, 取值范围：

  NO_UNICAST

  LOCAL_ADDRESS

  MULTICAST

  PROPAGATE

  RE_ADVER

  HOST

- **IsisSrPrefixAttributeFlag** (*list*) -- ISIS SR 前缀属性标志, 类型为：list, 默认值：NODE, 取值范围：

  EXTERNAL_PREFIX

  RE_ADVER

  NODE

- **SrSourceIpv4Id** (*str*) -- SR IPv4 源路由 ID, 类型为：有效的 ipv4 地址, 默认值：192.168.1.0

- **SrSourceIpv6Id** (*str*) -- SR IPv6 源路由 ID, 类型为：有效的 ipv6 地址, 默认值：2000::1

- **Ospfv2SrRangeFlag** (*list*) -- OSPFv2 SR 范围标志, 类型为：list, 默认值：INTER_AREA, 取值范围：

  INTER_AREA

- **IsisSrRangeFlag** (*list*) -- ISIS SR 范围标志, ISIS SR 前缀 SID 标志, 类型为：list, 默认值：ATTACHED, 取值范围：

  ADDRESS_FAMILY

  MIRROR_CONTEXT

  S_FLAG

  D_FLAG

  ATTACHED

- **SrRangeSubTlv** (*str*) -- SR 范围 Sub-TLVs, 类型为：string, 默认值：”0 Range Sub-TLV”

- **Srv6LocatorFlag** (*str*) -- SRv6 Locator 标志, 类型为：string, 默认值：NONE, 取值范围：

  NONE

  D_FLAG

- **Srv6LocatorAlgorithm** (*int*) -- SRv6 Locator 算法, 类型为：number, 取值范围：1-255, 默认值：0

- **Srv6LocatorMetric** (*int*) -- SRv6 Locator 度量值, 类型为：number, 取值范围：1-4294967295, 默认值：0

返回 Bgp Link States Prefix 对象, 类型：object / list

返回类型 (BgpLsPrefixConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${LinkState} | Create Bgp Link States | Sessions=${Session} |
| Create Bgp Link States Prefix | Sessions=${Session} | LinkState=$
↪{LinkState} |
```

static create_bgp_link_states_prefix_sr_range_sub_tlv(*Session*, *LinkStatePrefix*, ***kwargs*)

创建 Bgp Link States Prefix Sr Range Sub Tlv 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **LinkStatePrefix** (BgpLsPrefixConfig) -- Bgp Link States prefix 对象列表, 类型为：object / list

关键字参数

- **Algorithm** (*int*) -- 算法, 类型为：number, 取值范围：1-255, 默认值：0
- **OspfSrPrefixSidFlag** (*list*) -- OSPF SR Prefix SID Flags, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  NO_PHP

  MAPPING_SERVER

EXPLICIT_NULL

VALUE

LOCAL

- **IsisSrPrefixSidFlag** (*list*) -- ISIS SR Prefix SID Flags, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  RE_ADVER

  NODESID

  NO_PHP

  EXPLICIT_NULL

  VALUE

  LOCAL

- **SidLabelIndex** (*int*) -- SID/Label/Index, 类型为：number, 取值范围：1-255, 默认值：0

返回 Bgp Link States Prefix Sr Range Sub Tlv 对象, 类型：object / list

返回类型 (BgpLsSrRangeSubTlvConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${LinkState} | Create Bgp Link States | Sessions=${Session} |
| ${prefix} | Create Bgp Link States Prefix | LinkState=${LinkState} |
| Create Bgp Link States Prefix Sr Range Sub Tlv | Sessions=${Session} |␣
↪LinkStatePrefix=${prefix} |
```

static create_bgp_link_states_srv6_sid(*Session*, *LinkState*, ***kwargs*)

创建 Bgp Link States Srv6 Sid 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **LinkState** (BgpLsNodeConfig) -- Bgp Link States 对象列表, 类型为：object / list

关键字参数

- **Srv6AttributeFlag** (*list*) -- SRv6 SID 属性标志, 类型为：list, 默认值：SRV6_ENDPOINT_BEHAVIOR, 取值范围：

  SRV6_ENDPOINT_BEHAVIOR

  SRV6_BGP_PEER_NODE_SID

  SRV6_SID_STRUCTURE

- **Srv6EndpointBehavior** (*str*) -- SRv6 Endpoint 功能指令类型, 类型为：string, 默认值：CUSTOM, 取值范围：

  END

  END_WITH_PSP

  END_WITH_USP

  END_WITH_PSP_USP

END_X

END_X_WITH_PSP

END_X_WITH_USP

END_X_WITH_PSP_USP

END_T

END_T_WITH_PSP

END_T_WITH_USP

END_T_WITH_PSP_USP

END_B6_ENCAPS

END_BM

END_DX6

END_DX4

END_DT6

END_DT4

END_DT46

END_DX2

END_DX2V

END_DT2U

END_DT2M

END_B6_ENCAPS_RED

END_WITH_USD

END_WITH_PSP_USD

END_WITH_USP_USD

END_WITH_PSP_USP_USD

END_X_WITH_USD

END_X_WITH_PSP_USD

END_X_WITH_USP_USD

END_X_WITH_PSP_USP_USD

END_T_WITH_USD

END_T_WITH_PSP_USD

END_T_WITH_USP_USD

END_T_WITH_PSP_USP_USD

CUSTOM

- **Srv6EndpointBehaviorFlag** (*list*) -- SRv6 Endpoint 功能指令标志, 类型为：list, 默认值：NONE, 取值范围：

  NONE

  UNUSED0

  UNUSED1

  UNUSED2

UNUSED3

UNUSED4

UNUSED5

UNUSED6

UNUSED7

- **Srv6EndpointBehaviorAlgorithm** (*int*) -- SRv6 Endpoint 功能指令算法, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BgpPeerNodeSidFlag** (*list*) -- EPE Peer Node SID 标志, 类型为：list, 默认值：0, 取值范围：

  NONE

  BACKUP_FLAG

  SET_FLAG

  PERSISTENT_FLAG

- **Srv6BgpPeerNodeSidWeight** (*int*) -- EPE Peer Node SID 权重, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BgpPeerNodeSidPeerAsNumber** (*int*) -- Peer 自治域, 类型为：number, 取值范围：1-4294967295, 默认值：1001
- **Srv6BgpPeerNodeSidPeerBgpId** (*int*) -- Peer BGP 标识符, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **Srv6SidStructureLbLength** (*int*) -- SRv6 Locator Block 长度, 类型为：number, 取值范围：1-255, 默认值：32
- **Srv6SidStructureLnLength** (*int*) -- SRv6 Locator Node 长度, 类型为：number, 取值范围：1-255, 默认值：32
- **Srv6SidStructureFunLength** (*int*) -- SRv6 Function 长度, 类型为：number, 取值范围：1-255, 默认值：32
- **Srv6SidStructureArgLength** (*int*) -- SRv6 Argument 长度, 类型为：number, 取值范围：1-255, 默认值：32
- **Srv6Sid** (*str*) -- 类型为：有效的 ipv6 地址, 默认值：::1
- **MultiTopologyId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-4095, 默认值：“”

返回 Bgp Link States Srv6 Sid 对象, 类型：object / list

返回类型 (BgpLsSrv6SidConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${LinkState} | Create Bgp Link States | Sessions=${Session} |
| Create Bgp Link States Srv6 Sid | Sessions=${Session} | LinkState=$
↪{LinkState} |
```

static create_bgp_random_route(*PortNumber*, *Type=None*, *Ipv4RouteNumber=300000*, *Ipv6RouteNumber=100000*, *Ipv4StartMask=23*, *Ipv6StartMask=32*, *Ipv4StartRoute='60.0.0.0'*, *Ipv6StartRoute='60::'*, ***kwargs*)

static create_bgp_route_pool_custom_path_attribute(*RoutePool*, ***kwargs*)

创建 BGP Route Pool Custom Path Attribute 对象, 类型为：object / list

参数 **RoutePool** (BgpIpv4RoutepoolConfig) -- 所属的 Bgp Route Pool 对象, 类型为：object / list

关键字参数

- **PathAttributeType** (*int*) -- 路径属性的类型, 类型为：number, 取值范围：1-255, 默认值：1
- **OptionalFlag** (*str*) -- 指定 Optional Flag 的值, 类型为：string, 默认值：OPTION, 取值范围：

  WELL_KNOWN

  OPTION
- **TransitiveFlag** (*str*) -- 指定 Transitive Flag 的值, 类型为：string, 默认值：NONTRANSITIVE, 取值范围：

  NONTRANSITIVE

  TRANSITIVE
- **PartialFlag** (*str*) -- 指定 Partial Flag 的值, 类型为：string, 默认值：PARTIAL, 取值范围：

  COMPLETE

  PARTIAL
- **ExtendedLengthFlag** (*bool*) -- 是否启用 Extended Length Flag, 类型为：bool, 默认值：False
- **AttributeExtendedLength** (*int*) -- 指定路径属性的长度, 类型为：number, 默认值：0
- **AttributeValue** (*str*) -- 指定路径属性的值, 类型为：string, 默认值：""

返回 Bgp Route Pool Custom Path Attribute 对象, 类型：object / list

返回类型 (BgpPathAttributeConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${RoutePool} | Create Bgp Ipv4 Route Pool | Session=${Session} |
| ${AttributeValue} | Set Variable |␣
↪0500220001001E00AAAA000100010001000000000000000100001300010006301014000000␣
↪|
| Create Bgp Route Pool Custom Path Attribute | RoutePool=${RoutePool} |␣
↪AttributeValue=${AttributeValue} |
```

static create_bgp_segement_sub_tlv(*Session*, *SegementList*, *Types*, ***kwargs*)

创建 Bgp Segement Sub Tlv 对象, 类型为：object

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **SegementList** (BgpSrTePolicySegmentList) -- BGP Sr Te Policy Segement List 对象, 类型为：object
- **Types** (*str*) -- BGP Sr Te Policy Segement Sub Tlv 类型, 类型为 string, 默认值: A, 取值范围：A-K

关键字参数

- **Flags** (*list*) -- 选择一个或多个 Flag, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  V_FLAG

  A_FLAG

  S_FLAG

  B_FLAG

- **Label** (*int*) -- 指定标签, 类型为：number, 取值范围：1-1048575, 默认值：1600
- **LabelStep** (*int*) -- 指定标签跳变, 类型为：number, 取值范围：1-1048575, 默认值：1
- **Tc** (*int*) -- 指定 TC（Traffic Class，通信量类）值, 类型为：number, 取值范围：0-7, 默认值：0
- **Sbit** (*int*) -- 指定栈底标志（S）的值, 类型为：number, 取值范围：0-1, 默认值：1
- **Ttl** (*int*) -- 指定 TTL 值, 类型为：number, 取值范围：1-1048575, 默认值：255
- **Srv6Sid** (*str*) -- 指定 SRv6 SID, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Srv6SidStep** (*str*) -- 指定 SRv6 SID 的跳变步长, 类型为：有效的 ipv6 地址, 默认值：::1
- **EndpointBehavior** (*int*) -- 指定 SRv6 Endpoint Behavior, 类型为：number, 取值范围：1-1048575, 默认值：0
- **LbLength** (*int*) -- SRv6 SID Locator Block 长度, 类型为：number, 取值范围：1-1048575, 默认值：0
- **LnLength** (*int*) -- SRv6 SID Locator Node 长度, 类型为：number, 取值范围：1-1048575, 默认值：0
- **FunLength** (*int*) -- SRv6 SID Function 长度, 类型为：number, 取值范围：1-1048575, 默认值：0
- **ArgLength** (*int*) -- SRv6 SID 参数长度, 类型为：number, 取值范围：1-1048575, 默认值：0
- **SrAlgorithm** (*int*) -- SR 算法, 类型为：number, 取值范围：1-1048575, 默认值：0
- **Ipv4NodeAddress** (*str*) -- IPv4 节点地址, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv4NodeAddressStep** (*str*) -- IPv4 节点地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **Ipv6NodeAddress** (*str*) -- IPv6 节点地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Ipv6NodeAddressStep** (*str*) -- IPv6 节点地址跳变, 类型为：有效的 ipv6 地址, 默认值：::1
- **LocalInterfaceId** -- 本地 Interface ID, 类型为：number, 取值范围：1-1048575, 默认值：0
- **LocalIpv4Address** (*str*) -- IPv4 节点地址, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1

- **LocalIpv4AddressStep** (*str*) -- IPv4 节点地址跳变, 类型为: 有效的 ipv4 地址, 默认值：0.0.0.1
- **RemoteIpv4Address** (*str*) -- 远端 IPv4 地址, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **RemoteIpv4AddressStep** (*str*) -- 远端 IPv4 地址跳变, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **LocalIpv6NodeAddress** (*str*) -- IPv6 本地节点地址, 类型为: 有效的 ipv6 地址, 默认值：2000::1
- **LocalIpv6NodeAddressStep** (*str*) -- IPv6 本地节点地址跳变, 类型为: 有效的 ipv6 地址, 默认值：::1
- **RemoteInterfaceId** (*int*) -- 远端 Interface ID, 类型为：number, 取值范围：1-1048575, 默认值：0
- **RemoteIpv6NodeAddress** (*str*) -- IPv6 远端节点地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **RemoteIpv6NodeAddressStep** (*str*) -- IPv6 远端节点地址跳变, 类型为: 有效的 ipv6 地址, 默认值：::1

返回 Bgp Segement Sub Tlv 对象, 类型：object / list

返回类型 (BgpSegmentSubTlvTypeA)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${SrTe} | Create Bgp Sr Te Policy | Session=${Session} |
| ${SrTeSegmentList} | Create Bgp Sr Te Policy Segement List | Session=$
↪{Session} | SrTePolicys=${SrTe} |
| Create Bgp Segement Sub Tlv | Session=${Session} | SrTePolicys=${SrTe}␣
↪| Types=B |
```

static create_bgp_sr_te_policy(*Session*, ***kwargs*)

创建 Bgp Sr Te Policy 对象, 类型为：object / list

参数 **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

关键字参数

- **BindingSidCount** (*int*) -- Binding SID 数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **PolicyColor** (*int*) -- 指定 SR Policy 的起始 Color 值, 类型为：number, 默认值：0
- **PolicyColorStep** (*int*) -- 指定 Policy Color 的跳变步长, 类型为：number, 默认值：1
- **IpVersion** (*str*) -- 指定 IP 前缀类型, 类型为：string, 默认值：IPV4, 取值范围：

  IPV4

  IPV6
- **EndpointCount** (*int*) -- 指定目的节点的数量。该参数值不能大于 Binding SID 数量。类型为：number, 取值范围：1-4294967295, 默认值：1
- **Ipv4Endpoint** (*str*) -- 指定 Policy 块中目的节点的起始 IP 地址, 类型为: 有效的 ipv4 地址, 默认值：192.0.0.1

- **Ipv4EndpointStep** (*str*) -- 指定 Policy 块中目的节点的 IP 地址的跳变步长, 类型为：有效的 ipv4 地址, 默认值：0.0.0.1
- **Ipv6Endpoint** (*str*) -- 指定 Policy 目的节点的 IP 地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Ipv6EndpointStep** (*str*) -- 指定 Policy 目的节点的 IP 地址, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **EndpointIncrementMode** (*str*) -- 指定 Policy 块中目的节点 IP 地址的生成方式, 类型为：string, 默认值：RoundRobin, 取值范围：

  RoundRobin

  Sequential
- **Origin** (*str*) -- 指定 ORIGIN 的值, 类型为：string, 默认值：IGP, 取值范围：

  IGP

  EGP

  INCOMPLETE
- **AsPath** (*str*) -- 指定 AS 路径的值, 类型为：string, 默认值：””
- **AsPathType** (*str*) -- AS Path 类型, 类型为：string, 默认值：SEQUENCE, 取值范围：

  SET

  SEQUENCE

  CONFED_SEQUENCE

  CONFED_SET
- **LocalPref** (*int*) -- 指定 Local_PREF 的值, 类型为：number, 默认值：10
- **UseSessionAddressAsNextHop** (*bool*) -- 使用会话地址作为下一跳地址, 类型为：bool, 默认值：True
- **Ipv4NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv6NextHop** (*str*) -- 下一跳地址，即 UPDATE 消息中 NEXT_HOP 的值, 类型为：有效的 ipv6 地址, 默认值：2001::1
- **Distinguisher** (*int*) -- 输入 SR Policy 标识, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **RouteTarget** (*str*) -- 指定重定向的路由目标。类型为：string, 默认值：””
- **Community** (*str*) -- 当 Type 为 NO_EXPORT 时，团体值为 0xffffff01；当 Type 为 NO_ADVERTISE 时，团体值为 0xffffff02；当 Type 为 LOCAL_AS 时，团体值为 0xffffff03, 类型为：string, 默认值：””
- **ExtendedCommunity** (*str*) -- 扩展团体，类型为：string，默认值：”0x03:0x0b:0:0”
- **SrTePolicySubTlv** (*list*) -- 选择一个或多个 TLV, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  REMOTE_ENDPOINT

  COLOR

  PREFERENCE

  BINDING_SID

ENLP

PRIORITY

EGRESS_ENDPOINT

POLICY_CP_NAME

SRV6_BSID

POLICY_NAME

- **RemoteEndpointAsn** (*int*) -- 远端 Endpoint 自治域号, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **Ipv4RemoteEndpoint** (*str*) -- IPv6 远端 Endpoint, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv6RemoteEndpoint** (*str*) -- IPv6 远端 Endpoint, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Color** (*int*) -- 指定 Color 扩展团体中 Color Value 字段的值, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **ColorFlags** (*list*) -- 指定 Color 扩展团体中 Flags 字段的值, 类型为：list, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  C_FLAG

  O_FLAG
- **Preference** (*int*) -- 指定 SR Policy 中候选路径的优先级, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **BsidFlags** (*str*) -- Binding SID Flags, 类型为：string, 默认值：NO_SHOW, 取值范围：

  NO_SHOW

  S_FLAG

  I_FLAG
- **BsidLength** (*str*) -- Binding SID 长度, 类型为：string, 默认值：NO_SHOW, 取值范围：

  OCTET_0

  OCTET_4

  OCTET_16
- **BsidLabel** (*int*) -- Binding SID 标签, 类型为：number, 取值范围：1-4294967295, 默认值：0
- **BsidLabelStep** (*int*) -- Binding SID 标签跳变, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **BsidTc** (*int*) -- 指定 BSID 中的 S 字段值, 类型为：number, 取值范围：0-7, 默认值：0
- **BsidS** (*int*) -- 指定 BSID 中的 S 字段值, 类型为：number, 取值范围：0-1, 默认值：0
- **BsidTtl** (*int*) -- 指定 BSID 中的 TTL 字段值, 类型为：number, 默认值：0
- **Ipv6Bsid** (*str*) -- 指定 IPv6 BSID, 类型为：有效的 ipv6 地址, 默认值：2000::1

- **Ipv6BsidStep** (*str*) -- 指定 IPv6 BSID 的跳变步长, 类型为：有效的 ipv6 地址, 默认值：::1
- **Enlp** (*str*) -- 指定显式空标签策略, 类型为：string, 默认值：IPV4, 取值范围：

  RESERVED0

  IPV4

  IPV6
- **PolicyPriority** (*int*) -- 指定拓扑改变后重新计算 SR Policy 时的 Policy 优先级, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BsidFlags** (*list*) -- 指定 SRv6 Binding SID sub-TLV 中包含的 Flag, 类型为：list, 默认值：NO_SHOW(0x0), 取值范围：

  NO_SHOW

  S_FLAG

  I_FLAG

  B_FLAG
- **Srv6Bsid** (*str*) -- 指定 SRv6 BSID, 类型为：有效的 ipv6 地址, 默认值：2000::1
- **Srv6BsidStep** (*str*) -- 指定 SRv6 BSID 的跳变步长, 类型为：有效的 ipv6 地址, 默认值：::1
- **Srv6BsidEndpointBehavior** (*int*) -- 指定 SRv6 SID Endpoint Behavior, 类型为：number, 取值范围：1-65535, 默认值：0
- **Srv6BsidLbLength** (*int*) -- 指定 SRv6 SID Function 长度。类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BsidLnLength** (*int*) -- 指定 SRv6 SID Locator Node 长度。单位：比特, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BsidFunLength** (*int*) -- 指定 SRv6 SID Function 长度。单位：比特, 类型为：number, 取值范围：1-255, 默认值：0
- **Srv6BsidArgLength** (*int*) -- 指定 SRv6 SID 参数长度。单位：比特, 类型为：number, 取值范围：1-255, 默认值：0
- **CandidatePathName** (*str*) -- Policy 候选路径名称, 类型为：string, 默认值：""
- **PolicyName** (*str*) -- 指定 Policy 的名称, 类型为：string, 默认值：""
- **TunnelEgressEndpointAfi** (*str*) -- 隧道出口端点 AFI, 类型为：string, 默认值：IPV4, 取值范围：

  RESERVED0

  IPV4

  IPV6
- **Ipv4TunnelEgressEndpoint** (*str*) -- IPv4 远端 Endpoint, 类型为：有效的 ipv4 地址, 默认值：192.0.0.1
- **Ipv6TunnelEgressEndpoint** (*str*) -- IPv6 远端 Endpoint, 类型为：有效的 ipv6 地址, 默认值：2000::1

返回 Bgp Sr Te Policy 对象, 类型：object / list

返回类型 (BgpSrTePolicyConfig)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| Create Bgp Sr Te Policy | Session=${Session} |
```

static create_bgp_sr_te_policy_Segement_list(*Session*, *SrTePolicy*, ***kwargs*)

创建 Bgp Sr Te Policy Segement List 对象, 类型为：object / list

参数

- **Session** (BgpRouter) -- BGP 协议会话对象列表, 类型为：object / list
- **SrTePolicy** (BgpSrTePolicyConfig) -- BGP Sr Te Policy 对象, 类型为：object / list

关键字参数

- **SubTlvs** (*list*) -- 选择一个或多个子 TLV, 类型为:list, 默认值:NO_SHOW, 取值范围:

  NO_SHOW

  WEIGHT

- **Weight** (*int*) -- 指定 Segment List（SID 列表）对应的权重, 类型为: number, 取值范围：1-4294967295, 默认值：1

返回 Bgp Sr Te Policy Segement List 对象, 类型：object / list

返回类型 (BgpSrTePolicySegmentList)

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${SrTe} | Create Bgp Sr Te Policy | Session=${Session} |
| Create Bgp Sr Te Policy Segement List | Session=${Session} | SrTePolicy=
↪${SrTe} |
```

static create_capture_byte_pattern(*Port*, ***kwargs*)

在指定端口上创建 Byte Pattern

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **CustomCapturePatternOperator** (*str*) -- 表达式位运算符：，类型为: string, 默认值：AND, 支持参数

  AND OR XOR

- **CustomCapturePatternNot** (*bool*) -- 表达式取反：，类型为：bool, 取值范围：True 或 False, 默认值：False
- **UseFrameLength** (*bool*) -- 使用 Frame 长度：，类型为：bool, 取值范围：True 或 False, 默认值：False
- **Data** (*str*) -- 最小值：，类型为：string, 取值范围：十六进制字符串, 默认值：0x0,
- **MaxData** (*str*) -- 最大值：，类型为：string, 取值范围：十六进制字符串, 默认值：0xff,
- **Mask** (*str*) -- 掩码：，类型为：string, 取值范围：十六进制字符串, 默认值：0xff,

- **Offset** (*int*) -- 偏移位：，类型为：number, 取值范围：0-16378, 默认值：0
- **MinFrameLength** (*int*) -- 最小长度, 当 UseFrameLength 为 True 有效：，类型为：number, 取值范围：64-16383, 默认值：64
- **MaxFrameLength** (*int*) -- 最大长度, 当 UseFrameLength 为 True 有效：，类型为：number, 取值范围：64-16383, 默认值：16383

返回 Byte Pattern 唯一索引字符串 string，例如：CaptureBytePattern_1

返回类型 str

实际案例

robotframework:

```
| Create Capture Byte Pattern | Port=${Port} | Data=0x0 0x01 | Mask=0xff␣
↪0xff | Offset=0 | CustomCapturePatternOperator=OR |␣
↪CustomCapturePatternNot=True |
```

static create_capture_pdu_pattern(*Port*, *HeaderTypes*, *FieldName*, ***kwargs*)

在指定端口上创建 Pdu Pattern

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **HeaderTypes** (*list*) -- 指定报文结构 EthernetII Vlan IPv4 IPv6 Igmpv1 Igmpv1Query Igmpv2 Igmpv2Query Igmpv3Report Igmpv3Query Icmpv4EchoRequest Icmpv4EchoReply
- **FieldName** (*str*) -- 过滤字段名
- **CustomCapturePatternOperator** (*str*) -- 表达式位运算符：，类型为：string, 默认值：AND, 支持参数

  AND OR XOR
- **CustomCapturePatternNot** (*bool*) -- 表达式取反：，类型为：bool, 取值范围：True 或 False, 默认值：False
- **Value** (*str*) -- 最小值：，类型为：string
- **MaxValue** (*str*) -- 最大值：，类型为：string
- **Mask** (*str*) -- 掩码：，类型为：string

返回 Pdu Pattern 唯一索引字符串 string，例如：CapturePduPattern_1

返回类型 str

实际案例

robotframework:

```
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoReply |
| Create Capture Pdu Pattern | Port=${Port} | HeaderTypes=${HeaderTypes}␣
↪| FieldName=Icmpv4EchoReply_1.code | Value=4 | MaxValue=5 |
```

static create_dhcp_client(*Port*, ***kwargs*)

创建 DHCPv4 客户端协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象

关键字参数

- **Name** (*str*) -- DHCP 协议会话名称
- **Mode** (*str*) -- DHCPv4 客户端模式，默认值：CLIENT, 支持的参数：

  CLIENT

  RELAY_AGENT
- **HostName** (*str*) -- 主机名字，默认值：XINERTEL
- **ParameterRequests** (*list*) -- 主机请求选项, 默认值: ['NONEOPTION', 'SUBNET_MASK', 'DOMAIN_NAME_SERVERS', 'DOMAIN_NAME', 'STATIC_ROUTES'], 支持的参数:

  SUBNET_MASK

  ROUTERS

  DOMAIN_NAME_SERVERS

  DOMAIN_NAME

  STATIC_ROUTES

  IP_LEASE_TIME

  SERVER_IDENTIFIER

  T1

  T2
- **EnableRouterOption** (*bool*) -- 启用路由选项, 默认值：False
- **VendorClassIdentifier** (*str*) -- 供应商识别, 默认值：XINERTEL
- **BroadcastFlag** (*str*) -- 默认值：BROADCAST, 支持参数：

  UNICAST

  BROADCAST
- **RelayAgentIp** (*str*) -- 代理端 IP, 取值范围：IPv4 地址, 默认值：1.1.1.1
- **ServerIp** (*str*) -- DHCPv4 服务端 IP, 取值范围：IPv4 地址, 默认值：2.1.1.3
- **EnableRelayAgentCircuitID** (*bool*) -- 使能代理电路标识, 默认值：False
- **RelayAgentCircuitID** (*str*) -- 代理电路标识, 默认值：""
- **EnableRelayAgentRemoteID** (*str*) -- 使能代理远程标识, 默认值：False
- **RelayAgentRemoteID** (*str*) -- 代理远程标识, 默认值：""
- **EnableSyncAddressToInterface** (*bool*) -- 使能同步地址到接口, 默认值：True
- **HostInterface** (Interface) -- 客户端接口对象, 类型：object

返回 DHCP 协议会话对象 Object

返回类型 (DhcpClient)

实际案例

```
| ${Session} | Create Dhcp Client | Port=${Port} | Name=DHCP_Client_1 |
```

static create_dhcp_client_custom_option(*Session*, ***kwargs*)

创建测试仪表 DHCP 协议会话 option 对象

参数 **Session** (DhcpClient) -- DHCPv4 Client 协议对象, 类型为: object / list

关键字参数

- **OptionTag** (*number*) -- 可选项类型标识, 默认值: 0, 取值范围: 0-255
- **OptionType** (*str*) -- 可选项数据类型, 默认值: HEX, 支持以下参数:

  HEX

  STRING

  BOOLEAN

  INT8

  INT16

  INT32

  IP
- **EnableOptionValueList** (*bool*) -- 可选项值列表, 默认值: False
- **OptionValue** (*str*) -- 可选项值, 默认值: ""
- **MessageType** (*str*) -- 携带 Option 消息类型, 默认值: DISCOVER, 支持以下参数:

  DISCOVER

  REQUEST

返回 测试仪表 DHCP 协议会话 option 对象 Object

返回类型 (Dhcpv4ClientOption)

实际案例

```
| Create Dhcp Client Custom Option | Session=${Sessions} |
```

static create_dhcp_server(*Port*, ***kwargs*)

创建 DHCP Server 会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象

关键字参数

- **Name** (*str*) -- DHCP Server 协议会话名称
- **LeaseTime** (*number*) -- 租约时间(秒), 默认值: 600, 范围: 1-4294967295
- **RenewTime** (*number*) -- T1 租约更新时间 (%), 默认值: 50, 范围: 0-200
- **RebindTime** (*number*) -- T2 租约更新时间 (%), 默认值: 87.5, 范围: 0-200
- **MinLeaseTime** (*number*) -- 最小允许租约时间 (秒), 默认值: 10, 范围: 1-4294967295
- **DeclineReserveTime** (*number*) -- 资源释放等待时间 (秒), 默认值: 10, 范围: 1-600

- **OfferReserveTime** (*number*) -- 租约申请超时 (秒), 默认值：10, 范围：1-600
- **ServerHostName** (*str*) -- 服务端名字
- **DuplicateAddressDetection** (*bool*) -- 重复地址检测（DAD）, 默认值：False
- **DuplicateAddressDetectionTimeout** (*number*) -- DAD 超时时间, 默认值：0.5, 范围：0-60

返回 DHCP Server 会话对象 Object

返回类型 (DhcpServer)

实际案例

```
| Create Dhcp Server | Port=${Port} | RenewTime=100 |
```

static create_dhcp_server_address_pool(*Sessions*, ***kwargs*)

创建 DHCP Server 会话对象地址池

参数 **Sessions** (DhcpServer) -- DHCPv4 Client 协议对象, 类型为：object / list

关键字参数

- **Name** (*str*) -- DHCP Server 协议会话名称
- **PoolAddressStart** (*str*) -- 开始地址, 取值范围：IPv4 地址, 默认值：1.1.1.2
- **PoolAddressStep** (*str*) -- 地址步长, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **PrefixLength** (*number*) -- 前缀长度, 默认值：24, 范围：0-32
- **RouterList** (*str*) -- 网关列表
- **LimitPoolCount** (*bool*) -- 限制地址池个数, 默认值：True
- **PoolAddressCount** (*number*) -- 地址池地址个数, 默认值：255, 范围：0-4294967295
- **DomainName** (*str*) -- 域名
- **DnsList** (*str*) -- DNS 地址列表

返回 DHCP 地址池对象列表 list

返回类型 (Dhcpv4AddressPool)

实际案例

```
| Create Dhcp Server Address Pool | Sessions=${Sessions} |␣
↪PoolAddressStart=2.2.2.2 |
```

static create_dhcp_server_custom_option(*Session*, ***kwargs*)

创建测试仪表 DHCP 协议会话 option 对象

参数 **Session** (DhcpClient) -- DHCPv4 Client 协议对象, 类型为：object / list

关键字参数

- **OptionTag** (*number*) -- 可选项类型标识, 默认值：0, 取值范围：0-255

- **OptionType** (*str*) -- 可选项数据类型, 默认值：HEX, 支持以下参数：

  HEX

  STRING

  BOOLEAN

  INT8

  INT16

  INT32

  IP

- **EnableOptionValueList** (*bool*) -- 可选项值列表, 默认值：False
- **OptionValue** (*str*) -- 可选项值, 默认值：""
- **MessageType** (*str*) -- 携带 Option 消息类型, 默认值：DISCOVER, 支持以下参数：

  DISCOVER

  REQUEST

返回 测试仪表 DHCP 协议会话 option 对象 Object

返回类型 (Dhcpv4ClientOption)

实际案例

```
| Create Dhcp Server Custom Option | Session=${Sessions} |
```

static create_dhcpv6_client(*Port*, ***kwargs*)

创建 DHCPv6 客户端会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- DHCPv6 客户端会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 DHCPv6 客户端会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EmulationMode** (*str*) -- 会话模式, 默认值：DHCPV6, 取值范围：

  DHCPV6

  DHCPV6PD

  DHCPV6ANDPD

- **EnableRenewMsg** (*bool*) -- 使能 Renew 消息, 默认值：True, 取值范围：True 或 False
- **EnableRebindMsg** (*bool*) -- 使能 Rebind 消息, 默认值：True, 取值范围：True 或 False
- **EnableReconfigAccept** (*bool*) -- 使能 Reconfigure 消息, 默认值：True, 取值范围：True 或 False
- **EnableSyncAddressInterface** (*bool*) -- 默认值：True, 取值范围：True 或 False
- **T1Timer** (*int*) -- T1 时刻(秒), 取值范围：0-2147483647, 默认值：302400
- **T2Timer** (*int*) -- T2 时刻(秒), 取值范围：0-2147483647, 默认值：483840

- **PreferredLifetime** (*int*) -- 首选生命周期（秒）, 取值范围：0-2147483647, 默认值：604800
- **ValidLifetime** (*int*) -- 有效生命周期（秒）, 取值范围：0-2147483647, 默认值：2592000
- **RapidCommitOptMode** (*str*) -- 快速交互模式, 默认值：DISABLE, 取值范围：

  DISABLE

  ENABLE

  SERVERSET
- **DuidType** (*str*) -- DUID 类型, 默认值：LL, 取值范围：

  LLT

  EN

  LL

  CUSTOM
- **DuidCustomValue** (*int*) -- 自定义 DUID 编号, 取值范围：0-65535, 默认值：1
- **DuidEnterpriseNumber** (*int*) -- DUID 企业编号, 取值范围：0-4294967295, 默认值：3456
- **DuidStartValue** (*str*) -- DUID 企业编号, 默认值：3456, 取值范围：匹配正则表达式”^([0-9a-fA-F]{1,256})$”
- **DuidStepValue** (*int*) -- DUID 标识符跳变, 取值范围：0-4294967295, 默认值：0x1
- **DestinationAddress** (*str*) -- 目的地址, 默认值：ALL, 取值范围：

  ALL

  SERVER
- **EnableRelayAgent** (*bool*) -- 使能中继代理, 默认值：False, 取值范围：True 或 False
- **RelayAgentIp** (*str*) -- 中继代理 IP, 默认值：2000::1, 取值范围：有效的 ipv6 地址
- **ServerIp** (*str*) -- 服务 IP, 默认值：2001::2, 取值范围：有效的 ipv6 地址
- **EnableUseRelayAgentMacForTraffic** (*bool*) -- 默认值：True
- **RequestPrefixLength** (*int*) -- 请求前缀长度, 取值范围：0-128, 默认值：64
- **RequestPrefixStartAddress** (*str*) -- 请求前缀地址, 默认值：’::’, 取值范围：有效的 ipv6 地址
- **ControlPlaneSrcIPv6Addr** (*str*) -- 默认值：LINKLOCAL, 取值范围：

  LINKLOCAL

  ROUTEADVERTISEMENT
- **RequestStartAddress** (*str*) -- 请求前缀地址, 默认值：’::’, 取值范围：有效的 ipv6 地址
- **EnableAuthentication** (*bool*) -- 使能认证, 默认值：False, 取值范围：True 或 False

- **AuthenticationProtocol** (*str*) -- 认证协议, 默认值：DELAYED, 取值范围：

  DELAYED

  RECONFIGURATION
- **DhcpRealm** (*str*) -- 认证域名称, 默认值：'xinertel.com'
- **AuthenticationKeyId** (*int*) -- 取值范围：0-4294967295, 默认值：0
- **AuthenticationKey** (*str*) -- 认证秘钥 ID, 默认值：''
- **AuthenticationKeyType** (*str*) -- 认证秘钥类型, 默认值：ASCII, 取值范围：

  ASCII

  HEX
- **EnableDad** (*bool*) -- 使能重复地址检测（DAD）, 默认值：False, 取值范围：True 或 False
- **DadTimeout** (*int*) -- DAD 超时时间（秒）, 取值范围：1-4294967295, 默认值：2
- **DadTransmits** (*int*) -- DAD 传输次数, 取值范围：1-4294967295, 默认值：1
- **HostInterface** (Interface) -- 接口对象, 类型：object

返回 DHCPv6 客户端会话对象, 类型：object

返回类型 (Dhcpv6Client)

实际案例

```
| Create Dhcpv6 Client | Port=${Port} | DadTransmits=10 |
```

static create_dhcpv6_client_custom_options(*Sessions*, ***kwargs*)

创建 DHCPv6 Client Custom Options 对象

Args：

Session (Dhcpv6Client): DHCPv6 客户端会话对象, 类型为：object / list

关键字参数

- **OptionVal** (*int*) -- 选项标识, 取值范围：0-255, 默认值：0
- **IncludeMsg** (*list*) -- 包含选项的消息类型, 默认值：['SOLICIT', 'REQUEST'], 取值范围：

  SOLICIT

  REQUEST

  CONFIRM

  RENEW

  REBIND

  RELEASE

  INFOREQUEST

  RELAYFORWARD
- **Wildcards** (*bool*) -- 使能通配符, 默认值：False, 取值范围：True 或 False

- **StringIsHexadecimal** (*bool*) -- 使能十六进制字符, 默认值：False, 取值范围：True 或 False
- **OptionPayload** (*str*) -- 选项载荷, 默认值：”, 取值范围：string length in [0,512]
- **OptionHexPayload** (*int*) -- 十六进制选项载荷, 默认值：””
- **RemoveOption** (*bool*) -- 默认值：False, 取值范围：True 或 False

返回 DHCPv6 Client Custom Options 对象, 类型：object / list

返回类型 (Dhcpv6ClientCustomOptionsConfig)

实际案例

```
| ${Dhcpv6} | Create Dhcpv6 Client | Port=${Port} |
| Create Dhcpv6 Client Custom Options | Sessions=${Dhcpv6} |␣
↪Wildcards=True |
```

static create_dhcpv6_server(*Port*, ***kwargs*)

创建 DHCPv6 服务端会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- DHCPv6 服务端会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 DHCPv6 服务端会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EmulationMode** (*str*) -- 默认值：DHCPV6, 取值范围：

  DHCPV6

  DHCPV6PD
- **RenewalTimer** (*int*) -- T1 租约更新时间（%）, 取值范围：1-200, 默认值：50
- **RebindingTimer** (*int*) -- T2 租约更新时间（%）, 取值范围：1-200, 默认值：80
- **DnsList** (*str*) -- DNS 地址列表, 取值范围：IPv6 地址, 默认值：”::”
- **EnableDelayedAuth** (*bool*) -- 使能延时认证, 取值范围：True 或 False, 默认值：False
- **DhcpRealm** (*str*) -- DHCP 认证域名, 默认值：”xinertel.com”
- **AuthenticationKeyId** (*int*) -- 认证秘钥 ID, 取值范围：0-4294967295, 默认值：0
- **AuthenticationKey** (*str*) -- 认证秘钥, 默认值：””
- **AuthenticationKeyType** (*str*) -- 认证秘钥类型, 默认值：ASCII, 取值范围：

  ASCII

  HEX
- **EnabledReconfigureKey** (*bool*) -- 使能重新配置认证, 取值范围：True 或 False, 默认值：False
- **ReconfigureKey** (*str*) -- 重新配置秘钥类型, 默认值：””

- **ReconfigureKeyType** (*str*) -- 重新配置秘钥类型, 默认值：ASCII, 取值范围：

  ASCII

  HEX

- **EnabledDhcpv6Only** (*bool*) -- 使能单独 DHCPv6, 取值范围：True 或 False, 默认值：False
- **EnabledTcp** (*bool*) -- 使能 TCP, 取值范围：True 或 False, 默认值：False
- **TcpPort** (*int*) -- TCP 端口号, 取值范围：1-65535, 默认值：547
- **LeaseQueryStatusCode** (*str*) -- 租借查询应答码, 默认值：SUCCESS, 取值范围：

  SUCCESS

  UNKNOWN_QUERY_TYPE

  MALFORMED_QUERY

  NOT_CONFIGURED

  NOT_ALLOWED

- **BulkLeaseQueryStatusCode** (*str*) -- 批量租借查询应答码, 默认值：SUCCESS, 取值范围：

  SUCCESS

  UNKNOWN_QUERY_TYPE

  MALFORMED_QUERY

  NOT_CONFIGURED

  NOT_ALLOWED

  QUERY_TERMINATED

- **ActiveLeaseQueryStatusCode** (*str*) -- 活动租借查询应答码, 默认值：SUCCESS, 取值范围：

  SUCCESS

  UNKNOWN_QUERY_TYPE

  MALFORMED_QUERY

  NOT_CONFIGURED

  NOT_ALLOWED

  QUERY_TERMINATED

  DATA_MISSING

  CATCH_UP_COMPLETE

  NOT_SUPPORTED

- **StartTlsStatusCode** (*str*) -- 启动 TLS 应答码, 默认值：SUCCESS, 取值范围：

  SUCCESS

  TLS_CONNECTION_REFUSED

返回 DHCPv6 服务端会话对象, 类型：object

返回类型 (Dhcpv6Server)

实际案例

```
| Create Dhcpv6 Server | Port=${Port} | DadTransmits=10 |
```

static create_dhcpv6_server_address_pool(*Sessions*, ***kwargs*)

创建 DHCPv6 Server Address Pool 对象

Args:

Session (Dhcpv6Server): DHCPv6 服务端会话对象, 类型为: object / list

关键字参数

- **PrefixLength** (*int*) -- 地址池前缀长度, 取值范围: 0-128, 默认值: 64
- **AssignMode** (*str*) -- 默认值: SUCCESS, 取值范围:

  CUSTOM

  EUI64
- **StartAddress** (*str*) -- 地址池起始地址, 取值范围: IPv6 地址, 默认值: ”2001::1”
- **HostStep** (*str*) -- 地址池地址跳变, 取值范围: IPv6 地址, 默认值: ”::1”
- **AddressCount** (*int*) -- 地址池地址总数, 取值范围: 1-4294967295, 默认值: 65535
- **PreferredLifetime** (*int*) -- 首选生命周期（秒）, 取值范围: 0-4294967295, 默认值: 604800
- **ValidLifetime** (*int*) -- 有效生命周期（秒）, 取值范围: 0-4294967295, 默认值: 2592000
- **MinIaidValue** (*int*) -- 最小 IAID 值, 取值范围: 0-4294967295, 默认值: 0
- **MaxIaidValue** (*int*) -- 最大 IAID 值, 取值范围: 0-4294967295, 默认值: 4294967295

返回 DHCPv6 Server Address Pool 对象, 类型: object / list

返回类型 (Dhcpv6AddressPoolsConfig)

实际案例

```
| ${Dhcpv6} | Create Dhcpv6 Server | Port=${Port} |
| Create Dhcpv6 Server Address Pool | Sessions=${Dhcpv6} |␣
↪MinIaidValue=10 |
```

static create_dhcpv6_server_custom_options(*Sessions*, ***kwargs*)

创建 DHCPv6 Server Custom Options 对象

Args:

Session (Dhcpv6Server): DHCPv6 服务端会话对象, 类型为: object / list

关键字参数

- **OptionVal** (*int*) -- 选项标识, 取值范围: 0-255, 默认值: 0
- **IncludeMsg** (*list*) -- 包含选项的消息类型, 默认值: [’ADVERTISE’, ’REPLY’], 取值范围:

  ADVERTISE

REPLY

RECONFIGURE

RELAYREPLY

- **Wildcards** (*bool*) -- 使能通配符, 取值范围：True 或 False, 默认值：False
- **StringIsHexadecimal** (*bool*) -- 使能十六进制字符串, 取值范围：True 或 False, 默认值：False
- **OptionPayload** (*str*) -- 选项负载, 默认值：""
- **OptionHexPayload** (*str*) -- 选项负载, 默认值：""

返回 DHCPv6 Server Custom Options 对象, 类型：object / list

返回类型 (Dhcpv6ServerCustomOptionsConfig)

实际案例

```
| ${Dhcpv6} | Create Dhcpv6 Server | Port=${Port} |
| Create Dhcpv6 Server Custom Options | Sessions=${Dhcpv6} |␣
↪Wildcards=True |
```

static create_dhcpv6_server_prefix_pool(*Sessions*, ***kwargs*)

创建 DHCPv6 Server Prefix Pool 对象

Args：

Session (Dhcpv6Server): DHCPv6 服务端会话对象, 类型为：object / list

关键字参数

- **PrefixLength** (*int*) -- 前缀池前缀长度, 取值范围：0-128, 默认值：64
- **PrefixPoolStart** (*str*) -- 前缀池起始地址, 取值范围：IPv6 地址, 默认值："2001::1"
- **PrefixPoolStep** (*str*) -- 前缀池地址跳变, 取值范围：IPv6 地址, 默认值："0:0:0:1::"
- **PrefixAddressCount** (*int*) -- 前缀池地址总数, 取值范围：1-65535, 默认值：16
- **PreferredLifetime** (*int*) -- 最优租期（秒）, 取值范围：0-4294967295, 默认值：604800
- **ValidLifetime** (*int*) -- 有效租期（秒）, 取值范围：0-4294967295, 默认值：2592000
- **MinIaidValue** (*int*) -- 最小 IAID 值, 取值范围：0-4294967295, 默认值：0
- **MaxIaidValue** (*int*) -- 最大 IAID 值, 取值范围：0-4294967295, 默认值：4294967295

返回 DHCPv6 Server Prefix Pool 对象, 类型：object / list

返回类型 (Dhcpv6PrefixPoolsConfig)

实际案例

```
| ${Dhcpv6} | Create Dhcpv6 Server | Port=${Port} |
| Create Dhcpv6 Server Prefix Pool | Sessions=${Dhcpv6} | MinIaidValue=10␣
↪|
```

static create_dot1x(*Port*, ***kwargs*)

创建 802.1x 会话对象

参数 Port (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- 802.1x 会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 801.1x 会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AuthMode** (*str*) -- 默认值：MD5, 取值范围：

  MD5

  TLS

  TTLS
- **Identity** (*str*) -- 默认值：xinertel, 取值范围：string length in [1,255]
- **Password** (*str*) -- 默认值：xinertel, 取值范围：string length in [1,255]
- **UseAuthenticatorMac** (*bool*) -- 使能 801.1x 会话, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AuthenticatorMac** (*str*) -- 默认值：01:80:c2:00:00:03, 取值范围：有效的 mac 地址
- **RetryCount** (*int*) -- 默认值：5, 取值范围：uint32
- **RetryTimeout** (*int*) -- 默认值：5, 取值范围：1-4294967295
- **RetransmitCount** (*int*) -- 默认值：5, 取值范围：uint32
- **RetransmitTimeout** (*int*) -- 默认值：5, 取值范围：1-4294967295
- **SupplicantCertificateName** (*str*) -- 默认值：‘’，取值范围：string length in [1,255]
- **CertificatePassword** (*str*) -- 默认值：°，取值范围：string length in [1,255]
- **DuplicateUserInfoToInner** (*bool*) -- 使能 801.1x 会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **InnerIdentity** (*str*) -- 默认值：xinertel, 取值范围：string length in [1,255]
- **InnerPassword** (*str*) -- 默认值：xinertel, 取值范围：string length in [1,255]
- **InnerTunnelAuthMode** (*str*) -- 默认值：AUTO, 取值范围：

  AUTO

  GTC

  MS_CHAPV2

  MD5
- **EnableClientCertificate** (*bool*) -- 使能 801.1x 会话, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 802.1x 会话对象, 类型：object

返回类型 (Dot1x)

实际案例

```
| Create Dot1x | Port=${Port} | DadTransmits=10 |
```

static create_igmp(*Port*, ***kwargs*)

创建 IGMP 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- IGMP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ICMP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：IGMPV2, 支持版本：

  IGMPV1

  IGMPV2

  IGMPV3
- **PackReports** (*bool*) -- 合并报告报文, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InitialJoin** (*bool*) -- 单个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustJoin** (*bool*) -- 多个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustnessVariable** (*int*) -- Robust 值, 类型为：number, 取值范围：2-255, 默认值：2
- **UnsolicitedReportInterval** (*int*) -- 发送初始报文的时间间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **ForceLeave** (*bool*) -- 强制发送 Leave 报文, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterPresentTimeout** (*int*) -- IGMPv1 路由器存在的超时时间 (秒), 类型为：number, 取值范围：0-65535, 默认值：400
- **NotFragment** (*bool*) -- 设置 IP 头报文分片标志位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **TosValue** (*bool*) -- 设置 IP 头 TOS 值 (Hex), 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 IGMP 协议会话对象, 类型：object

返回类型 (Igmp)

实际案例

```
| Create Igmp | Port=${Port} | Version=IGMPV3 |
```

static create_igmp_querier(*Port*, ***kwargs*)

创建 IGMP Querier 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- IGMP Querier 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 IGMP Querier 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：IGMPV2, 支持版本：

  IGMPV1

  IGMPV2

  IGMPV3
- **RobustnessVariable** (*int*) -- 健壮系数, 取值范围：2-255, 默认值: 2
- **Interval** (*int*) -- 查询时间间隔（秒）, 取值范围：0-4294967295, 默认值: 125
- **QueryResponseInterval** (*int*) -- 查询响应时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 10000
- **StartupQueryCount** (*int*) -- 初始查询报文个数, 取值范围：1-255, 默认值：2
- **LastMemberQueryInterval** (*int*) -- 最后成员查询时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 1000
- **LastMemberQueryCount** (*bool*) -- 最后成员查询次数, 取值范围：0-255, 默认值: 2
- **IPv4DoNotFragment** (*bool*) -- 设置 IP 头报文分片标志位, 取值范围：True 或 False, 默认值: False
- **IPv4TosValue** (*str*) -- 设置 IP 头 TOS 值, 类型为：bool, 取值范围：0x0-0xff, 默认值: 0xc0

返回 IGMP 协议会话对象, 类型：object

返回类型 (IgmpQuerier)

实际案例

```
| Create Igmp Querier| Port=${Port} | Version=IGMPV3 |
```

static create_imix(*Name*, *Seed=None*)

创建流量 Imix 模板

参数

- **Name** (*str*) -- 创建的 Imix 模板名称
- **Seed** (*int*) -- Imix 模板随机种子

返回 Imix 模板对象

返回类型 (Imix)

实际案例

```
| Create Imix | Name=Imix_1 | Seed=10121112 |
```

static create_interface(*Port*, *Layers=None*)

在指定端口上創建接口

参数

- **Port** (*Port*) -- 测试仪仪表端口 Port 对象
- **Layers** (*list*) -- 接口封装类型, 支持的有:

  eth

  ipv4

  ipv6

返回 接口 interface 对象

返回类型 (Interface)

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 |
| ${Layers} | Create List | eth | ipv4 |
| ${Port} | Reserve Ports | ${Ports} | ${Location} |
| ${Interface} | Create Interface | ${Port} | ${Layers} |
```

static create_isis(*Port*, ***kwargs*)

创建 ISIS 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为: object

关键字参数

- **Name** (*str*) -- ISIS 协会话名称, 类型为: string
- **Enable** (*bool*) -- 使能 ISIS 协议会话, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IpVersion** (*str*) -- IP 版本, 类型为: string, 默认值: IPV4, 支持版本:

  IPV4

  IPV6

  IPV4IPV6

- **Level** (*str*) -- 区域类型, 类型为: string, 默认值: L2, 支持版本:

  L1

  L2

  L1L2

- **NetworkType** (*str*) -- 网络类型, 类型为: string, 默认值: BROADCAST, 支持参数:

  BROADCAST

  P2P

- **SystemId** (*str*) -- 系统 ID, 类型为: string, 取值范围: MAC 地址, 默认值: 00:00:00:00:00:01

- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-127, 默认值：0
- **AuthMethod** (*str*) -- 认证方式, 类型为：string, 默认值：NONE, 支持参数：

  NONE

  SIMPLE

  MD5
- **Password** (*str*) -- 4 字节自治域跳变, 类型为：string, 默认值：Xinertel
- **CircuitId** (*int*) -- 电路 ID, 类型为：number, 取值范围：0-255, 默认值：1
- **Area1** (*str*) -- 区域 ID 1, 类型为：hex number, 取值范围：0x0-0xff, 默认值：0x10
- **Area2** (*str*) -- 区域 ID 2, 类型为：hex number, 取值范围：0x0-0xff, 默认值：空
- **Area3** (*str*) -- 区域 ID 3, 类型为：hex number, 取值范围：0x0-0xff, 默认值：空
- **MetricMode** (*str*) -- 度量模式, 类型为：string, 默认值：NARROWWIDE, 支持参数：

  NARROW

  WIDE

  NARROWWIDE
- **TeRouterId** (*str*) -- TE 路由器 ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1
- **TeRouterIdIpv6** (*str*) -- IPv6 TE 路由器 ID, 类型为：string, 取值范围：IPv6 地址, 默认值：3000::1
- **HelloInterval** (*int*) -- Hello PDU 发送间隔 (秒), 类型为：number, 取值范围：1-300, 默认值：10
- **HelloMultiplier** (*int*) -- Hello 时间间隔倍数, 类型为：number, 取值范围：1-100, 默认值：3
- **PsnInterval** (*int*) -- PSNP 发送间隔 (秒), 类型为：number, 取值范围：1-20, 默认值：2
- **LspRefreshTime** (*int*) -- LSP 刷新时间 (秒), 类型为：number, 取值范围：1-65535, 默认值：900
- **RetransInterval** (*int*) -- LSP 重传间隔 (秒), 类型为：number, 取值范围：1-100, 默认值：5
- **HelloPadding** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LspSize** (*int*) -- LSP 大小, 类型为：number, 取值范围：100-1492, 默认值：1492
- **ValidateIpAddr** (*bool*) -- 使能接口校验, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **EnableBFD** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MtParams** (*int*) -- 多拓扑参数数量, 类型为：number, 取值范围：0-2, 默认值：0
- **PerPduAuthentication** (*int*) -- Per PDU 认证数量, 类型为：number, 取值范围：0-4, 默认值：0
- **ReportLabel** (*bool*) -- 使能 ReportLabel, 类型为：bool, 默认值：True
- **LearnRoute** (*bool*) -- 使能 LearnRoute, 类型为：bool, 默认值：True
- **RecordLspNextSequenceNum** (*bool*) -- 使能 Record Lsp Next Sequence Number, 类型为：bool, 默认值：True
- **L1NarrowMetric** (*int*) -- L1 Narrow Metric, 类型为：number, 取值范围：0-63, 默认值：1
- **L1WideMetric** (*int*) -- L1 Wild Metric, 类型为：number, 取值范围：0-16777214, 默认值：1
- **L2NarrowMetric** (*int*) -- L2 Narrow Metric, 类型为：number, 取值范围：0-63, 默认值：1
- **L2WideMetric** (*int*) -- L2 Wide Metric, 类型为：number, 取值范围：0-16777214, 默认值：1

返回 ISIS 协议会话对象

返回类型 (IsisRouter)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${MtId} | Create List | IPV4 | IPV6 |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True | MtParams=1 |
```

static create_isis_binding_sr_sid_sub_tlv(*Binding*, ***kwargs*)

创建 ISIS Capability Srms Preference Sub Tlv 对象

参数 **Binding** (IsisSrBindingTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **ValueType** (*str*) -- 选择标识符（SID 或标签）, 默认值：BIT32, 取值范围：

  BIT20

  BIT32

- **Sid** (*int*) -- 值类型为 20bit 时，指定起始标签; 值类型为 32bit 时，指定起始 SID, 默认值：12000

返回 ISIS Capability Srms Preference Sub TLV 对象

返回类型 (IsisSrSRMSPrefSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Binding} | Create Isis Binding Tlv | Lsp=${LSP} | Ipv4Version=False |
| Create Isis Binding Sr sid Sub Tlv | Binding=${Binding}
```

static create_isis_capability_sr_algorithm_sub_tlv(*Capability*, ***kwargs*)

创建 ISIS Capability Sr Algorithm Sub Tlv 对象

参数 **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数 **Algorithm** (*list*) -- 算法值, 默认值：0, 取值范围：int

返回 ISIS Capability Sr Algorithm Sub TLV 对象

返回类型 (IsisSrAlgorithmSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.
↪1.1 |
| Create Isis Capability Sr Algorithm Sub Tlv | Capability=${Capability} |
```

static create_isis_capability_sr_capability_sub_tlv(*Session*, *Capability*, ***kwargs*)

创建 ISIS Capability Sr Capability Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：['NOSHOW', 'IPv4_CAPABLE'], 取值范围：

  NOSHOW

  IPv4_CAPABLE

  IPv6_CAPABLE

- **ValueType** (*str*) -- 选择标识符（SID 或标签）, 默认值：BIT32, 取值范围：

  BIT20

  BIT32

返回 ISIS Capability Sr Capability Sub TLV 对象

返回类型 (IsisSrCapabilitySubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.
↪1.1 |
| Create Isis Capability Sr Capability Sub Tlv | Session={Session} |␣
↪Capability=${Capability} |
```

static create_isis_capability_sr_fad_sub_tlv(*Session*, *Capability*, ***kwargs*)

创建 ISIS Capability Sr Fad Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **FlexAlgo** (*int*) -- 灵活算法 ID, 默认值：128, 取值范围：128-255
- **MetricType** (*str*) -- 指定算路使用的度量类型, 默认值：IGP_METRIC, 取值范围：

  IGP_METRIC

  MIN_LINK_DELAY

  TE_METRIC
- **CalType** (*int*) -- 指定特定 IGP 算法的计算类型, 默认值：0, 取值范围：0-255
- **Priority** (*int*) -- 指定该 Sub TLV 的优先级, 默认值：0
- **FlexAlgoSubTlv** (*list*) -- 选择灵活算法路径计算要遵循的约束条件, 默认值：['UNKNOWN'], 取值范围：

  UNKNOWN

  EXCLUDE_ADMIN

  INCLUDE__ANY_ADMIN

  INCLUDE_ALL_ADMIN

  DEFINITION_FLAGS

  EXCLUDE_SRLG
- **ExcludeAdmin** (*int*) -- 类型为：number, 默认值：0, 取值范围：0-4294967295
- **IncludeAnyAdmin** (*int*) -- 类型为：number, 默认值：0, 取值范围：0-4294967295
- **IncludeAllAdmin** (*int*) -- 类型为：number, 默认值：0, 取值范围：0-4294967295
- **DefinitionFlags** (*list*) -- 类型为：hex int, 默认值：0x80, 取值范围：0-FF
- **ExcludeSRLG** (*list*) -- 类型为：hex int, 默认值：0x10020000, 取值范围：0-4294967295

返回 ISIS Capability Sr Fad Sub TLV 对象

返回类型 (IsisFelxAlgoDefinitionSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.
↪1.1 |
| Create Isis Capability Sr Fad Sub Tlv | Session={Session} | Capability=$
↪{Capability} |
```

static create_isis_capability_sr_node_msd_sub_tlv(*Session*, *Capability*, ***kwargs*)

创建 ISIS Capability Sr Node Msd Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  MAX_SEG_LELT

  MAX_END_POP

  MAX_T_INSERT

  MAX_T_ENCAPS

  MAX_END_D

- **MaxSegmentLeft** (*int*) -- 在应用与 SID 关联的 SRv6 Endpoint Function 指令之前，指定接收报文的 SRH 中 SL（Segment Left）字段的最大值, 默认值：8, 取值范围：0-255
- **MaxEndPop** (*int*) -- 指定 SRH 栈的顶端 SRH 中 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxInsert** (*int*) -- 指定执行 T.Insert 行为时可包含 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxEncap** (*int*) -- 指定执行 T.Encap 行为时可包含 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 默认值：8, 取值范围：0-255

返回 ISIS Capability Sr Node Msd Sub TLV 对象

返回类型 (IsisSrMsdSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.
↪1.1 |
| Create Isis Capability Sr Node Msd Sub Tlv | Capability=${Capability} |
```

static create_isis_capability_srms_preference_sub_tlv(*Capability*, ***kwargs*)

创建 ISIS Capability Srms Preference Sub Tlv 对象

参数 **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数 **Preference** (*int*) -- 指定本节点作为 SR Mapping Server 的优先级, 取值范围：0-255, 默认值：0

返回 ISIS Capability Srms Preference Sub TLV 对象

返回类型 (IsisSrSRMSPrefSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.
↪1.1 |
| Create Isis Capability Srms Preference Sub Tlv | Capability=$
↪{Capability} |
```

static create_isis_capability_srv6_capability_sub_tlv(*Session*, *Capability*, ***kwargs*)

创建 ISIS Capability Srv6 Capability Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Capability** (IsisCapabilityTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数 **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：UNKNOWN, 取值范围：

UNKNOWN

UNUSED0

O_BIT

UNUSED2

UNUSED3

UNUSED4

UNUSED5

UNUSED6

UNUSED7

UNUSED8

UNUSED9

UNUSED10

UNUSED11

UNUSED12

UNUSED13

UNUSED14

UNUSED15

返回 ISIS Capability Srv6 Capability Sub TLV 对象

返回类型 (IsisSrv6CapabilitySubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Capability} | Create Isis Capability Tlv | Lsp=${LSP} | SystemId=1.1.
↪1.1 |
| Create Isis Capability Srv6 Capability Sub Tlv | Session=Session |␣
↪Capability=${Capability} |
```

static create_isis_capability_tlv(*Session*, *Lsp*, ***kwargs*)

创建 ISIS Capability TLV 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **Option** (*list*) -- 选项, 默认值：['NOSHOW', 'SBIT'], 取值范围：

  NOSHOW

  SBIT

  DBIT

- **RouterId** (*str*) -- 路由器 ID, 默认值："192.0.0.1", 取值范围：有效 IPv4 地址

返回 ISIS Neighbor TLV 对象

返回类型 (IsisCapabilityTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Capability Tlv | Session=${Session} | Lsp=${LSP} |␣
↪RouterId=1.1.1.1 |
```

static create_isis_ipv4_tlv(*Lsp*, ***kwargs*)

创建 ISIS IPv4 TLV 对象

参数 **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **Name** (*str*) -- ISIS IPv4 TLV 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 ISIS IPv4 TLV, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- 路由类型, 类型为：string, 默认值：INTERNAL, 支持参数：

  INTERNAL

  EXTERNAL
- **RouteCount** (*int*) -- 路由数量, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **MetricType** (*str*) -- 度量类型, 类型为：string, 默认值：INTERNAL, 支持参数：

  INTERNAL

  EXTERNAL
- **WideMetric** (*int*) -- 扩展度量, 类型为：number, 取值范围：0-16777214, 默认值：10
- **UpDownBit** (*bool*) -- Up/Down 位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **StartIpv4Prefix** (*str*) -- 起始 IPv4 路由前缀, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1
- **PrefixLength** (*int*) -- 前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **NarrowMetric** (*int*) -- 默认度量, 类型为：number, 取值范围：0-63, 默认值：10

返回 ISIS IPv4 TLV 对象

返回类型 (IsisIpv4TlvConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Ipv4 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
```

static create_isis_ipv6_tlv(*Lsp*, ***kwargs*)

创建 ISIS IPv6 TLV 对象

参数 **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **Name** (*str*) -- ISIS IPv6 TLV 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 ISIS IPv6 TLV, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- 路由类型, 类型为：string, 默认值：INTERNAL, 支持参数：

  INTERNAL

EXTERNAL

- **RouteCount** (*int*) -- 路由数量, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **Increment** (*int*) -- 步长, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **MetricType** (*str*) -- 度量类型, 类型为: string, 默认值: INTERNAL, 支持参数:

  INTERNAL

  EXTERNAL
- **WideMetric** (*int*) -- 扩展度量, 类型为: number, 取值范围: 0-16777214, 默认值: 10
- **UpDownBit** (*bool*) -- Up/Down 位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **StartIpv6Prefix** (*str*) -- 起始 IPv4 路由前缀, 类型为: string, 取值范围: IPv6 地址, 默认值: 2000::1
- **PrefixLength** (*int*) -- 前缀长度, 类型为: number, 取值范围: 1-32, 默认值: 24

返回 ISIS IPv6 TLV 对象

返回类型 (IsisIpv6TlvConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Ipv6 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
```

static create_isis_lsp(*Session*, ***kwargs*)

创建 ISIS LSP 对象

参数 **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为: object

关键字参数

- **Name** (*str*) -- ISIS LSP 对象名称, 类型为: string
- **Enable** (*bool*) -- 使能 ISIS LSP, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **SystemId** (*str*) -- 系统 ID, 类型为: string, 取值范围: MAC 地址, 默认值: 00:00:00:00:00:01
- **Level** (*str*) -- 区域类型, 类型为: string, 默认值: L2, 支持版本:

  L1

  L2
- **PseudonodeId** (*int*) -- 伪节点 ID, 类型为: number, 取值范围: 1-100, 默认值: 0
- **TeRouterId** (*str*) -- TE 路由器 ID, 类型为: string, 取值范围: IPv4 地址, 默认值: 192.168.1.1
- **TeRouterIdIpv6** (*str*) -- IPv6 TE 路由器 ID, 类型为: string, 取值范围: IPv6 地址, 默认值: 3000::1

- **SequenceNumber** (*int*) -- 序列号, 类型为：number, 取值范围：1-300, 默认值：10
- **RemainingLifeTime** (*int*) -- 剩余生存时间, 类型为：number, 取值范围：1-100, 默认值：3
- **Checksum** (*int*) -- 使能正确校验和, 类型为：number, 取值范围：1-20, 默认值：2
- **AttachedBit** (*int*) -- 区域关联位, 类型为：number, 取值范围：1-65535, 默认值：900
- **OverloadBit** (*int*) -- 过载位, 类型为：number, 取值范围：1-100, 默认值：5

返回 ISIS LSP 对象

返回类型 (IsisLspConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| Create Isis Lsp | Session=${Session} | SystemId=00:00:00:00:00:02 |
```

static create_isis_neighbor_custom_sub_tlv(*SubTlv*, ***kwargs*)

创建 Isis Neighbor Custom Sub Tlv 对象

参数 **SubTlv** (IsisSrLinkMsdSubTlv) -- ISIS Neighbor Sr Link Msd Sid Sub TLV 对象, 类型为：object

关键字参数

- **SubType** (*int*) -- 该 Sub-TLV 的 Type 字段值, 默认值：0, 取值范围：0-255
- **SubValue** (*int*) -- 该 Sub-TLV 的 Value 字段值, 取值范围：十六进制值。默认值：08

返回 Isis Neighbor Sr Link Msd Sid Sub Tlv 对象

返回类型 (IsisSrLinkMsdSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Msd} | Create Isis Neighbor Sr Link Msd Sid Sub Tlv | Neighbor=$
↪{Neighbor} |
| Isis Neighbor Custom Sub Tlv | SubTlv=${Msd} |
```

static create_isis_neighbor_sr_adj_sid_sub_tlv(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Sr Adj Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位，默认值: ['NOSHOW', 'VALUE', 'LOCAL'], 取值范围:

  NOSHOW

  ADDRESS

  BACKUP

  VALUE

  LOCAL

  SET

  PERSISTENT

- **Sid** (*int*) -- Flags 中包含 L.Local 和 V.Value 时，指定标签值; Flags 中不包含 Value/Index 时，指定 SID/Label 范围内的标签偏移值, 默认值：0, 取值范围：0-4294967295
- **Weight** (*int*) -- 指定 Adj-SID 权重，用于负载分担, 默认值：0, 取值范围：0-255

返回 Isis Neighbor Sr Adj Sid Sub Tlv 对象

返回类型 (IsisSrAdjSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Sr Adj Sid Sub Tlv | Neighbor=${Neighbor} |
```

static create_isis_neighbor_sr_lan_adj_sid_sub_tlv(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Sr Lan Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为: object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位，默认值: ['NOSHOW', 'VALUE', 'LOCAL'], 取值范围:

  NOSHOW

  ADDRESS

  BACKUP

  VALUE

  LOCAL

  SET

  PERSISTENT

- **Sid** (*int*) -- Flags 中包含 L.Local 和 V.Value 时，指定标签值; Flags 中不包含 Value/Index 时，指定 SID/Label 范围内的标签偏移值, 默认值：0, 取值范围：0-4294967295
- **Weight** (*int*) -- 指定 Adj-SID 权重，用于负载分担, 默认值：0, 取值范围：0-255
- **SystemId** (*str*) -- 指定 LAN 上邻居的系统 ID, 默认值："00:00:00:00:00:01"

返回 Isis Neighbor Sr Adj Sid Sub Tlv 对象

返回类型 (IsisSrAdjSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Sr Lan Adj Sid Sub Tlv | Neighbor=${Neighbor} |
```

static create_isis_neighbor_sr_link_msd_sub_tlv(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Sr Link Msd Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  MAX_SEG_LELT

  MAX_END_POP

  MAX_T_INSERT

  MAX_T_ENCAPS

  MAX_END_D

- **MaxSegmentLeft** (*int*) -- 在应用与 SID 关联的 SRv6 Endpoint Function 指令之前，指定接收报文的 SRH 中 SL（Segment Left）字段的最大值, 默认值：8, 取值范围：0-255
- **MaxEndPop** (*int*) -- 指定 SRH 栈的顶端 SRH 中 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxInsert** (*int*) -- 指定执行 T.Insert 行为时可包含 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxEncap** (*int*) -- 指定执行 T.Encap 行为时可包含 SID 的最大数量, 默认值：8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 默认值：8, 取值范围：0-255

返回 Isis Neighbor Sr Link Msd Sid Sub Tlv 对象

返回类型 (IsisSrLinkMsdSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Sr Link Msd Sid Sub Tlv | Neighbor=${Neighbor} |
```

static create_isis_neighbor_srv6_endx_sid_sub_tlv(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Srv6 EndX Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  BACKUP

  SET

  PERSISTENT

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

  UNUSED7

- **Algorithm** (*int*) -- 指定 SID 关联的算法, 默认值：0, 取值范围：0-255
- **Weight** (*int*) -- 指定 End.X SID 的权重，用于负载分担, 默认值：100, 取值范围：0-255
- **EndpointFunc** (*str*) -- 端点行为, 默认值：END_NO, 取值范围：

  END_NO

  END_PSP

  END_USP

  END_PSP_USP

  END_X_NO

  END_X_PSP

  END_X_USP

  END_X_PSP_USP

END_T_NO
END_T_PSP
END_T_USP
END_T_PSPS_USP
END_B6
END_B6_ENCAPS
END_BM
END_DX6
END_DX4
EDN_DT6
END_DT4
END_DT46
END_DX2
END_DX2V
END_DX2U
END_DX2M
END_S
END_B6_RED
END_B6_ENCAPS_RED
END_WITH_USD
END_PSP_USD
END_USP_USD
END_PSP_USP_USD
END_X_USD
END_X_PSP_USD
END_X_USP_USD
END_X_PSP_USP_USD
END_T_USD
END_T_PSP_USD
END_T_USP_USD
END_T_PSP_USP_USD

- **EnableCustom** (*bool*) -- 使能自定义端点行为, 默认值：False
- **CustomFunc** (*int*) -- 自定义端点行为, 默认值：0, 取值范围：0-65535
- **SID** (*str*) -- 指定通告的 SRv6 SID, 默认值：”::1”, 取值范围：有效 IPv6 地址

返回 Isis Neighbor Srv6 EndX Sid Sub Tlv 对象

返回类型 (IsisSrv6EndXSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Srv6 EndX Sid Sub Tlv | Neighbor=${Neighbor} |
```

static create_isis_neighbor_srv6_lan_endx_sid_sub_tlv(*Session*, *Neighbor*, ***kwargs*)

创建 Isis Neighbor Srv6 Lan EndX Sid Sub Tlv 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为：object

关键字参数

- **SystemIdLan** (*str*) -- LAN 系统标识, 默认值：”00:10:96:00:00:01”
- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  BACKUP

  SET

  PERSISTENT

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

  UNUSED7
- **Algorithm** (*int*) -- 指定 SID 关联的算法, 默认值：0, 取值范围：0-255
- **Weight** (*int*) -- 指定 End.X SID 的权重，用于负载分担, 默认值：100, 取值范围：0-255
- **EndpointFunc** (*list*) -- 端点行为, 默认值：END_NO, 取值范围：

  END_NO

  END_PSP

  END_USP

  END_PSP_USP

  END_X_NO

  END_X_PSP

  END_X_USP

  END_X_PSP_USP

  END_T_N

  END_T_PSP

END_T_USP
END_T_PSPS_USP
END_B6
END_B6_ENCAPS
END_BM
END_DX6
END_DX4
EDN_DT6
END_DT4
END_DT46
END_DX2
END_DX2V
END_DX2U
END_DX2M
END_S
END_B6_RED
END_B6_ENCAPS_RED
END_WITH_USD
END_PSP_USD
END_USP_USD
END_PSP_USP_USD
END_X_USD
END_X_PSP_USD
END_X_USP_USD
END_X_PSP_USP_USD
END_T_USD
END_T_PSP_USD
END_T_USP_USD
END_T_PSP_USP_USD

- **EnableCustom** (*bool*) -- 使能自定义端点行为, 默认值：False
- **CustomFunc** (*int*) -- 自定义端点行为, 默认值：0
- **SID** (*str*) -- 指定通告的 SRv6 SID, 默认值：”::1”, 取值范围：有效 IPv6 地址

返回 Isis Neighbor Srv6 Lan EndX Sid Sub Tlv 对象

返回类型 (IsisSrv6LanEndXSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Srv6 Lan EndX Sid Sub Tlv | Neighbor=${Neighbor} |
```

static create_isis_neighbor_te_config(*Neighbor*, ***kwargs*)

创建 ISIS 邻居 TLV 的 Te Config 对象

参数 **Neighbor** (IsisNeighborConfig) -- ISIS Neighbor TLV 对象, 类型为: object

关键字参数

- **EnableInterfaceIp** (*bool*) -- 是否包含本地 IPv4 地址, 默认值：False
- **InterfaceIp** (*str*) -- 本地 IPv4 地址, 取值范围：有效的 ip 地址, 默认值：'0.0.0.0'
- **EnableNeighborIp** (*bool*) -- 是否包含邻居 IPv4 地址, 默认值：False
- **NeighborIp** (*int*) -- 邻居 IPv4 地址, 取值范围：有效的 ip 地址, 默认值：10
- **EnableInterfaceIpv6** (*bool*) -- 是否包含本地 IPv6 地址, 默认值：False
- **InterfaceIpv6** (*str*) -- 本地 IPv6 地址, 取值范围：有效的 ipv6 地址, 默认值：'2000::1'
- **EnableNeighborIpv6** (*bool*) -- 是否包含邻居 IPv6 地址, 默认值：False
- **NeighborIpv6** (*str*) -- 邻居 IPv6 地址, 取值范围：有效的 ipv6 地址, 默认值：'2000::1'
- **EnableTeGroup** (*bool*) -- 是否包含 TE 组, 默认值：False
- **TeGroup** (*int*) -- TE 组, 取值范围：0-4294967295, 默认值：1
- **EnableMaxBandwidth** (*bool*) -- 是否包含最大带宽值, 默认值：False
- **MaximunLink** (*int*) -- 最大带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：1000
- **EnableResBandwidth** (*bool*) -- 是否包含预留带宽值, 默认值：False
- **MaximumReservableLink** (*int*) -- 最大预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：1000
- **EnableUnresBandwidth** (*bool*) -- 是否包含未预留带宽优先级, 默认值：False
- **UnreservedBandwidth0** (*int*) -- 优先级 0 的未预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth1** (*int*) -- 优先级 1 的未预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth2** (*int*) -- 优先级 2 的未预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth3** (*int*) -- 优先级 3 的未预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth4** (*int*) -- 优先级 4 的未预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：0

- **UnreservedBandwidth5** (*int*) -- 优先级 5 的未预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth6** (*int*) -- 优先级 6 的未预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth7** (*int*) -- 优先级 7 的未预留带宽值 (字节/秒), 取值范围：0-4294967295, 默认值：0

返回 ISIS Neighbor TLV Te Config 对象

返回类型 (IsisTEConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Neighbor} | Create Isis Neighbor Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Te Config | Neighbor=${Neighbor} |
```

static create_isis_neighbor_tlv(*Lsp*, ***kwargs*)

创建 ISIS 邻居 TLV 对象

参数 **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **SystemId** (*str*) -- 邻居系统 ID, 取值范围：有效的 MAC 地址, 默认值："00:00:00:00:00:01"
- **PseudonodeSystemId** (*int*) -- 伪节点 ID, 取值范围：0-255, 默认值：0
- **NarrowMetric** (*int*) -- 默认度量, 取值范围：0-63, 默认值：1
- **WideMetric** (*int*) -- 扩展度量, 取值范围：0-16777214, 默认值：10

返回 ISIS Neighbor TLV 对象

返回类型 (IsisNeighborConfig)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Neighbor Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02 |
```

static create_isis_sr_binding_tlv(*Session*, *Lsp*, ***kwargs*)

创建 ISIS Binding TLV 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **EnableMt** (*bool*) -- 使能多拓扑, 默认值：False

- **MtId** (*str*) -- 多拓扑 ID, 默认值：STANDARD, 取值范围：

  STANDARD

  IPV6_ROUTING
- **Flags** (*list*) -- 标签, 默认值：['NOSHOW'], 取值范围：

  NOSHOW

  FBIT

  MBIT

  SBIT

  DBIT

  ABIT
- **Weight** (*int*) -- 权重, 默认值：0, 取值范围：0-255
- **Range** (*int*) -- 范围, 默认值：1, 取值范围：0-65535
- **Ipv4Version** (*bool*) -- 默认值：True
- **Ipv4Prefix** (*str*) -- IPv4 前缀, 默认值："192.0.0.1"
- **Ipv4PrefixLength** (*int*) -- IPv4 前缀长度, 默认值：1, 取值范围：1-32
- **Ipv6Prefix** (*str*) -- IPv6 前缀, 默认值："2000::1
- **Ipv6PrefixLength** (*int*) -- IPv6 前缀长度, 默认值：64, 取值范围：1-128

返回 ISIS Neighbor TLV 对象

返回类型 (IsisSrBindingTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Binding Tlv | Session={Session} | Lsp=${LSP} |␣
↪Ipv4Version=False |
```

static create_isis_srv6_end_sid_sub_tlv(*Session*, *Location*, ***kwargs*)

创建 ISIS Capability Srms Preference Sub Tlv 对象

参数 **Location** (IsisSrv6LocatorTlv) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：['UNKNOWN'], 取值范围：

  UNKNOWN

  UNUSED0

  UNUSED1

  UNUSED2

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

UNUSED7

- **EndpointFunc** (*str*) -- 端点行为, 默认值：END_NO, 取值范围：
  END_NO
  END_PSP
  END_USP
  END_PSP_USP
  END_X_NO
  END_X_PSP
  END_X_USP
  END_X_PSP_USP
  END_T_NO
  END_T_PSP
  END_T_USP
  END_T_PSPS_USP
  END_B6
  END_B6_ENCAPS
  END_BM
  END_DX6
  END_DX4
  EDN_DT6
  END_DT4
  END_DT46
  END_DX2
  END_DX2V
  END_DX2U
  END_DX2M
  END_S
  END_B6_RED
  END_B6_ENCAPS_RED
  END_WITH_USD
  END_PSP_USD
  END_USP_USD
  END_PSP_USP_USD
  END_X_USD
  END_X_PSP_USD
  END_X_USP_USD
  END_X_PSP_USP_USD
  END_T_USD
  END_T_PSP_USD

END_T_USP_USD

END_T_PSP_USP_USD

- **EnableCustom** (*bool*) -- 使能自定义端点行为, 默认值：False
- **CustomFunc** (*int*) -- 自定义端点行为, 默认值：0, 取值范围：0-65535
- **SID** (*str*) -- 指定通告的 SRv6 SID, 默认值：”::1”, 取值范围：IPv6 地址

返回 ISIS Capability Srms Preference Sub TLV 对象

返回类型 (IsisSrv6EndSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Location} | Create Isis Srv6 Location Tlv | Lsp=${LSP} | Algorithm=1 |
| Create Isis Srv6 End Sid Sub Tlv | Session={Session} | Location=$
↪{Location} |
```

static create_isis_srv6_location_tlv(*Session*, *Lsp*, ***kwargs*)

创建 ISIS Binding TLV 对象

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Lsp** (IsisLspConfig) -- ISIS LSP 对象, 类型为：object

关键字参数

- **MtId** (*str*) -- 多拓扑 ID, 默认值：STANDARD, 取值范围：

  UNKNOWN

  D_BIT

  A_BIT

  UNUSED2

  UNUSED3

  UNUSED4

  UNUSED5

  UNUSED6

  UNUSED7

- **Metric** (*int*) -- 指定度量值, 默认值：0, 取值范围：0-4294967295
- **Flags** (*list*) -- 标签, 默认值：[’UNKNOWN’], 取值范围：
- **Algorithm** (*int*) -- Locator 关联算法, 类型为：number, 取值范围：0-255, 默认值：0
- **NumLocator** (*int*) -- Locator 数量, 取值范围：0-4294967295, 默认值：1
- **LocatorSize** (*int*) -- 定位器大小, 取值范围：1-128, 默认值：64
- **Locator** (*str*) -- 定位器, 默认值：”aaaa:1:1:1::”, 取值范围：IPv6 地址
- **LocatorStep** (*int*) -- 定位器步长, 默认值：1, 取值范围：0-65535

返回 ISIS Srv6 Location TLV 对象

返回类型 (IsisSrv6LocatorTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Srv6 Location Tlv | Session={Session} | Lsp=${LSP} |␣
↪Algorithm=1 |
```

static create_isis_tlv_bier_Mpls_sub_sub_tlv(*Bier*, ***kwargs*)

创建 ISIS Tlv Bier Mpls Sub Sub Tlv 对象

参数 **Bier** (IsisBierSubTlv) -- ISIS Bierv6 Sub Tlv 对象, 类型为：object

关键字参数

- **MaxSI** (*int*) -- 指定最大 Set ID, 默认值：1, 取值范围：0-255
- **LabelorBiftId** (*int*) -- 指定标签范围中的起始标签值, 默认值：100, 取值范围：0-4294967295
- **BSLength** (*int*) -- 指定本地比特串的长度, 默认值：1, 取值范围：0-15

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisBierMplsSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Ipv6} | Create Isis Ipv6 Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Bier Sub Tlv | Tlv=${Ipv6} |
| Create Isis Tlv Bier Mpls Sub Sub Tlv | Bier=${Ipv6} |
```

static create_isis_tlv_bier_sub_tlv(*Tlv*, ***kwargs*)

创建 ISIS Tlv Bier Sub Tlv 对象

参数 **Tlv** (IsisIpv4TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **BFRId** (*int*) -- 指定 BFR（Bit Forwarding Router，比特转发路由器）ID, 取值范围：1-65535, 默认值：1
- **SubDomainId** (*int*) -- 指定 BIER 子域 ID, 取值范围：0-255, 默认值：1
- **IgpAlgorithm** (*int*) -- IGP 算法, 取值范围：0-255, 默认值：0
- **BierAlgorithm** (*int*) -- BIER 算法, 取值范围：0-255, 默认值：0

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisBierSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Ipv4} | Create Isis Ipv4 Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Bier Sub Tlv | Tlv=${Ipv4} |
```

static create_isis_tlv_bierv6_bift_id_sub_tlv(*Bier*, ***kwargs*)

创建 ISIS Tlv Bierv6 Bift Id Sub Tlv 对象

参数 **Bier** (IsisIpv6TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Type** (*int*) -- 指定 Type 字段值, 默认值：7, 取值范围：0-255
- **MPRA** (*str*) -- 指定 MPRA 地址, 默认值：'::1', 取值范围：有效 IPv6 地址

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisBierBiftIdSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Ipv6} | Create Isis Ipv6 Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Bierv6 Bift Id Sub Tlv | Tlv=${Ipv6} |
```

static create_isis_tlv_bierv6_sub_sub_tlv(*Bier*, ***kwargs*)

创建 ISIS Tlv Bierv6 Sub Sub Tlv 对象

参数 **Bier** (IsisBierSubTlv) -- ISIS Bierv6 Sub Tlv 对象, 类型为：object

关键字参数

- **MaxSI** (*int*) -- 指定最大 Set ID, 默认值：1, 取值范围：0-255
- **LabelorBiftId** (*int*) -- 指定标签范围中的起始标签值, 默认值：100, 取值范围：0-4294967295
- **BSLength** (*int*) -- 指定本地比特串的长度, 默认值：1, 取值范围：0-15

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisBierMplsSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Ipv6} | Create Isis Ipv6 Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Bier Sub Tlv | Tlv=${Ipv6} |
| Create Isis Tlv Bierv6 Sub Sub Tlv | Bier=${Ipv6} |
```

static create_isis_tlv_end_bier_sub_tlv(*Bier*, ***kwargs*)

创建 ISIS Tlv End Bier Sub Tlv 对象

参数 **Bier** (IsisIpv6TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Type** (*int*) -- 指定 Type 字段值。取值范围：0-255, 默认值：3
- **EndBierAddr** (*str*) -- 指定 End.BIER SID, 默认值：”::1”, 取值范围：有效 IPv6 地址

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisEndBierSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Ipv6} | Create Isis Ipv6 Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Tlv End Bier Sub Tlv | Tlv=${Ipv6} |
```

static create_isis_tlv_flex_algorithm_prefix_metric_sub_tlv(*Tlv*, ***kwargs*)

创建 ISIS Tlv Flex Algorithm Prefix Metric Sid Sub Tlv 对象

参数 **Tlv** (IsisIpv4TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Algorithm** (*int*) -- Locator 关联算法, 类型为：number, 取值范围：128-255, 默认值：128
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisFlexAlgoPrefixMetricSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Ipv4} | Create Isis Ipv4 Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Flex Algorithm Prefix Metric Sub Tlv | Tlv=${Ipv4} |
```

static create_isis_tlv_prefix_sid_sub_tlv(*Session*, *Tlv*, ***kwargs*)

创建 ISIS Tlv Prefix Sid Sub Tlv 对象

参数 **Tlv** (IsisIpv4TlvConfig) -- ISIS Tlv 对象, 类型为：object

关键字参数

- **Flags** (*list*) -- 选择一个或多个包含在 TLV 中的标志位, 默认值：['NOSHOW', 'NOPHP'], 取值范围：

  NOSHOW

ADVERTISEMENT

NODESID

NOPHP

EXPLICIT

VALUE

LOCAL

- **Sid** (*int*) -- SID/Label, 默认值：0, 取值范围：0-4294967295
- **Algorithm** (*int*) -- 指定计算到其他节点/前缀的可达信息的算法，指定 SID 关联的算法, 默认值：0, 取值范围：0-255
- **PrefixSidStep** (*int*) -- 默认值：1

返回 ISIS Prefix Sid Sub TLV 对象

返回类型 (IsisSrPrefixSidSubTlv)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${Ipv4} | Create Isis Ipv4 Tlv | Lsp=${LSP} |␣
↪SystemId=00:00:00:00:00:02 |
| Create Isis Tlv Prefix Sid Sub Tlv | Tlv=${Ipv4} |
```

static create_l2tp(*Port*, ***kwargs*)

创建 L2tp 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- L2tp 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 L2tp 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EmulationMode** (*str*) -- L2TP 角色, 默认值：LAC, 取值范围：

  LAC

  LNS
- **TunnelCountPerNode** (*int*) -- 每 LAC/LNS 隧道数, 取值范围：1-32768, 默认值：1
- **SessionCountPerTunnel** (*int*) -- 每隧道会话数, 取值范围：0-65535, 默认值：0
- **TunnelStartingId** (*int*) -- 隧道起始 ID, 取值范围：1-65535, 默认值：1
- **SessionStartingId** (*int*) -- 会话起始 ID, 取值范围：1-65535, 默认值：1
- **UdpSourcePort** (*int*) -- UDP 源端口, 取值范围：1-65535, 默认值：1701
- **UdpChecksumEnabled** (*bool*) -- 使能 UDP 校验和, 默认值：True
- **RetryTunnelCreationEnabled** (*bool*) -- 使能隧道重试, 默认值：False
- **TunnelCreationTimeout** (*int*) -- 隧道建立超时 (secs), 取值范围：1-65535, 默认值：5

- **MaxTunnelCreationTimes** (*int*) -- 隧道建立最大尝试次数, 取值范围：1-65535, 默认值：5
- **HostName** (*str*) -- 主机名, 取值范围：string length in [1,255], 默认值：xinertel
- **EnableAuthentication** (*bool*) -- 使能认证, 默认值：True
- **IncomingTunnelPassword** (*str*) -- Incoming 隧道密码, 取值范围：string length in [1,255], 默认值：xinertel
- **OutgoingTunnelPassword** (*str*) -- Outgoing 隧道密码, 取值范围：string length in [1,255], 默认值：xinertel
- **HelloEnabled** (*bool*) -- 使能 Hello, 默认值：False
- **HelloInterval** (*int*) -- Hello 间隔 (secs), 取值范围：1-255, 默认值：60
- **TxBitRate** (*int*) -- 发送 bps 速率 (bits/sec), 取值范围：1-65535, 默认值：56000
- **BearerCapabilities** (*str*) -- 负载能力, 默认值：ANALOG, 取值范围：

  DIGITAL

  ANALOG

  BOTH
- **BearerType** (*str*) -- 负载类型, 默认值：ANALOG, 取值范围：

  DIGITAL

  ANALOG
- **FrameCapabilities** (*str*) -- 帧能力, 默认值：SYNC, 取值范围：

  SYNC

  ASYNC

  BOTH
- **FrameType** (*str*) -- 帧类型, 默认值：SYNC, 取值范围：

  SYNC

  ASYNC
- **CallingNumberEnabled** (*bool*) -- 使能 Calling Number, 默认值：False
- **CallingNumber** (*str*) -- 隧道的 Calling Number, 默认值：xinertel
- **RxWindowSize** (*int*) -- 接收窗口大小, 取值范围：1-65535, 默认值：4
- **UseGatewayAsRemoteIp** (*bool*) -- 使用网关作为远端地址, 默认值：True
- **RemoteIpv4Address** (*str*) -- 远端 IPv4 地址, 取值范围：IPv4 地址, 默认值：2.1.1.1
- **RemoteIpv4AddressStep** (*str*) -- 远端 IPv4 地址跳变, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **RemoteIpv6Address** (*str*) -- 远端 IPv6 地址, 取值范围：IPv6 地址, 默认值：2000::1
- **RemoteIpv6AddressStep** (*str*) -- 远端 IPv6 地址跳变, 取值范围：IPv6 地址, 默认值：::1
- **LcpProxyMode** (*str*) -- LCP 代理模式, 默认值：NONE, 取值范围：

  NONE

  LCP

LCP_AUTH

- **ForceLcpRenegotiation** (*bool*) -- 强制 LCP 重协商, 默认值：False
- **Ipv4TosValue** (*hex int*) -- IPv4 TOS 值, 取值范围：1-65535, 默认值：0xc0
- **Ipv6TrafficClassValue** (*hex int*) -- IPv6 Traffic Class Value, 取值范围：1-65535, 默认值：0x0
- **HideFramingCapabilities** (*bool*) -- 默认值：False
- **HideBearerCapabilities** (*bool*) -- 默认值：False
- **HideAssignedTunnelId** (*bool*) -- 默认值：False
- **HideChallenge** (*bool*) -- 默认值：False
- **HideChallengeResponse** (*bool*) -- 默认值：False
- **HideAssignedSessionId** (*bool*) -- 默认值：False
- **HideCallSerialNumber** (*bool*) -- 默认值：False
- **HideFramingType** (*bool*) -- 默认值：False
- **HideCallingNumber** (*bool*) -- 默认值：False
- **HideTxConnectSpeed** (*bool*) -- 默认值：False
- **HideLastSentLcpConfReq** (*bool*) -- 默认值：False
- **HideLastReceivedLcpConfReq** (*bool*) -- 默认值：False
- **HideProxyAuthenType** (*bool*) -- 默认值：False
- **HideProxyAuthenName** (*bool*) -- 默认值：False
- **HideProxyAuthenChallenge** (*bool*) -- 默认值：False
- **HideProxyAuthenId** (*bool*) -- 默认值：False
- **HideProxyAuthenResponse** (*bool*) -- 默认值：False

返回 L2tp 协议会话对, 类型：object

返回类型 (L2tp)

实际案例

```
| Create L2tp | Port=${Port} |
```

static create_ldp(*Port*, ***kwargs*)

创建 LDP 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **HelloType** (*str*) -- Hello 类型, 类型为：string, 默认值：DIRECT, 取值范围:

  DIRECT

  TARGETED

  DIRECT_TARGETED

- **LabelAdvertType** (*str*) -- 标签分配方式, 类型为：string, 默认值：DIRECT, 取值范围:

  DU

  DOD

- **TransportMode** (*str*) -- Transport Address TLV 模式, 类型为：string, 默认值：TESTER_IP, 取值范围:

  TESTER_IP

  ROUTER_ID

  NONE

- **DutIpv4Address** (*int*) -- DUT IPv4 地址, 类型为：number, 型为：string, 默认值：2.1.1.1, 取值范围：IPv4 地址

- **DirectHelloInterval** (*int*) -- 直连 Hello 发送间隔 (sec), 类型为：number, 默认值：5, 取值范围：1-21845

- **TargetedHelloInterval** (*int*) -- 远端 Hello 发送间隔 (sec), 类型为：number, 默认值：15, 取值范围：1-21845

- **KeepAliveInterval** (*int*) -- 保活间隔 (sec), 类型为：number, 默认值：60, 取值范围：1-21845

- **LabelReqRetryCount** (*int*) -- 标签请求间隔 (sec), 类型为：number, 默认值：10, 取值范围：1-65535

- **LabelReqRetryInterval** (*int*) -- 标签请求重试次数, 类型为：number, 默认值：60, 取值范围：1-65535

- **Authentication** (*str*) -- 鉴权类型, 类型为：string, 默认值：DIRECT, 取值范围:

  NONE

  MD5

- **Password** (*str*) -- 密码, 类型为：string, 默认值：xinertel

- **EgressLabel** (*str*) -- 出标签方式, 类型为：string, 默认值：DIRECT, 取值范围:

  NEXT_AVAILABLE

  IMPLICIT

  EXPLICIT

- **MinLabel** (*int*) -- 最小标签值, 类型为：number, 默认值：16, 取值范围：0-1048575

- **EnableLspResult** (*bool*) -- LSP 统计使能, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **EnablePseudowireLspResult** (*bool*) -- 伪线 LSP 统计使能, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **LspBindMode** (*str*) -- LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

- **VcLspBindMode** (*str*) -- 虚拟电路 LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

- **GeneralizedLspBindMode** (*str*) -- 通用伪线 LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

返回 LDP 协议会话对象, 类型：object

返回类型 (Ldp)

实际案例

```
| Create Ldp | Port=${Port} |
```

static create_ldp_fec_128(*Session*, ***kwargs*)

创建 LDP FEC 128 对象

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP FEC 128 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP FEC 128, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **ControlWordEnable** (*bool*) -- 控制字使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Encapsulation** (*str*) -- 封装类型 (hex), 类型为：string, 默认值：PREFIX_FEC, 取值范围:

  ETHERNET_TAGGED_MODE

  ETHERNET

  CEM

- **GroupId** (*int*) -- 组 ID, 类型为：number, 默认值：1, 取值范围：1-65535
- **InterfaceMtu** (*int*) -- 接口 MTU, 类型为：number, 默认值：1500, 取值范围：1-65535
- **IncludePwStatusTlv** (*bool*) -- 伪线状态码使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **PwStatusCode** (*list*) -- 伪线状态码, 类型为：list, 默认值：PW_NOT_FORWARDING, 取值范围:

  PW_NOT_FORWARDING

  LOCAL_AC_RX_FAULT

  LOCAL_AC_TX_FAULT

LOCAL_PSN_PW_RX_FAULT

LOCAL_PSN_PW_TX_FAULT

- **UseCustomPwStatusTlv** (*bool*) -- 自定义伪线状态码使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CustomPwStatusCode** (*int*) -- 自定义伪线状态码, 类型为：number, 默认值：0, 取值范围: 0-4294967295
- **VcCount** (*int*) -- VC 数量, 类型为：number, 默认值：1, 取值范围：0-4294967295
- **StartVcId** (*int*) -- 起始 VC, 类型为：number, 默认值：1, 取值范围：0-4294967295
- **VcIdStep** (*int*) -- VC 跳变步长, 类型为：number, 默认值：1, 取值范围: 0-4294967295

返回 LDP FEC 128 对象列表, 类型: list

返回类型 (LdpFec128LspConfig)

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Ingress} | Create Ldp Fec 128 | Session=${Session} |
```

static create_ldp_fec_129(*Session*, ***kwargs*)

创建 LDP FEC 129 对象

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP FEC 129 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP FEC 129, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **ControlWordEnable** (*bool*) -- 控制字使能, 类型为: bool, 取值范围: True 或 False, 默认值：False
- **Encapsulation** (*str*) -- 封装类型 (hex), 类型为：string, 默认值：PREFIX_FEC, 取值范围:

  ETHERNET_TAGGED_MODE

  ETHERNET

  CEM

- **GroupId** (*int*) -- 组 ID, 类型为：number, 默认值：1, 取值范围：1-65535
- **InterfaceMtu** (*int*) -- 接口 MTU, 类型为：number, 默认值：1500, 取值范围: 1-65535
- **IncludePwStatusTlv** (*bool*) -- 伪线状态码使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **PwStatusCode** (*list*) -- 伪线状态码, 类型为: list, 默认值: PW_NOT_FORWARDING, 取值范围:

  PW_NOT_FORWARDING

  LOCAL_AC_RX_FAULT

  LOCAL_AC_TX_FAULT

LOCAL_PSN_PW_RX_FAULT

LOCAL_PSN_PW_TX_FAULT

- **UseCustomPwStatusTlv** (*bool*) -- 自定义伪线状态码使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **CustomPwStatusCode** (*int*) -- 自定义伪线状态码, 类型为：number, 默认值：0, 取值范围: 0-4294967295
- **PwCount** (*int*) -- PW 数量, 类型为:number, 默认值:1, 取值范围: 0-65535
- **Agi** (*str*) -- 起始 Agi, 型为：string, 默认值：100:1, 取值范围: IPv6 地址
- **AgiStep** (*str*) -- Agi 跳变步长, 型为：string, 默认值：0:1, 取值范围: IPv6 地址
- **Saii** (*str*) -- 起始 Saii, 型为：string, 默认值：10.0.0.1, 取值范围: IPv4 地址
- **SaiiStep** (*str*) -- Saii 跳变步长, 型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **Taii** (*str*) -- 起始 Taii, 型为：string, 默认值：192.0.0.1, 取值范围: IPv4 地址
- **TaiiStep** (*str*) -- Taii 跳变步长, 型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址

返回 LDP FEC 129 对象列表, 类型: list

返回类型 (LdpFec129LspConfig)

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Ingress} | Create Ldp Fec 129 | Session=${Session} |
```

static create_ldp_ipv4_egress(*Session*, ***kwargs*)

创建 LDP IPv4 Egress 对象

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP IPv4 Egress 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP IPv4 Egress, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **FecType** (*str*) -- Fec 类型, 类型为：string, 默认值：PREFIX_FEC, 取值范围:

  PREFIX_FEC

  HOST_FEC
- **LspCount** (*int*) -- Lsp 数量, 类型为：number, 默认值：1, 取值范围：1-65535
- **StartIpv4Prefix** -- Lsp IPv4 前缀地址, 型为:string, 默认值:192.0.1.0, 取值范围: IPv4 地址
- **PrefixLength** (*int*) -- Lsp IPv4 前缀长度, 类型为：number, 默认值：24, 取值范围: 1-32

- **PrefixStep** (*int*) -- Lsp IPv4 前缀跳变步长, 类型为：number, 默认值：1, 取值范围：1-65535
- **Ipv4PrefixStep** -- Lsp IPv4 前缀地址跳变步长, 型为：string, 默认值：0.0.1.0, 取值范围：IPv4 地址

返回 LDP IPv4 Egress 对象列表, 类型：list

返回类型 (LdpIpv4EgressLspConfig)

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Ldp Ipv4 Egress | Session=${Session} |
```

static create_ldp_ipv4_ingress(*Session*, ***kwargs*)

创建 LDP IPv4 Ingress 对象

参数 **Session** (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP IPv4 Ingress 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP IPv4 Ingress, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **FecType** (*str*) -- Fec 类型, 类型为：string, 默认值：PREFIX_FEC, 取值范围:

  PREFIX_FEC

  HOST_FEC
- **LspCount** (*int*) -- Lsp 数量, 类型为：number, 默认值：1, 取值范围：1-65535
- **StartIpv4Prefix** -- Lsp IPv4 前缀地址, 型为：string, 默认值：192.0.1.0, 取值范围：IPv4 地址
- **PrefixLength** (*int*) -- Lsp IPv4 前缀长度, 类型为：number, 默认值：24, 取值范围：1-32
- **PrefixStep** (*int*) -- Lsp IPv4 前缀跳变步长, 类型为：number, 默认值：1, 取值范围：1-65535
- **Ipv4PrefixStep** -- Lsp IPv4 前缀地址跳变步长, 型为：string, 默认值：0.0.1.0, 取值范围：IPv4 地址

返回 LDP IPv4 Ingress 对象列表, 类型：list

返回类型 (LdpIpv4IngressLspConfig)

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Ingress} | Create Ldp Ipv4 Ingress | Session=${Session} |
```

static create_lsp_ping(*Port*, ***kwargs*)

创建 Lsp Ping 会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- Lsp Ping 会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 Lsp Ping 会话, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Lsp Ping 会话对象, 类型：object

返回类型 (LspPing)

实际案例

```
| Create Lsp Ping | Port=${Port} |
```

static create_lsp_ping_echo_request(*Sessions*, ***kwargs*)

创建 Lsp Ping Echo Request 对象

Args：

Session (LspPing): Lsp Ping 会话对象, 类型为：object / list

关键字参数

- **OperationMode** (*list*) -- Operation 模式, 默认值：['PING'], 取值范围：

  PING

  TRACE

- **ReplyMode** (*str*) -- Echo Reply 模式, 默认值：REPLYVIAUDP, 取值范围：

  NOTREPLY

  REPLYVIAUDP

- **PingInterval** (*int*) -- Ping 发送测试包的时间间隔（秒）, 默认值：4, 取值范围：1-65535
- **PingTimeOut** (*int*) -- Ping 探测超时时间（秒）, 默认值：2, 取值范围：1-60
- **TraceInterval** (*int*) -- Trace 发送测试包的时间间隔（秒）, 默认值：120, 取值范围：1-65535
- **TraceTimeOut** (*int*) -- Trace 探测超时时间（秒）, 默认值：2, 取值范围：1-60
- **InnerLabel** (*str*) -- 标签, 默认值：NONE, 取值范围：

  NONE

  LDPIPv4

  VPNIPv4

  SEGMENT_ROUTING

- **OuterLabel** (*str*) -- 标签, 默认值：NONE, 取值范围：

  NONE

  LDPIPv4

  VPNIPv4

  SEGMENT_ROUTING
- **TimeToLive** (*int*) -- 生存时间, 默认值：255, 取值范围：1-255
- **ExpBits** (*int*) -- 实验比特位的值, 默认值：0, 取值范围：0-7
- **PadMode** (*str*) -- 填充模式, 默认值：WITHOUT_PAD, 取值范围：

  WITHOUT_PAD

  DROP_PAD

  COPY_PAD
- **Data** (*int*) -- 填充数据, 默认值：”, 取值范围：0-255
- **DesIpv4Addr** (*str*) -- 目的地址, 默认值：”127.0.0.1”，取值范围：有效的 ipv4 地址
- **ValidateFecStack** (*bool*) -- 校验 FEC Stack, 默认值：False, 取值范围：True 或 False
- **DownstreamMappingTlvType** (*str*) -- Downstream Mapping TLV 类型, 默认值：DOWNSTREAM_DETAILED_MAPPING_TLV, 取值范围：

  DOWNSTREAM_MAPPING_TLV

  DOWNSTREAM_DETAILED_MAPPING_TLV

返回 Lsp Ping Echo Request 对象, 类型：object / list

返回类型 (LspPingEchoRequestConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| Create Lsp Ping Echo Request | Sessions=${LspPing} |
```

static create_lsp_ping_fec_ldp_ipv4(*EchoRequests*, ***kwargs*)

创建 Lsp Ping Fec Ldp Ipv4 对象

Args：

EchoRequests (LspPingEchoRequestConfig): Lsp Ping Echo Request 对象, 类型为：object / list

关键字参数

- **Count** (*int*) -- 数量, 默认值：1, 取值范围：1-65535
- **StartAddr** (*str*) -- IPv4 地址, 默认值：”172.0.0.1”，取值范围：有效的 ipv4 地址
- **PrefixLength** (*int*) -- 前缀长度, 默认值：24, 取值范围：1-32
- **Step** (*int*) -- 步长, 默认值：1, 取值范围：1-255

返回 Lsp Ping Fec Ldp Ipv4 对象, 类型：object / list

返回类型 (LspPingFecLdpIpv4PrefixConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| Create Lsp Ping Fec Ldp Ipv4 | EchoRequests=${EchoRequest} |
```

static create_lsp_ping_fec_segment_routing(*EchoRequests*, ***kwargs*)

创建 Lsp Ping Fec Segment Routing 对象

Args:

EchoRequests (LspPingEchoRequestConfig): Lsp Ping Echo Request 对象, 类型为: object / list

关键字参数

- **IgpProtocol** (*str*) -- FEC 校验使用的 IGP 协议, 默认值: ISIS, 取值范围:

  OSPF

  ISIS

- **PrefixCount** (*int*) -- 前缀数量, 默认值: 1, 取值范围: 1-4294967295
- **PrefixAddrIncrement** (*int*) -- 地址步长, 默认值: 1, 取值范围: 1-4294967295

返回 Lsp Ping Fec Segment Routing 对象, 类型: object / list

返回类型 (LspPingFecSrConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| Create Lsp Ping Fec Segment Routing | EchoRequests=${EchoRequest} |
```

static create_lsp_ping_fec_sr_adjacency(*Srs*, ***kwargs*)

创建 Lsp Ping Fec Sr Adjacency 对象

Args:

Srs (LspPingFecSrConfig): Lsp Ping Fec Segment Routing 对象, 类型为: object / list

关键字参数

- **IsisSystemId** (*str*) -- ISIS 系统 ID, 默认值: "00:00:94:00:00:01", 取值范围: 有效的 mac 地址
- **IsisLanSystemId** (*str*) -- ISIS LAN 系统 ID, 默认值: "00:00:00:00:00:00", 取值范围: 有效的 mac 地址
- **IsisNeighborId** (*str*) -- ISIS 邻居 ID, 默认值: "00:00:94:00:00:01", 取值范围: 有效的 mac 地址
- **IsisNodeId** (*int*) -- ISIS 节点 ID, 默认值: 0, 取值范围: uint8
- **OspfLinkType** (*str*) -- OSPF 链路类型, 默认值: P2P, 取值范围:

  P2P

  TRANSITNETWORK

  STUBNETWORK

  VIRTUALLINK

- **OspfLinkId** (*str*) -- OSPF 链路 ID, 默认值：”0.0.0.0”，取值范围：有效的 ipv4 地址
- **OspfLinkData** (*str*) -- OSPF 链路数据, 默认值：”0.0.0.0”，取值范围：有效的 ipv4 地址
- **LocalRouterId** (*str*) -- 本地路由器 ID, 默认值：”192.168.1.1”，取值范围：有效的 ipv4 地址
- **RemoteRouterId** (*str*) -- 远端路由器 ID, 默认值：”192.168.1.1”，取值范围：有效的 ipv4 地址
- **LocalInterfaceId** (*str*) -- 本地接口 ID, 默认值：”0.0.0.0”，取值范围：有效的 ipv4 地址
- **RemoteInterfaceId** (*str*) -- 远端接口 ID, 默认值：”0.0.0.0”，取值范围：有效的 ipv4 地址

返回 Lsp Ping Fec Sr Detail 对象, 类型：object / list

返回类型 (LspPingFecSrDetailConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| ${Sr} | Create Lsp Ping Fec Segment Routing | EchoRequests=$
↪{EchoRequest} |
| Create Lsp Ping Fec Sr Ajacency | Srs=${Sr} |
```

static create_lsp_ping_fec_sr_prefix(*Srs*, ***kwargs*)

创建 Lsp Ping Fec Sr Prefix 对象

Args：

Srs (LspPingFecSrConfig): Lsp Ping Fec Segment Routing 对象, 类型为: object / list

关键字参数

- **Prefix** (*str*) -- 前缀地址, 默认值：”192.0.0.1”，取值范围：有效的 ipv4 地址
- **Length** (*int*) -- 前缀地址长度, 默认值：24, 取值范围：1-32
- **Algorithm** (*int*) -- 算法, 默认值：0, 取值范围：uint8

返回 Lsp Ping Fec Sr Detail 对象, 类型：object / list

返回类型 (LspPingFecSrDetailConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| ${Sr} | Create Lsp Ping Fec Segment Routing | EchoRequests=$
↪{EchoRequest} |
| Create Lsp Ping Fec Sr Prefix | Srs=${Sr} |
```

static create_lsp_ping_fec_vpn_ipv4(*EchoRequests*, ***kwargs*)

创建 Lsp Ping Fec Vpn Ipv4 对象

Args：

EchoRequests (LspPingEchoRequestConfig): Lsp Ping Echo Request 对象, 类型为：object / list

关键字参数

- **Count** (*int*) -- 数量, 默认值：1, 取值范围：1-65535
- **StartAddr** (*str*) -- IPv4 地址, 默认值："172.0.0.1"，取值范围：有效的 ipv4 地址
- **PrefixLength** (*int*) -- 前缀长度, 默认值：24, 取值范围：1-32
- **Step** (*int*) -- 步长, 默认值：1, 取值范围：1-255
- **RouteDistinguisher** (*str*) -- 路由标识, 默认值："100:1"，取值范围：匹配格式"uint16:uint32 | ipv4:uint16 | uint32:uint16 | uint16.uint16:uint16"

返回 Lsp Ping Fec Vpn Ipv4 对象, 类型：object / list

返回类型 (LspPingFecVPNIpv4PrefixConfig)

实际案例

```
| ${LspPing} | Create Lsp Ping | Port=${Port} |
| ${EchoRequest} | Create Lsp Ping Echo Request | Sessions=${LspPing} |
| Create Lsp Ping Fec Vpn Ipv4 | EchoRequests=${EchoRequest} |
```

static create_memberships(*Session*, ***kwargs*)

创建组播协议和组播组绑定关系对象

参数 **Session** (Mld, *Igmp*) -- IGMP/MLD 协会话对象, 类型为：object

关键字参数

- **DeviceGroupMapping** (*str*) -- 主机和组播组映射关系, 类型为：str, 默认值：MANYTOMANY, 取值范围：

  MANYTOMANY

  ONETOONE

  ROUNDROBIN

- **SourceFilterMode** (*str*) -- 源地址过滤模式, 类型为：str, 默认值：EXCLUDE, 取值范围：

  INCLUDE

  EXCLUDE

- **UserDefinedSources** (*bool*) -- 自定义源地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SpecifySourcesAsList** (*bool*) -- 配置离散源地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SourceAddressList** (*list*) -- 离散源地址列表, 类型为：list, 取值范围：ipv4 or ipv6 string list
- **NumberOfSources** (*int*) -- 组播组地址掩码, 类型为：number, 取值范围：0-16777215 默认值：1
- **StartingSourceIp** (*str*) -- 组播组起始源地址, 类型为：string, 取值范围：ipv4 or ipv6 string list，, 默认值 ipv4: 192.0.1.0 , ipv6: 2000::1
- **PrefixLength** (*int*) -- 组跳变位, 类型为：number, 取值范围：ipv4: 1-32 默认值：32, ipv6: 1-128 默认值：128

- **Increment** (*int*) -- 跳变步长, 类型为：number, 取值范围：0-16777215 默认值：1

返回 组播协议和组播组绑定关系对象, 类型：object

返回类型 (MldMembershipsConfig)

实际案例

```
| ${Session} | Create Mld | Port=${Port} |
| Create Memberships | Session=${Session} | Start=225.0.1.1 |␣
↪DeviceGroupMapping=ONETOONE |
```

static create_mld(*Port*, ***kwargs*)

创建 MLD 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- MLD 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 MLD 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：MLDV1, 支持版本：

  MLDV1

  MLDV2
- **PackReports** (*bool*) -- 合并报告报文, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InitialJoin** (*bool*) -- 单个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustJoin** (*bool*) -- 多个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustnessVariable** (*int*) -- Robust 值, 类型为：number, 取值范围：2-255, 默认值：2
- **UnsolicitedReportInterval** (*int*) -- 发送初始报文的时间间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **ForceLeave** (*bool*) -- 强制发送 Leave 报文, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TrafficClass** (*hex int*) -- IP 头的 Traffic Class 值, 型为：string, 取值范围：0x0-0xff, 默认值：0xc0

返回 MLD 协议会话对象, 类型：object

返回类型 (Mld)

实际案例

```
| Create Mld | Port=${Port} | Version=MLDV2 |
```

static create_mld_querier(*Port*, ***kwargs*)

创建 MLD Querier 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- MLD Querier 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 MLD Querier 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：MLDV1, 支持版本：

  MLDV1

  MLDV2
- **RobustnessVariable** (*int*) -- 健壮系数, 取值范围：2-255, 默认值: 2
- **Interval** (*int*) -- 查询时间间隔（秒）, 取值范围：0-4294967295, 默认值: 125
- **QueryResponseInterval** (*int*) -- 查询响应时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 10000
- **StartupQueryCount** (*int*) -- 初始查询报文个数, 取值范围：1-255, 默认值：2
- **LastMemberQueryInterval** (*int*) -- 最后成员查询时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 1000
- **LastMemberQueryCount** (*bool*) -- 最后成员查询次数, 取值范围：0-255, 默认值: 2
- **IPv6TrafficClassValue** (*str*) -- 设置 IPv6 头 TrafficClass 值, 取值范围：0x0-0xff, 默认值: 0x0

返回 MLD 协议会话对象, 类型：object

返回类型 (MldQuerier)

实际案例

```
| Create Mld Querier| Port=${Port} | Version=MLDV3 |
```

static create_mpls_wizard(*Type*)

测试仪表创建 MPLS 向导

参数 **Type** (*str*) -- mpls 向导类型, 支持: mpls_ip_vpn mpls_6vpe ldp_vpls

返回 bool：布尔值 (范围：True / False)

实际案例

```
| Create Mpls Wizard | Type=mpls_ip_vpn |
```

static create_multicast_group(*Version='IPv4'*, ***kwargs*)

创建全局组播组对象

参数 **Version** (*str*) -- 组播组 IP 版本，类型 string, 支持 ipv4 和 ipv6

关键字参数

- **Count** (*int*) -- 组播组数量, 类型为：number, 取值范围：0-65535, 默认值：1
- **Mode** (*str*) -- 组播组地址模式, 类型为：string, 默认值：RANGE, 取值范围：

  RANGE

  LIST

  RFC_4814
- **Start** (*str*) -- 组播组地址起始值, 类型为：ipv4/ipv6 string, 默认值：225.0.0.1 或 ff1e::1
- **Number** (*int*) -- 组播组地址数量, 类型为：number, 取值范围：1-268435456, 默认值：1
- **Increment** (*int*) -- 组播组地址步长, 类型为：number, 取值范围：1-268435456, 默认值：1
- **Prefix** (*int*) -- 组播组地址掩码, 类型为：number, 取值范围：ipv4：1-32 默认值：32, ipv6：1-128 默认值：128

返回 全局组播组对象, 类型：object

返回类型 (MldSelectMulticastGroupCommand)

实际案例

```
| Create Multicast Group | Version=IPV4 | Start=225.0.1.1 | Number=20 |
```

static create_ospf(*Port*, ***kwargs*)

创建 OSPFv2 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- OSPFv2 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 OSPFv2 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AreaId** (*str*) -- 区域 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 取值范围：Broadcast 或 P2P, 默认值：Broadcast
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **Cost** (*int*) -- 接口开销, 类型为：number, 取值范围：1-65535, 默认值：10

- **AuthenticationType** (*str*) -- 类型为：string, 取值范围：None Simple 或 MD5, 默认值：None
- **Password** (*str*) -- 密码, 类型为：string, 默认值：Xinertel
- **Md5KeyId** (*int*) -- MD5 密钥, 类型为：number, 取值范围：0-255, 默认值：1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT
- **EnableOspfv2Mtu** (*bool*) -- 使能 OSPF MTU, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GracefulRestartReason** (*str*) -- 平滑重启原因, 类型为：string, 默认值：UNKNOWN, 支持的原因：

  UNKNOWN

  SOFTWARE

  RELOADORUPGRADE

  SWITCH
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **HelloInterval** (*int*) -- Hello 包间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **RouterDeadInterval** (*int*) -- 路由器失效间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：40
- **LsaRetransInterval** (*int*) -- LSA 重传间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：5
- **LsaRefreshTime** (*int*) -- LSA 刷新间隔 (秒), 类型为：number, 取值范围：1-1800, 默认值：1800
- **EnableSrManagement** (*bool*) -- 启用 SR, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 OSPFv2 协议会话对象, 类型：object

返回类型 (OspfRouter)

实际案例

```
| Create Ospf | Port=${Port} |
```

static create_ospf_adj_sid_sub_tlv(*Session*, *ExtendedLinkTlv*, ***kwargs*)

创建 OSPFv2 Adj Sid Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **ExtendedLinkTlv** (Ospfv2ExtendedLinkTlvConfig) -- OSPFv2 Extended Link Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Adj Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- 选项, 类型为：list, 默认值：['ValueIndexFlag', 'LocalGlobalFlag', 'NONE'], 支持选项有：

  BackupFlag

  ValueIndexFlag

  LocalGlobalFlag

  GroupFlag

  PersistentFlag

  NONE
- **MultiTopologyId** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：0-255, 默认值：0
- **Weight** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：0-255, 默认值：0
- **SidLabel** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：1-255, 默认值：1

返回 OSPFv2 Adj Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv2AdjSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | | Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa}␣
↪|
| Create Ospf Adj Sid Sub Tlv | Session=${Session} | ExtendedLinkTlv=$
↪{Tlv} |
```

static create_ospf_asbr_summary_lsa(*Session*, ***kwargs*)

创建 OSPFv2 Asbr Summary LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Summary LSA 的名称, 类型为：string

- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **AsbrRouterId** (*int*) -- 路由个数, 类型为：number, 取值范围：1-1000000, 默认值：1
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Asbr Summary LSA 对象, 类型：object

返回类型 (Ospfv2AsbrSummaryLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Asbr Summary Lsa | Session=${Session} | Age=20 |
```

static create_ospf_bier_mpls_encap_sub_tlv(*Tlv*, ***kwargs*)

创建 OSPFv2 Bier Mpls Encap Sub Tlv 对象

参数 **Tlv** (*Port*) -- OSPFv2 Bier Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Bier Mpls Encap Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TlvType** (*int*) -- Type 字段值, 类型为：number, 取值范围：0-255, 默认值：10
- **MaxSi** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：1-255, 默认值：1

- **Label** (*int*) -- 标签范围中的起始标签值, 类型为：number, 取值范围：0-1048575, 默认值：100

返回 OSPFv2 Bier Mpls Encap Sub Tlv 对象, 类型：object

返回类型 (Ospfv2BierMplsEncapSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| ${SubTlv}| Create Ospf Bier Sub Tlv | Tlv=${Tlv} |
| Create Ospf Bier Mpls Encap Sub Tlv | Tlv=${SubTlv} |
```

static create_ospf_bier_sub_tlv(*Tlv*, ***kwargs*)

创建 OSPFv2 Bier Sub Tlv 对象

参数 **Tlv** (*Port*) -- OSPFv2 Ext Prefix Range Tlv / Ospfv2 Ext Prefix Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Bier Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TlvType** (*int*) -- Type 字段值, 类型为：number, 取值范围：0-255, 默认值：9
- **SubDomainId** (*int*) -- BIER 子域 ID, 类型为：number, 取值范围：1-255, 默认值：1
- **MtId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-255, 默认值：1
- **BfrId** (*int*) -- BFR（Bit Forwarding Router，比特转发路由器）ID, 类型为：number, 取值范围：1-65535, 默认值：1
- **Bar** (*int*) -- BIER 算法, 类型为：number, 取值范围：0-255, 默认值：0
- **Ipa** (*int*) -- IGP 算法, 类型为：number, 取值范围：0-255, 默认值：0

返回 OSPFv2 Bier Sub Tlv 对象, 类型：object

返回类型 (Ospfv2BierSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| Create Ospf Bier Sub Tlv | Tlv=${Tlv} |
```

static create_ospf_custom_sub_tlv(*SrLinkMsdSubTlv*, ***kwargs*)

创建 OSPFv2 Custom Sub Tlv 对象

参数 **SrLinkMsdSubTlv** (Ospfv2SrLinkMsdSubTlvConfig) -- OSPFv2 Sr Link Msd Sub Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Custom Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SubType** (*int*) -- 类型为：number, 取值范围：0-255, 默认值：0
- **SubValue** (*int*) -- 类型为：number, 取值范围：0-255, 默认值：8

返回 OSPFv2 Custom Sub Tlv 对象, 类型：object

返回类型 (Ospfv2CustomMsdSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | | Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa}␣
↪|
| ${SubTlv} | Create Ospf Sr Link Msd Sub Tlv | SrLinkMsdSubTlv=${Tlv} |
| Create Ospf Custom Sub Tlv | SrLinkMsdSubTlv = ${SubTlv} |
```

static create_ospf_ext_prefix_range_tlv(*Session*, *OpaqueExtendedPrefixLsa*, ***kwargs*)

创建 OSPFv2 Ext Prefix Range Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **OpaqueExtendedPrefixLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Extended Prefix LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Ext Prefix Range Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PrefixLength** (*int*) -- 前缀长度, 类型为：number, 取值范围：0-32, 默认值：24
- **AF** (*str*) -- 前缀的地址族, 类型为：string, 默认值：IPv4Unicast, 取值范围：

  IPv4Unicast
- **ExtendedPrefixRange** (*int*) -- 要生成的前缀的数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **ExtendedPrefixFlags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NoneFlag, 支持选项有：

  NoneFlag

  IAInterArea
- **AddressPrefix** (*str*) -- 起始地址前缀, 类型为：string, 默认值：192.0.1.0, 取值范围：有效的 ipv4 地址

返回 OSPFv2 Ext Prefix Range Tlv 对象, 类型：object

返回类型 (Ospfv2ExtPrefixRangeTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Prefix Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Ext Prefix Range Tlv | Session=${Session} |␣
↪OpaqueExtendedPrefixLsa=${Lsa} |
```

static create_ospf_ext_prefix_tlv(*Session*, *OpaqueExtendedPrefixLsa*, ***kwargs*)

创建 OSPFv2 Ext Prefix Range Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **OpaqueExtendedPrefixLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Extended Prefix LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Ext Prefix Range Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- 起始地址前缀, 类型为：string, 默认值：Unspecified, 取值范围：

  Unspecified

  IntraArea

  InterArea

  AsExternal

  NssaExternal
- **AddressPrefix** (*str*) -- 起始地址前缀, 类型为：string, 默认值：192.0.1.0, 取值范围：有效的 ipv4 地址
- **PrefixLength** (*int*) -- 要生成的前缀的数量, 类型为：number, 取值范围：0-32, 默认值：24
- **PrefixTlvBlockCount** (*int*) -- 要生成的前缀的数量, 类型为：number, 取值范围：0-32, 默认值：1
- **AF** (*str*) -- 起始地址前缀, 类型为：string, 默认值：IPv4Unicast, 取值范围：

  IPv4Unicast
- **ExtendedPrefixFlags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NoneFlag, 支持选项有：

  NoneFlag

  AttachFlag

  NodeFlag

返回 OSPFv2 Ext Prefix Tlv 对象, 类型：object

返回类型 (Ospfv2ExtPrefixTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Prefix Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Ext Prefix Range Tlv | Session=${Session} |␣
↪OpaqueExtendedPrefixLsa=${Lsa} |
```

static create_ospf_extended_link_tlv(*OpaqueExtendedLinkLsa*, ***kwargs*)

创建 OSPFv2 Extended link Tlv 对象

参数 **OpaqueExtendedLinkLsa** (Ospfv2OpaqueSrExtLinkLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Extended link Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LinkType** (*str*) -- 前缀的地址族, 类型为：string, 默认值：P2P, 取值范围：

  P2P

  TRANSITNETWORK

  STUBNETWORK

  VIRTUALLINK
- **LinkId** (*str*) -- 起始地址前缀, 类型为：string, 默认值：0.0.0.0, 取值范围：有效的 ipv4 地址
- **LinkData** (*str*) -- 起始地址前缀, 类型为：string, 默认值：0.0.0.0, 取值范围：有效的 ipv4 地址

返回 OSPFv2 Extended link Tlv 对象, 类型：object

返回类型 (Ospfv2ExtendedLinkTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa} |
```

static create_ospf_external_lsa(*Session*, ***kwargs*)

创建 OSPFv2 External LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 External LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1

- **LsType** (*str*) -- LSA 类型, 类型为：string, 默认值：ExtLsaLsType1, 支持选项有：

  ExtLsaLsType1: AS-External(5)

  ExtLsaLsType2: NSSA(7)

- **RouteCount** (*int*) -- 路由个数, 类型为：number, 取值范围：1-1000000, 默认值：1
- **StartNetworkPrefix** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：192.0.1.0
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-32, 默认值：24
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **MetricType** (*str*) -- 选项, 类型为：string, 默认值：ExtLsaLsMetricType1, 支持选项有：

  ExtLsaLsMetricType1

  ExtLsaLsMetricType2

- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：10
- **ForwardingAddress** (*str*) -- 转发地址, 即：LSA 中携带的转发地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：192.0.1.0
- **RouterTag** (*int*) -- 路由标签, 类型为：number, 取值范围：0-2147483647, 默认值：0
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **LsaAutomaticConversion** (*bool*) -- LSA 自动转换, 即：当配置的会话为 NSSA 会话时,Renix 会自动将此处配置的外部 LSA 转换为 NSSA-LSA 进行发送; 当配置的会话为非 NSSA 会话时,Renix 会自动将此处配置的 NSSA-LSA 转换为外部 LSA 进行发送, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 External LSA 对象, 类型：object

返回类型 (Ospfv2ExternalLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf External Lsa | Session=${Session} | Age=20 |
```

static create_ospf_lan_adj_sid_sub_tlv(*Session*, *ExtendedLinkTlv*, ***kwargs*)

创建 OSPFv2 Lan Adj Sid Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **ExtendedLinkTlv** (Ospfv2ExtendedLinkTlvConfig) -- OSPFv2 Extended Link Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Lan Adj Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- 选项, 类型为：list, 默认值：['ValueIndexFlag', 'LocalGlobalFlag', 'NONE'], 支持选项有：

  BackupFlag

  ValueIndexFlag

  LocalGlobalFlag

  GroupFlag

  PersistentFlag

  NONE
- **MultiTopologyId** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：0-255, 默认值：0
- **Weight** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：0-255, 默认值：0
- **NeighborId** (*str*) -- 起始地址前缀, 类型为：string, 默认值：0.0.0.0, 取值范围：有效的 ipv4 地址
- **SidLabel** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：1-255, 默认值：1

返回 OSPFv2 Lan Adj Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv2LanSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | | Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa}␣
↪|
| Create Ospf Lan Adj Sid Sub Tlv | Session=${Session} | Tlv=${Tlv} |
```

static create_ospf_network_atch_router(*NetworkLsa*, ***kwargs*)

创建 OSPFv2 Network LSA Atch Router 对象

参数 **NetworkLsa** (Ospfv2NetworkLsaConfig) -- 测试仪表 OSPFv2 Network LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Network LSA Atch Router 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AttachedRouter** (*str*) -- 附加路由器的 IP 地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0

返回 OSPFv2 Network LSA Atch Router 对象, 类型：object

返回类型 (Ospfv2NetworkAtchRouterConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${NetworkLsa} | Create Ospf Network Lsa | Session=${Session} | Age=20 |
| Create Ospf Network Lsa Atch Router | NetworkLsa=${NetworkLsa} |␣
↪Metric=65535 |
```

static create_ospf_network_lsa(*Session*, ***kwargs*)

创建 OSPFv2 Network LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Network LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **DrIpAddress** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-32, 默认值：24
- **Options** (*list*) -- 选项, 类型为：list, 默认值：NONTBIT | EBIT, 支持选项有：

  NONTBIT

  TOSBIT

EBIT

MCBIT

NPBIT

EABIT

DCBIT

OBIT

DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Network LSA 对象, 类型：object

返回类型 (Ospfv2NetworkLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Network Lsa | Session=${Session} | Age=20 |
```

static create_ospf_opaque_extended_link_lsa(*Session*, ***kwargs*)

创建 OSPFv2 Opaque Extended Link LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 Opaque Extended Link LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Scope** (*str*) -- Tlv 类型, 类型为：string, 默认值：AreaLocal, 支持选项有：

  LinkLocal

  AreaLocal

  AreaSystemWide
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：192.0.0.1
- **Instance** (*int*) -- 实例, 类型为：number, 取值范围：0-16777215, 默认值：1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

DCBIT

OBIT

DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Opaque Extended Link LSA 对象, 类型：object

返回类型 (OspfvOpaqueSrExtLinkLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Opaque Extended Link LSA | Session=${Session} | Age=20 |
```

static create_ospf_opaque_extended_prefix_lsa(*Session*, ***kwargs*)

创建 OSPFv2 Opaque Extended Prefix LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Opaque Extended Prefix LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Scope** (*str*) -- LSA 的泛洪区域, 类型为：string, 默认值：AreaLocal, 取值范围：

  LinkLocal

  AreaLocal

  AreaSystemWide

- **AdvertisingRouterId** (*str*) -- 通告路由器 ID, 类型为：string, 默认值：192.0.0.1, 取值范围：有效的 ipv4 地址
- **Instance** (*int*) -- 指定 LSA 中 Instance 字段的值, 类型为：number, 默认值：1, 取值范围：0-16777215
- **Options** (*list*) -- 类型为：list, 默认值：['NONTBIT', 'EBIT'], 取值范围：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT

- **Age** (*int*) -- LSA 的老化时间。单位为秒, 类型为：number, 默认值：1, 取值范围：0-3600
- **SequenceNumber** (*int*) -- LSA 的序列号, 类型为：number, 默认值：0x80000001, 取值范围：0-4294967295
- **Checksum** (*bool*) -- LSA 的校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Opaque Extended Prefix LSA 对象, 类型：object

返回类型 (Ospfv2OpaqueSrExtPrefixLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Opaque Extended Prefix Lsa | Session=${Session} | Age=20 |
```

static create_ospf_opaque_router_info_lsa(*Session*, ***kwargs*)

创建 OSPFv2 Opaque Router Info LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Opaque Router Info LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Opaque Router Info LSA 对象, 类型：object

返回类型 (Ospfv2OpaqueRouterInfoLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Opaque Router Info Lsa | Session=${Session} | Age=20 |
```

static create_ospf_prefix_sid_sub_tlv(*Session*, *Tlv*, ***kwargs*)

创建 OSPFv2 Prefix Sid Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **Tlv** (Ospfv2ExtPrefixTlvConfig) -- OSPFv2 Ext Prefix Range Tlv / Ospfv2 Ext Prefix Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Prefix Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PrefixSidTlvFlags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NoPhp, 取值范围：

  NoPhp

  MappingServer

  ExplicitNull

  ValueIndex

  LacalGlobal

- **MultiTopologyId** (*int*) -- 指定 MT-ID 的值, 类型为：number, 取值范围：0-255, 默认值：0
- **Algorithm** (*int*) -- 计算到其他节点/前缀的可达信息的算法, 类型为：number, 默认值：0
- **SidIndexLabel** (*int*) -- Flags 中包含 Value/Index 时, 指定标签值; Flags 中不包含 Value/Index 时，指定 SID/Label 范围内的标签偏移值, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **SidIndexLabelStep** (*int*) -- SidIndexLabel 跳变, 类型为：number, 取值范围：0-4294967295, 默认值：1

返回 OSPFv2 Prefix Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv2PrefixSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| Create Ospf Prefix Sid Sub Tlv | Session=${Session} | Tlv=${Tlv} |
```

static create_ospf_router_info_capability_tlv(*OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Router Info Capability Tlv 对象

参数 **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Router Info Capability Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **InfoCapability** (*int*) -- 指定 TLV 值, 类型为：number, 默认值：1, 取值范围：0-255

返回 OSPFv2 Router Info Capability Tlv 对象, 类型：object

返回类型 (Ospfv2RouterInfoCapabilityTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Router Info Capability Tlv | OpaqueRouterInfoLsa=${Lsa} |
```

static create_ospf_router_lsa(*Session*, ***kwargs*)

创建 OSPFv2 Router LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Router LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：192.0.0.1
- **RouterType** (*list*) -- 路由器类型, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  ABR

  ASBR

  VLE

- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Router LSA 对象, 类型：object

返回类型 (Ospfv2RouterLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Router Lsa | Session=${Session} | Age=20 |
```

static create_ospf_router_lsa_link(*RouterLsa*, ***kwargs*)

创建 OSPFv2 Router LSA Link 对象

参数 **RouterLsa** (Ospfv2RouterLsaConfig) -- 测试仪表 OSPFv2 Router LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Router LSA Link 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LinkType** (*str*) -- 链路类型, 类型为：string, 默认值：P2P, 支持选项有：

  P2P

  TRANSITNETWORK

  STUBNETWORK

  VIRTUALLINK
- **LinkId** (*str*) -- 链路状态 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **LinkData** (*str*) -- 链路数据, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：0-65535, 默认值：1

返回 OSPFv2 Router LSA Link 对象, 类型：object

返回类型 (Ospfv2RouterLsaLinksConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${RouterLsa} | Create Ospf Router Lsa | Session=${Session} | Age=20 |
| Create Ospf Router Lsa Link | RouterLsa=${RouterLsa} | Metric=65535 |
```

static create_ospf_sid_label_binding_sub_tlv(*Session*, *Tlv*, ***kwargs*)

创建 OSPFv2 Sid Label Binding Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

- **Tlv** (Ospfv2ExtPrefixTlvConfig) -- OSPFv2 Ext Prefix Range Tlv / Ospfv2 Ext Prefix Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sid Label Binding Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SidLabelBindingTlvFlags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NoneFlag, 取值范围：

  NoneFlag

  MirroringContext
- **Weight** (*int*) -- 进行负载均衡时的权重, 类型为：number, 取值范围：0-255, 默认值：0
- **MultiTopologyId** (*int*) -- 指定 MT-ID 的值, 类型为：number, 取值范围：0-255, 默认值：0
- **SidLabelType** (*str*) -- 标识符（SID 或者标签）, 类型为：string, 默认值：Bit20, 取值范围：

  Bit20

  Bit32
- **SidLabel** (*int*) -- SID/Label Type 为 20-Bit Label 时，指定标签值; SID/Label Type 为 32-Bit SID 时，指定 SID, 类型为：number, 取值范围：0-255, 默认值：16

返回 OSPFv2 Sid Label Binding Sub Tlv 对象, 类型：object

返回类型 (Ospfv2SidLabelBindingSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| Create Ospf Sid Label Binding Sub Tlv | Session=${Session} | Tlv=${Tlv}␣
↪|
```

static create_ospf_sr_algorithm_tlv(*OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Algorithm Tlv 对象

参数 **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Algorithm Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Algorithms** (*int*) -- 类型为：number, 默认值：0

返回 OSPFv2 Sr Algorithm Tlv 对象, 类型：object

返回类型 (Ospfv2SrAlgorithmTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Sr Algorithm Tlv | OpaqueRouterInfoLsa=${Lsa} |
```

static create_ospf_sr_fad_tlv(*Session*, *OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Fad Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Fad Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **FlexAlgo** (*int*) -- 灵活算法 ID, 类型为：number, 默认值：128, 取值范围：128-255
- **MetricType** (*str*) -- 指定算路使用的度量类型, 类型为：str, 默认值：IGP_METRIC, 取值范围：

  IGP_METRIC

  MIN_LINK_DELAY

  TE_METRIC
- **CalcType** (*int*) -- 指定特定 IGP 算法的计算类型, 类型为：number, 默认值：0, 取值范围：0-127
- **Priority** (*int*) -- 指定该 Sub-TLV 的优先级, 类型为：number, 默认值：0, 取值范围：0-255
- **FlexAlgoSubTlv** (*list*) -- 选择灵活算法路径计算要遵循的约束条件, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  EXCLUDE_ADMIN

  INCLUDE_ANY_ADMIN

  INCLUDE_ALL_ADMIN

  DEFINITION_FLAGS

  EXCLUDE_SRLG
- **ExcludeAdmin** (*int*) -- 类型为：number, 默认值：0, 取值范围：0-4294967295
- **IncludeAnyAdmin** (*int*) -- 类型为：number, 默认值：0, 取值范围：0-4294967295
- **IncludeAllAdmin** (*int*) -- 类型为：number, 默认值：0, 取值范围：0-4294967295
- **DefinitionFlags** (*list*) -- 类型为：hex int, 默认值：0x80, 取值范围：0-FF

- **ExcludeSRLG** (*list*) -- 类型为：hex int, 默认值：0x10020000, 取值范围：0-4294967295

返回 OSPFv2 Sr Fad Tlv 对象, 类型：object

返回类型 (Ospfv2FlexAlgoDefinitionTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Sr Fad Tlv | Session=${Session} | OpaqueRouterInfoLsa=${Lsa}
↪ |
```

static create_ospf_sr_fapm_sub_tlv(*Tlv*, ***kwargs*)

创建 OSPFv2 Sr Fapm Sub Tlv 对象

参数 **Tlv** (*Port*) -- OSPFv2 Ext Prefix Range Tlv / Ospfv2 Ext Prefix Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Fapm Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Algorithm** (*int*) -- 灵活算法 ID, 类型为：number, 取值范围：128-255, 默认值：128
- **Flags** (*int*) -- 单字节值, 类型为：number, 取值范围：0-255, 默认值：0
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 OSPFv2 Sr Fapm Sub Tlv 对象, 类型：object

返回类型 (Ospfv2SrFapmSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | Create Ospf Ext Prefix Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
| Create Ospf Sr Fapm Sub Tlv | Tlv=${Tlv} |
```

static create_ospf_sr_link_msd_sub_tlv(*Session*, *ExtendedLinkTlv*, ***kwargs*)

创建 OSPFv2 Sr Link Msd Sub Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **ExtendedLinkTlv** (Ospfv2ExtendedLinkTlvConfig) -- OSPFv2 Extended Link Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Link Msd Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **Flags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：UNKNOWN, 支持选项有：

  UNKNOWN

  MAX_SEG_LEFT

  MAX_END_POP

  MAX_T_INSERT

  MAX_T_ENCAPS

  MAX_END_D

- **MaxSegmentLeft** (*int*) -- 指定接收报文的 SRH 中 SL（Segment Left）字段的最大值, 类型为：number, 取值范围：0-255, 默认值：8
- **MaxEndPop** (*int*) -- 指定 SRH 栈的顶端 SRH 中 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaxInsert** (*int*) -- 指定执行 T.Insert 行为时可包含 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaxEncap** (*int*) -- 指定执行 T.Encap 行为时可包含 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaxEndD** (*int*) -- 指定执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8

返回 OSPFv2 Sr Link Msd Sub Tlv 对象, 类型：object

返回类型 (Ospfv2SrLinkMsdSubTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Extended Link Lsa | Session=${Session} |␣
↪Age=20 |
| ${Tlv} | | Create Ospf Extended link Tlv | OpaqueExtendedLinkLsa=${Lsa}␣
↪|
| Create Ospf Sr Link Msd Sub Tlv | Session=${Session} | Tlv=${Tlv} |
```

static create_ospf_sr_node_msd_tlv(*Session*, *OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Node Msd Tlv 对象

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object
- **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Node Msd Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- TLV 中的标志位, 类型为：list, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  MAX_SEG_LEFT

  MAX_END_POP

MAX_T_INSERT

MAX_T_ENCAPS

MAX_END_D

- **MaxSegmentLeft** (*int*) -- 在应用与 SID 关联的 SRv6 Endpoint Function 指令之前，指定接收报文的 SRH 中 SL（Segment Left）字段的最大值, 类型为：number, 默认值：0, 取值范围：0-255
- **MaxEndPop** (*int*) -- 指定 SRH 栈的顶端 SRH 中 SID 的最大数量, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxInsert** (*int*) -- 指定执行 T.Insert 行为时可包含 SID 的最大数量, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxEncap** (*int*) -- 指定执行 T.Encap 行为时可包含 SID 的最大数量, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 类型为：number, 默认值：8, 取值范围：0-255

返回 OSPFv2 Sr Node Msd Tlv 对象, 类型：object

返回类型 (Ospfv2SrNodeMsdTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Sr Node Msd Tlv | Session=${Session} | OpaqueRouterInfoLsa=$
↪{Lsa} |
```

static create_ospf_sr_sid_label_range_tlv(*OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Sid Label Range Tlv 对象

参数 **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Sid Label Range Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SidLabelType** (*str*) -- 类型为：str, 默认值：Bit20, 取值范围：

  Bit20

  Bit32

- **SidLabelBase** (*int*) -- SID/Label Type 为 20-Bit Label 时，指定起始标签; SID/Label Type 为 32-Bit SID 时，指定起始 SID, 类型为：number, 默认值：0, 取值范围：1-4294967295
- **SidLabelRange** (*int*) -- 指定要创建的 SID/标签的数量, 类型为：number, 默认值：0, 取值范围：1-16777215

返回 OSPFv2 Sr Sid Label Range Tlv 对象, 类型：object

返回类型 (Ospfv2SidLabelRangeTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Sr Sid Label Range Tlv | OpaqueRouterInfoLsa=${Lsa} |
```

static create_ospf_sr_srms_preference_tlv(*OpaqueRouterInfoLsa*, ***kwargs*)

创建 OSPFv2 Sr Srms Preference Tlv 对象

参数 **OpaqueRouterInfoLsa** (Ospfv2OpaqueRouterInfoLsaConfig) -- OSPFv2 Opaque Router Info LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Sr Srms Preference Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Preference** (*int*) -- 指定路由器作为 SR Mapping Server 的优先级, 类型为：number, 默认值：1, 取值范围：0-255

返回 OSPFv2 Sr Srms Preference Tlv 对象, 类型：object

返回类型 (Ospfv2SrmsPreferenceTlvConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${Lsa} | Create Ospf Opaque Router Info Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospf Sr Srms Preference Tlv | OpaqueRouterInfoLsa=${Lsa} |
```

static create_ospf_summary_lsa(*Session*, ***kwargs*)

创建 OSPFv2 Summary LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Summary LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **RouteCount** (*int*) -- 路由个数, 类型为：number, 取值范围：1-1000000, 默认值：1
- **StartNetworkPrefix** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：192.0.1.0
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-32, 默认值：24
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：10

- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Summary LSA 对象, 类型：object

返回类型 (Ospfv2SummaryLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Summary Lsa | Session=${Session} | Age=20 |
```

static create_ospf_te_lsa(*Session*, ***kwargs*)

创建 OSPFv2 Te LSA 对象

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Te LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **TlvType** (*str*) -- Tlv 类型, 类型为：string, 默认值：LsaLink, 支持选项有：

  LsaRouter

  LsaLink

- **RouterId** (*str*) -- 路由器 ID, 类类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **LinkId** (*str*) -- Link ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **LinkType** (*str*) -- Link 类型, 类型为：string, 默认值：LsaLink, 支持选项有：

  LinkP2P

LinkMultiaccess

- **Instance** (*int*) -- 实例, 类型为：number, 取值范围：0-16777215, 默认值：1
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：0-16777215, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

  DNBIT
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv2 Te LSA 对象, 类型：object

返回类型 (Ospfv2TeLsaConfig)

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| Create Ospf Te Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3(*Port*, ***kwargs*)

创建 OSPFv3 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- OSPFv3 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 OSPFv3 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **InstanceId** (*int*) -- 实例 ID, 类型为：number, 取值范围：0-255, 默认值：0
- **AreaId** (*str*) -- 区域 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableExtendedLsa** (*bool*) -- 使能扩展 LSA, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **ExtendedLsaMode** (*str*) -- 扩展 LSA 模式, 类型为：string, 默认值：Full, 取值范围：

  NONE

  MixedModeOriginateOnly

  MixedModeOriginateSPF

  Full

- **AreaExtendedLsaMode** (*str*) -- 扩展区域 LSA 模式, 类型为：string, 默认值：InheritGlobal, 取值范围：

  InheritGlobal

  NONE

  MixedModeOriginateOnly

  MixedModeOriginateSPF

  Full

- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 取值范围：Broadcast 或 P2P, 默认值：Broadcast
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **InterfaceId** (*int*) -- 接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：10
- **Cost** (*int*) -- 接口开销, 类型为：number, 取值范围：1-65535, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'V6BIT', 'EBIT', 'RBIT'], 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

  Unused15

  Unused14

  Unused13

  Unused12

  Unused11

  Unused10

  Unused9

  Unused8

Unused7

Unused6

Unused5

Unused4

Unused3

Unused2

Unused1

Unused0

- **EnableOspfv3Mtu** (*bool*) -- 使能 OSPFv3 MTU, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GracefulRestartReason** (*str*) -- 平滑重启原因, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  SOFTWARE

  RELOADORUPGRADE

  SWITCH

- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **HelloInterval** (*int*) -- Hello 包间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **RouterDeadInterval** (*int*) -- 路由器失效间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：40
- **LsaRetransInterval** (*int*) -- LSA 重传间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：5
- **LsaRefreshTime** (*int*) -- LSA 刷新间隔 (秒), 类型为：number, 取值范围：1-1800, 默认值：1800

返回 OSPFv3 协议会话对象, 类型：object

返回类型 (Ospfv3Router)

实际案例

```
| Create Ospfv3 | Port=${Port} |
```

static create_ospfv3_as_external_lsa(*Session*, ***kwargs*)

创建 OSPFv3 As External LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 As External LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **PrefixCount** (*int*) -- 前缀个数, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-128, 默认值：64
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  NUBIT

  LABIT

  MCBIT

  PBIT

  DNBit

  NBit

  Unused1

  Unused0
- **IsExternalMetric** (*bool*) -- 是否外部度量值, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **EnableForwardingAddress** (*bool*) -- 使能转发地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ForwardingAddress** (*str*) -- 转发地址, 即：LSA 中携带的转发地址, 类型为：string, 取值范围：IPv6 地址, 默认值：'::'
- **AdminTag** (*int*) -- 管理标签, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ReferencedLsType** (*int*) -- 参考链路状态类型, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0x0
- **ReferencedLinkStateId** (*int*) -- 参考链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **LsaAutomaticConversion** (*bool*) -- LSA 自动转换, 即：当配置的会话为 NSSA 会话时,Renix 会自动将此处配置的外部 LSA 转换为 NSSA-LSA 进行发送; 当配置的会话为非 NSSA 会话时,Renix 会自动将此处配置的 NSSA-LSA 转换为外部 LSA 进行发送, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 As External LSA 对象, 类型：object

返回类型 (Ospfv3AsExternalLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 As External Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3_bier_mpls_encap_sub_tlv(*SubTlv*, ***kwargs*)

创建 OSPFv3 Bier Mpls Encap Sub Tlv 对象

参数 **SubTlv** (Ospfv3BierSubTlvConfig) -- Ospfv3 Bier Sub Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Bier Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TlvType** (*int*) -- Type 字段值, 类型为：number, 取值范围：0-255, 默认值：10
- **MaxSi** (*int*) -- 最大 Set ID, 类型为：number, 取值范围：1-255, 默认值：1
- **Label** (*int*) -- 标签范围中的起始标签值, 类型为：number, 取值范围：0-1048575, 默认值：100
- **BsLen** (*int*) -- 本地比特串的长度, 类型为：number, 取值范围：0-15, 默认值：4

返回 Ospfv3 Bier Mpls Encap Sub Tlv 对象, 类型：object

返回类型 (Ospfv3BierMplsEncapSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Inter Area Prefix Lsa | Session=${Session} |␣
↪Age=20 |
| ${SubTlv} | Create Ospfv3 Bier Sub Tlv | Lsa=${Lsa} |
| Create Ospfv3 Bier Mpls Encap Sub Tlv | SubTlv=${SubTlv} |
```

static create_ospfv3_bier_sub_tlv(*Lsa*, ***kwargs*)

创建 OSPFv3 Bier Sub Tlv 对象

参数 **Lsa** (Ospfv3InterAreaRouterLsaConfig) -- Ospfv3 Inter Area Router LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Bier Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TlvType** (*int*) -- Type 字段值, 类型为：number, 取值范围：0-255, 默认值：9

- **SubDomainId** (*int*) -- BIER 子域 ID, 类型为：number, 取值范围：1-255, 默认值：1
- **MtId** (*int*) -- 多拓扑 ID, 类型为：number, 取值范围：1-255, 默认值：1
- **BfrId** (*int*) -- BFR（Bit Forwarding Router，比特转发路由器）ID, 类型为：number, 取值范围：1-65535, 默认值：1
- **Bar** (*int*) -- BIER 算法, 类型为：number, 取值范围：0-255, 默认值：0
- **Ipa** (*int*) -- IGP 算法, 类型为：number, 取值范围：0-255, 默认值：0

返回 Ospfv3 Bier Sub Tlv 对象, 类型：object

返回类型 (Ospfv3BierSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Inter Area Prefix Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospfv3 Bier Sub Tlv | Lsa=${Lsa} |
```

static create_ospfv3_endx_sid_structure_sub_tlv(*SubTlv*, ***kwargs*)

创建 OSPFv3 Endx Sid Structure Sub Tlv 对象

参数 **SubTlv** (Ospfv3Srv6EndXSidSubTlvConfig) -- OSPFv3 Srv6 EndX Sid Sub Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Link Msd Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LbLength** (*int*) -- SRv6 SID Locator Block 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **LnLength** (*int*) -- SRv6 SID Locator Node 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **FunctionLength** (*int*) -- SRv6 SID Function 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **ArgumentLength** (*int*) -- SRv6 SID Argument 长度, 类型为：number, 取值范围：0-128, 默认值：32

返回 Ospfv3 Endx Sid Structure Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6SidStructureSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| ${SubTlv} | Create Ospfv3 Srv6 Msd Sid Sub Tlv | RouterLsaLink=$
↪{LsaLink} |
| Create Ospfv3 Endx Sid Structure Sub Tlv | SubTlv=${SubTlv} |
```

static create_ospfv3_inter_area_prefix_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Inter Area Prefix LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Inter Prefix LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ExtendedLsaTlvs** (*list*) -- 扩展 LSA TLVs, 类型为：list, 默认值：Ipv6InterAreaPrefix, 支持选项有：

  NONE

  Ipv6InterAreaPrefix
- **PrefixCount** (*int*) -- 前缀个数, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-128, 默认值：64
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  NUBIT

  LABIT

  MCBIT

  PBIT
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Ospfv3 Inter Area Prefix LSA 对象, 类型：object

返回类型 (Ospfv3InterAreaPrefixLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Inter Area Prefix Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3_inter_area_router_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Inter Area Router LSA 对象

参数 Session (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Router LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **AsbrId** (*str*) -- ASBR ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：NONTBIT | V6BIT | EBIT, 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

  Unused15

  Unused14

  Unused13

  Unused12

  Unused11

  Unused10

  Unused9

  Unused8

  Unused7

  Unused6

  Unused5

Unused4

Unused3

Unused2

Unused1

Unused0

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Ospfv3 Inter Area Router LSA 对象, 类型：object

返回类型 (Ospfv3InterAreaRouterLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Inter Area Router Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3_intra_area_prefix_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Intra Area Prefix LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Intra Prefix LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ReferencedLsType** (*str*) -- 参考 LS 类型, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0x0
- **ReferencedAdvertisingRouterId** (*str*) -- 参考通告路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **ReferencedLinkStateId** (*int*) -- 参考链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ExtendedLsaTlvs** (*list*) -- 扩展 LSA TLVs, 类型为：list, 默认值：Ipv6IntraAreaPrefix, 支持选项有：

  NONE

  Ipv6IntraAreaPrefix
- **PrefixCount** (*int*) -- 前缀个数, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1

- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-128, 默认值：64
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  NUBIT

  LABIT

  MCBIT

  PBIT

  DNBit

  NBit

  Unused1

  Unused0
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Ospfv3 Intra Area Prefix LSA 对象, 类型：object

返回类型 (Ospfv3IntraAreaPrefixLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Intra Area Prefix Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3_lan_endx_sid_structure_sub_tlv(*SubTlv*, ***kwargs*)

创建 OSPFv3 Lan Endx Sid Structure Sub Tlv 对象

参数 **SubTlv** (Ospfv3Srv6LanEndXSidSubTlvConfig) -- OSPFv3 Srv6 Lan EndX Sid Sub Tlv 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Link Msd Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LbLength** (*int*) -- SRv6 SID Locator Block 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **LnLength** (*int*) -- SRv6 SID Locator Node 长度, 类型为：number, 取值范围：0-128, 默认值：32
- **FunctionLength** (*int*) -- SRv6 SID Function 长度, 类型为：number, 取值范围：0-128, 默认值：32

- **ArgumentLength** (*int*) -- SRv6 SID Argument 长度, 类型为：number, 取值范围：0-128, 默认值：32

返回 Ospfv3 Lan Endx Sid Structure Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6LanEndXSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| ${SubTlv} | Create Ospfv3 Srv6 Msd Sid Sub Tlv | RouterLsaLink=$
↪{LsaLink} |
| Create Ospfv3 Lan Endx Sid Structure Sub Tlv | SubTlv=${SubTlv} |
```

static create_ospfv3_link_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Link LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Link LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ExtendedLsaTlvs** (*list*) -- 扩展 LSA TLVs, 类型为：list, 默认值：Ipv6IntraAreaPrefix, 支持选项有：

  NONE

  Ipv6IntraAreaPrefix

  Ipv6LinkLocalAddr

  Ipv4LinkLocalAddr
- **PrefixCount** (*int*) -- 前缀个数, 类型为：number, 取值范围：1-4294967295, 默认值：1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为：number, 取值范围：1-128, 默认值：64
- **Increment** (*int*) -- 步长, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为：list, 默认值：NONTBIT, 支持选项有：

  NONTBIT

  NUBIT

  LABIT

MCBIT

PBIT

DNBit

NBit

Unused1

Unused0

- **LinkLocalInterfaceAddress** (*str*) -- 本地链路接口地址, 类型为: string, 取值范围: IPv6 地址, 默认值: fe80::1
- **Ipv4LinkLocalInterfaceAddress** (*str*) -- 本地链路接口地址, 类型为: string, 取值范围: IPv4 地址, 默认值: 0.0.0.0
- **RouterPriority** (*int*) -- 路由优先级, 类型为: number, 取值范围: 1-255, 默认值: 1
- **Options** (*list*) -- 选项, 类型为: list, 默认值: NONTBIT | V6BIT | EBIT, 支持选项有:

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

  Unused15

  Unused14

  Unused13

  Unused12

  Unused11

  Unused10

  Unused9

  Unused8

  Unused7

  Unused6

  Unused5

  Unused4

  Unused3

  Unused2

  Unused1

  Unused0

- **Age** (*int*) -- 路由器优先级, 类型为: number, 取值范围: 0-3600, 默认值: 0

- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Link LSA 对象, 类型：object

返回类型 (Ospfv3LinkLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Link Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3_network_atch_router(*Lsa*, ***kwargs*)

创建 OSPFv3 Network LSA Atch Router 对象

参数 **Lsa** (Ospfv3NetworkLsaConfig) -- 测试仪表 OSPFv3 Network LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Network LSA Atch Router 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AttachedRouter** (*str*) -- 附加路由器的 IP 地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0

返回 OSPFv3 Network LSA Atch Router 对象, 类型：object

返回类型 (Ospfv3NetworkAtchRouterConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Network Lsa | Session=${Session} | Age=20 |
| Create Ospfv3 Network Lsa Atch Router | Lsa=${RouterLsa} | Metric=65535␣
↪|
```

static create_ospfv3_network_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Network LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Network LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

  Unused15

  Unused14

  Unused13

  Unused12

  Unused11

  Unused10

  Unused9

  Unused8

  Unused7

  Unused6

  Unused5

  Unused4

  Unused3

  Unused2

  Unused1

  Unused0

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Network LSA 对象, 类型：object

返回类型 (Ospfv3NetworkLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Network Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3_nssa_external_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Nssa External LSA 对象

参数 Session (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为: object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Nssa External LSA 的名称, 类型为: string
- **Enable** (*bool*) -- 使能, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即: 指定最初发布 LSA 的路由器 ID, 类型为: string, 取值范围: 0.0.0.0-255.255.255.255, 默认值: 1.1.1.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为: number, 取值范围: 0-4294967295, 默认值: 0
- **ExtendedLsaTlvs** (*list*) -- 扩展 LSA TLVs, 类型为: list, 默认值: Ipv6ExternalPrefix, 支持选项有:

  NONE

  Ipv6ExternalPrefix
- **ExtendedLsaSubTlvs** (*list*) -- 扩展 LSA Sub-TLV, 类型为: list, 默认值: NONTBIT, 支持选项有:

  NONE

  Ipv6ForwardingAddr

  Ipv4ForwardingAddr

  RouteTag
- **PrefixCount** (*int*) -- 前缀个数, 类型为: number, 取值范围: 1-4294967295, 默认值: 1
- **StartPrefixAddress** (*str*) -- 起始网络前缀, 类型为: string, 取值范围: IPv6 地址, 默认值: 2000::1
- **PrefixLength** (*int*) -- 路由器优先级, 类型为: number, 取值范围: 1-128, 默认值: 64
- **Increment** (*int*) -- 步长, 类型为: number, 取值范围: 0-4294967295, 默认值: 1
- **PrefixOptions** (*list*) -- 前缀选项, 类型为: list, 默认值: NONTBIT, 支持选项有:

  NONTBIT

  NUBIT

  LABIT

  MCBIT

  PBIT

  DNBit

  NBit

Unused1

Unused0

- **IsExternalMetric** (*bool*) -- 是否外部度量值, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **EnableForwardingAddress** (*bool*) -- 使能转发地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ForwardingAddress** (*str*) -- Ipv6 转发地址, 即：LSA 中携带的转发地址, 类型为：string, 取值范围：IPv6 地址, 默认值：'::'
- **Ipv4ForwardingAddress** (*str*) -- IPv4 转发地址, 取值范围：IPv4 地址, 默认值：0.0.0.0
- **AdminTag** (*int*) -- 管理标签, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **ReferencedLsType** (*int*) -- 参考链路状态类型, 类型为：hex number, 取值范围：0x0-0xFFFF, 默认值：0x0
- **ReferencedLinkStateId** (*int*) -- 参考链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LsaAutomaticConversion** (*bool*) -- LSA 自动转换, 即：当配置的会话为 NSSA 会话时,Renix 会自动将此处配置的外部 LSA 转换为 NSSA-LSA 进行发送; 当配置的会话为非 NSSA 会话时,Renix 会自动将此处配置的 NSSA-LSA 转换为外部 LSA 进行发送, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Nssa External LSA 对象, 类型：object

返回类型 (Ospfv3NssaExternalLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Nssa External Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3_opaque_router_info_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Opaque Router Info LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Opaque Router Info LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Scope** (*str*) -- 起始网络前缀, 类型为：string, 默认值：AreaLocal, 取值范围：

LinkLocal

AreaLocal

AreaSystemWide

- **AdvertisingRouterId** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.0.1
- **LinkStateId** (*int*) -- 路由优先级, 类型为：number, 取值范围：1-255, 默认值：0
- **TlvsFlag** (*list*) -- 前缀选项, 类型为：list, 默认值：['NONTBIT', 'InfoCapabilities'], 支持选项有：

  NONEBIT

  InfoCapabilities

  FuncCapabilities

- **InformationalCapabilities** (*list*) -- 前缀选项, 类型为：list, 默认值：NONEBIT, 支持选项有：

  NONEBIT

  RcBit

  RhBit

  SrsBit

  TesBit

  PolBit

  Etbit

  MiBit

  SrhBit

  Unused8

  Unused9

  Unused10

  Unused11

  Unused12

  Unused13

  Unused14

  Unused15

  Unused16

  Unused17

  Unused18

  Unused19

  Unused20

  Unused22

  Unused21

  Unused23

  Unused24

Unused25

Unused26

Unused27

Unused28

Unused29

Unused30

- **FunctionalCapabilities** (*list*) -- 前缀选项，类型为：list，默认值：NONEBIT，支持选项有：

  NONEBIT

  Unused0

  Unused1

  Unused2

  Unused3

  Unused4

  Unused5

  Unused6

  Unused7

  Unused8

  Unused9

  Unused10

  Unused11

  Unused12

  Unused13

  Unused14

  Unused15

  Unused16

  Unused17

  Unused18

  Unused19

  Unused20

  Unused22

  Unused21

  Unused23

  Unused24

  Unused25

  Unused26

  Unused27

  Unused28

  Unused29

Unused30

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Opaque Router Info LSA 对象, 类型：object

返回类型 (Ospfv3OpaqueRouterInfoLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Opaque Router Info LSA | Session=${Session} | Age=20 |
```

static create_ospfv3_router_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Router LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Router LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AdvertisingRouterId** (*str*) -- 通告路由器 ID 即：指定最初发布 LSA 的路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.0.0.1
- **LinkStateId** (*int*) -- 链路状态 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **RouterType** (*str*) -- 路由器类型, 类型为：string, 默认值：NONTBIT, 支持选项有：

  NONEBIT

  RouterTypeABR

  RouterTypeASBR

  RouterTypeVirtype
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

Unused15

Unused14

Unused13

Unused12

Unused11

Unused10

Unused9

Unused8

Unused7

Unused6

Unused5

Unused4

Unused3

Unused2

Unused1

Unused0

- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Router LSA 对象, 类型：object

返回类型 (Ospfv3RouterLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
```

static create_ospfv3_router_lsa_link(*RouterLsa*, ***kwargs*)

创建 OSPFv3 Router LSA Link 对象

参数 **RouterLsa** (Ospfv3RouterLsaConfig) -- 测试仪表 OSPFv3 Router LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Router LSA Link 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LinkType** (*str*) -- 链路类型, 类型为：string, 默认值：P2P, 支持选项有：

  P2P

  TRANSITNETWORK

  VIRTUALLINK

- **InterfaceId** (*int*) -- 接口 ID, 即该 ID 用于唯一标识 simulated router 的接口, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **NeighborInterfaceId** (*int*) -- 邻居接口 ID, 即该 ID 用于唯一标识邻居路由器的接口, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **NeighborRouterId** (*str*) -- 邻居路由器 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：1.0.0.1
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-65535, 默认值：1

返回 OSPFvv3 Router LSA Link 对象, 类型：object

返回类型 (Ospfv3RouterLsaLinksConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} | Metric=65535 |
```

static create_ospfv3_sr_algorithm_tlv(*Lsa*, ***kwargs*)

创建 OSPFv3 Sr Algorithm Tlv 对象

参数 **Lsa** (Ospfv3OpaqueRouterInfoLsaConfig) -- Ospfv3 Opaque Router Info LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Sr Algorithm Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Algorithms** (*int*) -- 算法, 类型为：number, 默认值：0

返回 Ospfv3 Sr Algorithm Tlv 对象, 类型：object

返回类型 (Ospfv3SrAlgorithmTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Opaque Router Info LSA | Session=${Session} |␣
↪Age=20 |
| Create Ospfv3 Sr Algorithm Tlv | Lsa=${Lsa} |
```

static create_ospfv3_sr_fad_tlv(*Session*, *Lsa*, ***kwargs*)

创建 OSPFv3 Sr Fad Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **Lsa** (Ospfv3OpaqueRouterInfoLsaConfig) -- Ospfv3 Opaque Router Info LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Sr Fad Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **FlexAlgorithm** (*int*) -- 灵活算法 ID, 类型为：number, 取值范围：128-255, 默认值：128
- **MetricType** (*str*) -- 度量类型, 类型为：string, 默认值：IGPMetric, 取值范围：

  IGPMetric

  MinUnidirectionalLinkDelay

  TEDefaultMetric
- **CalculationType** (*int*) -- 特定 IGP 算法的计算类型, 类型为：number, 取值范围：0-127, 默认值：0
- **Priority** (*int*) -- 该 TLV 的优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **FlexAlgorithmSubTlvs** (*list*) -- 灵活算法路径计算要遵循的约束条件, 类型为：list, 默认值：NONEBIT, 支持选项有：

  NONEBIT

  ExcludeAdminGroups

  IncludeAnyAdminGroups

  IncludeAllAdminGroups

  DefinitionFlags

  ExcludeSRLG
- **ExcludeAdminGroups** (*int*) -- 算法, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **IncludeAnyAdminGroups** (*int*) -- 算法, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **IncludeAllAdminGroups** (*int*) -- 算法, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **DefinitionFlags** (*int*) -- 算法, 类型为：number, 取值范围：0-FF, 默认值：0x80
- **ExcludeSRLG** (*int*) -- 算法, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Ospfv3 Sr Fad Tlv 对象, 类型：object

返回类型 (Ospfv3SrFadTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Opaque Router Info LSA | Session=${Session} |␣
↪Age=20 |
| Create Ospfv3 Sr Fad Tlv | Lsa=${Lsa} |
```

static create_ospfv3_sr_fapm_sub_tlv(*Lsa*, ***kwargs*)

创建 OSPFv3 Sr Fapm Sub Tlv 对象

参数 Lsa (Ospfv3InterAreaRouterLsaConfig) -- Ospfv3 Inter Area Router LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Sr Fapm Sub Tlv 的名称, 类型为：string

- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Algorithm** (*int*) -- 灵活算法 ID, 类型为：number, 取值范围：128-255, 默认值：128
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：128-255, 默认值：0

返回 Ospfv3 Sr Fapm Sub Tlv 对象, 类型：object

返回类型 (Ospfv3SrFapmSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Inter Area Prefix Lsa | Session=${Session} |␣
↪Age=20 |
| Create Ospfv3 Sr Fapm Sub Tlv | Lsa=${Lsa} |
```

static create_ospfv3_srv6_capabilities_tlv(*Session*, *Lsa*, ***kwargs*)

创建 OSPFv3 Srv6 Capabilities Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **Lsa** (Ospfv3OpaqueRouterInfoLsaConfig) -- Ospfv3 Opaque Router Info LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Capabilities Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Flags** (*list*) -- 类型为：list, 默认值：NONEBIT, 支持选项有：

  NONEBIT

  Unused0

  OFlag

  Unused2

  Unused3

  Unused4

  Unused5

  Unused6

  Unused7

  Unused8

  Unused9

  Unused10

  Unused11

  Unused12

  Unused13

  Unused14

Unused15

返回 Ospfv3 Srv6 Capabilities Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6CapabilitiesTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Opaque Router Info LSA | Session=${Session} |␣
↪Age=20 |
| Create Ospfv3 Srv6 Capabilities Tlv | Lsa=${Lsa} |
```

static create_ospfv3_srv6_endx_sid_sub_tlv(*Session*, *RouterLsaLink*, ***kwargs*)

创建 OSPFv3 Srv6 EndX Sid Sub Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **RouterLsaLink** (Ospfv3RouterLsaLinksConfig) -- OSPFv3 Router Lsa LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 EndX Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EndpointBehaviorId** (*int*) -- SRv6 SID 的端点行为 ID, 类型为：number, 默认值：0
- **Flags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NONEBIT, 取值范围：

  NONEBIT

  Unused0

  Unused1

  Unused2

  Unused3

  Unused4

  PersistentFlag

  SetFlag

  BackupFlag

- **Algorithm** (*int*) -- SID 关联的算法, 类型为：number, 取值范围：0-255, 默认值：0
- **Weight** (*int*) -- END.X SID / LAN END.X SID 的权重，用于负载分担, 类型为：number, 取值范围：0-255, 默认值：1
- **Sid** (*str*) -- 通告的 SRv6 SID, 邻居路由器 ID, 类型为：string, 取值范围：有效的 ipv6 地址, 默认值：'aaaa:1:1:1::'

返回 OSPFv3 Srv6 EndX Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6EndXSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| Create Ospfv3 Srv6 Endx Sid Sub Tlv | RouterLsaLink=${LsaLink} |
```

static create_ospfv3_srv6_lan_endx_sid_sub_tlv(*Session*, *RouterLsaLink*, ***kwargs*)

创建 OSPFv3 Srv6 Lan EndX Sid Sub Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **RouterLsaLink** (Ospfv3RouterLsaLinksConfig) -- OSPFv3 Router Lsa LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Lan EndX Sid Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EndpointBehaviorId** (*int*) -- SRv6 SID 的端点行为 ID, 类型为：number, 默认值：0
- **Flags** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NONEBIT, 取值范围：

  NONEBIT

  Unused0

  Unused1

  Unused2

  Unused3

  Unused4

  PersistentFlag

  SetFlag

  BackupFlag
- **Algorithm** (*int*) -- SID 关联的算法, 类型为：number, 取值范围：0-255, 默认值：0
- **Weight** (*int*) -- END.X SID / LAN END.X SID 的权重，用于负载分担, 类型为：number, 取值范围：0-255, 默认值：1
- **Sid** (*str*) -- 通告的 SRv6 SID, 邻居路由器 ID, 类型为：string, 取值范围：有效的 ipv6 地址, 默认值：'aaaa:1:1:1::'

返回 OSPFv3 Srv6 Lan EndX Sid Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6LanEndXSidSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| Create Ospfv3 Srv6 Lan Endx Sid Sub Tlv | RouterLsaLink=${LsaLink} |
```

static create_ospfv3_srv6_link_msd_sub_tlv(*Session*, *RouterLsaLink*, ***kwargs*)

创建 OSPFv3 Srv6 Link Msd Sub Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **RouterLsaLink** (Ospfv3RouterLsaLinksConfig) -- OSPFv3 Router Lsa LSA 列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Link Msd Sub Tlv 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Msds** (*list*) -- 包含在 TLV 中的标志位, 类型为：list, 默认值：NONEBIT, 取值范围：

  NONTBIT

  MaxiSegmentLeft

  MaxiEndPop

  MaxiTInsert

  MaxiTEncaps

  MaxiEndD
- **MaximumEndDSrh** (*int*) -- 接收报文的 SRH 中 SL（Segment Left）字段的最大值, 类型为：number, 取值范围：0-255, 默认值：8
- **MaximumEndPop** (*int*) -- SRH 栈的顶端 SRH 中 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaximumSegmentsLeft** (*int*) -- 执行 T.Insert 行为时可包含 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaximumTEncapSrh** (*int*) -- 执行 T.Encap 行为时可包含 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8
- **MaximumTInsertSrh** (*int*) -- 执行 End.DX6 和 End.DT6 功能时，SRH 中 SID 的最大数量, 类型为：number, 取值范围：0-255, 默认值：8

返回 OSPFv3 Srv6 Link Msd Sub Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6MsdSubTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${RouterLsa} | Create Ospfv3 Router Lsa | Session=${Session} | Age=20 |
| ${LsaLink} | Create Ospfv3 Router Lsa Link | RouterLsa=${RouterLsa} |␣
↪Metric=65535 |
| Create Ospfv3 Srv6 Msd Sid Sub Tlv | RouterLsaLink=${LsaLink} |
```

static create_ospfv3_srv6_location_lsa(*Session*, ***kwargs*)

创建 OSPFv3 Srv6 Location LSA 对象

参数 **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Location LSA 的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Scope** (*str*) -- 起始网络前缀, 类型为：string, 默认值：AreaLocal, 取值范围：

  LinkLocal

  AreaLocal

  AreaSystemWide
- **AdvertisingRouterId** (*str*) -- 起始网络前缀, 类型为：string, 取值范围：IPv4 地址, 默认值：192.0.0.1
- **LinkStateId** (*int*) -- 路由优先级, 类型为：number, 取值范围：1-255, 默认值：0
- **Age** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-3600, 默认值：0
- **SequenceNumber** (*int*) -- 序列号, 类型为：hex number, 取值范围：0x1-0xFFFFFFFF, 默认值：0x80000001
- **Checksum** (*bool*) -- 校验和, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 OSPFv3 Srv6 Location LSA 对象, 类型：object

返回类型 (Ospfv3Srv6LocatorLsaConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| Create Ospfv3 Srv6 Location LSA | Session=${Session} | Age=20 |
```

static create_ospfv3_srv6_location_tlv(*Session*, *Lsa*, ***kwargs*)

创建 OSPFv3 Srv6 Location Tlv 对象

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：object
- **Lsa** (Ospfv3Srv6LocatorLsaConfig) -- Ospfv3 Srv6 Location LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv3 Srv6 Location Tlv 的名称, 类型为：string

- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterType** (*str*) -- 路由器类型, 类型为：string, 默认值：IntraArea, 支持选项有：

  IntraArea

  InterArea

  ASExternal

  NSSAExternal
- **Algorithm** (*int*) -- Locator 关联算法, 类型为：number, 取值范围：0-255, 默认值：0
- **LocatorLength** (*int*) -- Locator 前缀长度, 类型为：number, 取值范围：0-128, 默认值：64
- **Flags** (*list*) -- 类型为：list, 默认值：NONEBIT, 支持选项有：

  NONEBIT

  Unused0

  Unused1

  Unused2

  Unused3

  Unused4

  Unused5

  ABit

  NFlag
- **Metric** (*int*) -- 度量值, 类型为：number, 取值范围：1-16777215, 默认值：1
- **Locator** (*str*) -- 通告的 Locator, 类型为：string, 取值范围：有效的 ipv6 地址, 默认值：'aaaa:1:1:1::'

返回 Ospfv3 Srv6 Location Tlv 对象, 类型：object

返回类型 (Ospfv3Srv6LocatorTlvConfig)

实际案例

```
| ${Session} | Create Ospfv3 | Port=${Port} |
| ${Lsa} | Create Ospfv3 Opaque Router Info LSA | Session=${Session} |␣
↪Age=20 |
| Create Ospfv3 Srv6 Location Tlv | Lsa=${Lsa} |
```

static create_pcep(*Port*, ***kwargs*)

创建 PCEP 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- PCEP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 PCEP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **Role** (*str*) -- PCEP 角色, 类型为：string, 默认值：PCE, 取值范围:

  PCE

  PCC

- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPv4, 取值范围:

  IPv4

  IPv6

- **UseGatewayAsDutIp** (*bool*) -- 使用网关地址作为 DUT 地址, 选中则使用接口上配置的网关 IP 地址作为 DUT 地址；未选中则自定义 DUT IP 地址, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **SessionIpAddress** (*str*) -- 会话 IP 地址, 用于 PCEP 连接的 IP 类型, 类型为：string, 默认值：Interface_IP, 取值范围:

  Interface_IP

  Router_ID

- **PeerIpv4Address** (*str*) -- DUT IPv4 地址, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv4 时可见, 指定 DUT 的 IPv4 地址, 类型为：string, 默认值：192.85.1.1, 取值范围：有效的 Ipv4 地址

- **PeerIpv4AddressStep** (*str*) -- DUT IPv4 地址跳变, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv4 时可见, 指定 DUT IPv4 地址的增量步长, 类型为：string, 默认值：0.0.0.1, 取值范围：有效的 Ipv4 地址

- **PeerIpv6Address** (*str*) -- DUT IPv6 地址, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv6 时可见, 指定 DUT 的 IPv6 地址, 类型为：string, 默认值：2000::1, 取值范围：有效的 Ipv6 地址

- **PeerIpv6AddressStep** (*str*) -- DUT IPv6 地址跳变, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv6 时可见, 指定 DUT 的 IPv6 地址的增量步长, 类型为：string, 默认值：::1, 取值范围：有效的 Ipv6 地址

- **SessionInitiator** (*bool*) -- 会话发起者, 选中则主动发起会话建立请求；未选中则监听对端的发起会话建立请求。双方均主动发起会话建立请求时，IP 地址大的一方优先级更高, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **Negotiation** (*bool*) -- 使能 Negotiation, 选中则对 Keepalive Timer 和 Dead Timer 的值进行协商, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **KeepAlive** (*str*) -- Keep Alive 间隔 (sec), KEEPALIVE 消息的发送间隔，以秒为单位, 类型为：string, 默认值：30, 0-65535

- **MinKeepAlive** (*int*) -- KEEPALIVE 消息发送间隔的最小值。以秒为单位, 类型为：number, 取值范围：0-255, 默认值：0

- **MaxKeepAlive** (*int*) -- KEEPALIVE 消息发送间隔的最大值，以秒为单位, 类型为：number, 取值范围：0-255, 默认值：255

- **Dead** (*str*) -- Dead 间隔 (sec), 从未收到对端消息到 PCEP 会话断开连接之间的时间间隔。类型为：string, 取值范围：0-65535, 默认值：120

- **MinDeadAlive** (*int*) -- 最小可接受 Dead 间隔 (sec), 类型为：number, 取值范围：0-255, 默认值：0

- **MaxDeadAlive** (*int*) -- 最大可接受 Dead 间隔 (sec), 类型为：number, 取值范围：0-255, 默认值：255

- **EnableStatefulCapability** (*bool*) -- 选中则 OPEN 消息中包含 Stateful PCE Capability TLV, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **StatefulCapability** (*list*) -- 使能 PCE Stateful Capability 选中时可见, 单击单元格并从下拉菜单中选择一个或多个能力, 类型为：list, 默认值：['LSP_UPDATE','LSP_INSTANTIATION'], 取值范围:

  LSP_UPDATE

  INCLUDE_DB_VERSION

  LSP_INSTANTIATION

  TRIGGERED_RESYNC

  DELTA_LSP_SYN

  TRIGGERED_INITIAL_SYNC

- **EnableSegmentRoutingCapability** (*list*) -- 选择段路由扩展，OPEN 消息中将包括该 Capability TLV, 类型为：list, 默认值：['SR'], 取值范围：

  SR

  SRv6

- **PathSetupTypeList** (*list*) -- 添加路径建立类型, 类型为：list, 默认值：[0,1]

- **SrCapabilityFlags** (*list*) -- 选择一个或多个 SR 能力标志, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SR 时可见, 类型为：list, 默认值：['NONTBIT','NFlag','XFlag'],

  NONTBIT

  NFlag

  XFlag

- **Srv6CapabilityFlags** (*list*) -- 选择一个或多个 SRv6 能力标志, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SRv6 时可见, 类型为：list, 默认值：['NONTBIT','NFlag','XFlag'],

  NONTBIT

  NFlag

  XFlag

- **MSDs** (*list*) -- 选择一个或多个 MSD 类型, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SRv6 时可见, 类型为：list, 默认值：['NONTBIT']

  NONTBIT

  MaxiSegmentLeft

  MaxiEndPop

  MaxiHEncaps

  MaxiEndD

- **MaximumSidDepth** (*int*) -- 指定 SID 的最大数量, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SR 时可见, 类型为：number, 默认值：0, 取值范围：0-255

- **MaxSegmentsLeft** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum Segments Left 时可见, 类型为：number, 默认值：8, 取值范围：0-255

- **MaxEndPop** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum End Pop 时可见, 类型为：number, 默认值：8, 取值范围：0-255

- **MaxHencaps** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum H.Encaps 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum End D 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **EnableDbVersionTlv** (*bool*) -- 选中则配置 DB version TLV, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LspStateDbVersion** (*int*) -- 指定 LSP 状态数据库的初始版本号, 选中使能 DB Version TLV 时可见, 类型为：number, 默认值：1, 取值范围：1-18446744073709551614

返回 PCEP 协议会话对象, 类型：object

返回类型 (Pcep)

实际案例

```
| Create Pcep | Port=${Port} |
```

static create_pcep_bw_object(*PcepLsps*, ***kwargs*)

创建 PCEP Bw Object 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Bw Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Bw Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Bandwidth** (*str*) -- 类型为：string, 取值范围：0-4294967295, 默认值：0

返回 Pcep Bw Object 对象列表, 类型：object / list

返回类型 (PcepBwObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Bw Object | PcepLsp=${Egress} |
```

static create_pcep_endpoint_object(*PcepLsps*, ***kwargs*)

创建 PCEP Endpoint Object 对象

参数 **PcepLsps** (PccLspConfig) -- ：PCEP LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Endpoint Object 对象名称, 类型为：string

- **Enable** (*bool*) -- PCEP Endpoint Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Endpoint Object 对象列表, 类型: object / list

返回类型 (PcepEndPointObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Endpoint Object | PcepLsp=${Egress} |
```

static create_pcep_lsp_auto_tx_parameters(*PcepAutoParameters*, ***kwargs*)

创建 PCEP PCE Lsp Auto Tx Parameters 对象

参数 **PcepAutoParameters** (BgpRouter) -- : PCEP LSP Auto Parameters 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Tx Parameters 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Auto Tx Parameters 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **ObjectCategory** (*str*) -- 选择对象类别, 类型为：string, 默认值：LSP, 取值范围：

  BANDWIDTH

  RP

  NO_PATH

  ENDPOINT

  METRIC

  ERO

  RRO

  LSPA

  SRP

  LSP

  XRO
- **SelectObjectHandle** (*str*) -- 选择对象, 类型为：string, 默认值：""

返回 Pcep Lsp Auto Tx Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoTxParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pce Auto Update Parameters | PcepLsp=$
↪{Egress} |
| ${LspAutoTx} | Create Pcep Lsp Auto Tx Parameters | PcepAutoParameters=$
↪{Parameter} |
```

static create_pcep_lspa_object(*PcepLsps*, ***kwargs*)

创建 PCEP Lspa Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Lspa Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Lspa Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **SetupPriority** (*int*) -- 指定 TE LSA 抢占资源的优先级。类型为: number, 取值范围: 0-7, 默认值: 0
- **HoldingPriority** (*int*) -- 指定 TE LSA 持有资源的优先级。类型为: number, 取值范围: 0-7, 默认值: 0
- **LFlag** (*bool*) -- LSPA L Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **Affinities** (*bool*) -- 选中则对 32 位掩码和链接属性进行比较，设置包含链接条件和排除链接条件, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **ExcludeAny** (*int*) -- Affinities 选中时启用, 排除与 32 位掩码中任何属性匹配的链接, 类型为: number, 取值范围: 0-4294967295, 默认值: 0
- **IncludeAny** (*int*) -- Affinities 选中时启用, 包含与 32 位掩码中任何属性匹配的链接, 类型为: number, 取值范围: 0-4294967295, 默认值: 0
- **IncludeAll** (*int*) -- Affinities 选中时启用, 包含与 32 位掩码中全部属性匹配的链接, 类型为: number, 取值范围: 0-4294967295, 默认值: 0

返回 Pcep Lspa Object 对象列表, 类型: object / list

返回类型 (PcepLspaObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Lspa Object | PcepLsp=${Egress} |
```

static create_pcep_metric_list(*PcepLsps*, ***kwargs*)

创建 PCEP Metric List 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Metric List 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Metric List 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False

返回 Pcep Metric List 对象列表, 类型: object / list

返回类型 (PcepMetricListConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Metric List | PcepLsp=${Egress} |
```

static create_pcep_metric_object(*PcepMetricLists*, ***kwargs*)

创建 PCEP Metric Object 对象

参数 **PcepMetricLists** (BgpRouter) -- : PCEP Metric List 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Metric Object 对象, 类型为: string
- **Enable** (*bool*) -- PCEP Metric Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **BoundFlag** (*bool*) -- PCReq 消息中 METRIC Object 的 B(Bound) 位是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **ComputedFlag** (*bool*) -- PCReq 消息中 METRIC Object 的 C(Computed Metric) 位是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **MetricType** (*str*) -- 指定度量值类型, 类型为: string, 默认值: MAX_SID_DEPTH, 取值范围:

  IGP_METRIC

  TE_METRIC

  HOP_COUNTS

  MAX_SID_DEPTH

- **MetricValue** (*int*) -- 指定最大度量值, 类型为：number, 取值范围：0-4294967295, 默认值：10

返回 Pcep Metric Object 对象列表, 类型: object / list

返回类型 (PcepMetricObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Metric List | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Metric Object | PcepMetricLists=${Object} |
```

static create_pcep_no_path_reason(*PcepLsps*, ***kwargs*)

创建 PCEP No Path Reason 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCE LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP No Path Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP No Path Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **NoPathType** (*str*) -- No-Path 类型, 类型为：string, 默认值：NOT_SATISFYING_CONSTRAINTS, 取值范围:

  NOT_SATISFYING_CONSTRAINTS

  PCE_CHAIN_BROKEN
- **CFlag** (*bool*) -- C Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NoPathReason** (*str*) -- No-Path 原因, 类型为: string, 默认值: NONTBIT, 取值范围:

  NONTBIT

  PCE_UNAVAILABLE

  UNKNOWN_DESTINATION

  UNKNOWN_SOURCE

返回 Pcep No Path Object 对象列表, 类型: object / list

返回类型 (PcepNoPathObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep No Path Object | PcepLsp=${Egress} |
```

static create_pcep_pcc_auto_delegation_parameters(*PcepLsps*, ***kwargs*)

创建 PCEP Auto Delegation Parameters 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Delegation Parameters 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Auto Delegation Parameters 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True

返回 Pcep Pcc Lsp Auto Delegation Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoDelegationParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pcc Auto Delegation Parameters | PcepLsp=$
↪{Egress} |
```

static create_pcep_pcc_auto_request_parameters(*PcepLsps*, ***kwargs*)

创建 PCEP PCC Lsp Auto Request Parameters 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Request Parameters 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Auto Request Parameters 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True

返回 Pcep Pce Lsp Auto Request Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoRequestParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pcc Auto Request Parameters | PcepLsp=$
↪{Egress} |
```

static create_pcep_pcc_auto_sync_parameters(*PcepLsps*, ***kwargs*)

创建 PCEP PCC Lsp Auto Sync Parameters 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Sync Parameters 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Auto Sync Parameters 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True

返回 Pcep Pce Lsp Auto Sync Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoSyncParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pcc Auto Sync Parameters | PcepLsp=${Egress}␣
↪|
```

static create_pcep_pcc_lsp(*Sessions*, ***kwargs*)

创建 PCEP PCC LSP 对象

参数 Sessions (Pcep) -- ：PCEP 协议会话对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP PCC LSP 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PCEP PCC LSP, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LspCount** (*int*) -- Lsp 数量, 类型为：number, 默认值：1, 取值范围：1-65535
- **AutoGenSymbolicName** (*bool*) -- 系统自动生成 Symbolic Name, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SymbolicName** (*str*) -- 设置 Symbolic Name, 类型为：string, 默认值：PLSP_@s
- **PathSetupType** (*str*) -- 建立 LSP 的方法, 类型为：string, 默认值：SEGMENT_ROUTING, 取值范围：

  SEGMENT_ROUTING

  SRv6
- **SourceIpv4Address** (*str*) -- 起始源 IP 地址, 类型为：string, 默认值：192.85.1.1, 取值范围：IPv4 地址
- **SourceIpv4AddressStep** (*str*) -- 源 IP 地址的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围：IPv4 地址
- **SourceIpv4AddressSessionOffset** -- 源 IPv4 地址接口跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围：IPv4 地址
- **DestinationIpv4Address** (*str*) -- 起始目的 IP 地址, 类型为：string, 默认值：193.85.1.1, 取值范围：IPv4 地址
- **DestinationIpv4AddressStep** (*str*) -- 目的 IP 地址的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围：IPv4 地址
- **DestinationIpv4AddressSessionOffset** (*str*) -- 目的 IPv4 地址跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围：IPv4 地址
- **SourceIpv6Address** (*str*) -- 源 IPv6 地址的跳变步长, 类型为：string, 默认值：2000::1, 取值范围：IPv6 地址
- **SourceIpv6AddressStep** (*str*) -- 源 IPv6 地址的跳变步长, 类型为：string, 默认值：::1, 取值范围：IPv6 地址
- **SourceIpv6AddressSessionOffset** (*str*) -- 源 IPv6 地址接口跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围：IPv6 地址
- **DestinationIpv6Address** (*str*) -- 起始目的 IPv6 地址, 类型为：string, 默认值：2001::1, 取值范围：IPv6 地址

- **DestinationIpv6AddressStep** (*str*) -- 目的 IPv6 地址的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **DestinationIpv6AddressSessionOffset** (*str*) -- 目的 IPv6 地址跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围: IPv6 地址
- **LspInitiateMethod** (*str*) -- LSP 初始方式, 类型为：string, 默认值：REPORT, 取值范围:

  REPORT

  PCE_INITIATE

  SYNCHRONIZATION

  REQUEST
- **ImmediateDelegation** (*bool*) -- 直接托管, 会话建立后自动将 LSP 托管给 PCE, LSP 初始方式为 Report Method、Synchronization Method 或 Request Method 时可见, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **DelegationInSynchronization** (*bool*) -- 同步中托管, LSP 初始方式为 Synchronization Method 时可见, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Pcc Lsp 对象列表, 类型: object / list

返回类型 (PccLspConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
```

static create_pcep_pcc_lsp_info(*PcepLsps*, ***kwargs*)

创建 PCEP PCC LSP INFO 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP PCC LSP INFO 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP PCC LSP INFO 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Administrator** (*bool*) -- 使能 Administrative, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **State** (*str*) -- 设置 Initial LSP State, 类型为：string, 默认值：GOING_UP, 取值范围：

  DOWN

  UP

  ACTIVE

  GOING_DOWN

  GOING_UP

  RESERVED_5

RESERVED_6

RESERVED_7

- **AutoGeneratedPlspId** (*bool*) -- 自动生成 PLSP-ID, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PlspId** (*int*) -- 指定起始 PLSP-ID, 类型为：number, 默认值：1, 取值范围：1-1048575
- **Step** (*int*) -- 指定 PLSP-ID 的跳变步长, 类型为：number, 默认值：1, 取值范围：1-1048575
- **LspId** (*int*) -- 指定起始 LSP ID, 类型为：number, 默认值：1, 取值范围：1-65535
- **LspIdStep** (*int*) -- 指定同一个会话中 LSP-ID 的跳变步长, 类型为：number, 默认值：1, 取值范围：1-65535
- **LspIdSessionOffset** (*int*) -- 指定 PCEP 会话块中 LSP-ID 在会话之间的跳变步长, 类型为：number, 默认值：1, 取值范围：1-65535
- **TunnelId** (*int*) -- 指定起始隧道 ID, 类型为：number, 默认值：1, 取值范围：1-65535
- **TunnelStep** (*int*) -- 指定同一个会话中隧道 ID 的跳变步长, 类型为：number, 默认值：1, 取值范围：1-65535
- **TunnelSessionOffset** (*int*) -- 指定 PCEP 会话块中隧道 ID 在会话之间的跳变步长, 类型为：number, 默认值：1, 取值范围：1-65535
- **ExtendedTunnelIPv4Id** (*str*) -- 指定扩展隧道 ID, 类型为：string, 默认值：10.0.0.1, 取值范围: IPv4 地址
- **ExtendedTunnelIPv4IdStep** (*str*) -- 指定同一个会话中扩展隧道 ID 的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **ExtendedTunnelIPv4IdSessionOffset** (*str*) -- 指定 PCEP 会话块中扩展隧道 ID 在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围: IPv4 地址
- **ExtendedTunnelIPv6Id** (*str*) -- 指定扩展隧道 ID, 类型为：string, 默认值：2000:1::1, 取值范围: IPv6 地址
- **ExtendedTunnelIPv6IdStep** (*str*) -- 指定同一个会话中扩展隧道 ID 的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **ExtendedTunnelIPv6IdSessionOffset** -- 指定 PCEP 会话块中扩展隧道 ID 在会话之间的跳变步长, 类型为: string, 默认值: ::1:0, 取值范围: IPv6 地址

返回 Pcep Pcc Lsp Info 对象列表, 类型: object / list

返回类型 (PccLspInfoObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${LspInfo} | Create Pcep Pcc Lsp Info | PcepLsp=${Egress} |
```

static create_pcep_pce_auto_initiate_parameters(*PcepLsps*, ***kwargs*)

创建 PCEP PCE Lsp Auto Initiate Parameters 对象

参数 **PcepLsps** (PceLspConfig) -- : PCEP PCE LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Initiate Parameters 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Auto Initiate Parameters 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True

返回 Pcep Pce Lsp Auto Initiate Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoInitiateParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pce Auto Initiate Parameters | PcepLsp=$
↪{Egress} |
```

static create_pcep_pce_auto_reply_parameters(*PcepLsps*, ***kwargs*)

创建 PCEP PCE Lsp Auto Reply Parameters 对象

参数 **PcepLsps** (PceLspConfig) -- : PCEP PCE LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Reply Parameters 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Auto Reply Parameters 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True

返回 Pcep Pce Lsp Auto Reply Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoReplyParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pce Auto Reply Parameters | PcepLsp=${Egress}
↪ |
```

static create_pcep_pce_auto_update_parameters(*PcepLsps*, ***kwargs*)

创建 PCEP PCE Lsp Auto Update Parameters 对象

参数 **PcepLsps** (PceLspConfig) -- : PCEP PCE LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Auto Update Parameters 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Auto Update Parameters 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True

返回 Pcep Pce Lsp Auto Update Parameters 对象列表, 类型: object / list

返回类型 (PcepAutoUpdateParametersConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Pce Auto Update Parameters | PcepLsp=$
↪{Egress} |
```

static create_pcep_pce_lsp(*Sessions*, ***kwargs*)

创建 PCEP PCE LSP 对象

参数 Sessions (Pcep) -- ：PCEP 协议会话对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP PCE LSP 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PCEP PCE LSP, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LspCount** (*int*) -- Lsp 数量, 类型为：number, 默认值：1, 取值范围：1-65535
- **AutoGenSymbolicName** (*bool*) -- 系统自动生成 Symbolic Name, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SymbolicName** (*str*) -- 设置 Symbolic Name, 类型为：string, 默认值：PLSP_@s
- **PathSetupType** (*str*) -- 建立 LSP 的方法, 类型为：string, 默认值：SEGMENT_ROUTING, 取值范围：

  SEGMENT_ROUTING

  SRv6
- **SourceIpv4Address** (*str*) -- 起始源 IP 地址, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **SourceIpv4AddressStep** (*str*) -- 源 IP 地址的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **SourceIpv4AddressSessionOffset** -- 源 IPv4 地址接口跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围: IPv4 地址
- **DestinationIpv4Address** (*str*) -- 起始目的 IP 地址, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **DestinationIpv4AddressStep** (*str*) -- 目的 IP 地址的跳变步长, 类型为：string, 默认值：0.0.0.1, 取值范围: IPv4 地址
- **DestinationIpv4AddressSessionOffset** -- 目的 IPv4 地址跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：0.0.1.0, 取值范围: IPv4 地址
- **SourceIpv6Address** (*str*) -- 源 IPv6 地址的跳变步长, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **SourceIpv6AddressStep** (*str*) -- 源 IPv6 地址的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **SourceIpv6AddressSessionOffset** -- 源 IPv6 地址接口跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围: IPv6 地址
- **DestinationIpv6Address** (*str*) -- 起始目的 IPv6 地址, 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址

- **DestinationIpv6AddressStep** (*str*) -- 目的 IPv6 地址的跳变步长, 类型为：string, 默认值：::1, 取值范围: IPv6 地址
- **DestinationIpv6AddressSessionOffset** -- 目的 IPv6 地址跳变, 指定 PCEP 会话块中源 IP 地址在会话之间的跳变步长, 类型为：string, 默认值：::1:0, 取值范围: IPv6 地址
- **LspInitiateMethod** (*str*) -- LSP 初始方式, 类型为：string, 默认值：UPDATE, 取值范围:

  UPDATE

  PCE_INITIATE

  REPLY
- **ImmediateUpdate** (*bool*) -- 直接更新, 类型为：bool, 取值范围：True 或 False, 默认值：True

返回 Pcep Pce Lsp 对象列表, 类型: object / list

返回类型 (PceLspConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
```

static create_pcep_pce_lsp_info(*PcepLsps*, ***kwargs*)

创建 PCEP PCE LSP INFO 对象

参数 **PcepLsps** (PceLspConfig) -- : PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP PCE LSP INFO 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP PCE LSP INFO 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Administrator** (*bool*) -- 使能 Administrative, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **State** (*str*) -- 设置 Initial LSP State, 类型为:string, 默认值:GOING_UP, 取值范围:

  DOWN

  UP

  ACTIVE

  GOING_DOWN

  GOING_UP

  RESERVED_5

  RESERVED_6

  RESERVED_7

返回 Pcep Pce Lsp Info 对象列表, 类型: object / list

返回类型 (PceLspInfoObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Info} | Create Pcep Pce Lsp Info | PcepLsp=${Egress} |
```

static create_pcep_rp_object(*PcepLsps*, ***kwargs*)

创建 PCEP PCC Rp Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP PCC Rp Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP PCC Rp Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **AutoGeneratedId** (*bool*) -- 自动生成 RP-ID 类型为:bool, 取值范围:True 或 False, 默认值: True
- **RpId** (*int*) -- 指定起始 RP-ID, Auto-Generated RP-ID 未选中时启用, 类型为: number, 取值范围: 0-4294967295, 默认值: 1
- **RpIdStep** (*int*) -- 指定 PLSP-ID 的跳变步长, 类型为: number, 取值范围: 0-4294967295, 默认值: 1
- **Priority** (*int*) -- 指定请求的优先级。数字越大，优先级越高, 类型为: number, 取值范围: 0-7, 默认值: 0
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCRep 消息中 I Flag 置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **BFlag** (*bool*) -- RP Object 的 B(Bi-directional) 位置位, 类型为: bool, 取值范围: True 或 False, 默认值: False
- **OFlag** (*bool*) -- RP Object 的 O(strict/loose) 位置位, 类型为: bool, 取值范围: True 或 False, 默认值: False

返回 Pcep Pcc Rp Object 对象列表, 类型: object / list

返回类型 (PcepRpObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Parameter} | Create Pcep Rp Object | PcepLsp=${Egress} |
```

static create_pcep_sr_ero_object(*PcepLsps*, ***kwargs*)

创建 PCEP Sr Ero Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Sr Ero Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Sr Ero Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True

- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep Metric List 对象列表, 类型: object / list

返回类型 (PcepSrEroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Sr Ero Object | PcepLsp=${Egress} |
```

static create_pcep_sr_ero_sub_object(*PcepSrEroObjects*, ***kwargs*)

创建 PCEP Sr Ero Sub Object 对象

参数 **PcepSrEroObjects** (PcepSrEroObjectConfig) -- ：PCEP PCE LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Sr Ero Sub Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Sr Ero Sub Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- PCReq 消息中 P Flag 是否置位, 类型为：string, 默认值：STRICT, 取值范围:

  STRICT

  LOOSE
- **NaiType** (*str*) -- PCReq 消息中 I Flag 是否置位, 类型为：string, 默认值：IPV4_NODE_ID, 取值范围:

  ABSENT

  IPV4_NODE_ID

  IPV6_NODE_ID

  IPV4_ADJACENCY

  IPV6_ADJACENCY_GLOBAL

  UNNUMBERED_ADJACENCY

  IPV6_ADJACENCY_LINK_LOCAL
- **MFlag** (*bool*) -- M Flag, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **CFlag** (*bool*) -- C Flag, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SFlag** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- F Flag, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SidLabel** (*int*) -- SID Label, 类型为：number, 取值范围：0-1048575, 默认值：16

- **SidLabelStep** (*int*) -- SID Label 跳变, 类型为：number, 取值范围：0-1048575, 默认值：0
- **SidLabelSessionOffset** (*int*) -- SID Label 接口间跳变, 类型为：number, 取值范围：0-1048575, 默认值：0
- **SidIndex** (*int*) -- SID Index（32 Bits）, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **SidIndexStep** (*int*) -- SID Index 跳变（32 Bits）, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **SidIndexSessionOffset** (*int*) -- SID Index 接口间跳变（32 Bits）, 类型为：number, 取值范围：0-1048575, 默认值：0
- **SidTrafficClass** (*int*) -- SID Traffic Class（3 bits）, 类型为：number, 取值范围：0-7, 默认值：0
- **SidTimeToLive** (*int*) -- SID Time To Liv, 类型为：number, 取值范围：0-255, 默认值：255
- **SidBottomOfStack** (*bool*) -- SID Bottom Of Stack Flag（1 Bit）, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NaiIpv4NodeId** (*str*) -- NAI IPv4 Node ID, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiIpv6NodeId** (*str*) -- NAI IPv6 Node ID, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiLocalIpv4Address** (*str*) -- NAI Local IPv4 Address, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiLocalIpv6Address** (*str*) -- NAI Local IPv6 Address, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiRemoteIpv4Address** (*str*) -- NAI Remote IPv4 Address, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **NaiRemoteIpv6Address** (*str*) -- NAI Remote IPv6 Address, 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **NaiLocalNodeId** (*str*) -- NAI Local Node-ID, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiLocalInterfaceId** (*int*) -- NAI Local Interface ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **NaiRemoteNodeId** (*str*) -- NAI Remote Node-ID, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **NaiRemoteInterfaceId** (*int*) -- NAI Remote Interface ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Pcep Sr Ero Sub Object 对象列表, 类型: object / list

返回类型 (PcepSrEroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Sr Ero Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Sr Ero Sub Object | PcepSrEroObjects=$
↪{Object} |
```

static create_pcep_sr_rro_object(*PcepLsps*, ***kwargs*)

创建 PCEP Sr Rro Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Sr Ero Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Sr Ero Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False

返回 Pcep Metric List 对象列表, 类型: object / list

返回类型 (PcepSrRroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Sr Rro Object | PcepLsp=${Egress} |
```

static create_pcep_sr_rro_sub_object(*PcepSrRroObjects*, ***kwargs*)

创建 PCEP Sr Rro Sub Object 对象

参数 **PcepLsps** (PcepSrRroObjectConfig) -- : PCEP PCE Sr Rro 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Sr Rro Sub Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Sr Rro Sub Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **NaiType** (*str*) -- 子对象中 NT 字段的值, 类型为: string, 默认值: IPV4_NODE_ID, 取值范围:

  ABSENT

  IPV4_NODE_ID

  IPV6_NODE_ID

  IPV4_ADJACENCY

  IPV6_ADJACENCY_GLOBAL

  UNNUMBERED_ADJACENCY

IPV6_ADJACENCY_LINK_LOCAL

- **MFlag** (*bool*) -- 子对象中的 M Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **CFlag** (*bool*) -- 子对象中的 C Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SFlag** (*bool*) -- 子对象中的 S Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- 子对象中的 F Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SidIndex** (*int*) -- 指定起始 SID Index, 类型为：number, 取值范围：0-1048575, 默认值：1
- **SidLabel** (*int*) -- 指定起始 SID Label, 类型为：number, 取值范围：0-1048575, 默认值：16
- **SidTrafficClass** (*int*) -- 指定流量类型字段的值, 类型为：number, 取值范围：0-7, 默认值：0
- **SidTimeToLive** (*int*) -- 指定 TTL 字段值, 类型为：number, 取值范围：0-255, 默认值：255
- **SidBottomOfStack** (*bool*) -- 是否指定的 SID Label 为标签栈的栈底标签, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NaiIpv4NodeId** (*str*) -- 指定 NAI IPv4 节点 ID, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiIpv6NodeId** (*str*) -- 指定 NAI IPv6 节点 ID, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiLocalIpv4Address** (*str*) -- 指定 NAI 本地 IPv4 地址, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiLocalIpv6Address** (*str*) -- 指定 NAI 本地 IPv6 地址, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiRemoteIpv4Address** (*str*) -- 指定 NAI 远端 IPv4 地址, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **NaiRemoteIpv6Address** (*str*) -- 指定 NAI 远端 IPv6 地址, 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **NaiLocalNodeId** (*str*) -- 指定 NAI 远端节点 ID, 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **NaiLocalInterfaceId** (*int*) -- 指定 NAI 本地接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **NaiRemoteNodeId** (*str*) -- 指定 NAI 远端节点 ID, 类型为：string, 默认值：193.85.1.1, 取值范围: IPv4 地址
- **NaiRemoteInterfaceId** (*int*) -- 指定 NAI 远端接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Pcep Sr Ero Sub Object 对象列表, 类型: object / list

返回类型 (PcepSrRroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Sr Rro Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Sr Rro Sub Object | PcepSrRroObjects=$
↪{Object} |
```

static create_pcep_srp_info(*PcepLsps*, ***kwargs*)

创建 PCEP Srp Info 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCE LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Srp Info 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Srp Info 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **AutoGeneratedId** (*bool*) -- 自动生成 SRP-ID, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **SrpId** (*int*) -- 指定起始 SRP-ID, 类型为: number, 取值范围: 0-4294967295, 默认值: 1
- **SrpIdStep** (*int*) -- 指定 SRP-ID 的跳变步长, 类型为: number, 取值范围: 0-4294967295, 默认值: 1

返回 Pcep Srp Info 对象列表, 类型: object / list

返回类型 (PcepSrpObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srp Info | PcepLsp=${Egress} |
```

static create_pcep_srv6_ero_object(*PcepLsps*, ***kwargs*)

创建 PCEP Srv6 Ero Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Srv6 Ero Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Srv6 Ero Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False

返回 Pcep Srv6 Ero Object 对象列表, 类型: object / list

返回类型 (PcepSrv6EroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srv6 Ero Object | PcepLsp=${Egress} |
```

static create_pcep_srv6_ero_sub_object(*PcepSrv6EroObjects*, ***kwargs*)

创建 PCEP Srv6 Ero Sub Object 对象

参数 PcepSrv6EroObjects (PcepSrv6EroObjectConfig) -- ： PCEP Srv6 Ero 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Srv6 Ero Sub Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Srv6 Ero Sub Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteType** (*str*) -- 建立 LSP 使用的路由类型, 类型为：string, 默认值：STRICT, 取值范围:

  STRICT

  LOOSE

- **NaiType** (*str*) -- 指定端点行为, 类型为：string, 默认值：IPV6_NODE_ID, 取值范围:

  ABSENT

  IPV4_NODE_ID

  IPV6_NODE_ID

  IPV4_ADJACENCY

  IPV6_ADJACENCY_GLOBAL

  UNNUMBERED_ADJACENCY

  IPV6_ADJACENCY_LINK_LOCAL

- **SFlag** (*bool*) -- 子对象中的 S Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- 子对象中的 F Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EndpointBehavior** (*str*) -- 指定端点行为, 类型为：string, 默认值：Invalid, 取值范围:

  Invalid

  EndNoPspUsp

  EndPsp

  EndUsp

  EndPspUsp

  EndXNoPspUsp

  EndXPsp

  EndXUsp

  EndXPspUsp

  EndTNoPspUsp

EndTPsp

EndTUsp

EndTPspUsp

EndB6Encaps

EndBM

EndDX6

EndDX4

EndDT6

EndDT4

EndDT46

EndDX2

EndDX2V

EndDT2U

EndDT2M

ENDB6EncapsRed

EndUSD

EndPSPUSD

EndUSPUSD

EndPSPUSPUSD

EndXUSD

EndXPSPUSD

EndXUSPUSD

EndXPSPUSPUSD

EndTUSD

EndTPSPUSD

EndTUSPUSD

EndTPSPUSPUSD

- **SRv6Sid** (*str*) -- 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiIpv6NodeId** (*str*) -- 类型为: string, 默认值: 2000::1, 取值范围: IPv6 地址
- **NaiLocalIpv6Address** (*str*) -- 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiRemoteIpv6Address** (*str*) -- 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **NaiLocalInterfaceId** (*int*) -- 类型为: number, 取值范围: 0-4294967295, 默认值：0
- **NaiRemoteInterfaceId** (*int*) -- 类型为: number, 取值范围: 0-4294967295, 默认值：0

返回 Pcep Srv6 Ero Sub Object 对象列表, 类型: object / list

返回类型 (PcepSrv6EroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srv6 Ero Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Srv6 Ero Sub Object | PcepSrEroObjects=$
↪{Object} |
```

static create_pcep_srv6_rro_object(*PcepLsps*, ***kwargs*)

创建 PCEP Srv6 Rro Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP PCC LSP 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Srv6 Rro Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Srv6 Rro Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为: bool, 取值范围: True 或 False, 默认值: False

返回 Pcep Srv6 Rro Onject 对象列表, 类型: object / list

返回类型 (PcepSrv6RroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srv6 Rro Object | PcepLsp=${Egress} |
```

static create_pcep_srv6_rro_sub_object(*PcepSrv6RroObjects*, ***kwargs*)

创建 PCEP Srv6 Rro Sub Object 对象

参数 **PcepSrv6RroObjects** (PcepSrv6RroObjectConfig) -- : PCEP Srv6 Rro 对象列表, 类型为: object / list

关键字参数

- **Name** (*str*) -- PCEP Srv6 Rro Sub Object 对象名称, 类型为: string
- **Enable** (*bool*) -- PCEP Srv6 Rro Sub Object 对象, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **NaiType** (*str*) -- 子对象中 NT 字段的值, 类型为: string, 默认值: IPV6_NODE_ID, 取值范围:

  ABSENT

  IPV4_NODE_ID

  IPV6_NODE_ID

  IPV4_ADJACENCY

  IPV6_ADJACENCY_GLOBAL

  UNNUMBERED_ADJACENCY

IPV6_ADJACENCY_LINK_LOCAL

- **SFlag** (*bool*) -- 子对象中的 S Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- 子对象中的 F Flag 置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EndpointBehavior** (*str*) -- 指定端点行为, 类型为：string, 默认值：Invalid, 取值范围:

  Invalid

  EndNoPspUsp

  EndPsp

  EndUsp

  EndPspUsp

  EndXNoPspUsp

  EndXPsp

  EndXUsp

  EndXPspUsp

  EndTNoPspUsp

  EndTPsp

  EndTUsp

  EndTPspUsp

  EndB6Encaps

  EndBM

  EndDX6

  EndDX4

  EndDT6

  EndDT4

  EndDT46

  EndDX2

  EndDX2V

  EndDT2U

  EndDT2M

  ENDB6EncapsRed

  EndUSD

  EndPSPUSD

  EndUSPUSD

  EndPSPUSPUSD

  EndXUSD

  EndXPSPUSD

  EndXUSPUSD

  EndXPSPUSPUSD

EndTUSD

EndTPSPUSD

EndTUSPUSD

EndTPSPUSPUSD

- **SRv6Sid** (*str*) -- 指定 SRv6 SID, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiIpv6NodeId** (*str*) -- 指定 NAI IPv6 节点 ID, 指定 NAI IPv6 节点 ID, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiLocalIpv6Address** (*str*) -- 指定 NAI 本地 IPv6 地址, 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **NaiRemoteIpv6Address** (*str*) -- 指定 NAI 远端 IPv6 地址, 类型为：string, 默认值：2001::1, 取值范围: IPv6 地址
- **NaiLocalInterfaceId** (*int*) -- 指定 NAI 本地接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **NaiRemoteInterfaceId** (*int*) -- 指定 NAI 远端接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 Pcep Srv6 Rro Sub Object 对象列表, 类型: object / list

返回类型 (PcepSrv6RroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pcc Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Srv6 Rro Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Srv6 Rro Sub Object | PcepSrv6RroObjects=$
↪{Object} |
```

static create_pcep_xro_object(*PcepLsps*, ***kwargs*)

创建 PCEP XRO Object 对象

参数 **PcepLsps** (PccLspConfig) -- : PCEP LSP 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP XRO Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP XRO Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **PFlag** (*bool*) -- PCReq 消息中 P Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IFlag** (*bool*) -- PCReq 消息中 I Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **FFlag** (*bool*) -- PCReq 消息中 F Flag 是否置位, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 Pcep XRO Object 对象列表, 类型: object / list

返回类型 (PcepXroObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep XRO Object | PcepLsp=${Egress} |
```

static create_pcep_xro_sub_object(*PcepXroObjects*, ***kwargs*)

创建 PCEP XRO Object 对象

参数 **PcepXroObjects** (PcepXroObjectConfig) -- ：PCEP XRO Object 对象列表, 类型为：object / list

关键字参数

- **Name** (*str*) -- PCEP Xro Sub Object 对象名称, 类型为：string
- **Enable** (*bool*) -- PCEP Xro Sub Object 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **XFlag** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Type** (*str*) -- 类型为：string, 默认值：IPv4_PREFIX, 取值范围:

  IPv4_PREFIX

  IPv6_PREFIX

  UNNUMBERED_INTERFACE_ID

  AUTONOMOUS_SYS_NUM

  SRLG
- **PrefixLength** (*int*) -- 类型为：number, 取值范围：0-32, 默认值：24
- **Attribute** (*str*) -- 类型为：string, 默认值：INTERFACE, 取值范围:

  INTERFACE

  NODE

  SRLG
- **Ipv4Address** (*str*) -- 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **Ipv6Address** (*str*) -- 类型为：string, 默认值：2000::1, 取值范围: IPv6 地址
- **TeRouterId** (*str*) -- 类型为：string, 默认值：192.85.1.1, 取值范围: IPv4 地址
- **InterfaceId** (*int*) -- 类型为：number, 默认值：0
- **AsNumber** (*int*) -- 类型为：number, 默认值：0
- **SrlgId** (*int*) -- 类型为：number, 默认值：0

返回 Pcep Xro Sub Object 对象列表, 类型: object / list

返回类型 (PcepXroSubObjectConfig)

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | HelloType=DIRECT_TARGETED |
| ${Egress} | Create Pcep Pce Lsp | Sessions=${Session} |
| ${Object} | Create Pcep Xro Object | PcepLsp=${Egress} |
| ${Subobject} | Create Pcep Xro Sub Object | PcepXroObjects=${Object} |
```

static create_peclsp_for_srte(*Excel*, *Session*, *TunnelCount=16000*, *PcelspCount=4000*, *SymbolicNameIdentification='Tunnel'*)

从 Excel 表格创建 SRTE 性能测试 PCE LSP

参数

- **Excel** -- Excel 文件完整路径
- **Session** -- PceLspConfig 对象
- **TunnelCount** -- 隧道数量
- **PcelspCount** -- pcelsp 数量

返回 布尔值 Bool (范围：True / False)

返回类型 bool

Examples:

static create_pim(*Port*, ***kwargs*)

创建 PIM 协议会话对象

参数 Port (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- PIM 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SessionMode** (*str*) -- 协议模式, 类型为：string, 默认值：SM, 支持版本：

  SM

  SSM
- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPV4, 支持版本：

  IPV4

  IPV6
- **DrPriority** (*int*) -- DR 优先级, 类型为：number, 取值范围：1-65535, 默认值：1
- **DrAddr** (*str*) -- DR 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.0
- **DrIpv6Addr** (*str*) -- DR IPv6 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：'::'
- **GenIdMode** (*str*) -- GenID 模式, 类型为：string, 默认值：FIXED, 支持参数：

  FIXED

  INCR

  RAND

- **RegisterEnable** (*bool*) -- Register 使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **BsrEnable** (*bool*) -- BSR 使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **BsrPriority** (*int*) -- BSR 优先级, 类型为：number, 取值范围：0-255, 默认值：1
- **BsrInterval** (*int*) -- BSR 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-3600, 默认值：60
- **HelloInterval** (*int*) -- Hello 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-3600, 默认值：30
- **HelloHoldTime** (*int*) -- Hello 消息超时时间（秒）, 类型为：number, 取值范围：1-65535, 默认值：105
- **JoinPruneInterval** (*int*) -- Join/Prune 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-65535, 默认值：60
- **JoinPruneHoldTime** (*int*) -- Join/Prune 消息超时时间（秒）, 类型为：number, 取值范围：1-65535, 默认值：210

返回 PIM 协议会话对象, 类型：object

返回类型 (PimRouter)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| ${DrAddr} | Set Variable | ${Session.DrAddr} |
```

static create_pim_group(*Session*, ***kwargs*)

创建 PIM Group 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM Group 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM Group 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **GroupCheck** (*bool*) -- 协议模式, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GroupType** (*str*) -- 组类别, 类型为：string, 默认值：ANY_G, 支持版本：

  ANY_G

  S_G

  S_G_RPT

  ANY_RP
- **GroupAddr** (*str*) -- 组地址, 类型为：string, 取值范围：IPv4 地址, 默认值：225.0.0.1
- **GroupCount** (*int*) -- 组数目, 类型为：number, 取值范围：1-65535（BigTao）1-500000（DarYu）, 默认值：1
- **GroupModifierStep** (*int*) -- 组地址增量步进, 类型为：number, 取值范围：0-65535, 默认值：1

- **GroupModifierBit** (*int*) -- 组地址增量位, 类型为：number, 取值范围：1-32, 默认值：32
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：10.10.10.10
- **JoinSrc** (*str*) -- Join 源地址, 类型为：string, 取值范围：IPv4 地址, 默认值：1.1.1.1
- **JoinMaskLen** (*int*) -- Join 掩码长度, 类型为：number, 取值范围：1-32, 默认值：32
- **PruneSrcAddr** -- Prune 源地址, 类型为：string, 取值范围：IPv4 地址, 默认值：1.1.1.1
- **PruneMaskLen** (*int*) -- Prune 掩码长度, 类型为：number, 取值范围：1-32, 默认值：32

返回 PIM Group 对象, 类型：object

返回类型 (PimGroupConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim Group | Session=${Session} | GroupAddr=255.0.0.2 |
```

static create_pim_ipv6_group(*Session*, ***kwargs*)

创建 PIM IPv6 Group 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv6 Group 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM IPv6 Group 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **GroupCheck** (*bool*) -- 协议模式, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GroupType** (*str*) -- 组类别, 类型为：string, 默认值：ANY_G, 支持版本：

  ANY_G

  S_G

  S_G_RPT

  ANY_RP
- **GroupAddr** (*str*) -- 组地址, 类型为：string, 取值范围：IPv6 地址, 默认值：ff1e::1
- **GroupCount** (*int*) -- 组数目, 类型为：number, 取值范围：1-65535（BigTao）1-500000（DarYu）, 默认值：1
- **GroupModifierStep** (*int*) -- 组地址增量步进, 类型为：number, 取值范围：0-65535, 默认值：1
- **GroupModifierBit** (*int*) -- 组地址增量位, 类型为：number, 取值范围：1-128, 默认值：128
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1

- **JoinSrc** (*str*) -- Join 源地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **JoinMaskLen** (*int*) -- Join 掩码长度, 类型为：number, 取值范围：1-128, 默认值：64
- **PruneSrcAddr** (*str*) -- Prune 源地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PruneMaskLen** (*int*) -- Prune 掩码长度, 类型为：number, 取值范围：1-128, 默认值：64

返回 PIM IPv6 Group 对象, 类型：object

返回类型 (PimIpv6GroupConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim IPv6 Group | Session=${Session} | GroupAddr=ff1e::2 |
```

static create_pim_ipv6_register_group(*Session*, ***kwargs*)

创建 PIM IPv6 Register Group 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv6 Register Group 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM IPv6 Register Group 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **MulticastGroupToSourceDistribution** (*str*) -- 组播地址和源地址映射方式, 类型为：string, 默认值：PAIR, 支持参数：

  PAIR

  BACKBONE
- **RegisterTransmitMode** (*str*) -- 注册发送模式, 类型为：string, 默认值：CONTINUOUS, 支持参数：

  FIXED

  CONTINUOUS
- **FixedModeCount** (*int*) -- 固定模式数量, 类型为：number, 取值范围：1-1000, 默认值：5
- **MulticastGroupCount** (*int*) -- 多播组数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartMulticastGroupAddr** (*str*) -- 多播组起始地址, 类型为：string, 取值范围：IPv6 地址, 默认值：ff1e::2
- **MulticastGroupStep** (*int*) -- 多播组步长, 类型为：number, 取值范围：1-255, 默认值：1
- **MulticastGroupPrefixLength** (*int*) -- 多播组前缀长度, 类型为：number, 取值范围：1-128, 默认值：64
- **MulticastSourceCount** (*int*) -- 组播源数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartMulticastSourceAddr** (*str*) -- 多播组起始地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2001::1

- **MulticastSourceStep** (*int*) -- 多播组步长, 类型为：number, 取值范围：1-255, 默认值：1
- **MulticastSourcePrefixLength** (*int*) -- 多播组前缀长度, 类型为：number, 取值范围：1-128, 默认值：64
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::2
- **RegisterTransmitInterval** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：10-180, 默认值：60

返回 Create Pim Ipv6 Register Group 对象, 类型：object

返回类型 (PimIpv6RegisterGroupConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim Ipv6 Register Group | Session=${Session} | RpAddr=3000::2 |
```

static create_pim_ipv6_rp_map(*Session*, ***kwargs*)

创建 PIM IPv6 Rp Map 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv6 Rp Map 对象名称, 类型为：string
- **Enable** (*bool*) -- 使能 PIM IPv6 Rp Map 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **MulticastGroupAddr** (*str*) -- 组播地址和源地址映射方式, 类型为：string, 取值范围：IPv6 地址, 默认值：ff1e::1
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **PrefixLength** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：0-128, 默认值：128
- **RpPriority** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：0-255, 默认值：0
- **RpHoldTime** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：1-65535, 默认值：150

返回 PIM IPv6 Rp Map 对象, 类型：object

返回类型 (PimIpv6RpMapConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim Ipv6 Rp Map | Session=${Session} | RpAddr=3000::1 |
```

static create_pim_register_group(*Session*, ***kwargs*)

创建 PIM IPv4 Register Group 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv4 Register Group 对象名称, 类型为：string

- **Enable** (*bool*) -- 使能 PIM IPv4 Register Group 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **MulticastGroupToSourceDistribution** (*str*) -- 组播地址和源地址映射方式, 类型为：string, 默认值：PAIR, 支持参数：

  PAIR

  BACKBONE
- **RegisterTransmitMode** (*str*) -- 注册发送模式, 类型为：string, 默认值：CONTINUOUS, 支持参数：

  FIXED

  CONTINUOUS
- **FixedModeCount** (*int*) -- 固定模式数量, 类型为：number, 取值范围：1-1000, 默认值：5
- **MulticastGroupCount** (*int*) -- 多播组数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartMulticastGroupAddr** (*str*) -- 多播组起始地址, 类型为：string, 取值范围：IPv4 地址, 默认值：225.0.1.1
- **MulticastGroupStep** (*int*) -- 多播组步长, 类型为：number, 取值范围：1-255, 默认值：1
- **MulticastGroupPrefixLength** (*int*) -- 多播组前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **MulticastSourceCount** (*int*) -- 组播源数量, 类型为：number, 取值范围：1-65535, 默认值：1
- **StartMulticastSourceAddr** (*str*) -- 多播组起始地址, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1
- **MulticastSourceStep** (*int*) -- 多播组步长, 类型为：number, 取值范围：1-255, 默认值：1
- **MulticastSourcePrefixLength** (*int*) -- 多播组前缀长度, 类型为：number, 取值范围：1-32, 默认值：24
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：10.10.10.20
- **RegisterTransmitInterval** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：10-180, 默认值：60

返回 PIM IPv4 Register Group 对象, 类型：object

返回类型 (PimRegisterGroupConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim Register Group | Session=${Session} | RpAddr=20.10.10.20 |
```

static create_pim_rp_map(*Session*, ***kwargs*)

创建 PIM IPv4 Rp Map 对象

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM IPv4 Rp Map 对象名称, 类型为：string

- **Enable** (*bool*) -- 使能 PIM IPv4 Rp Map 对象, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **MulticastGroupAddr** (*str*) -- 组播地址和源地址映射方式, 类型为：string, 取值范围：IPv4 地址, 默认值：255.0.0.1
- **RpAddr** (*str*) -- RP 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：10.10.10.10
- **PrefixLength** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：0-32, 默认值：32
- **RpPriority** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：0-255, 默认值：0
- **RpHoldTime** (*int*) -- Register 发送时间间隔（秒）, 类型为：number, 取值范围：1-65535, 默认值：150

返回 PIM IPv6 Rp Map 对象, 类型：object

返回类型 (PimIpv6RpMapConfig)

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Create Pim IPv4 Rp Map | Session=${Session} | RpAddr=20.10.10.10 |
```

static create_pppoe(*Port*, *EmulationMode='CLIENT'*, ***kwargs*)

创建 PPPoE 协议会话对象

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：object
- **EmulationMode** (*str*) -- PPPoE 角色, 默认值：CLIENT, 取值范围：

  CLIENT SERVER

关键字参数

- **Name** (*str*) -- PPPoE 协会话名称
- **Enable** (*bool*) -- 使能 PPPoE 协议会话, 默认值：True
- **AuthenticationType** (*str*) -- 认证方式, 默认值：NO_AUTHENTICATION, 取值范围：

  NO_AUTHENTICATION NEGOTIATION CHAP_MD5 PAP
- **Username** (*str*) -- 用户名, 默认值：xinertel, 取值范围：string length in [1,126]
- **Password** (*str*) -- 密码, 默认值：xinertel, 取值范围：string length in [1,126]
- **ServiceName** (*str*) -- 服务名, 默认值：””, 取值范围：string length in [0,255]
- **EnableMaxPayloadTag** (*bool*) -- 使能最大净荷标签, 默认值：False
- **MaxPayloadBytes** (*int*) -- 最大净荷 (字节), 取值范围：1-65535, 默认值：1500
- **LcpConfigReqTimeout** (*int*) -- LCP Configure-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **LcpConfigReqMaxAttempts** (*int*) -- LCP Configure-Request 最大尝试次数, 取值范围：1-65535, 默认值：10

- **LcpTermReqTimeout** (*int*) -- LCP Terminate-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **LcpTermReqMaxAttempts** (*int*) -- LCP Terminate-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **NcpConfigReqTimeout** (*int*) -- NCP Configure-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **NcpConfigReqMaxAttempts** (*int*) -- NCP Configure-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **LcpNcpMaxNak** (*int*) -- LCP/NCP 最大 Nak 数量, 取值范围：1-65535, 默认值：5
- **EnableMruNegotiation** (*bool*) -- 使能 MRU 协商, 默认值：True
- **MruSize** (*int*) -- MRU(字节), 取值范围：128-65535, 默认值：1492
- **EnableEchoRequest** (*bool*) -- 使能 Echo-Request 报文, 默认值：False
- **EchoRequestInterval** (*int*) -- Echo-Request 间隔 (sec), 取值范围：1-65535, 默认值：10
- **EchoRequestMaxAttempts** (*int*) -- Echo-Request 最大尝试次数, 取值范围：1-65535, 默认值：3
- **EnableMagicNumber** (*bool*) -- 使能 Magic Number, 默认值：True
- **PadiTimeout** (*int*) -- Client 参数, PADI 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PadiMaxAttempts** (*int*) -- Client 参数, PADI 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PadrTimeout** (*int*) -- Client 参数, PADR 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PadrMaxAttempts** (*int*) -- Client 参数, PADR 最大尝试次数, 取值范围：1-65535, 默认值：10
- **EnableRelayAgent** (*bool*) -- Client 参数, 启用中继代理, 默认值：False
- **RelayAgentDestMac** (*str*) -- Client 参数, 中继代理 MAC 地址, 取值范围：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **RelayAgentDestMacStep** (*str*) -- Client 参数, 中继代理 MAC 地址跳变, 取值范围：有效的 mac 地址, 默认值：00:00:00:00:00:01
- **UseRelayAgentPadi** (*bool*) -- Client 参数, PADI 中包含中继代理信息, 默认值：True
- **UseRelayAgentPadr** (*bool*) -- Client 参数, PADR 中包含中继代理信息, 默认值：True
- **RelayAgentType** (*str*) -- Client 参数, 中继代理类型, 默认值：RFC2516, 取值范围：

  RFC2516 DSL_FORUM
- **RelaySessionId** (*str*) -- Client 参数, 中继会话 ID, 取值范围：string length in [0,12], 默认值：""
- **CircuitId** (*str*) -- Client 参数, 环路 ID, 取值范围：string length in [0,63], 默认值：@s
- **RemoteId** (*str*) -- Client 参数, 远程 ID, 取值范围：string length in [0,63], 默认值：@m-@p
- **ChapChalReqTimeout** (*int*) -- Client 参数, CHAP Challenge Request 超时时间 (sec), 取值范围：1-65535, 默认值：3

- **ChapAckTimeout** (*int*) -- Client 参数, CHAP Ack 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapMaxReplyAttempts** (*int*) -- Client 参数, CHAP Reply 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PapReqTimeout** (*int*) -- Client 参数, PAP Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PapReqMaxAttempts** (*int*) -- Client 参数, PAP Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **EnableAutoRetry** (*bool*) -- Client 参数, 使能 PPPoE 协议会话, 默认值：False
- **AutoRetryCount** (*int*) -- Client 参数, 重连次数, 取值范围：1-65535, 默认值：65535
- **LcpDelay** (*int*) -- Client 参数, LCP 推迟时间 (ms), 取值范围：1-65535, 默认值：0
- **EnableAutoFillIpv6** (*bool*) -- Client 参数, 启用获取 Global IPv6 地址, 默认值：True
- **AcName** (*str*) -- Server 参数, 访问集中器名称, 默认值：Xinertel
- **ChapReplyTimeout** (*int*) -- Server 参数, CHAP Reply 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapMaxChalAttempts** (*int*) -- Server 参数, CHAP Challenge 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PapPeerReqTimeout** (*int*) -- Server 参数, 等待 PAP Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **Ipv4Start** (*str*) -- Server 参数, IPv4 起始地址, 默认值：192.0.1.0
- **Ipv4Step** (*str*) -- Server 参数, IPv4 地址步长, 默认值：0.0.0。1
- **Ipv4Count** (*int*) -- Server 参数, IPv4 地址数量, 取值范围：1-65535, 默认值：3
- **Ipv6InterfaceId** (*str*) -- Server 参数, 起始 Interface ID, 默认值：”::2”
- **Ipv6InterfaceIdStep** (*str*) -- Server 参数, Interface ID 跳变步长, 默认值：”::1”
- **Ipv6PrefixStart** (*str*) -- Server 参数, IPv6 起始前缀, 默认值：”2002::”
- **Ipv6PrefixStep** (*str*) -- Server 参数, IPv6 前缀跳变步长, 默认值：”0:0:0:1::”
- **Ipv6Count** (*int*) -- Server 参数, IPv6 前缀数量, 取值范围：1-65535, 默认值：1
- **EnableForceConnectMode** (*bool*) -- 强制重连模式, 默认值：False
- **UnconnectedSessionThreshold** (*int*) -- 未连接会话门限值, 取值范围：1-65535, 默认值：1
- **MAndOFlag** (*str*) -- Server 参数, M 与 O 标志位, 默认值：M0_O0, 支持 M0_O0 M0_O1 M1

返回 PPPoE 协议会话对象, 类型：object

返回类型 (PppoeClent)

实际案例

```
| Create Pppoe | Port=${Port} |
| Create Pppoe | Port=${Port} | EmulationMode=Server |
```

static create_pppoe_custom_option(*Session*, ***kwargs*)

编辑 PPPoE 自定义选项

参数 **Session** (PppoeClent) or (PppoeServer) -- PPPoE 协议会话对象

关键字参数

- **OptionValue** (*int*) -- 选项标识符, 默认值: 0, 取值范围: 0-65535
- **SubPortocolType** (*str*) -- 包含选项的消息类型, 默认值：1 sec, 支持类型:

  LinkControlProtocol

  IPControlProtocol

  IPv6ControlProtocol

  PPPoEPADIandPADR
- **UseWildcards** (*bool*) -- 使用通配符, 默认值：False
- **StringIsHexadecimal** (*int*) -- 使能十六进制字符, 默认值：False
- **OptionData** (*str*) -- 十进制选项载荷
- **OptionHexData** (*int*) -- 十六进制选项载荷

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Create Pppoe Custom Option | Sessions=${Sessions} | OptionValue=1 |␣
↪SubPortocolType=LinkControlProtocol | OptionData=55 |
```

static create_rip(*Port*, ***kwargs*)

创建 RIP 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- RIP 协议会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 RIP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- RIP 版本, 类型为：string, 默认值：RIPV2, 支持版本：

  RIPV1

  RIPV2

  RIPNG
- **UpdateType** (*str*) -- 仿真路由器指定发送 RIP 消息的通信方式, 类型为: string, 默认值：MULTICAST, 支持方式：

  BROADCAST

  MULTICAST

  UNICAST

- **DutIpv4Address** (*str*) -- 指定接收 RIP 消息的路由器的 IP 地址, 当 RIP 版本为 RIPv1 或者 RIPv2 时, 该选项可配。类型为: string, 默认值: 224.0.0.9
- **DutIpv6Address** (*str*) -- 指定接收 RIP 消息的路由器的 IP 地址, 当 RIP 版本为 RIPng 并且更新类型指定为 Unicast 时，该选项可配。类型为：string, 默认值：ff02::9
- **AuthMethod** (*str*) -- 认证方式, 当 RIP 版本为 RIPv2 时配置该选项。类型为：string, 默认值：NONE, 支持方式：

  NONE

  SIMPLE

  MD5
- **Password** (*str*) -- 当认证方式为 Simple/MD5 时，输入的认证密码, 类型为：string, 默认值：Xinetel
- **Md5KeyId** (*int*) -- 当认证方式为 MD5 时，输入的 MD5 密钥, 类型为: number, 取值范围：0-255, 默认值：1
- **UpdateInterval** (*int*) -- 发送 RIP 更新消息的时间间隔，单位为秒, 类型为：number, 取值范围：1-65535, 默认值：30
- **UpdateJitter** (*int*) -- 发送 RIP 更新消息的时间抖动, 类型为：number, 取值范围：0-5, 默认值：0
- **MaxRoutePerUpdate** (*int*) -- 更新消息中可携带的最大路由数, 类型为: number, 取值范围：1-70, 默认值：25
- **SplitHorizon** (*bool*) -- 是否开启水平分割功能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableViewRoutes** (*bool*) -- 是否需要查看学到的路由信息, 类型为: bool, 取值范围：True 或 False, 默认值：False
- **EnableIpAddrValidation** (*bool*) -- 验证收到的 IP 地址是否和本地地址在同一网段, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 RIP 协议会话对象, 类型：object

返回类型 (RipRouter)

实际案例

```
| Create Rip | Port=${Port} | EnableIpAddrValidation=True |
```

static create_rip_ipv4_route(*Session*, ***kwargs*)

创建 RIP IPv4 路由对象

参数 **Session** (RipRouter) -- Rip 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- RIP IPv4 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 RIP IPv4 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteCount** (*str*) -- 路由池中包含的路由的个数, 类型为：string, 取值范围：1-16777215, 默认值：1
- **StartIpv4Prefix** (*str*) -- 指定起始 IPv4 地址, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：192.168.1.0
- **PrefixLength** (*int*) -- 地址前缀长度, 类型为：number, 取值范围：1-32, 默认值：24

- **Increment** (*str*) -- 增量步长, 类型为：string, 取值范围：1-255, 默认值：1
- **NextHop** (*str*) -- 指定路由下一跳, 类型为：string, 取值范围：有效的 ipv4 地址, 默认值：0.0.0.0
- **Metric** (*int*) -- 路由度量, 16 表示不可达。类型为：number, 取值范围：1-16, 默认值：1
- **RouteTag** (*int*) -- 路由标签域的值, 0 表示没有 tag. 类型为：number, 取值范围：0-65535, 默认值：0

返回 RIP IPv4 路由对象, 类型：object

返回类型 (RipIpv4RouteConfig)

实际案例

```
| ${Session} | Create Rip | Port=${Port} |
| Create Rip Ipv4 Route | Session=${Session} | Metric=10 |
```

static create_rip_ipv6_route(*Session*, ***kwargs*)

创建 RIP IPv6 路由对象

参数 **Session** (RipRouter) -- Rip 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- RIP IPv6 路由名称, 类型为：string
- **Enable** (*bool*) -- 使能 RIP IPv6 路由, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouteCount** (*int*) -- 路由池中包含的路由的个数, 类型为：number, 取值范围：1-2147483647, 默认值：1
- **StartIpv6Prefix** (*str*) -- 起始 IPv6 地址, 类型为：string, 取值范围：有效的 IPv6 地址, 默认值：'2000::'
- **RouteStep** (*str*) -- IP 地址的增加步长, 类型为：string, 取值范围：有效的 IPv6 地址, 默认值：'0:0:0:1::'
- **PrefixLength** (*int*) -- 地址的前缀长度, 类型为：number, 取值范围：1-128, 默认值：64
- **NextHop** (*str*) -- 路由下一跳, 类型为：string, 取值范围：有效的 IPv6 地址, 默认值：'::'
- **Metric** (*int*) -- 路由度量, 类型为：number, 取值范围：1-16, 默认值：1
- **RouteTag** (*int*) -- 路由标签域的值, 0 表示没有 tag, 类型为：number, 取值范围：0-65535, 默认值：0

返回 RIP IPv6 路由对象, 类型：object

返回类型 (RipIpv6RouteConfig)

实际案例

```
| ${Session} | Create Rip | Port=${Port} |
| Create Rip Ipv6 Route | Session=${Session} |
```

static create_stream_header(*Stream*, *HeaderTypes*, *Index=None*)

创建流量报文头部

参数

- **Stream** (StreamTemplate) --
- **Index** (*int*) -- 报文头部创建在当前流量头部的序号, 类型为：number, 取值范围 None 或 0-16383, 当 Index 为 None 表示重新创建流量报文类型
- **HeaderTypes** (*list*) -- 报文头部类型列表, 类型为：list, 支持的报文头部(不区分大小写):

  ethernetii

  vlan

  vxlan

  arp

  ipv4

  ipv6

  tcp

  udp

  l2tpv2data

  ppp

  pppoe

  icmpv4echorequest

  destunreach

  icmpv4echoreply

  informationreply

  informationrequest

  icmpv4parameterproblem

  icmpv4redirect

  sourcequench

  timeexceeded

  timestampreply

  timestamprequest

  icmpmaskrequest

  icmpmaskreply

  destinationunreachable

  icmpv6echoreply

  icmpv6echorequest

  packettoobig

icmpv6parameterproblem
timeexceed
routersolicit
routeradvertise
icmpv6redirect
neighborsolicit
neighboradvertise
mldv1query
mldv1report
mldv1done
mldv2query
mldv2report
igmpv1
igmpv1query
igmpv2
igmpv2query
igmpv3report
igmpv3query
custom
ospfv2linkstateupdate
ospfv2linkstaterequest
ospfv2databasedescription
ospfv2linkstateacknowledge
ospfv2unknown
ospfv2hello
mpls

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${HeaderTypes} | Create List | EthernetII | IPv4 | TCP |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
```

static create_vxlan(*Port*, ***kwargs*)

创建 Vxlan 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- Vxlan 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 Vxlan 协议会话, 默认值：True
- **AutoUdpSourcePort** (*bool*) -- 自动计算 UDP 源端口, 默认值：True

- **UdpSourcePort** (*int*) -- 配置 UDP 源端口, 取值范围：3-4095, 默认值：1025
- **EnableUdpChecksum** (*bool*) -- 使能计算 UDP 校验和, 默认值：False
- **EvpnLearning** (*bool*) -- 使能 EVPN 学习, 默认值：False
- **OvsdbLearning** (*bool*) -- 使能 OVSDB 学习, 默认值：False
- **MulticastType** (*str*) -- 组播类型, 默认值：IGMP, 取值范围：

  IGMP

  PIM

  MLD
- **VtepTunnelIp** (*str*) -- VTEP 隧道 IP 地址, 默认值：INTERFACEIP, 取值范围：

  INTERFACEIP

  ROUTERID
- **EnableIrb** (*bool*) -- 默认值：False
- **RPAddress** (*str*) -- 选择 PIM 的 RP 地址, 取值范围：IPv4 地址, 默认值：192.0.0.1
- **RPIpv6Address** (*str*) -- 选择 PIM 的 RP 地址, 取值范围：IPv6 地址, 默认值：2000::1
- **IrbMode** (*str*) -- 默认值：Symmetric, 取值范围：

  Symmetric

返回 Vxlan 协议会话对象, 类型：object

返回类型 (Vxlan)

实际案例

```
| Create Vxlan | Port=${Port} |
```

static create_vxlan_segment(**kwargs*)

创建 Vxlan Segment 对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- Vxlan 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 Vxlan 协议会话, 默认值：True
- **StartVni** (*int*) -- 起始 VNI, 取值范围：0-16777215, 默认值：0
- **VniCount** (*int*) -- VNI 个数, 取值范围：1-65535, 默认值：1
- **VniStep** (*int*) -- VNI 跳变步长, 取值范围：1-65535, 默认值：1
- **CommunicationType** (*str*) -- 学习方式, 默认值：UNICAST, 取值范围：

  UNICAST

  MULTICAST

  VxlanEVPN

- **VniDistributionType** (*str*) -- VNI 在 VPN 之间的分配方式, 默认值: ROUNDROBIN, 取值范围:

  ROUNDROBIN

  LINEAR
- **EnableL3Vni** (*bool*) -- 使能 L3VNI, 默认值: False
- **StartL3Vni** (*int*) -- 起始 L3VNI, 取值范围: 1-16777215, 默认值: 1
- **L3VniStep** (*int*) -- L3VNI 跳变步长, 取值范围: 1-16777215, 默认值: 1
- **L3VniCount** (*int*) -- L3 VNI 数量, 取值范围: 1-65535, 默认值: 1
- **VniTrafficType** (*str*) -- 流端点模式, 默认值: ROUNDROBIN, 取值范围:

  L2VNI

  L3VNI

  L2AndL3VNI
- **EnableVmArp** (*bool*) -- 使能 VM ARP, 默认值: False

返回 Vxlan Segment 对象, 类型: object

返回类型 (VxlanSegmentConfig)

实际案例

```
| Create Vxlan Segment | Port=${Port} |
```

static del_benchmark()

static del_imix_distribution_frame(*IMix*, *Index=None*)

在 Imix 模板删除指定自定义帧长

参数

- **IMix** (Imix) -- 测试仪表 Imix 模板对象
- **Index** (*int*) -- 测试仪表 Imix 模板自定义帧长序号

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| ${Imix} | Create Imix | Name=Imix_1 | Seed=10121112 |
| Add IMix Distribution Frame | IMix=${Imix} | Type=random | Min=64 |␣
↪Max=128 | Weight=50 |
| Add IMix Distribution Frame | IMix=${Imix} | Type=random | Min=128 |␣
↪Max=256 | Weight=50 |
| Bind Stream Imix | Stream=${Stream} | Imix=${Imix} |
| Del IMix Distribution Frame | IMix=${Imix} | Index=1
```

static del_objects(*Objects*)

删除测试仪表相关对象

:param : param Objects: 测试仪表相关对象:param : type Objects: 类型为: list, 测试仪表相关对象 object 列表

Returns: 布尔值 Bool (范围: True / False)

实际案例

robotframework:

```
| ${Port} | Get Ports |
| Del Objects | Port=${Port} |
```

static del_port(*Ports=None*)

删除测试仪端口

参数 **Ports** -- 测试仪表端口对象列表, 类型为：list

返回 Returns：布尔值 Bool (范围：True / False)

实际案例

robotframework:

```
| Del Port | Ports=${Port_1} |
| Del Port |
```

static del_stream(*Ports=None*, *Streams=None*)

删除测试仪流量

参数

- **Ports** (list (*Port*)) -- 测试仪表端口对象列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量对象列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream_1} | Add Stream | Ports=${Port_1}
| ${Stream_2} | Add Stream | Ports=${Port_2}
| ${Stream_2} | Add Stream | Ports=${Port_3}
| Del Stream | Streams=${Stream_1} |
| Del Stream | Ports=${Port_2} |
| Del Stream |
```

static dhcp_abort(*Sessions*)

终止测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Abort | Sessions=${Sessions} |
```

static dhcp_bind(*Sessions*)

Bind 测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Bind | Sessions=${Sessions} |
```

static dhcp_rebind(*Sessions*)

重新 Bind 测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Rebind | Sessions=${Sessions} |
```

static dhcp_reboot(*Sessions*)

重新启动测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Reboot | Sessions=${Sessions} |
```

static dhcp_release(*Sessions*)

释放测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Release | Sessions=${Sessions} |
```

static dhcp_renew(*Sessions*)

Renew 测试仪表 DHCP 协议会话

参数 **Sessions** (DhcpClient) -- DHCPv4 Client 协议对象

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcp Renew | Sessions=${Sessions} |
```

static dhcpv6_client_abort(*Sessions*)

中断 DHCPv6/PD 客户端

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Abort | Sessions=${Sessions} |
```

static dhcpv6_client_active_lease_query(*Sessions*)

DHCPv6 客户端活动租借查询

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Active Lease Query | Sessions=${Sessions} |
```

static dhcpv6_client_bind(*Sessions*)

DHCPv6 客户端绑定地址

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Bind | Sessions=${Sessions} |
```

static dhcpv6_client_bulk_lease_query(*Sessions*, ***kwargs*)

DHCPv6 客户端批量租借查询

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list

Keyword Args：

QueryType (str): Bulk Leasequery 消息中 query-type 类型, 默认值: QUERY_BY_ADDRESS, 取值范围：

QUERY_BY_ADDRESS

QUERY_BY_CLIENTID

QUERY_BY_RELAY_ID

QUERY_BY_LINK_ADDRESS

QUERY_BY_REMOTE_ID

ClientAddress (str): 指定客户端 IPv6 地址, 默认值：'2000::1', 取值范围：有效的 ipv6 地址

ClientId (str): 客户端 ID, 默认值：", 取值范围：匹配正则表达式"^([0-9a-fA-F]{0,512})$"

RelayIdentifier (str): 中继器 ID, 默认值：", 取值范围：匹配正则表达式"^([0-9a-fA-F]{0,512})$"

LinkAddress (str): 链路地址, 默认值：'2000::1', 取值范围：有效的 ipv6 地址

RemoteId (str): 中继代理 Remote-ID 值, 默认值："

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Bulk Lease Query | Sessions=${Sessions} |
```

static dhcpv6_client_confirm(*Sessions*)

DHCPv6 客户端确认参数

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Confirm | Sessions=${Sessions} |
```

static dhcpv6_client_info_request(*Sessions*)

DHCPv6 客户端请求信息

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Info Request | Sessions=${Sessions} |
```

static dhcpv6_client_lease_query(*Sessions*, ***kwargs*)

DHCPv6 客户端租借查询

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

关键字参数

- **QueryType** (*str*) -- Leasequery 消息中 query-type 类型, 默认值: QUERY_BY_ADDRESS, 取值范围:

  QUERY_BY_ADDRESS

  QUERY_BY_CLIENTID

- **ClientAddress** (*str*) -- 客户端 IPv6 地址, 默认值: '2000::1', 取值范围: 有效的 ipv6 地址
- **ClientId** (*str*) -- 客户端 ID, 默认值: '', 取值范围: 匹配正则表达式"^([0-9a-fA-F]{0,512})$"

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Lease Query | Sessions=${Sessions} |
```

static dhcpv6_client_rebind(*Sessions*)

DHCPv6 客户端广播续租地址

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Rebind | Sessions=${Sessions} |
```

static dhcpv6_client_release(*Sessions*)

DHCPv6 客户端释放地址

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Release | Sessions=${Sessions} |
```

static dhcpv6_client_renew(*Sessions*)

DHCPv6 客户端单播续租地址

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Renew | Sessions=${Sessions} |
```

static dhcpv6_client_start_tls(*Sessions*)

DHCPv6 客户端启动 TLS

参数 **Sessions** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Client Start Tls | Sessions=${Sessions} |
```

static dhcpv6_server_abort(*Sessions*)

中断 DHCPv6/PD 服务器

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Abort | Sessions=${Sessions} |
```

static dhcpv6_server_reconfigure_rebind(*Sessions*)

DHCPv6 服务端重新配置 Rebind

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Reconfigure Rebind | Sessions=${Sessions} |
```

static dhcpv6_server_reconfigure_renew(*Sessions*)

DHCPv6 服务端重新配置 Renew

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Reconfigure Renew | Sessions=${Sessions} |
```

static dhcpv6_server_start(*Sessions*)

启动 DHCPv6 服务端

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Start | Sessions=${Sessions} |
```

static dhcpv6_server_stop(*Sessions*)

停止 DHCPv6 服务端

参数 **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为: object / list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Dhcpv6 Server Stop | Sessions=${Sessions} |
```

static disconnect_bgp(*Sessions*)

断开 BGP 协议会话连接

参数 **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Disconnect Bgp | Sessions=${Sessions} |
```

static disconnect_l2tp(*Sessions*)

断开 L2tp 协议会话

参数 **Sessions** (L2tp) -- L2tp 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Disconnect L2tp | Sessions=${Sessions} |
```

static disconnect_pppoe(*Sessions*)

断开 PPPoE 协议会话

参数 **Sessions** (list (PppoeClent)) -- PPPoE 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Disconnect Pppoe | Sessions=${Sessions} |
```

static dot1x_delete_certificate(*Sessions*)

删除 802.1x 证书

参数 **Sessions** (Dot1x) -- 802.1x 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dot1x Upload Certificate | Sessions=${Sessions} |
```

static dot1x_upload_certificate(*Sessions*, *Folder*)

上传 802.1x 证书

参数

- **Sessions** (Dot1x) -- 802.1x 会话对象, 类型为：object / list
- **Folder** (*str*) -- 证书所在路径，e.g.'c:/CertificateFolder'

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Dot1x Upload Certificate | Sessions=${Sessions} | Folder=${Folder} |
```

static download_packages(*Port*, *FileDir*, *FileName*, *MaxCount=0*, *TimeOut=30*)

下载指定端口捕获到的数据包

参数

- **Port** (*Port*) -- 测试仪表端口对象 object
- **FileDir** (*str*) -- 报文保存的路径, ("D:/test")
- **FileName** (*str*) -- 报文保存的文件的名称
- **MaxCount** (*int*) -- 下载报文最大数量，默认值 0, 表示下载端口上捕获到的所有报文
- **TimeOut** (*int*) -- 下载报文的超时时间单位秒，超时时间内为下载完成则下载失败, 默认值 30

返回 下载数据包文件的绝对路径 (例如："D:test10.0.5.10_1_1dowload.pcap")

返回类型 (str)

实际案例

robotframework:

```
| ${Port} | Get Ports |
| &{File} | Download Packages | Port=${Port} | FileDir=D:/test |␣
↪FileName=download |
```

static edit_benchmark_address_learning_capacity(*Config*, *MinAddressCount=1*, *MaxAddressCount=65536*, *InitAddressCount=20480*, *Resolution=2*, *AgingTime=15*, *LearningRate=10000*)

编辑 RFC2889 测试套件地址容量测试项参数

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object

- **MinAddressCount** (*int*) -- 学习地址最小值, 默认值: 1, 范围: 1-16777216
- **MaxAddressCount** (*int*) -- 习地址最小值默认值: 65536, 范围: 1-16777216
- **InitAddressCount** (*int*) -- 20480, 范围: 1-16777216
- **Resolution** (*int*) -- 精度 (%), 默认值: 2, 范围: 1-100
- **AgingTime** (*int*) -- 老化时间 (秒), 默认值: 50, 范围: 1-3600
- **LearningRate** (*int*) -- 地址学习速率 (帧/秒), 默认值: 10000, 范围: 1-148809523

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_address_learning_rate(*Config*, *MinRateCount=1488*, *MaxRateCount=1488*, *InitRateCount=1488*, *Resolution=2*, *AgingTime=15*, *AddressCount=1000*)

编辑 RFC2889 测试套件地址容量测试项参数

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **MinRateCount** (*int*) -- 学习地址最小值, 默认值: 1488, 范围: 1-148809523
- **MaxRateCount** (*int*) -- 习地址最小值默认值: 1488, 范围: 1-148809523
- **InitRateCount** (*int*) -- 1488, 范围: 1-148809523
- **Resolution** (*int*) -- 精度 (%), 默认值: 2, 范围: 1-100
- **AgingTime** (*int*) -- 老化时间 (秒), 默认值: 50, 范围: 1-3600
- **AddressCount** (*int*) -- 地址数量, 默认值: 1000, 范围: 1-4294967295

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_burst_count_loop(*Config*, *Mode='step'*, *Start=1*, *End=1*, *Step=1*, *Custom=[1, 2]*)

编辑 RFC2889 测试套件突发帧数，设置测试项: 广播帧转发测试、拥塞控制测试、错误帧过滤测试、转发测试

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Mode** (*str*) -- 负载单位, 默认值: step, 支持类型:

  step

  custom

- **Start** (*int*) -- 开始帧数, 默认值: 1, 范围: 1-65535
- **End** (*int*) -- 结束帧数, 默认值: 1, 范围: 1-65535
- **Step** (*int*) -- 1, 范围: 1-65535
- **Custom** (*list[int]*) -- 自定义帧数, 默认值: [1, 2]

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_duration(*Config*, *Trial=1*, *Mode='second'*, *Count=100*)

编辑测试套件测试时长设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Trial** (*int*) -- 测试验次数, 默认值: 1
- **Mode** (*str*) -- 模式, 默认值: second, 支持 second 和 burst
- **Count** (*int*) -- 突发包个数 (帧) 或时长 (秒), 默认值: 100, 范围: 1-80000000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_errored_frame_filtering(*Config*, *CrcTested=True*, *CrcFrameLength=64*, *UndersizedTested=True*, *UndersizedFrameLength=60*, *OversizedTested=True*, *OversizedFrameLength=1519*, *MaxLegalFrameLength=1518*, *BurstSize=1*)

编辑 RFC2889 测试套件错误帧过滤测试项参数

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **CrcTested** (*bool*) -- 使能错误类型 CRC, 默认值: True
- **CrcFrameLength** (*int*) -- CRC 帧长度, 默认值: 64, 范围: 64-10000
- **UndersizedTested** (*bool*) -- 使能超短帧长度, 默认值: True
- **UndersizedFrameLength** (*int*) -- 60, 范围: 58-63
- **OversizedTested** (*int*) -- 使能超长帧长度, 默认值: True
- **OversizedFrameLength** (*int*) -- 超长帧长度, 默认值: 1519, 范围: 1519-16383
- **MaxLegalFrameLength** (*int*) -- 最大合法帧长, 默认值: 1518, 范围: 1-4294967295

- **BurstSize** (*int*) -- 地址数量, 默认值: 1000, 范围: 1-4294967295

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_frame(*Config*, *Type='custom'*, *Length=128*, *Min=128*, *Max=256*, *Start=128*, *End=256*, *Step=128*, *Custom=None*, *ImixTemplates=None*)

编辑测试套件帧长度设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Type** (*str*) -- 帧长度类型, 默认值: custom, 支持类型:

  fixed

  random

  step

  custom

  fixed

  imix
- **Length** (*int*) -- 固定帧长值, 默认值: 128, 范围: 58-16383
- **Min** (*int*) -- 最小帧长值, 默认值: 128, 范围: 58-16383
- **Max** (*int*) -- 最大帧长值, 默认值: 128, 范围: 58-16383
- **Start** (*int*) -- 开始帧长值, 默认值: 128, 范围: 58-16383
- **End** (*int*) -- 结束帧长值, 默认值: 128, 范围: 58-16383
- **Step** (*int*) -- 跳变帧步长值, 默认值: 128, 范围: 58-16383
- **Custom** (*int*) -- 自定义帧长列表, 默认值: [64, 128, 256, 512, 1024, 1280, 1518], 范围: 58-16383
- **ImixTemplates** (*list*) -- IMix 模板列表, 默认值: None

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| @{Items} | Create List | throughput | frameloss |
| @{FrameSize} | Create List | 256 | 1024 | 16383 |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items}
↪ |
| Edit Benchmark Path | Configs=${Config} | Path=C:/test |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} |␣
↪Mode=meshed | Mapping=roundrobin |
```

(下页继续)

(续上页)

```
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=1000 |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@
↪{FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

static edit_benchmark_latency(*Configs*, *, *Type='FIFO'*, *DelayBefore=2*, *DelayAfter=10*)

编辑测试套件时间参数

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Type** (*str*) -- 时延类型, 支持以下类型:

  LIFO: (Store and forward)

  FIFO: (Bit forwarding);

  LILO

  FILO
- **DelayBefore** (*int*) -- 启动流前延迟时间，单位：秒，，默认值：2，范围：1-3600
- **DelayAfter** (*int*) -- 停止流后延迟时间，单位：秒，，默认值：10，范围：1-3600

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_learning(*Configs*, *Frequency*, *, *EnableLearning=True*, *LearningRate=1000*, *LearningRepeat=5*, *DelayBefore=2*, *EnableArp=False*, *ArpRate=1000*, *ArpRepeat=5*)

编辑测试套件地址学习设置

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Frequency** (*str*) -- 学习频率, 支持以下类型:

  once

  trial

  frame

  iter
- **EnableLearning** (*bool*) -- 使能地址学习, 默认值: True
- **LearningRate** (*int*) -- 地址学习速率，单位：帧/秒, 默认值：1000, 范围：1-14880952
- **LearningRepeat** (*int*) -- 学习重复次数, 默认值: 5, 范围: 1-65536
- **DelayBefore** (*int*) -- 学习延迟时间，单位：秒, 默认值: 2, 范围: 1-65536

- **EnableArp** (*bool*) -- 使能三层 ARP 学习, 默认值: True
- **ArpRate** (*int*) -- ARP 学习速率，单位：秒, 默认值: 1000, 范围: 1-14880952
- **ArpRepeat** (*int*) -- ARP 学习重复次数, 默认值: 5, 范围: 1-65536

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_multicast_base_parameters(*Configs*, *Version='igmpv2'*, *Ipv4GroupAddressStart='225.0.0.1'*, *Ipv4GroupAddressStep='0.1.0.0'*, *Ipv4PrefixLength=32*, *Ipv6GroupAddressStart='ff1e::1'*, *Ipv6GroupAddressStep='0:0:0:1::'*, *Ipv6PrefixLength=128*, *GroupIncrement=1*, *JoinGroupDelay=15*, *LeaveGroupDelay=15*, *JoinLeaveSendRate=1000*, *GroupDistributeMode='even'*)

编辑 RFC3918 测试套件-组播参数

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Version** (*str*) -- 负载单位, 默认值: percent, 支持类型:

  igmpv1

  igmpv2

  igmpv3

  mldv1

  mldv2
- **Ipv4GroupAddressStart** (*str*) -- 起始 IP 地址, 默认值: 225.0.0.1
- **Ipv4GroupAddressStep** (*str*) -- 起始 IP 步长, 默认值: 0.1.0.0
- **Ipv4PrefixLength** (*int*) -- IP 前缀长度, 默认值: 32, 范围: 1-32
- **Ipv6GroupAddressStart** (*str*) -- 起始 IPv6 地址, 默认值: ffle:
- **Ipv6GroupAddressStep** (*str*) -- 起始 IPv6 步长, 默认值: 0:0:0:1:
- **Ipv6PrefixLength** (*int*) -- IPv6 前缀长度, 默认值: 128, 范围: 1-128
- **GroupIncrement** (*int*) -- 组跳变步长, 默认值: 1, 范围: 1-4294967295
- **JoinGroupDelay** (*int*) -- 加入组延迟 (秒), 默认值: 15, 范围: 0-4294967295
- **LeaveGroupDelay** (*int*) -- 离开组延迟 (秒), 默认值: 15, 范围: 0-4294967295
- **JoinLeaveSendRate** (*int*) -- 组播发消息速率 (包/秒), 默认值: 1000, 范围: 0-1000000000

- **GroupDistributeMode** (*str*) -- 组播组分布模式, 默认值: even, 范围:

  even: 平均

  weigh: 权重

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_multicast_group_count_loop(*Config*, *LoopMode='step'*, *FixedGroup=10*, *MinGroup=10*, *MaxGroup=50*, *StartGroup=10*, *EndGroup=50*, *StepGroup=10*, *CustomGroup=(10, 20, 100)*)

RFC3918 测试套件-组播组

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **LoopMode** (*str*) -- 模式, 默认值: step, 支持:

  fixed random step custom
- **FixedGroup** (*str*) -- 固定, 默认值: 10
- **MinGroup** (*str*) -- 最小, 默认值: 10
- **MaxGroup** (*int*) -- 最大, 默认值: 50
- **StartGroup** (*int*) -- 开始, 默认值: 10
- **EndGroup** (*int*) -- 结束, 默认值: 50
- **StepGroup** (*int*) -- 步长, 默认值: 10
- **CustomGroup** (*list[int]*) -- 自定义比例, 默认值: (10, 20, 100)

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_multicast_join_leave_delay(*Config*, *DelayBetweenJoinAndStartStream=10*, *DelayBetweenJoinAndLeave=10*)

RFC3918 测试套件-配置加入离开组时延

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object

- **DelayBetweenJoinAndStartStream** (*int*) -- 从启动流量开始到发送加入组报文之间的时间间隔。单位是秒, 默认值: 10, 范围: 0-3600
- **DelayBetweenJoinAndLeave** (*int*) -- 发送加入组报文和发送离开组报文之间的时间间隔。单位是秒, 默认值: 10, 范围: 0-3600

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_multicast_mixed_throughput_unicast_streams(*Config*, *Streams*)

RFC3918 测试套件组播混合吞吐量-配置单播流量

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Streams** (list[(StreamTemplate)]) -- 仪表测试流模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_multicast_other(*Configs*, *StopTestWhenFailed=True*, *VerifyFreq='topo_changed'*, *DurationMode='second'*, *TimeDurationCount=1*, *BurstDurationCount=100*, *TxFrameRate=1000*)

编辑 RFC3918 测试套件-组播参数-其他

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **StopTestWhenFailed** (*bool*) -- 如果流验证失败停止测试, 默认值: True
- **VerifyFreq** (*str*) -- 验证频率, 默认值: topo_changed, 支持: none topo_changed frame iteration
- **DurationMode** (*str*) -- 时长模式, 默认值: second, 支持: second burst
- **TimeDurationCount** (*int*) -- 发送秒速, 默认值: 1000
- **BurstDurationCount** (*int*) -- 帧发送数量, 默认值: 100
- **TxFrameRate** (*int*) -- 帧发送速率 (帧/秒), 默认值: 1000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_multicast_stream_tos(*Configs*, *Tos=0*, *FlowLabel=0*, *TTL=7*, *Priority=0*)

编辑 RFC3918 测试套件-组播参数-流配置

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Tos** (*int*) -- IPv4 TOS 值, 默认值: 0
- **FlowLabel** (*int*) -- IPv6 Flow Label 值, 默认值: 0
- **TTL** (*int*) -- IPv4 TTL 值, 默认值: 10
- **Priority** (*int*) -- VLAN 优先级, 默认值: 0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_multicast_traffic_ratio_loop(*Config*, *LoopMode='step'*, *FixedRatio=10*, *MinRatio=10*, *MaxRatio=50*, *StartRatio=10*, *EndRatio=50*, *StepRatio=10*, *CustomRatio=(10, 20, 100)*)

RFC3918 测试套件配置组播混合吞吐量-组播百分比设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **LoopMode** (*str*) -- 模式, 默认值: step, 支持:
  fixed random step custom
- **FixedRatio** (*str*) -- 固定比例, 默认值: 10
- **MinRatio** (*str*) -- 最小比例, 默认值: 10
- **MaxRatio** (*int*) -- 最大比例, 默认值: 50
- **StartRatio** (*int*) -- 开始比例, 默认值: 10
- **EndRatio** (*int*) -- 结束比例, 默认值: 50
- **StepRatio** (*int*) -- 步长, 默认值: 10
- **CustomRatio** (*list[int]*) -- 自定义比例, 默认值: (10, 20, 100)

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_path(*Configs*, *Path*)

编辑测试套件地址学习设置

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **Path** (*str*) -- 测试结果 DB 文件保存的绝对路径

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_result_file_name(*Config*, *EnableCustomResult=False*, *ResultFileName=None*, *AddTimeStamp=True*)

配置自定义测试结果名称

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object 列表
- **EnableCustomResult** (*bool*) -- 使用自定义测试结果名称, 默认值: False
- **ResultFileName** (*str*) -- 结果名称
- **AddTimeStamp** (*bool*) -- 添加时间戳, 默认值: False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_search(*Config*, *Mode='binary'*, *Lower=1*, *Upper=100*, *Init=10*, *Step=10*, *Resolution=1*, *Ratio=50*, *Acceptance=0*, *Ignore=False*, *EnableLatency=False*, *Maxlatency=30*)

编辑测试套件测试负载设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **Mode** (*str*) -- 负载类型, 默认值: binary, 支持类型:

  binary

  step

  combo
- **Lower** (*int*) -- 速率下限 (%), 默认值: 1, 范围: 0.001-100
- **Upper** (*int*) -- 速率上限 (%), 默认值: 100, 范围: 0.001-100
- **Init** (*int*) 初始速率 (%) -- 10, 范围: 0.001-100

- **Step** (*int*) -- 步长 (%), 默认值: 10, 范围: 0.001-100
- **Resolution** (*int*) -- 精度 (%), 默认值: 1, 范围: 0.001-100
- **Ratio** (*int*) -- 二分法查找百分比 (%), 默认值: 50, 范围: 0.001-99.9999
- **Acceptance** (*int*) -- 可接受的丢包率, 默认值: 0, 范围: 0-100
- **Ignore** (*bool*) -- 忽略上下限, 默认值: False
- **EnableLatency** (*bool*) -- 使能时延吞吐量, 默认值: False
- **Maxlatency** (*int*) -- 最大允许时延, 默认值: 30

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_traffic_load_loop(*Config*, *LoadUnit='percent'*, *LoadMode='custom'*, *FixedLoad=10*, *LoadMin=10*, *LoadMax=50*, *LoadStart=10*, *LoadEnd=50*, *LoadStep=10*, *LoadCustom=(10, 20, 50)*)

编辑测试套件负载设置

参数

- **Config** (config) -- 仪表测试测试套件测试项对象 object
- **LoadUnit** (*str*) -- 负载单位, 默认值: percent, 支持类型:

  percent

  fps

  mbps

  kbps

  bps

  Bps

  ifg
- **LoadMode** (*str*) -- 负载类型, 默认值: custom, 支持类型:

  fixed

  random

  step

  custom
- **FixedLoad** (*int*) -- 固定负载, 默认值: 10, 范围: 0.001-100
- **LoadMin** (*int*) -- 10, 范围: 0.001-100
- **LoadMax** (*int*) -- 最大负载, 默认值: 50, 范围: 0.001-100
- **LoadStart** (*int*) -- 开始负载, 默认值: 10, 范围: 0.001-100
- **LoadEnd** (*int*) -- 结束负载, 默认值: 50, 范围: 0.001-100
- **LoadStep** (*int*) -- 负载步长, 默认值: 10, 范围: 0.001-100

- **LoadCustom** (*list[int]*) -- 自定义负载, 默认值: [10, 20, 50]

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_benchmark_transport_layer(*Configs*, *HeaderType=None*, *EnableRandomPort=True*, *SrcPortBase=7*, *SrcPortStep=1*, *SrcPortCount=0*, *DstPortBase=7*, *DstPortStep=1*, *DstPortCount=0*)

编辑测试套件使用已存在流量

参数

- **Configs** (list (config)) -- 仪表测试测试套件测试项对象 object 列表
- **HeaderType** (*str*) -- 报文头类型, 默认值: none, 支持: none tcp udp
- **EnableRandomPort** (*bool*) -- 使能随机端口, 默认值: True
- **SrcPortBase** (*int*) -- 源端口起始值, 默认值: 7, 范围: 0-65535
- **SrcPortStep** (*int*) -- 源端口步长, 默认值: 1, 范围: 0-65535
- **SrcPortCount** (*int*) -- 源端口数量, 默认值: 0, 范围: 0-65535
- **DstPortBase** (*int*) -- 目的端口起始值, 默认值: 7, 范围: 0-65535
- **DstPortStep** (*int*) -- 1, 范围: 0-65535
- **DstPortCount** (*int*) -- 目的端口数量, 默认值: 0, 范围: 0-65535

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

static edit_bfd(*Session*, ***kwargs*)

编辑 BFD 协议会话对象参数

参数 **Session** (BfdRouter) -- BFD 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BFD 协议会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 BFD 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterRole** (*str*) -- BFD 会话的角色, 类型为：string, 默认值：Active, 支持角色：

  Active

  Passive

- **TimeIntervalUnit** (*str*) -- 时间间隔的单位。类型为：string, 默认值：milliseconds, 支持单位：

  milliseconds

  microseconds

- **DesiredMinTXInterval** (*int*) -- 期望的最小发送时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：50
- **RequiredMinRXInterval** (*int*) -- 需要的最小接收时间间隔。类型为：number, 取值范围：1-10000 (milliseconds); 1-10000000 (microseconds), 默认值：50
- **DetectMultiple** (*int*) -- 用于检测超时的时间因子, 类型为：number, 取值范围：2-100, 默认值：3
- **AuthenticationType** (*str*) -- 认证方式, 类型为：string, 默认值：None, 支持的方式：

  NONE

  SIMPLE_PASSWORD

  KEYED_MD5

  METICULOUS_KEYED_MD5

  KEYED_SHA1

  METICULOUS_KEYED_SHA1

- **Password** (*str*) -- 当认证方式不为 NONE 时，在该单元格输入认证密码。密码可以是数字、字母或者数字和字母的组合，最长为 16 位。类型为：string, 默认值：Xinertel
- **KeyID** (*int*) -- 当认证方式不为 NONE 时，在该单元格输入 Key ID, 类型为：number, 取值范围：0-255, 默认值：1

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Bfd | Session=${Session} | EnableViewRoutes=True |
```

static edit_bgp(*Session*, ***kwargs*)

编辑 Bgp 协议会话对象参数

参数 **Session** (BgpRouter) -- Bgp 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- BGP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 BGP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **BgpInitiator** (*bool*) -- BGP 会话发起者, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AsNumber** (*int*) -- 自治域, 类型为：number, 取值范围：1-65535, 默认值：1
- **AsNumberStep** (*int*) -- 自治域跳变, 类型为：number, 取值范围：0-65535, 默认值：1

- **Enable4ByteAs** (*bool*) -- 使能 4 字节自治域, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **AsNumber4Byte** (*int*) -- 4 字节自治域, 类型为：number, 取值范围：0.1-65535.65535, 默认值：1.1
- **AsNumber4ByteStep** (*int*) -- 4 字节自治域跳变, 类型为：number, 取值范围：0.1-65535.65535, 默认值：0.1
- **DutAsNumber** (*int*) -- DUT 自治域, 类型为：number, 取值范围：1-65535, 默认值：1
- **DutAsNumberStep** (*int*) -- DUT 自治域跳变, 类型为：number, 取值范围：1-65535, 默认值：1
- **Enable4ByteDutAs** (*bool*) -- 使能 DUT4 字节自治域, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Dut4ByteAsNumber** (*int*) -- DUT4 字节自治域, 类型为：number, 取值范围：0.1-65535.65535, 默认值：1.1
- **Dut4ByteAsNumberStep** (*int*) -- DUT4 字节自治域跳变, 类型为：number, 取值范围：0.1-65535.65535, 默认值：0.1
- **BgpType** (*str*) -- BGP 类型, 类型为：string, 取值范围：EBGP, IBGP, 默认值：IBGP
- **UseGatewayAsDutIp** (*bool*) -- 使用网关地址作为 DUT 地址, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **BgpSessionIpAddressType** (*str*) -- 会话 IP 类型, 类型为：string, 取值范围：INTERFACE_IP, ROUTE_ID, 默认值：INTERFACE_IP
- **DutIpv4Address** (*str*) -- DUT IPv4 地址, 当 IP 版本为 IPv4，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID, 类型为：string, 取值范围：IPv4 地址, 默认值：2.1.1.1
- **DutIpv4AddressStep** (*str*) -- DUT IPv4 地址跳变, 当 IP 版本为 IPv4，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID 增量步长, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **DutIpv6Address** (*str*) -- DUT IPv4 地址, 当 IP 版本为 IPv6，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID, 类型为：string, 取值范围：IPv6 地址, 默认值：2000::1
- **DutIpv6AddressStep** (*str*) -- DUT IPv4 地址跳变, 当 IP 版本为 IPv6，并且使用网关地址作为 DUT 地址未选中时，需配置该选项指定 DUT 的 Router ID 增量步长, 类型为：string, 取值范围：IPv6 地址, 默认值：::1
- **HoldTime** (*int*) -- Hold Time 间隔 (sec), 类型为：number, 取值范围：3-65535, 默认值：90
- **KeepaliveTime** (*int*) -- Keep Alive 间隔 (sec), 类型为：number, 取值范围：1-65535, 默认值：30
- **ConnectRetryCount** (*int*) -- 重连次数, 取值范围：0-65535, 默认值：0
- **ConnectRetryInterval** (*int*) -- 重连间隔 (sec), 取值范围：10-300, 默认值：30
- **MaxRoutesPerUpdateMessage** (*int*) -- Update 报文中最大路由数量, 取值范围：10-300, 默认值：2000
- **RouteRefreshMode** (*str*) -- Route Refresh 模式, 类型为：string, 取值范围：None; Route Refresh, 默认值：None
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **RestartTime** (*int*) -- 平滑重启时间（秒）, 取值范围：3-4095, 默认值：90
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Authentication** (*str*) -- 使用的认证类型, 类型为：string, 取值范围：None 或 MD5, 默认值：None
- **Password** (*str*) -- 认证密码, 类型为：类型为：string, 取值范围：字符串，由 1-255 个数字、字母或特殊字符组成, 默认值：xinertel
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableSr** (*bool*) -- 使能 SR, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Bgp | Session=${Session} |
```

static edit_bgp_port_config(*Ports*, ***kwargs*)

修改 Bgp 端口统计对象

参数 **Ports** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **BgpUpdateMessageRate** (*int*) -- BGP Update Messages Tx Rate (messages/sec), 取值范围：1-1000, 默认值：10
- **SessionConnectDelay** (*int*) -- BGP Session Connect Delay (ms), 取值范围：1-10000, 默认值：100
- **SessionDisconnectDelay** (*int*) -- BGP Session Disconnect Delay (ms), 取值范围：1-10000, 默认值：100
- **BgpSrVersion** (*str*) -- BGP SR Version, 默认值：RFC8669, 取值范围：RFC8669
- **SrgbBase** (*int*) -- Global SRGB Base, 取值范围：0-16777215, 默认值：16000
- **SrgbRange** (*int*) -- Global SRGB Range, 取值范围：0-16777215, 默认值：1000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Bgp Port Config | Ports=${Ports} | BgpUpdateMessageRate=100 |
```

static edit_capture(*Ports*, ***kwargs*)

编辑端口数据捕获参数

参数 **Ports** -- 测试仪表端口对象列表, 类型为：list

关键字参数

- **Name** (*str*) -- 端口捕获名称, 类型为：string
- **CaptureMode** (*str*) -- 捕获模式：, 类型为：string, 默认值：ALL, 支持参数

  ALL CTRL_PLANE RealTime_All
- **CacheCapacity** (*str*) -- 缓存容量, 类型为：string, 默认值：Cache_Max, 支持参数

  Cache_Max Cache_32KB Cache_64KB Cache_128KB Cache_256KB Cache_512KB Cache_1MB Cache_2MB Cache_4MB Cache_8MB Cache_16MB Cache_32MB Cache_64MB Cache_128MB Cache_256MB Cache_512MB Cache_1GB
- **FilterMode** (*str*) -- 筛选模式, 类型为：string, 默认值：BYTE, 支持选项有：

  BYTE PDU
- **BufferFullAction** (*str*) -- 缓存区满后执行动作, 类型为：string, 默认值：STOP, 支持选项有：

  STOP WRAP
- **StartingFrameIndex** -- 下载报文的起始编号, 类型为：number, 默认值：1
- **AttemptDownloadPacketCount** -- 下载报文数量, 类型为：number, 默认值：0, 0 表示下载所有报文
- **FcsError** -- Fec 错误, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Ipv4ChecksumError** -- Ipv4 Checksum 错误, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **PayloadError** -- Payload 错误, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableRealtimeCapture** -- 捕获模式为 Control Plane (Tx and Rx) 或 RealTime All(Control Plane tx/rx and data plane rx), 类型为：bool, 取值范围：True 或 False, 默认值：False
- **SliceMode** (*str*) -- 切片模式, 类类型为：string, 默认值：DISABLE, 支持选项有：

  DISABLE ENABLE
- **SliceByteSize** -- 切片字节大小, 类型为：number, 取值范围：32-16383, 默认值：128

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Capture | Ports=${Ports} | CaptureMode=CTRL_PLANE |
```

static edit_capture_event(*Port*, *EventType='QUALIFY'*, ***kwargs*)

在指定端口上设置帧捕获条件

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：object
- **EventType** (*str*) -- 帧捕获类型：, 类型为：string, 支持参数：

  QUALIFY START START

关键字参数

- **LogicRelation** -- 指定捕获事件之间的逻辑关系 And 或 Or
- **PatternMatch** -- 指定捕获事件之间的逻辑关系 And 或 Or
- **FcsError** -- FCS 错误，支持 IGNORE、INCLUDE 或 EXCLUDE
- **PrbsError** -- FCS 错误，支持 IGNORE、INCLUDE 或 EXCLUDE
- **Ipv4ChecksumError** -- IPv4 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **TcpChecksumError** -- TCP 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **UdpChecksumError** -- UDP 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **IgmpChecksumError** -- Igmp 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **IcmpChecksumError** -- Icmp 校验和错误帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **SequenceError** -- 序列号错误帧,支持 IGNORE、INCLUDE 或 EXCLUDE
- **UndersizedFrame** -- 超短帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **OversizedFrame** -- 超长帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **JumboFrame** -- Jumbo 帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **FrameLength** -- 特定长度帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **FrameLengthValue** -- 特定帧长度，默认值：0
- **SignaturePresent** -- 带有签名的流帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **StreamIdMatch** -- 特定流号的流帧，支持 IGNORE、INCLUDE 或 EXCLUDE
- **StreamId** -- 特定流号，默认值：0
- **Ipv4Packets** -- IPv4 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **TcpPackets** -- TCP 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **UdpPackets** -- UDP 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **Ipv6Packets** -- IPv6 报文，支持 IGNORE、INCLUDE 或 EXCLUDE
- **IgmpPackets** -- IGMP 报文，支持 IGNORE、INCLUDE 或 EXCLUDE

- **PayloadError** -- PRBS 错误帧, 支持 IGNORE、INCLUDE 或 EXCLUDE

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Capture Event | Port=${Port} | EventType=QUALIFY |␣
↪LogicRelation=Or | PrbsError=INCLUDE | FrameLength=INCLUDE |␣
↪FrameLengthValue=128 |
```

static edit_capture_filter(*Port*, *Expression*)

在指定端口上设置报文过滤逻辑表达式

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：object
- **Expression** (*str*) -- 过滤逻辑表达式：, 类型为：string

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${BytePattern} | Create Capture Byte Pattern | Port=${Port} |␣
↪CustomCapturePatternOperator=OR | CustomCapturePatternNot=True |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoReply |
| ${PduPattern} | Create Capture Pdu Pattern | Port=${Port} | HeaderTypes=
↪${HeaderTypes} | FieldName=Icmpv4EchoReply_1.code | Value=4 |␣
↪MaxValue=5 | CustomCapturePatternOperator=OR |␣
↪CustomCapturePatternNot=True |
| Edit Capture Filter | Port=${Port} | Expression=${BytePattern} && $
↪{PduPattern} |
```

static edit_capture_pattern(*Pattern*, ***kwargs*)

修改 Capture Pattern 参数

参数 **Pattern** (*str*) -- Capture Pattern 的标识, 类型为：sting, 例如：CaptureBytePattern_1 或 CapturePduPattern_1

关键字参数

- **Pattern** 支持的 **Args** (*Pdu*) -- CustomCapturePatternOperator (str): 表达式位运算符：, 类型为：string, 默认值：AND, 支持参数

  AND OR XOR

  CustomCapturePatternNot (bool): 表达式取反：, 类型为：bool, 取值范围：True 或 False, 默认值：False

  UseFrameLength (bool): 使用 Frame 长度：, 类型为：bool, 取值范围：True 或 False, 默认值：False

  Data (str): 最小值：, 类型为：string, 取值范围：十六进制字符串, 默认值：0x0,

  MaxData (str): 最大值：, 类型为：string, 取值范围：十六进制字符串, 默认值：0xff,

Mask (str): 掩码：, 类型为：string, 取值范围：十六进制字符串, 默认值：0xff,

Offset (int): 偏移位：, 类型为：number, 取值范围：0-16378, 默认值：0

MinFrameLength (int): 最小长度, 当 UseFrameLength 为 True 有效：, 类型为：number, 取值范围：64-16383, 默认值：64

MaxFrameLength (int): 最大长度, 当 UseFrameLength 为 True 有效：, 类型为：number, 取值范围：64-16383, 默认值：16383

- **Pattern** 支持的 **Args** -- CustomCapturePatternOperator (str): 表达式位运算符：, 类型为：string, 默认值：AND, 支持参数

  AND OR XOR

  CustomCapturePatternNot (bool): 表达式取反：, 类型为：bool, 取值范围：True 或 False, 默认值：False

  Value (str): 最小值：, 类型为：string

  MaxValue (str): 最大值：, 类型为：string

  Mask (str): 掩码：, 类型为：string

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${BytePattern} | Create Capture Byte Pattern | Port=${Port} |␣
↪CustomCapturePatternOperator=OR | CustomCapturePatternNot=True |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoReply |
| ${PduPattern} | Create Capture Pdu Pattern | Port=${Port} | HeaderTypes=
↪${HeaderTypes} | FieldName=Icmpv4EchoReply_1.code | Value=4 |␣
↪MaxValue=5 | CustomCapturePatternOperator=OR |␣
↪CustomCapturePatternNot=True |
| Edit Capture Pattern | Pattern=${BytePattern} |␣
↪CustomCapturePatternOperator=XOR|
| Edit Capture Pattern | Pattern=${PduPattern} |␣
↪CustomCapturePatternOperator=XOR|
```

static edit_configs(*Configs*, ***kwargs*)

static edit_dhcp_client(*Session*, ***kwargs*)

编写 DHCP 协议会话对象参数

参数 **Session** (DhcpClient) -- DHCPv4 Client 协议对象

关键字参数

- **Name** (*str*) -- DHCP 协议会话名称
- **Mode** (*str*) -- DHCPv4 客户端模式，默认值：CLIENT, 支持的参数：

  CLIENT

  RELAY_AGENT

- **HostName** (*str*) -- 主机名字，默认值：XINERTEL
- **ParameterRequests** (*list*) -- 主机请求选项, 默认值：['NONEOPTION', 'SUBNET_MASK', 'DOMAIN_NAME_SERVERS', 'DOMAIN_NAME', 'STATIC_ROUTES'], 支持的参数:

SUBNET_MASK

ROUTERS

DOMAIN_NAME_SERVERS

DOMAIN_NAME

STATIC_ROUTES

IP_LEASE_TIME

SERVER_IDENTIFIER

T1

T2

- **EnableRouterOption** (*bool*) -- 启用路由选项, 默认值：False
- **VendorClassIdentifier** (*str*) -- 供应商识别, 默认值：XINERTEL
- **BroadcastFlag** (*str*) -- 默认值：BROADCAST, 支持参数：

  UNICAST

  BROADCAST
- **RelayAgentIp** (*str*) -- 代理端 IP, 取值范围：IPv4 地址, 默认值：1.1.1.1
- **ServerIp** (*str*) -- DHCPv4 服务端 IP, 取值范围：IPv4 地址, 默认值：2.1.1.3
- **EnableRelayAgentCircuitID** (*bool*) -- 使能代理电路标识, 默认值：False
- **RelayAgentCircuitID** (*str*) -- 代理电路标识, 默认值：""
- **EnableRelayAgentRemoteID** (*str*) -- 使能代理远程标识, 默认值：False
- **RelayAgentRemoteID** (*str*) -- 代理远程标识, 默认值：""
- **EnableSyncAddressToInterface** (*bool*) -- 使能同步地址到接口, 默认值：True
- **HostInterface** (Interface) -- 客户端接口对象, 类型：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Dhcp Client | Session=${Session} | RelayAgentIp=2.2.2.2 |
```

static edit_dhcp_client_port_config(*Ports*, ***kwargs*)

修改 DHCP 客户端端口配置对象

Args：

Ports (*Port*): 测试仪表端口对象, 类型为：object / list

关键字参数

- **SetupRate** (*int*) -- 建立速率, 默认值: 100, 范围: 1-65535
- **TeardownRate** -- 拆除速率, 默认值: 100, 范围: 1-65535
- **MaxOutstanding** -- 最大同时请求个数速率, 默认值: 100, 范围: 1-65535
- **LeaseTime** -- 期望租约时间（秒）, 默认值: 600, 范围: 1-4294967295

- **SessionRetryCount** -- 创建会话尝试次数, 默认值: 0, 范围: 0-65535
- **MessageRetryCount** -- 消息发送超时尝试次数, 默认值: 5, 范围: 0-65535
- **MessageTimeout** -- 消息超时时间（秒）, 默认值: 10, 范围: 1-99999
- **MaxMessageSize** -- 允许最大有效负荷（字节）, 默认值: 576, 范围: 291-1500

返回 Dhcpv4 Port Config 对象, 类型：object / list

返回类型 (Dhcpv4PortConfig)

实际案例

```
| Edit Dhcp Client Port Config | Port=${Port} | SetupRate=65535 |
```

static edit_dhcp_server(*Session*, ***kwargs*)

编写 DHCP Server 会话对象参数

参数 **Session** (DhcpServer) -- DHCPv4 Client 协议对象

关键字参数

- **Name** (*str*) -- DHCP Server 协议会话名称
- **LeaseTime** (*number*) -- 租约时间（秒）, 默认值: 600, 范围: 1-4294967295
- **RenewTime** (*number*) -- T1 租约更新时间 (%), 默认值：50, 范围：0-200
- **RebindTime** (*number*) -- T2 租约更新时间 (%), 默认值: 87.5, 范围: 0-200
- **MinLeaseTime** (*number*) -- 最小允许租约时间 (秒), 默认值：10, 范围：1-4294967295
- **DeclineReserveTime** (*number*) -- 资源释放等待时间 (秒), 默认值：10, 范围：1-600
- **OfferReserveTime** (*number*) -- 租约申请超时 (秒), 默认值：10, 范围：1-600
- **ServerHostName** (*str*) -- 服务端名字
- **DuplicateAddressDetection** (*bool*) -- 重复地址检测（DAD）, 默认值: False
- **DuplicateAddressDetectionTimeout** (*number*) -- DAD 超时时间, 默认值：0.5, 范围：0-60

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Dhcp Server | Session=${Session} | LeaseTime=1000 |
```

static edit_dhcp_server_port_config(*Ports*, ***kwargs*)

修改 DHCP 服务器端口配置

Args：

Ports (*Port*): 测试仪表端口对象, 类型为：object / list

关键字参数

- **RenewRate** (*int*) -- 强制单播续租速度, 默认值: 100, 范围: 1-65535
- **MaxOutstanding** -- 最大请求个数, 默认值: 1000, 范围: 1-65535

返回 Dhcpv4 Server Port Config 对象, 类型：object / list

返回类型 (Dhcpv4ServerPortConfig)

实际案例

```
| Edit Dhcp Server Port Config | Port=${Port} | RenewRate=1000 |
```

static edit_dhcpv6_client_port_config(*Ports*, ***kwargs*)

修改 DHCPv6 端口配置对象

Args：

Ports (*Port*): 测试仪表端口对象, 类型为：object

关键字参数

- **RequestRate** (*int*) -- Request 速率（会话/秒）, 取值范围：1-10000, 默认值：100
- **ReleaseRate** (*int*) -- Release 速率（会话/秒）, 取值范围：1-10000, 默认值：100
- **RenewRate** (*int*) -- Renew 速率（会话/秒）, 取值范围：1-10000, 默认值：100
- **MaxOutstanding** (*int*) -- 最大会话数量, 取值范围：1-2048, 默认值：1000
- **SolicitInitialTimeout** (*int*) -- Solicit 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：5
- **SolicitMaxTimeout** (*int*) -- Solicit 消息最大超时时间（秒）, 取值范围：1-99999, 默认值：120
- **SolicitRetryCount** (*int*) -- Solicit 消息重发次数, 取值范围：0-32, 默认值：10
- **SolicitIndefiniteRetry** (*bool*) -- Solicit 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **SolicitDisableRetries** (*bool*) -- Solicit 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **RequestInitialTimeout** (*int*) -- Request 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：2
- **RequestMaxTimeout** (*int*) -- Request 消息最大超时时间（秒）, 取值范围：1-99999, 默认值：30
- **RequestRetryCount** (*int*) -- Request 消息重发次数, 取值范围：0-32, 默认值：10
- **RequestIndefiniteRetry** (*bool*) -- Request 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **RequestDisableRetries** (*bool*) -- Request 消息禁止重发, 默认值：False, 取值范围：True 或 False
- **ConfirmInitialTimeout** (*int*) -- Confirm 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：2
- **ConfirmMaxTimeout** (*int*) -- Confirm 消息最大超时时间（秒）, 取值范围：1-99999, 默认值：4

- **ConfirmMaxDuration** (*int*) -- Confirm 消息最大重发次数, 取值范围：0-32, 默认值：5
- **RenewInitialTimeout** (*int*) -- Renew 消息初始超时时间（秒）, 取值范围：0-32, 默认值：10
- **RenewMaxTimeout** (*int*) -- Renew 消息最大超时时间（秒）, 取值范围：1-99999, 默认值：600
- **RenewRetryCount** (*int*) -- Renew 消息重发次数, 取值范围：0-32, 默认值：5
- **RenewIndefiniteRetry** (*bool*) -- Renew 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **RenewDisableRetries** (*bool*) -- Renew 消息禁止重发, 默认值：False, 取值范围：True 或 False
- **RebindInitialTimeout** (*int*) -- Rebind 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：10
- **RebindMaxTimeout** (*int*) -- Rebind 消息最大超时时间（秒）, 取值范围：1-99999, 默认值：600
- **RebindRetryCount** (*int*) -- Rebind 消息重发次数, 取值范围：0-32, 默认值：5
- **RebindIndefiniteRetry** (*bool*) -- Rebind 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **RebindDisableRetries** (*bool*) -- Rebind 消息禁止重发, 默认值：False, 取值范围：True 或 False
- **ReleaseInitialTimeout** (*int*) -- Release 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：1
- **ReleaseRetryCount** (*int*) -- Release 消息重发次数, 取值范围：0-32, 默认值：3
- **ReleaseIndefiniteRetry** (*bool*) -- Release 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **ReleaseDisableRetries** (*bool*) -- Release 消息禁止重发, 默认值：False, 取值范围：True 或 False
- **DeclineInitialTimeout** (*int*) -- Decline 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：1
- **DeclineRetryCount** (*int*) -- Decline 消息重发次数, 取值范围：0-32, 默认值：5
- **DeclineIndefiniteRetry** (*bool*) -- Decline 消息无限次重发, 默认值：False, 取值范围：True 或 False
- **DeclineDisableRetries** (*bool*) -- Decline 消息禁止重发, 默认值：False, 取值范围：True 或 False
- **InfoRequestInitialTimeout** (*int*) -- Information-Request 消息初始超时时间（秒）, 取值范围：1-99999, 默认值：1
- **InfoRequestMaxTimeout** (*int*) -- Information-Request 消息最大超时时间（秒）, 取值范围：1-99999, 默认值：120
- **InfoRequestRetryCount** (*int*) -- Information-Request 消息重发次数, 取值范围：0-32, 默认值：5
- **InfoRequestIndefiniteRetry** (*bool*) -- Information-Request 消息无限次重发, 默认值：False, 取值范围：True 或 False

- **InfoRequestDisableRetries** (*bool*) -- Information-Request 消息禁止重发, 默认值：False, 取值范围：True 或 False
- **TcpServerPort** (*int*) -- TCP 服务端口号, 取值范围：1-65535, 默认值：547

返回 DHCPv6 Client Custom Options 对象, 类型：object / list

返回类型 (Dhcpv6PortRateConfig)

实际案例

```
| Edit Dhcpv6 Client Port Config | Ports=${Port} | TcpServerPort=10 |
```

static edit_dot1x_port_config(*Ports*, ***kwargs*)

修改 802.1x 端口配置对象

Args：

Ports (*Port*): 测试仪表端口对象, 类型为：object

关键字参数

- **AuthenticationRate** (*int*) -- 默认值：100, 取值范围：1-16384
- **LogoutRate** (*int*) -- 默认值：100, 取值范围：1-16384
- **OutstandingSessions** (*int*) -- 默认值：100, 取值范围：1-10000

返回 802.1x 端口配置对象, 类型：object / list

返回类型 (Dot1xPortConfig)

实际案例

```
| Edit dot1x Port Config | Ports=${Port} | OutstandingSessions=10 |
```

static edit_header_arp(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 ARP 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 vlan 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **HardwareType** (*int*) --
- **ProtocolType** (*str*) --
- **HardwareSize** (*int*) --
- **ProtocolSize** (*int*) --
- **Opcode** (*int*) --
- **SendMac** (*str*) --
- **SendIpv4** (*str*) --
- **TargetMac** (*str*) --
- **TargetIpv4** (*str*) --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Arp | Stream=${Stream} | Level=0 |␣
↪SendMac=00:00:01:01:01:01 |
```

static edit_header_custom(*Stream*, *Level=0*, *Index=None*, ***kwargs*)

修改测试仪表流量模板中 Custom 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 Custom 头部在流量模板中所有 Custom 头部的序列号, 默认值: 0, 范围: 0-65535
- **Index** (*int*) -- Custom 头部在中需要修改 pattern 位于所有 PatternByte 和 Checksum 的序号

关键字参数

- **Pattern** --
- **Checksum** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Custom |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_custom | Stream=${Stream} | Pattern=1212121212 |
| edit_header_custom | Stream=${Stream} | Checksum=Auto |
| edit_header_custom | Stream=${Stream} | Pattern=343434343␣
↪|Checksum=Auto |
| edit_header_custom | Stream=${Stream} | Index=0 |Pattern=565656565 |
| edit_header_custom | Stream=${Stream} | Index=2 |Pattern=787878787 |
```

static edit_header_ethernet(*Stream*, *Level=0*, *DestMacAdd=None*, *SourceMacAdd=None*, *ProtocolType=None*)

修改测试仪表流量模板中 Ethernet 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 vlan 头部的序列号, 默认值: 0, 范围: 0-65535
- **DestMacAdd** (*str*) -- 目的 mac 地址
- **SourceMacAdd** (*str*) -- 源 mac 地址
- **ProtocolType** (*str*) -- 上层协议类型

Returns：

bool: 布尔值 Bool (范围：True / False)

实际案例

```
| Edit Header Ethernet | Stream=${Stream} | Level=0 |␣
↪DestMacAdd=00:01:01:01:01:02 |
```

static edit_header_gre(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Ethernet 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 vlan 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **ChecksumPresent** (*int*) --
- **Routing** (*int*) --
- **KeyPresent** (*int*) --
- **SequenceNumberPresent** (*int*) --
- **Reserved** (*int*) --
- **Version** (*int*) --
- **Protocol** (*str*) --
- **EnableKeepAlive** (*int*) --
- **KeepAlivePeriod** (*int*) --
- **KeepAliveRetries** (*int*) --
- **Checksum** (*dict*) -- e.g: {'checksum': 123, 'reserved': 123}
- **Key** (*int*) --
- **SequenceNumber** (*int*) --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | GRE |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Gre | Stream=${Stream} | Level=0 | KeepAlivePeriod=100 |␣
↪KeepAliveRetries=200 |
```

static edit_header_icmp_dest_unreach(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Dest Unreach 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Dest Unreach 头部在流量模板中所有 Icmp Dest Unreach 头部的序列号

关键字参数

- **Type** --

- **Code** --
- **Checksum** --
- **Unused** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Dest Unreach | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

static edit_header_icmp_echo_reply(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Echo Reply 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Icmp Echo Reply 头部在流量模板中所有 Icmp Echo Reply 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-255
- **Code** (*int*) -- 范围: 0-255
- **Checksum** (*str*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **Identifier** (*int*) -- 范围: 0-65535
- **SequenceNumber** (*int*) -- 范围: 0-65535

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoReply |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Echo Reply | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

static edit_header_icmp_echo_request(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Echo Request 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Icmp Echo Request 头部在流量模板中所有 Icmp Echo Request 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-255
- **Code** (*int*) -- 范围: 0-255
- **Checksum** (*str*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **Identifier** (*int*) -- 范围: 0-65535
- **SequenceNumber** (*int*) -- 范围: 0-65535

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4EchoRequest |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_icmp_echorequest | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

static edit_header_icmp_information_reply(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Information Reply 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Information Reply 头部在流量模板中所有 Icmp Information Reply 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Information Reply | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

static edit_header_icmp_information_request(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Information Request 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Information Request 头部在流量模板中所有 Icmp Information Request 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Information Request | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

static edit_header_icmp_mask_reply(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Mask Reply 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Mask Reply 头部在流量模板中所有 Icmp Mask Reply 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --
- **AddrMask** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | Icmpv4MaskReply |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Mask Reply | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

static edit_header_icmp_mask_request(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Mask Request 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Mask Request 头部在流量模板中所有 Icmp Mask Request 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --

- **Identifier** --
- **SequenceNumber** --
- **AddrMask** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Mask Request | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

static edit_header_icmp_parameter_problem(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Parameter Problem 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Parameter Problem 头部在流量模板中所有 Icmp Parameter Problem 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Pointer** --
- **Reserve** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --

- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Parameter Problem | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

static edit_header_icmp_redirect(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Redirect 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Redirect 头部在流量模板中所有 Icmp Redirect 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **GatewayAddress** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --

- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Redirect | Stream=${Stream} | Level=0 | Identifier=100␣
↪|
```

static edit_header_icmp_source_quench(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Source Quench 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Source Quench 头部在流量模板中所有 Icmp Source Quench 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --

- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Source Quench | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

static edit_header_icmp_time_exceeded(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Time Exceeded 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Time Exceeded 头部在流量模板中所有 Icmp Time Exceeded 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **Data** --
- **Ipv4HeaderVersion** --
- **Ipv4HeaderHeadLen** --
- **Ipv4HeaderTosPrecedence** --
- **Ipv4HeaderTosDelay** --
- **Ipv4HeaderTosThroughput** --
- **Ipv4HeaderTosReliability** --
- **Ipv4HeaderTosMonetaryCost** --
- **Ipv4HeaderTosReserved** --
- **Ipv4HeaderDiffservDscp** --
- **Ipv4HeaderDiffserveCodePointPrecedence** --
- **Ipv4HeaderDiffserveClassSelectorPrecedence** --
- **Ipv4HeaderDiffservDscpDrop** --
- **Ipv4HeaderDiffservDscpUndefine** --
- **Ipv4HeaderDiffservEcn** --
- **Ipv4HeaderTosByte** --
- **Ipv4HeaderTotalLength** --
- **Ipv4HeaderID** --
- **Ipv4HeaderFlags** --
- **Ipv4HeaderOffset** --
- **Ipv4HeaderTTL** --
- **Ipv4HeaderProtocol** --
- **Ipv4HeaderChecksum** --
- **Ipv4HeaderSource** --
- **Ipv4HeaderDestination** --
- **Ipv4HeaderHeaderOption** --
- **Ipv4HeaderPadding** --
- **Ipv4HeaderGateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Time Exceeded | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

static edit_header_icmp_time_stamp_reply(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Time Stamp Reply 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Time Stamp Reply 头部在流量模板中所有 Icmp Time Stamp Reply 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --
- **originateTimestamp** --
- **receiveTimestamp** --
- **transmitTimestamp** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Time Stamp Reply | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

static edit_header_icmp_time_stamp_request(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmp Time Stamp Request 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmp Time Stamp Request 头部在流量模板中所有 Icmp Time Stamp Request 头部的序列号

关键字参数

- **Type** --

- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --
- **OriginateTimestamp** --
- **ReceiveTimestamp** --
- **TransmitTimestamp** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmp Time Stamp Request | Stream=${Stream} | Level=0 |␣
↪Identifier=100 | SequenceNumber=200 |
```

static edit_header_icmpv6_destination_unreachable(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Destination Unreachable 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Destination Unreachable 头部在流量模板中所有 Icmpv6 Destination Unreachable 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **HeaderData** --
- **Version** --
- **TrafficClass** --
- **FlowLable** --
- **PayloadLength** --
- **NextHeader** --
- **HopLimit** --
- **Source** --
- **Destination** --
- **Gateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 |␣
↪DestinationUnreachable |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Destination Unreachable | Stream=${Stream} | Level=0␣
↪| Identifier=100 |
```

static edit_header_icmpv6_echo_reply(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Echo Reply 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Echo Reply 头部在流量模板中所有 Icmpv6 Echo Reply 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Icmpv6EchoReply |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Echo Reply | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

static edit_header_icmpv6_echo_request(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Echo Request 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Echo Request 头部在流量模板中所有 Icmpv6 Echo Request 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Identifier** --
- **SequenceNumber** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Icmpv6EchoRequest |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Echo Request | Stream=${Stream} | Level=0 |␣
↪Identifier=100 |
```

static edit_header_icmpv6_group_records(*Stream*, *Level=0*, *Index=0*, *Header='mldv2report'*, ***kwargs*)

修改测试仪表流量模板中 ICMPv6 Mldv2 Report 报文头部 Group Records 内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 ICMPv6 头部在流量模板中所有 IGMPv3 头部的序列号
- **Index** -- 要修改的 ICMPv6 Group Records 头部在流量模板中所有 ICMPv6 Group Records 的序列号
- **Header** -- 要修改的流量头部，默认修改 mldv2report 头部对象，其他对象包括：

  mldv2report

- **Args** (*Keyword*) -- recordType: 类型为 int，默认值：1

  auxDataLen: 类型为 int，默认值：0

  numberOfSources: 类型为 int，默认值：1

  multicastAddress: 类型为 list, 组播 ipv4 地址，默认值：225.0.0.1

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Report | Stream=${Stream} | Level=0 | Type=1 |
| Edit Header Icmpv6 Group Records | Stream=${Stream} | recordType=10 |
```

static edit_header_icmpv6_header_option(*Stream*, *Option*, *Level=0*, *Index=0*, *Header='routersolicit'*, ***kwargs*)

修改测试仪表流量模板中 ICMPv6 报文头部 Header Option 内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Option** -- ICMPv6 报文头部 Header Option 类型, 类型为：string, 支持的类型包括：

  optionSourceLinkLayerAddress

  optionTargetLinkLayerAddress

  optionPrefixInformation

  optionMTU

  generalTLV

generalWildcardTLV

- **Level** -- 要修改的 ICMPv6 头部在流量模板中所有 IGMPv3 头部的序列号
- **Index** -- 要修改的 ICMPv6 Header Option 头部在流量模板中所有 ICMPv6 Header Option 的序列号
- **Header** -- 要修改的流量头部，默认修改 mldv2report 头部对象，其他对象包括：

  routersolicit

  routeradvertise

  icmpv6redirect

  neighborsolicit

  neighboradvertise

- **Args** (*Keyword*) -- optionSourceLinkLayerAddress:

    type

    length

    address

  optionTargetLinkLayerAddress:

    type

    length

    address

  optionPrefixInformation:

    type

    length

    prefixLength

    onLinkFlag

    autonomousFlag

    reserved

    validLifetime

    preferredLifetime

    reserved2

    prefixAddress

  optionMTU:

    type

    length

    reserved3

    mtu

  generalTLV:

    type

    length

    value

  generalWildcardTLV:

type

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Report | Stream=${Stream} | Level=0 | Type=1 |
| Edit Header Icmpv6 Group Records | Stream=${Stream} | recordType=10 |
```

static edit_header_icmpv6_mldv1_done(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv1 Done 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv1 Done 头部在流量模板中所有 Icmpv6 Mldv1 Done 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **MaxRespDelay** --
- **Reserved** --
- **MulticastAddress** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv1Done |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv1 Done | Stream=${Stream} | Level=0 | Code=1 |
```

static edit_header_icmpv6_mldv1_query(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv1 Query 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv1 Query 头部在流量模板中所有 Icmpv6 Mldv1 Query 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --

- **MaxRespDelay** --
- **Reserved** --
- **MulticastAddress** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv1Query |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv1 Query | Stream=${Stream} | Level=0 | Code=1 |
```

static edit_header_icmpv6_mldv1_report(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv1 Report 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv1 Report 头部在流量模板中所有 Icmpv6 Mldv1 Report 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **MaxRespDelay** --
- **Reserved** --
- **MulticastAddress** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv1Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv1 Report | Stream=${Stream} | Level=0 | Code=1 |
```

static edit_header_icmpv6_mldv2_query(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv2 Query 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv2 Query 头部在流量模板中所有 Icmpv6 Mldv2 Query 头部的序列号

关键字参数

- **Type** --
- **Code** --

- **Checksum** --
- **MaxRespCode** --
- **Reserved** --
- **GroupAddress** --
- **Resv** --
- **Sflag** --
- **Qrv** --
- **Qqic** --
- **NumberOfSources** --
- **SourceAddressList** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Query |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Query | Stream=${Stream} | Level=0 | Code=1 |
```

static edit_header_icmpv6_mldv2_report(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Mldv2 Report 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Mldv2 Report 头部在流量模板中所有 Icmpv6 Mldv2 Report 头部的序列号

关键字参数

- **Type** --
- **Unused** --
- **Checksum** --
- **Reserved** --
- **NumberOfGroupRecords** --
- **GroupRecords** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Report | Stream=${Stream} | Level=0 | Type=1 |
```

static edit_header_icmpv6_neighbor_advertise(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Neighbor Advertise 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Neighbor Advertise 头部在流量模板中所有 Icmpv6 Neighbor Advertise 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Rflag** --
- **Sflag** --
- **Oflag** --
- **Reserve** --
- **TargetAddress** --
- **HeaderOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | NeighborAdvertise |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Neighbor Advertise | Stream=${Stream} | Level=0 |␣
↪Code=1 |
```

static edit_header_icmpv6_neighbor_solicitation(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Neighbor Solicitation 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Neighbor Solicitation 头部在流量模板中所有 Icmpv6 Neighbor Solicitation 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --

- **Reserve** --
- **TargetAddress** --
- **HeaderOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | NeighborSolicitation␣
↪|
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Neighbor Solicitation | Stream=${Stream} | Level=0 |␣
↪Code=1 |
```

static edit_header_icmpv6_packet_too_big(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Packet Too Big 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Packet Too Big 头部在流量模板中所有 Icmpv6 Packet Too Big 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Mtu** --
- **HeaderData** --
- **Version** --
- **TrafficClass** --
- **FlowLable** --
- **PayloadLength** --
- **NextHeader** --
- **HopLimit** --
- **Source** --
- **Destination** --
- **Gateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | PacketTooBig |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Packet Too Big | Stream=${Stream} | Level=0 | Code=1␣
↪|
```

static edit_header_icmpv6_parameter_problem(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Parameter Problem 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Parameter Problem 头部在流量模板中所有 Icmpv6 Parameter Problem 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Pointer** --
- **HeaderData** --
- **Version** --
- **TrafficClass** --
- **FlowLable** --
- **PayloadLength** --
- **NextHeader** --
- **HopLimit** --
- **Source** --
- **Destination** --
- **Gateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 |␣
↪Icmpv6ParameterProblem |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Parameter Problem | Stream=${Stream} | Level=0 |␣
↪Code=1 |
```

static edit_header_icmpv6_redirect(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Redirect 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object

- **Level** -- 要修改的 Icmpv6 Redirect 头部在流量模板中所有 Icmpv6 Redirect 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **TargetAddress** --
- **DestAddress** --
- **HeaderOption** --
- **RedirectedHdrOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Icmpv6Redirect |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Redirect | Stream=${Stream} | Level=0 | Code=1 |
```

static edit_header_icmpv6_redirected_header(*Stream*, *Level=0*, *Index=0*, *Header='icmpv6redirect'*, ***kwargs*)

修改测试仪表流量模板中 ICMPv6 Redirected 报文头部 Header 内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 ICMPv6 头部在流量模板中所有 IGMPv3 头部的序列号
- **Index** -- 要修改的 ICMPv6 Redirected Header 头部在流量模板中所有 ICMPv6 Redirected Header 的序列号
- **Header** -- 要修改的流量头部，默认修改 icmpv6redirect 头部对象，支持对象包括：

  icmpv6redirect

- **Args** (*Keyword*) -- type: 类型为 int，默认值：4, 取值范围：

  Source Link-Layer Address：1

  Target Link-Layer Address：2

  Prefix Information：3

  Redirected Header：4

  MTU：5

  length: 类型为 int，默认值：4

  reserved1: 类型为 int，默认值：0

  reserved2: 类型为 int，默认值：0

  Version

TrafficClass

FlowLable

PayloadLength

NextHeader

HopLimit

Source

Destination

Gateway

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | Mldv2Report |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Mldv2 Report | Stream=${Stream} | Level=0 | Type=1 |
| Edit Header Icmpv6 Redirected Header | Stream=${Stream} | type=1 |
```

static edit_header_icmpv6_router_advertise(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Router Advertise 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Router Advertise 头部在流量模板中所有 Icmpv6 Router Advertise 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **CurHopLimit** --
- **ManagedAddrFlag** --
- **OtherConfigFlag** --
- **Reserved** --
- **RouterLifetime** --
- **ReachableTime** --
- **RetransTime** --
- **HeaderOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | routeradvertise |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Router Advertise | Stream=${Stream} | Level=0 |␣
↪Code=1 |
```

static edit_header_icmpv6_router_solicitation(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Router Solicitation 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Router Solicitation 头部在流量模板中所有 Icmpv6 Router Solicitation 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --
- **HeaderOption** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | routersolicit |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Router Solicitation | Stream=${Stream} | Level=0 |␣
↪Code=1 |
```

static edit_header_icmpv6_time_exceed(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Icmpv6 Time Exceed 报文头部内容

参数

- **Stream** -- 测试仪表流量对象 object, 类型为：object
- **Level** -- 要修改的 Icmpv6 Time Exceed 头部在流量模板中所有 Icmpv6 Time Exceed 头部的序列号

关键字参数

- **Type** --
- **Code** --
- **Checksum** --
- **Reserve** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv6 | TimeExceed |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Icmpv6 Time Exceed | Stream=${Stream} | Level=0 | Code=1 |
```

static edit_header_igmpv1_query(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv1 Query 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv1 Query 头部在流量模板中所有 IGMPv1 Query 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 11
- **Unused** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv1Query |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv1_query | Stream=${Stream} | Level=0 |␣
↪GroupAddress=225.0.1.1 |
```

static edit_header_igmpv1_report(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv1 Report 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv1 Report 头部在流量模板中所有 IGMPv1 Report 头部的序列号, 范围 0-65535

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 12
- **Unused** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv1 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv1_report | Stream=${Stream} | Level=0 |␣
↪GroupAddress=225.0.1.1 |
```

static edit_header_igmpv2_query(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv2 Query 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv2 Query 头部在流量模板中所有 IGMPv2 Query 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 12
- **MaxResponseTime** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv2Query |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv2_query | Stream=${Stream} | Level=0 |␣
↪MaxResponseTime=15 | GroupAddress=225.0.1.1 |
```

static edit_header_igmpv2_report(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv2 Report 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv2 Report 头部在流量模板中所有 IGMPv2 Report 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 11
- **MaxResponseTime** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv2 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv2_report | Stream=${Stream} | Level=0 |␣
↪MaxResponseTime=15 | GroupAddress=225.0.1.1 |
```

static edit_header_igmpv3_group_records(*Stream*, *Level=0*, *Index=0*, *Header='igmpv3report'*, *SourceAddressList=[]*, *ExceedauxDataList=[]*, ***kwargs*)

修改测试仪表流量模板中 IGMPv3 Report 报文头部 Group Records 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv3 头部在流量模板中所有 IGMPv3 头部的序列号
- **Index** (*int*) -- 要修改的 IGMPv3 Group Records 头部在流量模板中所有 IGMPv3 Group Records 的序列号
- **Header** (*str*) -- 要修改的流量头部，默认修改 igmpv3report 头部对象，其他对象包括：

  igmpv3report
- **SourceAddressList** (*list*) -- Source Address List, 传入 ipv4 地址列表, 类型为 list
- **ExceedauxDataList** (*list*) -- Exceed Aux Data List, 传入 data 列表, 类型为 list
- **Args** (*Keyword*) -- RecordType (int): 类型为 int，默认值：1

  AuxDataLen (int): 类型为 int，默认值：0

  NumberOfSources (int): 类型为 int，默认值：1

  MulticastAddress (str): 类型为组播 ipv4 地址，默认值：225.0.0.1

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv3 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv3_report | Stream=${Stream} | Level=0 |␣
↪NumGroupRecords=15 |
| Edit Header IPv4 Option | Stream=${Stream} | recordType=10 |
```

static edit_header_igmpv3_query(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv3 Query 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv3 Query 头部在流量模板中所有 IGMPv3 Query 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 11
- **MaxResponseTime** (*int*) -- 范围: 0-255, 默认值: 0
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **GroupAddress** (*int*) -- 范围: ipv4 地址
- **Reserved** (*str*) -- 范围: 00-ff, 默认值: 00
- **SuppressFlag** (*int*) -- 范围: 0 or 1
- **Qrv** (*str*) -- 范围: 000 or 111
- **Qqic** (*int*) -- 范围: 0 or 255
- **NumberOfSources** (*int*) -- 范围: 0-65535
- **SourceAddressList** (*list*) -- 范围: ipv4 address list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv3Query |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv3_query | Stream=${Stream} | Level=0 |␣
↪MaxResponseTime=15 | GroupAddress=225.0.1.1 |
```

static edit_header_igmpv3_report(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IGMPv2 Report 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 IGMPv3 Report 头部在流量模板中所有 IGMPv3 Report 头部的序列号

关键字参数

- **Type** (*int*) -- 范围: 0-99, 默认值: 22
- **Reserved1** (*str*) -- 范围: 范围: 00-ff, 默认值: 00
- **Checksum** (*int*) -- 范围: 0000-ffff、AUTO, 其中 ffff 表示产生错误, AUTO 表示自动计算 Checksum
- **Reserved2** (*str*) -- 范围: 范围: 00-ff, 默认值: 00
- **NumGroupRecords** (*int*) -- 范围: 0-65535、AUTO, AUTO 表示自动计算
- **GroupRecords** (*int*) -- 范围: 0-65535

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | IGMPv3 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| edit_header_igmpv3_report | Stream=${Stream} | Level=0 |␣
↪NumGroupRecords=15 |
```

static edit_header_ipv4(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IPv4 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 IPv4 头部的序列号，默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **HeadLen** --
- **Tos** --
- **TotalLength** --
- **Flags** --
- **ID** --
- **Offset** --
- **TTL** --
- **Protocol** --
- **Checksum** --
- **Source** --
- **Destination** --
- **Padding** --
- **Gateway** --
- **TosPrecedence** --
- **HeaderOption** -- 插入 HeaderOption 字段，支持传入列表，支持的参数有：

  EndOfOption

  Nop

  Security

  LooseSourceRoute

  StrictSourceRoute

  RouterAlert

  RecordRoute

  TimeStamp

  StreamIdentifier

  General

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| ${HeaderOption} | Create List | EndOfOption | LooseSourceRoute |
| Edit Header IPv4 | Stream=${Stream} | Level=0 | Source=192.168.1.1 |␣
↪HeaderOption=${HeaderOption} |
```

static edit_header_ipv4_option(*Stream*, *Type*, *Level=0*, *Index=0*, *Header='ipv4'*, ***kwargs*)

修改测试仪表流量模板中 IPv4 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 IPv4 头部在流量模板中所有 IPv4 头部的序列号, 默认值: 0, 范围: 0-65535
- **Index** (*int*) -- 要修改的 IPv4 Option 头部在流量模板中所有 IPv4 Option 的序列号, 默认值: 0, 范围: 0-65535
- **Header** -- 要修改的流量头部，默认修改 ipv4 头部对象，支持头部对象包括:

  ipv4 destunreach parameterproblem redirect sourcequench timeexceeded
- **Type** (*list*) -- 插入 HeaderOption 字段，支持传入列表，支持的参数有:

  EndOfOption

  Nop

  Security

  LooseSourceRoute

  StrictSourceRoute

  RouterAlert

  RecordRoute

  TimeStamp

  StreamIdentifier

  General
- **Args** (*Keyword*) -- EndOfOption 类型支持:

  optiontype

  Nop 类型支持:

  optiontype

  Security 类型支持:

  optiontype

  length

  security

  compartments

handlingRestrictions
txControlCode
LooseSourceRoute 类型支持：
optiontype
length
pointer
addressList
StrictSourceRoute 类型支持：
optiontype
length
pointer
addressList
RouterAlert 类型支持：
optiontype
length
routerAlertValue
RecordRoute 类型支持：
optiontype
length
pointer
addressList
TimeStamp 类型支持：
optiontype
length
pointer
overflow
flag
timeStamp
timeStampSet
StreamIdentifier 类型支持：
optiontype
length
systemId
General 类型支持：
optiontype
length
value

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |
| create_stream_header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header IPv4 | Stream=${Stream} | Level=0 | Source=192.168.1.1 |␣
↪HeaderOption=RouterAlert |
| Edit Header IPv4 Option | Stream=${Stream} | Type=RouterAlert |␣
↪routerAlertValue=1 |
```

static edit_header_ipv6(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 IPv6 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 IPv6 头部在流量模板中所有 IPv6 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **TrafficClass** --
- **FlowLabel** --
- **PayloadLength** --
- **NextHeader** --
- **HopLimit** --
- **Source** --
- **Destination** --
- **Gateway** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header IPv6 | Stream=${Stream} | Level=0 | Source=2000::1 |
```

static edit_header_isis_area_address_entry(*Stream*, *Level=0*, *TlvIndex=0*, *EntryIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Hello/Lsp 报文中 Tlv 头部 Area Address Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Hello/Lsp 头部在流量模板中所有 ISIS L1/L2 Hello/Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis Area Address Entry 节点在流量模板中所有 Isis Area Address Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **tlvLength** (*hex*) -- Length, 默认值：1, 取值范围：1-255
- **AreaAddress** (*hex*) -- Area Address, 默认值：00, 长度：0-255byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1hello | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} |␣
↪Index=0 | lspEntries=1 |
| Edit Header Isis Area Address Entry | Stream=${Stream} | TlvIndex=0 |␣
↪EntryIndex=0 | remainTime=10 |
```

static edit_header_isis_csnp(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Isis L1/L2 Scnp 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Isis L1/L2 Scnp 头部在流量模板中所有 Isis L1/L2 Scnp 头部的序列号

关键字参数

- **InterRoutingProtocolDiscriminator** (*int*) -- Intradomain Routeing Protocol Discriminator, 默认值：<AUTO>83
- **lengthIndicator** (*int*) -- Length Indicator, 默认值：<AUTO>33
- **versionIdExtend** (*int*) -- Version/Protocol ID Extension, 默认值：<AUTO>1
- **idLength** (*int*) -- ID Length, 默认值：<AUTO>6
- **reserved1** (*int*) -- 默认值：3, 取值范围：0-7
- **pDUType** (*int*) -- PDU Type, 默认值：<AUTO>24, 取值范围：0-31
- **version** (*int*) -- 默认值：1, 取值范围：0-255
- **reserved2** (*int*) -- 默认值：3, 取值范围：0-255
- **maxAreaAddress** (*int*) -- Maximum Area Addresses, 默认值：3, 取值范围：0-3
- **pDULength** (*int*) -- PDU Length, 默认值：<AUTO>33
- **lspId** (*hex int*) -- Source ID, 默认值：00000000000000, 取值范围：00000000000000-FFFFFFFFFFFFFF
- **startLspId** (*hex int*) -- Start LSP-ID, 默认值：0000000000000000, 取值范围：0000000000000000-FFFFFFFFFFFFFFFF
- **endLspId** (*hex int*) -- End LSP-ID, 默认值：0000000000000000, 取值范围：0000000000000000-FFFFFFFFFFFFFFFF
- **CsnpDataTlvOptionHeader** (*list*) -- 可插入的选项，默认无选项，可选值：

  isIsLspEntries

authentionInfo

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis csnp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
```

static edit_header_isis_external_metric_entry(*Stream*, *Level=0*, *TlvIndex=0*, *EntryIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1 Lsp 报文中 Tlv 头部 Metric Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Lsp 头部在流量模板中所有 ISIS L1/L2 Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis External Metric Entry 节点在流量模板中所有 Isis External Metric Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **reserved** (*hex*) -- Distribution, 默认值：0, 取值范围：0-1
- **defaultMetricIEbit** (*int*) -- Default Metric I/E Bit, 默认值：0, 取值范围：0-1
- **defaultMetric** (*int*) -- Default Metric, 默认值：0, 取值范围：0-63
- **delayMetricsbit** (*int*) -- Delay Metric S bit, 默认值：0, 取值范围：0-1
- **delayMetricSBit** (*int*) -- Delay Metric R bit, 默认值：0, 取值范围：0-1
- **delayMetric** (*int*) -- Delay Metric, 默认值：0, 取值范围：0-63
- **expenseMetricsBit** (*int*) -- Expense Metric S Bit, 默认值：0, 取值范围：0-1
- **expenseMetricIEbit** (*int*) -- Expense Metric R Bit, 默认值：0, 取值范围：0-1
- **expenseMetric** (*int*) -- Expense Metric, 默认值：0, 取值范围：0-63
- **errorMetricsBit** (*int*) -- Error Metric S Bit, 默认值：0, 取值范围：0-1
- **errorMetricIEbit** (*int*) -- Error Metric R Bit, 默认值：0, 取值范围：0-1
- **errorMetric** (*int*) -- Error Metric, 默认值：0, 取值范围：0-63
- **ipAddress** (*str*) -- IP Address, 默认值：192.168.0.2, 取值范围：有效的 ipv4 地址
- **subMask** (*hex*) -- Subnet Mask, 默认值：00000000, 长度：4byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} |␣
↪Index=0 | lspEntries=1 |
| Edit Header Isis External Metric Entry | Stream=${Stream} | TlvIndex=0␣
↪| EntryIndex=0 | errorMetricIEbit=1 |
```

static edit_header_isis_hello(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Isis L1/L2 Hello 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Isis L1/L2 Hello 头部在流量模板中所有 Isis L1/L2 Hello 头部的序列号

关键字参数

- **InterRoutingProtocolDiscriminator** (*int*) -- Intradomain Routeing Protocol Discriminator, 默认值：<AUTO>83
- **lengthIndicator** (*int*) -- Length Indicator, 默认值：<AUTO>33
- **versionIdExtend** (*int*) -- Version/Protocol ID Extension, 默认值：<AUTO>1
- **idLength** (*int*) -- ID Length, 默认值：<AUTO>6
- **commonReserved1** (*int*) -- 默认值：3, 取值范围：0-7
- **pDUType** (*int*) -- PDU Type, 默认值：<AUTO>24, 取值范围：0-31
- **version** (*int*) -- 默认值：1, 取值范围：0-255
- **commonReserved2** (*int*) -- 默认值：3, 取值范围：0-255
- **maxAreaAddress** (*int*) -- Maximum Area Addresses, 默认值：3, 取值范围：0-3
- **fixedReserve1** (*int*) -- Reserved, 默认值：0, 取值范围：0-63
- **circuitType** (*int*) -- Circuit Type, 默认值：1, 取值范围：1-3
- **senderSystemID** (*hex*) -- Source ID, 默认值：000000000001, 长度: 6byte
- **holderTimer** (*int*|*hex*) -- Holding Timer, l1 默认值：51, 取值范围：0-65535; l2 默认值：0033, 长度：2byte
- **pDULength** (*int*) -- PDU Length, 默认值：<AUTO>27, 取值范围：0-65535
- **fixedReserve2** (*int*) -- Reserved, 默认值：0, 取值范围：0-1
- **priority** (*int*) -- Priority, 默认值：0, 取值范围：0-127
- **designatedSystemID** (*hex*) -- LAN ID, 默认值：00000000010001, 长度为 7byte

- **isIsTlv** (*list*) -- TLV Header, 默认值：”, 取值范围：
  isIsAreaAddress (l1) / areaAddress (l2)
  padding
  authentionInfo
  protocolSupport
  ipInterfaceAddress
  neighbor
  restartSignal
  Ipv6InterfaceAddress

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1helloHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis hello | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
```

static edit_header_isis_internal_metric_entry(*Stream*, *Level=0*, *TlvIndex=0*, *EntryIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Lsp 报文中 Tlv 头部 Metric Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Lsp 头部在流量模板中所有 ISIS L1/L2 Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis Internal Metric Entry 节点在流量模板中所有 Isis Internal Metric Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **reserved** (*hex*) -- Distribution, 默认值：0, 取值范围：0-1
- **defaultMetricIEbit** (*int*) -- Default Metric I/E Bit, 默认值：0, 取值范围：0-1
- **defaultMetric** (*int*) -- Default Metric, 默认值：0, 取值范围：0-63
- **delayMetricsbit** (*int*) -- Delay Metric S bit, 默认值：0, 取值范围：0-1
- **delayMetricSBit** (*int*) -- Delay Metric R bit, 默认值：0, 取值范围：0-1
- **delayMetric** (*int*) -- Delay Metric, 默认值：0, 取值范围：0-63
- **expenseMetricsBit** (*int*) -- Expense Metric S Bit, 默认值：0, 取值范围：0-1
- **expenseMetricIEbit** (*int*) -- Expense Metric R Bit, 默认值：0, 取值范围：0-1

- **expenseMetric** (*int*) -- Expense Metric, 默认值：0, 取值范围：0-63
- **errorMetricsBit** (*int*) -- Error Metric S Bit, 默认值：0, 取值范围：0-1
- **errorMetricIEbit** (*int*) -- Error Metric R Bit, 默认值：0, 取值范围：0-1
- **errorMetric** (*int*) -- Error Metric, 默认值：0, 取值范围：0-63
- **ipAddress** (*str*) -- IP Address, 默认值：192.168.0.2, 取值范围：有效的 ipv4 地址
- **subMask** (*hex*) -- Subnet Mask, 默认值：00000000, 长度：4byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} |␣
↪Index=0 | lspEntries=1 |
| Edit Header Isis Internal Metric Entry | Stream=${Stream} | TlvIndex=0␣
↪| EntryIndex=0 | errorMetricIEbit=1 |
```

static edit_header_isis_lsp(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Isis L1/L2 Lsp 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Isis L1/L2 Hello 头部在流量模板中所有 Isis L1/L2 Hello 头部的序列号

关键字参数

- **InterRoutingProtocolDiscriminator** (*int*) -- Intradomain Routeing Protocol Discriminator, 默认值：<AUTO>83, 取值范围：00-FF
- **lengthIndicator** (*int*) -- Length Indicator, 默认值：<AUTO>27, 取值范围：0-255
- **versionIdExtend** (*int*) -- Version/Protocol ID Extension, 默认值：<AUTO>1, 取值范围：0-255
- **idLength** (*int*) -- ID Length, 默认值：<AUTO>6, 取值范围：0-255
- **reserved1** (*int*) -- Reserved, 默认值：0, 取值范围：0-7
- **pDUType** (*int*) -- PDU Type, 默认值：<AUTO>18, 取值范围：0-31
- **version** (*int*) -- Version, 默认值：1, 取值范围：0-255
- **reserved2** (*int*) -- Reserved, 默认值：0, 取值范围：0-255
- **maxAreaAddress** (*int*) -- Maximum Area Addresses, 默认值：3, 取值范围：0-3
- **pDULength** (*int*) -- Reserved, 默认值：0, 取值范围：0-63
- **remainTime** (*int*) -- Circuit Type, 默认值：1, 取值范围：1-3
- **lspId** (*hex*) -- Source ID, 默认值：0000000000000000, 长度: 8byte

- **seqcenceNum** (*hex*) -- Seqcence Number, 默认值：00000000, 长度: 4byte
- **checksum** (*hex*) -- Checksum, 默认值：<AUTO>0000, 长度: 2byte
- **partitionRepair** (*int*) -- Partition Repair Bit, 默认值：0, 取值范围：0-1
- **attchment** (*int*) -- Attchment, 默认值：0, 取值范围：0-15
- **OverloadBit** (*int*) -- Overload Bit, 默认值：0, 取值范围：0-1
- **TypeOfIntermediateSystem** (*int*) -- Type of Intermediate System, 默认值：0, 取值范围：0-3
- **LspisIsTlvOptionSet** (*list*) -- TLV Header, 默认值：”, 取值范围：

  isIsAreaAddress

  isIsReachability

  extendedReachability

  isIsIpInterReachability

  isIsProtocolsSupported

  isIsIPExternalReachability

  ipInterfaceAddress

  Ipv6InterfaceAddress

  isIsIpv6Reachability

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1helloHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
```

static edit_header_isis_lsp_entry(*Stream*, *Level=0*, *TlvIndex=0*, *LspIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Csnp 报文中 Tlv 头部 Lsp Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Csnp 头部在流量模板中所有 ISIS L1/L2 Csnp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **LspIndex** (*int*) -- 要修改的 Isis Lsp Entry 节点在流量模板中所有 Isis Lsp Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **remainTime** (*int*) -- 默认值：1, 取值范围：0-65535

- **lspId** (*hex*) -- 默认值: 0000000000000001, 取值范围: 0000000000000001-FFFFFFFFFFFFFFFF
- **lspSeqcenceNum** (*hex*) -- 默认值: 00000001, 取值范围: 00000001-FFFFFFFF
- **checksum** (*hex*) -- 默认值: <AUTO>0000

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1csnp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} |␣
↪Index=0 | lspEntries=1 |
| Edit Header IsisLsp Entry | Stream=${Stream} | TlvIndex=0 | LspIndex=0␣
↪| remainTime=10 |
```

static edit_header_isis_metric_entry(*Stream*, *Level=0*, *TlvIndex=0*, *EntryIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Lsp 报文中 Tlv 头部 Metric Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为: object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Lsp 头部在流量模板中所有 ISIS L1/L2 Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis Metric Entry 节点在流量模板中所有 Isis Metric Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **reserved** (*hex*) -- Distribution, 默认值: 0, 取值范围: 0-1
- **defaultMetricIEbit** (*int*) -- Default Metric I/E Bit, 默认值: 0, 取值范围: 0-1
- **defaultMetric** (*int*) -- Default Metric, 默认值: 0, 取值范围: 0-63
- **delayMetricsbit** (*int*) -- Delay Metric S bit, 默认值: 0, 取值范围: 0-1
- **delayMetricSBit** (*int*) -- Delay Metric R bit, 默认值: 0, 取值范围: 0-1
- **delayMetric** (*int*) -- Delay Metric, 默认值: 0, 取值范围: 0-63
- **expenseMetricsBit** (*int*) -- Expense Metric S Bit, 默认值: 0, 取值范围: 0-1
- **expenseMetricIEbit** (*int*) -- Expense Metric R Bit, 默认值: 0, 取值范围: 0-1
- **expenseMetric** (*int*) -- Expense Metric, 默认值: 0, 取值范围: 0-63
- **errorMetricsBit** (*int*) -- Error Metric S Bit, 默认值: 0, 取值范围: 0-1
- **errorMetricIEbit** (*int*) -- Error Metric R Bit, 默认值: 0, 取值范围: 0-1

- **errorMetric** (*int*) -- Error Metric, 默认值：0, 取值范围：0-63
- **isNeighbor** (*hex*) -- IS Neighbor, 默认值：00000000000000, 长度：7byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} |␣
↪Index=0 | lspEntries=1 |
| Edit Header Isis Metric Entry | Stream=${Stream} | TlvIndex=0 |␣
↪EntryIndex=0 | errorMetricIEbit=1 |
```

static edit_header_isis_nlpid_entry(*Stream*, *Level=0*, *TlvIndex=0*, *NlpidIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Hello/Lsp 报文中 Tlv 头部 NLPID Entry 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 ISIS L1/L2 Hello/Lsp 头部在流量模板中所有 ISIS L1/L2 Hello/Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **NlpidIndex** (*int*) -- 要修改的 Isis NLPID Entry 节点在流量模板中所有 Isis NLPID Entry 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **tlvLength** (*hex*) -- Length, 默认值：1, 取值范围：1-255
- **entryId** (*hex*) -- Area Address, 默认值：01, 长度：0-255byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1hello | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} |␣
↪Index=0 | lspEntries=1 |
| Edit Header Isis Nlpid Entry | Stream=${Stream} | TlvIndex=0 |␣
↪NlpidIndex=0 | remainTime=10 |
```

static edit_header_isis_psnp(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 Isis L1/L2 Pcnp 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 范围: 0-65535: 要修改的 Isis L1/L2 Pcnp 头部在流量模板中所有 Isis L1/L2 Pcnp 头部的序列号

关键字参数

- **InterRoutingProtocolDiscriminator** (*int*) -- Intradomain Routeing Protocol Discriminator, 默认值：<AUTO>83
- **lengthIndicator** (*int*) -- Length Indicator, 默认值：<AUTO>33
- **versionIdExtend** (*int*) -- Version/Protocol ID Extension, 默认值：<AUTO>1
- **idLength** (*int*) -- ID Length, 默认值：<AUTO>6
- **reserved1** (*int*) -- 默认值：3, 取值范围：0-7
- **pDUType** (*int*) -- PDU Type, 默认值：<AUTO>24, 取值范围：0-31
- **version** (*int*) -- 默认值：1, 取值范围：0-255
- **reserved2** (*int*) -- 默认值：3, 取值范围：0-255
- **maxAreaAddress** (*int*) -- Maximum Area Addresses, 默认值：3, 取值范围：0-3
- **pDULength** (*int*) -- PDU Length, 默认值：<AUTO>33
- **sourceId** (*hex*) -- Source ID, 默认值：000000000000, 长度: 6byte
- **reserved** (*hex*) -- Reserved, 默认值：00, 长度: 1byte
- **CsnpDataTlvOptionHeader** (*list*) -- 可插入的选项，默认无选项，可选值：

  isIsLspEntries

  authentionInfo

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis psnp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
```

static edit_header_isis_sub_tlv(*Stream*, *SubTlv*, *Level=0*, *TlvIndex=0*, *SubTlvIndex=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Lsp 报文中 Tlv 头部 Sub Tlv 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **SubTlv** (*str*) -- Isis Sub Tlv 节点类型，支持：

  adGroupSubtlv

  ipv4InterfaceAddressSubtlv

  ipv4NeighborAddressSubtlv

maxLinkBandwidthSubtlv

ReservableLinkBandwidthSubtlv

unReservedBandwidthSubtlv

interfaceIpv6Subtlv

neigbhorIpv6Subtlv

- **Level** (*int*) -- 要修改的 ISIS L1/L2 Lsp 头部在流量模板中所有 ISIS L1/L2 Lsp 头部的序列号, 默认值: 0, 范围: 0-65535
- **TlvIndex** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535
- **EntryIndex** (*int*) -- 要修改的 Isis Sub Tlv 节点在流量模板中所有 Isis Sub Tlv 节点的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **adGroupSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：3, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  adminGroupValue (int): Length, 默认值：0, 取值范围：0-4294967245

- **ipv4InterfaceAddressSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：7, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  ipv4InterfaceAddressValue (str): IP Interface Address, 默认值: 192.168.0.2, 取值范围：有效的 ipv4 地址

- **ipv4NeighborAddressSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：8, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  ipv4NeighborAddressValue (str): IP Neighbor Address, 默认值: 192.168.0.2, 取值范围：有效的 ipv4 地址

- **maxLinkBandwidthSubtlv** 选项支持： -- tlvCode (int): Type, 默认值: 9, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  maxBandwidthValue (int): Maximum Link Bandwidth, 默认值：0, 取值范围：0-4294967245

- **ReservableLinkBandwidthSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：10, 取值范围：0-255

  tlvLength (int): Length, 默认值：4, 取值范围：0-255

  ReservableLinkBandwidthValue (int): Reservable Link Bandwidth, 默认值：0, 取值范围：0-4294967245

- **unReservedBandwidthSubtlv** 选项支持： -- tlvCode (int): Type, 默认值：11, 取值范围：0-255

  tlvLength (int): Length, 默认值：32, 取值范围：0-255

  resBandwidth0Value (int): Unreserved ResBandwidth Priority0, 默认值：0, 取值范围：0-4294967245

  resBandwidth1Value (int): Unreserved ResBandwidth Priority1, 默认值：0, 取值范围：0-4294967245

resBandwidth2Value (int): Unreserved ResBandwidth Priority2, 默认值：0, 取值范围：0-4294967245

resBandwidth3Value (int): Unreserved ResBandwidth Priority3, 默认值：0, 取值范围：0-4294967245

resBandwidth4Value (int): Unreserved ResBandwidth Priority4, 默认值：0, 取值范围：0-4294967245

resBandwidth5Value (int): Unreserved ResBandwidth Priority5, 默认值：0, 取值范围：0-4294967245

resBandwidth6Value (int): Unreserved ResBandwidth Priority6, 默认值：0, 取值范围：0-4294967245

resBandwidth7Value (int): Unreserved ResBandwidth Priority7, 默认值：0, 取值范围：0-4294967245

- **interfaceIpv6Subtlv** 选项支持： -- tlvCode (int): Type, 默认值：12, 取值范围：0-255

  tlvLength (int): Length, 默认值：16, 取值范围：0-255

  interfaceIpv6Value (str): Interface IPv6 Value, 默认值：2001::2, 取值范围：有效的 ipv6 地址

- **neigbhorIpv6Subtlv** 选项支持： -- tlvCode (int): Type, 默认值：13, 取值范围：0-255

  tlvLength (int): Length, 默认值：16, 取值范围：0-255

  neighboripv6Value (str): Neighbor IPv6 Value, 默认值：2001::2, 取值范围：有效的 ipv6 地址

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1lsp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} |␣
↪Index=0 | lspEntries=1 |
| Edit Header Isis Metric Entry | Stream=${Stream} | TlvIndex=0 |␣
↪EntryIndex=0 | errorMetricIEbit=1 |
```

static edit_header_isis_tlv_header(*Stream*, *Option*, *Level=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 ISIS L1/L2 Csnp/Hello/Lsp/Psnp 报文中 Tlv 头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Option** (*str*) -- ISIS L1/L2 Csnp 支持：

  isIsLspEntries

  authentionInfo

  ISIS L1/L2 Hello 支持：

isIsAreaAddress

padding

authentionInfo

protocolSupport

ipInterfaceAddress

neighbor

restartSignal

Ipv6InterfaceAddress

ISIS L1/L2 Lsp 支持:

isIsAreaAddress

isIsReachability

extendedReachability

isIsIpInterReachability

isIsProtocolsSupported

isIsIPExternalReachability

ipInterfaceAddress

Ipv6InterfaceAddress

isIsIpv6Reachability

- **Level** (*int*) -- 要修改的 ISIS L1/L2 Csnp/Hello 头部在流量模板中所有 ISIS L1/L2 Csnp/Hello 头部的序列号, 默认值: 0, 范围: 0-65535
- **Index** (*int*) -- 要修改的 Isis Tlv 头部在流量模板中所有 Isis Tlv 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **isIsLspEntries** 选项支持: (*ISIS L1/L2 Csnp/Psnp*) -- tlvCode (int): 默认值: <AUTO>9

  length (int): 默认值: <AUTO>0

  lspEntries (int): lsp entry 个数, 默认值: 0

- **authentionInfo** 选项支持: (*ISIS L1/L2 Hello*) -- tlvCode (int): 默认值: <AUTO>10

  length (int): 默认值: <AUTO>0

  authenticationType (int): 默认值: 1, 取值范围: 0-255

  authenticationLength (int): 默认值: 1, 取值范围: 0-255

  authentication (hex int): 默认值: 01, 取值范围: 长度 0-255

- **isIsAreaAddress** 选项支持: (*ISIS L1/L2 Hello/Lsp*) -- tlvCode (int): 默认值: <AUTO>9, 取值范围: 0-255

  tlvLength (int): 默认值: <AUTO>0, 取值范围: 0-255

  AreaAddressEntries (int): area address entry 个数, 默认值: 0

- **padding** 选项支持: (*ISIS L1/L2 Hello*) -- tlvCode (int): 默认值: <AUTO>9, 取值范围: 0-255

  tlvLength (int): 默认值: <AUTO>0, 取值范围: 0-255

  padding (hex): 默认值: 00, 取值范围: 00-FF

- **authentionInfo** 选项支持： -- tlvCode (int): 默认值：<AUTO>10

  length (int): 默认值：<AUTO>0

  authenticationType (int): Authentication Type, 默认值：1, 取值范围：0-255

  authenticationLength (int): Authentication Length, 默认值：1, 取值范围：0-255

  authentication (hex): Authentication, 默认值：00, 最大长度 255byte

- **protocolSupport** 选项支持： (*ISIS L1/L2 Hello/Lsp*) -- tlvCode (int): 默认值：<AUTO>129

  length (int): 默认值：<AUTO>2

  NlPIDEntriesField (int): NLPID Entries 个数

- **ipInterfaceAddress** 选项支持: (*ISIS L1/L2 Hello/Lsp*) -- tlvCode (int): 默认值：<AUTO>9, 取值范围：0-255

  tlvLength (int): 默认值：<AUTO>0, 取值范围：0-255

  ipv4InterfaceAddress (list): IPv4 Interface Address, 列表长度最大 1024, 元素默认值：192.168.0.2, 取值范围：有效的 ipv4 地址

- **neighbor** 选项支持： (*ISIS L1/L2 Hello*) -- tlvCode (int): 默认值：<AUTO>6, 取值范围：0-255

  tlvLength (int): 默认值：<AUTO>2, 取值范围：0-255

  MacAdd (list): IS Neighbors, 列表长度最大 10, 元素默认值：00:00:00:13:40:20, 取值范围：有效的 mac 地址

- **restartSignal** 选项支持： (*ISIS L1/L2 Hello*) -- tlvCode (int): Type, 默认值：<AUTO>211, 取值范围：0-255

  tlvLength (int): Length, 默认值：<AUTO>3, 取值范围：0-255

  reserved1 (int): Reserved, 默认值：0, 取值范围：0-31

  suppressAdjacency (int): Suppress Adjacency Advertisement, 默认值：0, 取值范围：0-1

  restartAck (int): Restart Acknowledgement, 默认值：0, 取值范围：0-1

  restartReq (int): Restart Request, 默认值：0, 取值范围：0-1

  remainTime (int): Remaining Time, 默认值：0, 取值范围：0-65535

  restartNeighborIdField (str): Restarting Neighbor ID, 默认：000000000000, 长度：6byte

- **Ipv6InterfaceAddress** 选项支持： (*ISIS L1/L2 Hello/Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>232, 取值范围：0-255

  tlvLength (int): Length, 默认值：<AUTO>16, 取值范围：0-255

  ipv6InterfaceAddress (list): IPv6 Interface Address, 列表最大长度 1024, 元素默认值：2001::2, 取值范围：有效的 ipv6 地址

- **isIsReachability** 选项支持： (*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>2, 取值范围：0-255

  tlvLength (int): Length, 默认值：<AUTO>12, 取值范围：0-255

  virtualFlag (int): Virtual Flag, 默认值：<AUTO>12, 取值范围：0-255

  metricEntries (int): Metric Entry 个数, 默认值：0, 最大：1024

- **extendedReachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>2, 取值范围：0-255

  length (int): Length, 默认值：<AUTO>12, 取值范围：0-255

  neighborID (hex): Neighbor ID, 默认值：00000000000000, 长度：7 byte

  metric (hex): metric, 默认值：000000, 长度：3 byte

  tlvLength (int): Sub-TLV Length, 默认值：1, 取值范围：0-255

  iisNeighborSubTlv (list): Sub-TLV 类型, 支持类型：

  adGroupSubtlv

  ipv4InterfaceAddressSubtlv

  ipv4NeighborAddressSubtlv

  maxLinkBandwidthSubtlv

  ReservableLinkBandwidthSubtlv

  unReservedBandwidthSubtlv

  interfaceIpv6Subtlv

  neigbhorIpv6Subtlv

- **isIsIpInterReachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>128, 取值范围：0-255

  length (int): Length, 默认值：<AUTO>12, 取值范围：0-255

  internalmetricEntries (int): Internal Metric Entry 个数, 默认值：0, 最大：1024

- **isIsIPExternalReachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>130, 取值范围：0-255

  length (int): Length, 默认值：<AUTO>12, 取值范围：0-255

  externalmetricEntries (int): External Metric Entry 个数, 默认值：0, 最大：1024

- **isIsIpv6Reachability** 选项支持：(*ISIS L1/L2 Lsp*) -- tlvCode (int): Type, 默认值：<AUTO>236, 取值范围：0-255

  tlvLength (int): Length, 默认值：<AUTO>16, 取值范围：0-255

  metric (int): metric, 默认值：0, 取值范围：0-4294967295

  ubit (int): Up/Down Bit, 默认值：0, 取值范围：0-1

  xbit (int): External Origin Bit, 默认值：0, 取值范围：0-1

  sbit (int): Sub-TLV Bit, 默认值：0, 取值范围：0-1

  reserved (int): Reserved Bit, 默认值：0, 取值范围：0-31

  prefixLength (int): Prefix Length, 默认值：0, 取值范围：0-255

  prefix (hex): Prefix, 默认值：00, 长度：0-255byte

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | l1csnpHeader |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Isis L1csnp | Stream=${Stream} | Level=0 | version=10 |␣
↪maxAreaAddress=3 |
| Edit Header Isis Tlv Header | Stream=${Stream} | Option=${Option} |␣
↪Index=0 | lspEntries=1 |
```

static edit_header_l2tpv2_data(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 L2tpv2 Data 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 L2tpv2 Data 头部在流量模板中所有 L2tpv2 Data 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **TunnelId** (*int*) --
- **SessionId** (*int*) --

返回 (范围: True / False)

返回类型 bool

实际案例

```
| Edit Header L2tpv2 Data | Stream=${Stream} | Level=0 | TunnelId=1000 |
```

static edit_header_mpls(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 MPLS 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 MPLS 头部在流量模板中所有 MPLS 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Label** (*int*) --
- **Exp** (*str*) --
- **Bottom** (*int*) --
- **TTL** (*int*) --

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | MPLS | IPv4 |
| Create Stream header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header MPLS | Stream=${Stream} | Level=0 | TTL=192.168.1.1 |␣
↪TTL=100 |
```

static edit_header_ospfv2_ack(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Link State Acknowledge 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Link State Acknowledge 头部在流量模板中所有 OSPFv2 Link State Acknowledge 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --
- **PacketOptionsReserved7** --
- **PacketOptionsReserved6** --
- **PacketOptionsDcBit** --
- **PacketOptionsEaBit** --
- **PacketOptionsNpBit** --
- **PacketOptionsMcBit** --
- **PacketOptionsEBit** --
- **PacketOptionsReserved0** --
- **LsaHeaderCount** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Ack | Stream=${Stream} | Level=0 | InterfaceMtu=9000␣
↪| LsaHeaderCount=1 |
```

static edit_header_ospfv2_dd(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Database Description 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Database Description 头部在流量模板中所有 OSPFv2 Database Description 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** (*str*) -- SimplePassword

  MD5

  UserDefined

  NoAuth
- **AuthValue1** --
- **AuthValue2** --
- **PacketOptionsReserved7** --
- **PacketOptionsReserved6** --
- **PacketOptionsDcBit** --
- **PacketOptionsEaBit** --
- **PacketOptionsNpBit** --
- **PacketOptionsMcBit** --
- **PacketOptionsEBit** --
- **PacketOptionsReserved0** --
- **InterfaceMtu** --
- **SequenceNumber** --
- **DdOptionsReserved7** --
- **DdOptionsReserved6** --
- **DdOptionsReserved5** --
- **DdOptionsReserved4** --

- **DdOptionsReserved3** --
- **DdOptionsIBit** --
- **DdOptionsMBit** --
- **DdOptionsMsBit** --
- **LsaHeaderCount** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪OSPFv2DatabaseDescription |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Dd | Stream=${Stream} | Level=0 | InterfaceMtu=9000␣
↪| LsaHeaderCount=1 |
```

static edit_header_ospfv2_hello(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Hello 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Hello 头部在流量模板中所有 OSPFv2 Hello 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --
- **PacketOptionsReserved7** --
- **PacketOptionsReserved6** --
- **PacketOptionsDcBit** --
- **PacketOptionsEaBit** --
- **PacketOptionsNpBit** --
- **PacketOptionsMcBit** --
- **PacketOptionsEBit** --
- **PacketOptionsReserved0** --
- **NetworkMask** --

- **HelloInterval** --
- **RouterPriority** --
- **RouterDeadInterval** --
- **DesignatedRouter** --
- **BackupDesignatedRouter** --
- **Neighbors** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | OSPFv2Hello |
| ${Neighbors} | Create List | 2.2.2.2 | 3.3.3.3 | 4.4.4.4 |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Hello | Stream=${Stream} | Level=0 | AuthType=2 |␣
↪Neighbors=${Neighbors} |
```

static edit_header_ospfv2_lsa(*Stream*, *HeaderType*, *Level=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 报文中 Lsa 头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **HeaderType** (*str*) -- Ospfv2LinkStateUpdateQueryHeader

  Ospfv2LinkStateRequestHeader

  Ospfv2DatabaseDescriptionHeader

  Ospfv2LinkStateAcknowledgeHeader
- **Level** (*int*) -- 要修改的 OSPFv2 HeaderType 头部在流量模板中所有 OSPFv2 HeaderType 头部的序列号, 默认值: 0, 范围: 0-65535
- **Index** (*int*) -- 要修改的 OSPFv2 HeaderType Lsa 头部在流量模板中所有 OSPFv2 HeaderType Lsa 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **LsaAge** --
- **Reserved7** --
- **Reserved6** --
- **DcBit** --
- **EaBit** --
- **NpBit** --
- **McBit** --
- **EBit** --
- **Reserved0** --
- **LsType** --
- **LinkStateId** --

- **AdvertisingRouter** --
- **LsSequenceNumber** --
- **LsChecksum** --
- **LsaLength** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪OSPFv2DatabaseDescription |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Dd | Stream=${Stream} | Level=0 | InterfaceMtu=9000␣
↪| LsaHeaderCount=2 |
| Edit Header Ospfv2 Lsa | Stream=${Stream} | Index=0 | LsaAge=10 |␣
↪LinkStateId=4.4.4.4 |
| Edit Header Ospfv2 Lsa | Stream=${Stream} | Index=1 | LsaAge=20 |␣
↪LinkStateId=5.5.5.5 |
```

static edit_header_ospfv2_request(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Link State Request 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Link State Request 头部在流量模板中所有 OSPFv2 Link State Request 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --
- **LsaHeaderCount** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | OSPFv2Unknown |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Request | Stream=${Stream} | Level=0 | AuthType=2 |␣
↪LsaHeaderCount=2 |
```

static edit_header_ospfv2_unknown(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Unknown 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Unknown 头部在流量模板中所有 OSPFv2 Unknown 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --
- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | OSPFv2Unknown |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Unknown | Stream=${Stream} | Level=0 | AuthType=2 |␣
↪AuthValue1=1 |
```

static edit_header_ospfv2_update(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Link State Update 报文头部内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 Link State Update 头部在流量模板中所有 OSPFv2 Link State Update 头部的序列号

关键字参数

- **Version** --
- **Type** --
- **PacketLength** --

- **RouterID** --
- **AreaID** --
- **Checksum** --
- **AuthType** --
- **AuthValue1** --
- **AuthValue2** --
- **NumberOfLsas** --
- **LsaHeaders** -- 支持的参数有：

  Router Network Summary SummaryAsbr AsExternal

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪ospfv2linkstateupdate |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr␣
↪| AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2␣
↪| LsaHeaders=${LsaHeaders} |
```

static edit_header_ospfv2_update_lsa(*Stream*, *Type*, *Level=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Update 报文中 Lsa 头部内容.

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Type** (*str*) -- OSPFv2 报文 lsa 类型支持：

  Router

  Network

  Summary

  SummaryAsbr

  External
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **Index** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号

关键字参数

- **LsaAge** (*str, optional*) --
- **Reserved7** (*int, optional*) --
- **Reserved6** --
- **DcBit** --
- **EaBit** --

- **NpBit** --
- **McBit** --
- **EBit** --
- **Reserved0** --
- **LsType** --
- **LinkStateId** --
- **AdvertisingRouter** --
- **LsSequenceNumber** --
- **LsChecksum** --
- **LsaLength** --
- **LSA** 的参数 (*External*) -- RouterLsaReserved1
  NumberOfLinks
  RouterLsaLinkCount
  reserved7Router
  reserved6Router
  dcBitRouter
  eaBitRouter
  npBitRouter
  mcBitRouter
  eBitRouter
  reserved0Router
- **LSA** 的参数 -- NetworkMask:
  AttachedRoute1:
  AttachedRouteCount:
- **LSA** 的参数 -- NetworkMask:
  LsaReserved1:
  LsaMetric:
  TosMetricsCount:
  NetworkMask:
  LsaReserved1:
  LsaMetric:
  TosMetricsCount:
- **LSA** 的参数 -- NetworkMask: External
  ExternalOptionsEBit: External
  ExternalOptionsReserved: External
  ExternalRouteMetric: External
  ForwardingAddress: External
  ExternalRouteTag: External
  TosMetricsCount: External

返回 (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr␣
↪| AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2␣
↪| LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=0 |␣
↪Type=Router | RouterLsaReserved1=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=1 |␣
↪Type=Network | NetworkMask=255.255.0.0 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=2 |␣
↪Type=Summary | LsaReserved1=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=3 |␣
↪Type=SummaryAsbr | LsaMetric=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=4 |␣
↪Type=AsExternal | ExternalOptionsEBit=1 |
```

static edit_header_ospfv2_update_nework_attached_route(*Stream*, *Level=0*, *LsaIndex=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Update 报文中 Nework Lsa 头部 Attached Route 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **LsaIndex** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号
- **Index** (*int*) -- 要修改的 OSPFv2 update Nework Lsa Attached Route 头部在流量模板中所有 Nework Lsa Attached Route 头部的序列号

关键字参数 **RouterID** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr␣
↪| AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2␣
↪| LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=1 |␣
↪Type=Network | AttachedRouteCount=2 |
| Edit Header Ospfv2 Update Nework Attached Route | Stream=${Stream} |␣
↪Level=0 | LsaIndex=1 | Index=0 | RouterID=2.2.2.2 |
```

static edit_header_ospfv2_update_route_link_tos_metric(*Stream*, *Level=0*, *LsaIndex=0*, *MetricIndex=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Update 报文中 Route Lsa 头部 Link 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **LsaIndex** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号
- **MetricIndex** (*int*) -- 要修改的 OSPFv2 update Route Lsa Link 头部在流量模板中所有 OSPFv2 update Route Lsa Link 头部的序列号
- **Index** (*int*) -- 要修改的 OSPFv2 update Route Lsa Link Tos Metric 头部在流量模板中所有 OSPFv2 update Route Lsa Link Tos Metric 头部的序列号

关键字参数

- **RouterLsaLinkType** --
- **RouterLsaMetricReserved** --
- **RouterTosLinkMetrics** --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr␣
↪| AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2␣
↪| LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=0 |␣
↪Type=Router | RouterLsaLinkCount=2 |
```

(下页继续)

(续上页)

```
| Edit Header Ospfv2 Update Route Lsa Link | Stream=${Stream} | Level=0 |␣
↪LsaIndex=0 | Index=0 | RouterLsaTosMetricsCount=2 |
| Edit Header Ospfv2 Update Route Link Tos Metric | Stream=${Stream} |␣
↪Level=0 | LsaIndex=0 | MetricIndex=0 | Index=0 |␣
↪RouterLsaMetricReserved=1 |
```

static edit_header_ospfv2_update_route_lsa_link(*Stream*, *Level=0*, *LsaIndex=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Update 报文中 Route Lsa 头部 Link 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **LsaIndex** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号
- **Index** (*int*) -- 要修改的 OSPFv2 update Route Lsa Link 头部在流量模板中所有 OSPFv2 update Route Lsa Link 头部的序列号

关键字参数

- **LinkId** (*int*) --
- **LinkData** (*int*) --
- **RouterLsaLinkType** (*int*) -- 1：Point-to-Point 2：Transit 3：Stub 4：Virtual
- **NumRouterLsaTosMetrics** (*int*) --
- **RouterLinkMetrics** (*int*) --
- **RouterLsaTosMetricsCount** (*int*) --

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr␣
↪| AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2␣
↪| LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=0 |␣
↪Type=Router | RouterLsaLinkCount=2 |
| Edit Header Ospfv2 Update Route Lsa Link | Stream=${Stream} | Level=0 |␣
↪LsaIndex=0 | Index=0 | LinkId=2.2.2.2 |
```

static edit_header_ospfv2_update_tos_metric(*Stream*, *Type*, *Level=0*, *LsaIndex=0*, *Index=0*, ***kwargs*)

修改测试仪表流量模板中 OSPFv2 Update 报文中 Summary、SummaryAsbr 或 AsExternal Lsa 头部 Tos Metric 内容

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Type** (*str*) -- OSPFv2 Update 报文中 Lsa 类型，支持:

  Summary SummaryAsbr AsExternal
- **Level** (*int*) -- 要修改的 OSPFv2 update 头部在流量模板中所有 OSPFv2 update 头部的序列号
- **LsaIndex** (*int*) -- 要修改的 OSPFv2 update Lsa 头部在流量模板中所有 OSPFv2 update Lsa 头部的序列号
- **Index** (*int*) -- 要修改的 Summary、SummaryAsbr 或 AsExternal Lsa 头部在流量模板中所有 Summary、SummaryAsbr 或 AsExternal Lsa 头部的序列号

关键字参数

- **Summary 或 SummaryAsbr,** 支持 **Args** -- MetricReserved LinkMetrics
- **AsExternal,** 支持 **Args** -- EBit RouteTos RouteMetrics ForwardingAddress RouteTag

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Stream} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 |␣
↪Ospfv2LinkStateAcknowledge |
| ${LsaHeaders} | Create List | Router | Network | Summary | SummaryAsbr␣
↪| AsExternal |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
| Edit Header Ospfv2 Update | Stream=${Stream} | Level=0 | AreaID=2.2.2.2␣
↪| LsaHeaders=${LsaHeaders} |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=2 |␣
↪Type=Summary | TosMetricsCount=2 |
| Edit Header Ospfv2 Update Tos Metric | Stream=${Stream} | Type=Summary␣
↪| Level=0 | LsaIndex=2 | Index=0 | MetricReserved=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=3 |␣
↪Type=SummaryAsbr | TosMetricsCount=2 |
| Edit Header Ospfv2 Update Tos Metric | Stream=${Stream} |␣
↪Type=SummaryAsbr | Level=0 | LsaIndex=3 | Index=0 | MetricReserved=1 |
| Edit Header Ospfv2 Update Lsa | Stream=${Stream} | Level=0 | Index=4 |␣
↪Type=AsExternal | TosMetricsCount=2 |
| Edit Header Ospfv2 Update Tos Metric | Stream=${Stream} |␣
↪Type=AsExternal | Level=0 | LsaIndex=4 | Index=0 | EBit=1 |
```

static edit_header_ppp(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 PPP 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 PPP 头部在流量模板中所有 PPP 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Addresses** (*str*) --

- **Controls** (*int*) --
- **Protocol** (*int*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Ppp | Stream=${Stream} | Level=0 | Addresses=192.168.0.1 |
```

static edit_header_pppoe(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 PPPoE 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 PPPoE 头部在流量模板中所有 PPPoE 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **Version** (*int*) --
- **Type** (*int*) --
- **Code** (*str*) --
- **SessionId** (*int*) --
- **PayloadLen** (*int*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Pppoe | Stream=${Stream} | Level=0 | Code=11 |
```

static edit_header_tcp(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 TCP 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 tcp 头部在流量模板中所有 tcp 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **SourcePort** (*str*) --
- **DestPort** (*str*) --
- **SeqNum** (*int*) --
- **AckNum** (*int*) --
- **DataOffset** (*int*) --
- **Reserved** (*str*) --
- **Flags** (*int*) --

- **WindowSize** (*int*) --
- **Checksum** (*str*) --
- **UrgentPointer** (*str*) --
- **Option** (*str*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Tcp | Stream=${Stream} | Level=0 | SourcePort=1024 |
```

static edit_header_udp(*Stream*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中 UDP 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 udp 头部在流量模板中所有 udp 头部的序列号, 默认值: 0, 范围: 0-65535

关键字参数

- **SourcePort** (*str*) --
- **DestPort** (*str*) --
- **Length** (*int*) --
- **Checksum** (*str*) --

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Udp | Stream=${Stream} | Level=0 | SourcePort=1024 |
```

static edit_header_vlan(*Stream*, *Level=0*, *ID=None*, *Priority=None*, *CFI=None*)

修改测试仪表流量模板中 vlan 报文头部内容

参数

- **Stream** (StreamTemplate) --
- **Level** (*int*) -- 要修改的 vlan 头部在流量模板中所有 vlan 头部的序列号, 默认值: 0, 范围: 0-65535
- **ID** (*int*) -- vlan id
- **Priority** (*int*) -- vlan 优先级
- **CFI** (*str*) -- vlan cfi 值

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Header Vlan | Stream=${Stream} | Level=0 | ID=4094 |
```

static edit_igmp(*Session*, ***kwargs*)

编辑 IGMP 协议会话对象

参数 **Session** (Igmp) -- IGMP 协会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- IGMP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ICMP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：IGMPV2, 支持版本：

  IGMPV1

  IGMPV2

  IGMPV3
- **PackReports** (*bool*) -- 合并报告报文, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InitialJoin** (*bool*) -- 单个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustJoin** (*bool*) -- 多个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustnessVariable** (*int*) -- Robust 值, 类型为：number, 取值范围：2-255, 默认值：2
- **UnsolicitedReportInterval** (*int*) -- 发送初始报文的时间间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **ForceLeave** (*bool*) -- 强制发送 Leave 报文, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **RouterPresentTimeout** (*int*) -- IGMPv1 路由器存在的超时时间 (秒), 类型为：number, 取值范围：0-65535, 默认值：400
- **NotFragment** (*bool*) -- 设置 IP 头报文分片标志位, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **TosValue** (*bool*) -- 设置 IP 头 TOS 值 (Hex), 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Igmp | Port=${Port} | Version=IGMPV1 | RouterPresentTimeout=500 |
```

static edit_igmp_querier(*Session*, ***kwargs*)

编辑 IGMP Querier 协议会话对象

参数 **Session** (IgmpQuerier) -- IGMP 协会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- IGMP Querier 协会话名称, 类型为：string

- **Enable** (*bool*) -- 使能 ICMP Querier 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：IGMPV2, 支持版本：

  IGMPV1

  IGMPV2

  IGMPV3
- **RobustnessVariable** (*int*) -- 健壮系数, 取值范围：2-255, 默认值: 2
- **Interval** (*int*) -- 查询时间间隔（秒）, 取值范围：0-4294967295, 默认值: 125
- **QueryResponseInterval** (*int*) -- 查询响应时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 10000
- **StartupQueryCount** (*int*) -- 初始查询报文个数, 取值范围：1-255, 默认值：2
- **LastMemberQueryInterval** (*int*) -- 最后成员查询时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 1000
- **LastMemberQueryCount** (*bool*) -- 最后成员查询次数, 取值范围：0-255, 默认值: 2
- **IPv4DoNotFragment** (*bool*) -- 设置 IP 头报文分片标志位, 取值范围：True 或 False, 默认值: False
- **IPv4TosValue** (*str*) -- 设置 IP 头 TOS 值, 类型为：bool, 取值范围：0x0-0xff, 默认值: 0xc0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Igmp Querier | Port=${Port} | Version=IGMPV3 | IPv4TosValue=0xff |
```

static edit_interface(*Interface*, *Layer=None*, *Level=None*, ***kwargs*)

修改测试仪表接口的参数

参数

- **Interface** (Interface) -- 测试仪仪表接口 Interface 对象
- **Layer** (*str*) -- 创建接口类型, 支持的有:

  EthIILayer

  VLANLayer

  IPv4Layer

  IPv6Layer
- **Level** (*int*) -- 要修改的 Layer 在 interface 的所有相同 Layer 的序号，默认值：None, 范围：0-1，为 None 表示修改所有 Layer

关键字参数

- **Count** --
- **EnableInterfaceCount** --
- **RouterIdMode** --

- **RouterId** --
- **RouterIdStep** --
- **RouterIdList** --
- **Ipv6RouterId** --
- **Ipv6RouterIdList** --
- **EnableVlansAssociation** --
- **EthIILayer** -- AddressMode

  Address

  Step

  AddressList

  EnableRandMac

  RandomSeed
- **VLANLayer** -- AddressMode

  VlanId

  Step

  VlanIdList

  Priority

  PriorityStep

  PriorityList

  Cfi

  Tpid
- **IPv4Layer** -- AddressMode

  Address

  Step

  AddressList

  PrefixLength

  Gateway

  GatewayStep

  GatewayList

  GatewayCount

  GatewayMac

  ResolvedMacList

  ResolvedState
- **IPv6Layer** -- AddressMode

  Address

  Step

  AddressList

  PrefixLength

  Gateway

GatewayStep

GatewayList

GatewayCount

GatewayMac

ResolvedMacList

ResolvedState

LinkLocalGenType

LinkLocal

LinkLocalStep

LinkLocalList

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 |
| ${Layers} | Create List | eth | ipv4 |
| ${Port} | Reserve Ports | ${Ports} | ${Location} |
| ${Interface} | Create Interface | ${Port} | ${Layers} |
| Edit Interface | Interfaces=${Interface} | Layer=IPv4Layer |␣
↪Gateway=192.168.1.1 |
```

static edit_interface_stack(*Interfaces*, *Layers=None*, *Tops=None*)

修改测试仪表接口的结构

参数

- **Interfaces** (list(Interface)) -- 测试仪仪表接口 Interface 对象列表
- **Layers** (*list*) -- 接口封装类型，支持的有：

  eth

  pppoe

  ppp

  vlan

  ipv4

  ipv6
- **Tops** (*list*) -- 接口最上层封装类型，支持的有：

  eth

  pppoe

  ppp

  vlan

  ipv4

  ipv6

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 |
| ${Layers} | Create List | eth |
| ${Tops} | Create List | vlan | ipv4 |
| ${Port} | Reserve Ports | ${Ports} | ${Location} |
| ${Interface} | Create Interface | ${Port} | ${Layers} |
| Edit Interface Stack | ${Interface} | ${Layers} | ${Tops} |
```

static edit_isis(*Session*, ***kwargs*)

编辑 ISIS 协议会话对象参数

参数 **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- ISIS 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ISIS 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPV4, 支持版本：

  IPV4

  IPV6

  IPV4IPV6
- **Level** (*str*) -- 区域类型, 类型为：string, 默认值：L2, 支持版本：

  L1

  L2

  L1L2
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 默认值：BROADCAST, 支持参数：

  BROADCAST

  P2P
- **SystemId** (*str*) -- 系统 ID, 类型为：string, 取值范围：MAC 地址, 默认值：00:00:00:00:00:01
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-127, 默认值：0
- **AuthMethod** (*str*) -- 认证方式, 类型为：string, 默认值：NONE, 支持参数：

  NONE

  SIMPLE

  MD5
- **Password** (*str*) -- 4 字节自治域跳变, 类型为：string, 默认值：Xinertel
- **CircuitId** (*int*) -- 电路 ID, 类型为：number, 取值范围：0-255, 默认值：1
- **Area1** (*str*) -- 区域 ID 1, 类型为：hex number, 取值范围：0x0-0xff, 默认值：0x10

- **Area2** (*str*) -- 区域 ID 2, 类型为：hex number, 取值范围：0x0-0xff, 默认值：空
- **Area3** (*str*) -- 区域 ID 3, 类型为：hex number, 取值范围：0x0-0xff, 默认值：空
- **MetricMode** (*str*) -- 度量模式, 类型为：string, 默认值：NARROWWIDE, 支持参数：

  NARROW

  WIDE

  NARROWWIDE
- **TeRouterId** (*str*) -- TE 路由器 ID, 类型为：string, 取值范围：IPv4 地址, 默认值：192.168.1.1
- **TeRouterIdIpv6** (*str*) -- IPv6 TE 路由器 ID, 类型为：string, 取值范围：IPv6 地址, 默认值：3000::1
- **HelloInterval** (*int*) -- Hello PDU 发送间隔 (秒), 类型为：number, 取值范围：1-300, 默认值：10
- **HelloMultiplier** (*int*) -- Hello 时间间隔倍数, 类型为：number, 取值范围：1-100, 默认值：3
- **PsnInterval** (*int*) -- PSNP 发送间隔 (秒), 类型为：number, 取值范围：1-20, 默认值：2
- **LspRefreshTime** (*int*) -- LSP 刷新时间 (秒), 类型为：number, 取值范围：1-65535, 默认值：900
- **RetransInterval** (*int*) -- LSP 重传间隔 (秒), 类型为：number, 取值范围：1-100, 默认值：5
- **HelloPadding** (*bool*) -- 类型为：bool, 取值范围：True 或 False, 默认值：True
- **LspSize** (*int*) -- LSP 大小, 类型为：number, 取值范围：100-1492, 默认值：1492
- **ValidateIpAddr** (*bool*) -- 使能接口校验, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableBFD** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MtParams** (*int*) -- 多拓扑参数数量, 类型为：number, 取值范围：0-2, 默认值：0
- **PerPduAuthentication** (*int*) -- Per PDU 认证数量, 类型为：number, 取值范围：0-4, 默认值：0
- **ReportLabel** (*bool*) -- 使能 ReportLabel, 类型为：bool, 默认值：True
- **LearnRoute** (*bool*) -- 使能 LearnRoute, 类型为：bool, 默认值：True
- **RecordLspNextSequenceNum** (*bool*) -- 使能 Record Lsp Next Sequence Number, 类型为：bool, 默认值：True
- **L1NarrowMetric** (*int*) -- L1 Narrow Metric, 类型为：number, 取值范围：0-63, 默认值：1

- **L1WideMetric** (*int*) -- L1 Wild Metric, 类型为：number, 取值范围：0-16777214, 默认值：1
- **L2NarrowMetric** (*int*) -- L2 Narrow Metric, 类型为：number, 取值范围：0-63, 默认值：1
- **L2WideMetric** (*int*) -- L2 Wide Metric, 类型为：number, 取值范围：0-16777214, 默认值：1

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${MtId} | Create List | IPV4 | IPV6 |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True | MtParams=1 |
```

static edit_isis_mt_params(*Session*, *Index=0*, *MtId=None*, *MtFlags=None*)

编辑 ISIS 协议会话 MT 参数

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Index** (*int*) -- ISIS 协议会话 MT 参数对象序号，类型为：number, 取值范围：0-1, 默认值：0
- **MtId** (*str*) -- 多拓扑 ID, 类型为：string, 默认值：IPV4, 支持参数：

  IPV4

  IPV6

- **MtFlags** (*list*) -- 多拓扑 Flags, 类型为：list, 默认值：NOSHOW, 支持参数：

  NOSHOW

  ABIT

  OBIT

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True | MtParams=1 |
| Edit Isis Mt Params | Session=${Session} | MtId=IPV6 | MtFlags=${
↪{MtFlags} |
```

static edit_isis_per_pdu(*Session*, *Index=0*, ***kwargs*)

编辑 ISIS 协议会话 Per Pdu Authentication 参数

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object

- **Index** (*int*) -- ISIS 协议会话 Per Pdu Authentication 参数对象序号，类型为：number, 取值范围：0-1, 默认值：0

关键字参数

- **PdusType** (*str*) -- PDU 类型, 类型为：string, 默认值：L1_HELLO, 支持参数：

  L1_HELLO

  L2_HELLO

  L1_AREA_PDUS

  L2_DOMAIN_PDUS

- **AuthMethod** (*str*) -- 认证类型, 类型为：string, 默认值：NONE, 支持参数：

  NONEReportLabel

  SIMPLE

  MD5

- **Password** (*str*) -- 认证密码, 类型为：string, 默认值：Xinertel.

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True |␣
↪PerPduAuthentication=1 |
| Edit Isis Per Pdu | Session=${Session} | PdusType=L2_HELLO |␣
↪AuthMethod=SIMPLE | Password=Test |
```

static edit_isis_port_config(*Ports*, ***kwargs*)

修改 Isis 端口统计对象

参数 **Ports** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数 **UpdateRoutesTransmitRate** (*int*) -- IS-IS Tx Hello Rate(messages/second) , 取值范围：1-10000, 默认值：1000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Isis Port Config | Ports=${Ports} | UpdateRoutesTransmitRate=100 |
```

static edit_l2tp(*Session*, ***kwargs*)

修改 L2tp 协议会话对象

参数 **Session** (L2tpProtocolConfig) -- L2tp 协议会话对, 类型为：object

关键字参数

- **Name** (*str*) -- L2tp 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 L2tp 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True

- **EmulationMode** (*str*) -- L2TP 角色, 默认值：LAC, 取值范围：

  LAC

  LNS

- **TunnelCountPerNode** (*int*) -- 每 LAC/LNS 隧道数, 取值范围：1-32768, 默认值：1
- **SessionCountPerTunnel** (*int*) -- 每隧道会话数, 取值范围：0-65535, 默认值：0
- **TunnelStartingId** (*int*) -- 隧道起始 ID, 取值范围：1-65535, 默认值：1
- **SessionStartingId** (*int*) -- 会话起始 ID, 取值范围：1-65535, 默认值：1
- **UdpSourcePort** (*int*) -- UDP 源端口, 取值范围：1-65535, 默认值：1701
- **UdpChecksumEnabled** (*bool*) -- 使能 UDP 校验和, 默认值：True
- **RetryTunnelCreationEnabled** (*bool*) -- 使能隧道重试, 默认值：False
- **TunnelCreationTimeout** (*int*) -- 隧道建立超时 (secs), 取值范围：1-65535, 默认值：5
- **MaxTunnelCreationTimes** (*int*) -- 隧道建立最大尝试次数, 取值范围：1-65535, 默认值：5
- **HostName** (*str*) -- 主机名, 取值范围：string length in [1,255], 默认值：xinertel
- **EnableAuthentication** (*bool*) -- 使能认证, 默认值：True
- **IncomingTunnelPassword** (*str*) -- Incoming 隧道密码, 取值范围：string length in [1,255], 默认值：xinertel
- **OutgoingTunnelPassword** (*str*) -- Outgoing 隧道密码, 取值范围：string length in [1,255], 默认值：xinertel
- **HelloEnabled** (*bool*) -- 使能 Hello, 默认值：False
- **HelloInterval** (*int*) -- Hello 间隔 (secs), 取值范围：1-255, 默认值：60
- **TxBitRate** (*int*) -- 发送 bps 速率 (bits/sec), 取值范围：1-65535, 默认值：56000
- **BearerCapabilities** (*str*) -- 负载能力, 默认值：ANALOG, 取值范围：

  DIGITAL

  ANALOG

  BOTH

- **BearerType** (*str*) -- 负载类型, 默认值：ANALOG, 取值范围：

  DIGITAL

  ANALOG

- **FrameCapabilities** (*str*) -- 帧能力, 默认值：SYNC, 取值范围：

  SYNC

  ASYNC

  BOTH

- **FrameType** (*str*) -- 帧类型, 默认值：SYNC, 取值范围：

  SYNC

  ASYNC

- **CallingNumberEnabled** (*bool*) -- 使能 Calling Number, 默认值：False
- **CallingNumber** (*str*) -- 隧道的 Calling Number, 默认值：xinertel
- **RxWindowSize** (*int*) -- 接收窗口大小, 取值范围：1-65535, 默认值：4
- **UseGatewayAsRemoteIp** (*bool*) -- 使用网关作为远端地址, 默认值：True
- **RemoteIpv4Address** (*str*) -- 远端 IPv4 地址, 取值范围：IPv4 地址, 默认值：2.1.1.1
- **RemoteIpv4AddressStep** (*str*) -- 远端 IPv4 地址跳变, 取值范围：IPv4 地址, 默认值：0.0.0.1
- **RemoteIpv6Address** (*str*) -- 远端 IPv6 地址, 取值范围：IPv6 地址, 默认值：2000::1
- **RemoteIpv6AddressStep** (*str*) -- 远端 IPv6 地址跳变, 取值范围：IPv6 地址, 默认值：::1
- **LcpProxyMode** (*str*) -- LCP 代理模式, 默认值：NONE, 取值范围：

  NONE

  LCP

  LCP_AUTH
- **ForceLcpRenegotiation** (*bool*) -- 强制 LCP 重协商, 默认值：False
- **Ipv4TosValue** (*hex int*) -- IPv4 TOS 值, 取值范围：1-65535, 默认值：0xc0
- **Ipv6TrafficClassValue** (*hex int*) -- IPv6 Traffic Class Value, 取值范围：1-65535, 默认值：0x0
- **HideFramingCapabilities** (*bool*) -- 默认值：False
- **HideBearerCapabilities** (*bool*) -- 默认值：False
- **HideAssignedTunnelId** (*bool*) -- 默认值：False
- **HideChallenge** (*bool*) -- 默认值：False
- **HideChallengeResponse** (*bool*) -- 默认值：False
- **HideAssignedSessionId** (*bool*) -- 默认值：False
- **HideCallSerialNumber** (*bool*) -- 默认值：False
- **HideFramingType** (*bool*) -- 默认值：False
- **HideCallingNumber** (*bool*) -- 默认值：False
- **HideTxConnectSpeed** (*bool*) -- 默认值：False
- **HideLastSentLcpConfReq** (*bool*) -- 默认值：False
- **HideLastReceivedLcpConfReq** (*bool*) -- 默认值：False
- **HideProxyAuthenType** (*bool*) -- 默认值：False
- **HideProxyAuthenName** (*bool*) -- 默认值：False
- **HideProxyAuthenChallenge** (*bool*) -- 默认值：False
- **HideProxyAuthenId** (*bool*) -- 默认值：False
- **HideProxyAuthenResponse** (*bool*) -- 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit L2tp | Session=${Session} |
```

static edit_l2tp_port_config(*Ports*, ***kwargs*)

修改 DHCPv6 端口配置对象

Args：

Ports (*Port*): 测试仪表端口对象, 类型为：object / list

关键字参数 TunnelConnectRate (*int*) -- Request 速率（会话/秒）, 取值范围：1-1000, 默认值：100

返回 L2tp 端口对象, 类型：object / list

返回类型 (L2tpPortConfig)

实际案例

```
| Edit L2tp Port Config | Ports=${Port} | TcpServerPort=10 |
```

static edit_ldp(*Session*, ***kwargs*)

编辑 LDP 协议会话对象参数

参数 Session (Ldp) -- LDP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- LDP 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 LDP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **HelloType** (*str*) -- Hello 类型, 类型为：string, 默认值：DIRECT, 取值范围:

  DIRECT

  TARGETED

  DIRECT_TARGETED
- **LabelAdvertType** (*str*) -- 标签分配方式, 类型为：string, 默认值：DIRECT, 取值范围:

  DU

  DOD
- **TransportMode** (*str*) -- Transport Address TLV 模式, 类型为：string, 默认值：TESTER_IP, 取值范围:

  TESTER_IP

  ROUTER_ID

  NONE
- **DutIpv4Address** (*int*) -- DUT IPv4 地址, 类型为：number, 型为：string, 默认值：2.1.1.1, 取值范围：IPv4 地址
- **DirectHelloInterval** (*int*) -- 直连 Hello 发送间隔 (sec), 类型为：number, 默认值：5, 取值范围：1-21845
- **TargetedHelloInterval** (*int*) -- 远端 Hello 发送间隔 (sec), 类型为：number, 默认值：15, 取值范围：1-21845

- **KeepAliveInterval** (*int*) -- 保活间隔 (sec), 类型为：number, 默认值：60, 取值范围：1-21845
- **LabelReqRetryCount** (*int*) -- 标签请求间隔 (sec), 类型为：number, 默认值：10, 取值范围：1-65535
- **LabelReqRetryInterval** (*int*) -- 标签请求重试次数, 类型为：number, 默认值：60, 取值范围：1-65535
- **Authentication** (*str*) -- 鉴权类型, 类型为：string, 默认值：DIRECT, 取值范围:

  NONE

  MD5
- **Password** (*str*) -- 密码, 类型为：string, 默认值：xinertel
- **EgressLabel** (*str*) -- 出标签方式, 类型为：string, 默认值：DIRECT, 取值范围:

  NEXT_AVAILABLE

  IMPLICIT

  EXPLICIT
- **MinLabel** (*int*) -- 最小标签值, 类型为：number, 默认值：16, 取值范围：0-1048575
- **EnableLspResult** (*bool*) -- LSP 统计使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnablePseudowireLspResult** (*bool*) -- 伪线 LSP 统计使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LspBindMode** (*str*) -- LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE
- **VcLspBindMode** (*str*) -- 虚拟电路 LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE
- **GeneralizedLspBindMode** (*str*) -- 通用伪线 LSP 绑定模式, 类型为：string, 默认值：TX_RX, 取值范围:

  TX_RX

  TX

  RX

  NONE

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Ldp | Port=${Port} |
| Edit Ldp | Session=${Session} | HelloType=DIRECT_TARGETED |
```

static edit_ldp_port_config(*Ports*, ***kwargs*)

修改 LDP 端口配置对象

Args:

Ports (*Port*): 测试仪表端口对象, 类型为: object

关键字参数

- **EstablishRate** (*int*) -- LDP 发送速率 (messages/sec), 取值范围: 1-10000, 默认值: 100
- **AdvertiseRate** (*int*) -- 会话建立速率 (sessions/sec), 取值范围: 1-10000, 默认值: 100
- **ReleaseRate** (*int*) -- 会话释放速率 (sessions/sec), 取值范围: 1-10000, 默认值: 100
- **FecPerLdpMsg** (*int*) -- 取值范围: 1-65535, 默认值: 65535

返回 LDP Port Config 对象, 类型: object / list

返回类型 (LdpPortConfig)

实际案例

```
| Edit LDP Port Config | Ports=${Port} | TcpServerPort=10 |
```

static edit_lsp_ping(*Wizard*, ***kwargs*)

配置 MPLS 流量 LSP Ping 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **EnableLspPing** (*bool*) -- False,
- **DestinationIpv4Address** (*str*) -- ' 127.0.0.1',
- **PingInterval** (*int*) -- 4,
- **PingTimeout** (*int*) -- 2,
- **TimeToLive** (*int*) -- 255,
- **LspExpValue** (*int*) -- 0,
- **ValidateFecStack** (*bool*) -- False,
- **PadMode** (*str*) -- 支持: TransmitWithoutPadTlv RequestPeerToDropPadTlv RequestPeerToCopyPadTlv
- **PadData** (*list*) -- [],

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_lsp_ping_port_config(*Ports*, ***kwargs*)

修改 Lsp Ping 端口配置对象

Args：

Ports (*Port*): 测试仪表端口对象, 类型为：object

关键字参数

- **UpdateTransmitRate** (*int*) -- 默认值：1000, 取值范围：1-10000
- **FrequencyPing** (*int*) -- 执行 Ping 测试时间间隔（秒）, 默认值：60, 取值范围：1-2147483647
- **FrequencyTrace** (*int*) -- 执行 Trace 测试时间间隔（秒）, 默认值：60, 取值范围：60-2147483647

返回 Lsp Ping 协议端口对象, 类型：object / list

返回类型 (LspPingPortConfig)

实际案例

```
| Edit Lsp Ping Port Config | Ports=${Port} | FrequencyTrace=10 |
```

static edit_mld(*Session*, ***kwargs*)

创建 MLD 协议会话对象

参数 **Session** (Mld) -- MLD 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- MLD 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 MLD 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：MLDV1, 支持版本：

  MLDV1

  MLDV2
- **PackReports** (*bool*) -- 合并报告报文, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **InitialJoin** (*bool*) -- 单个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustJoin** (*bool*) -- 多个初始报文加入组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RobustnessVariable** (*int*) -- Robust 值, 类型为：number, 取值范围：2-255, 默认值：2
- **UnsolicitedReportInterval** (*int*) -- 发送初始报文的时间间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **ForceLeave** (*bool*) -- 强制发送 Leave 报文, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **TrafficClass** (*hex int*) -- IP 头的 Traffic Class 值, 型为：string, 取值范围：0x0-0xff, 默认值：0xc0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Mld | Port=${Port} | Version=MLDV2 |
```

static edit_mld_querier(*Session*, ***kwargs*)

编辑 MLD Querier 协议会话对象

参数 **Session** (MldQuerier) -- MLD 协会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- MLD Querier 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 ICMP Querier 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Version** (*str*) -- 版本, 类型为：string, 默认值：MLDV1, 支持版本：

  MLDV1

  MLDV2
- **RobustnessVariable** (*int*) -- 健壮系数, 取值范围：2-255, 默认值: 2
- **Interval** (*int*) -- 查询时间间隔（秒）, 取值范围：0-4294967295, 默认值: 125
- **QueryResponseInterval** (*int*) -- 查询响应时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 10000
- **StartupQueryCount** (*int*) -- 初始查询报文个数, 取值范围：1-255, 默认值：2
- **LastMemberQueryInterval** (*int*) -- 最后成员查询时间间隔（毫秒）, 取值范围：0-4294967295, 默认值: 1000
- **LastMemberQueryCount** (*bool*) -- 最后成员查询次数, 取值范围：0-255, 默认值: 2
- **IPv6TrafficClassValue** (*str*) -- 设置 IPv6 头 TrafficClass 值, 取值范围：0x0-0xff, 默认值: 0x0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Mld Querier | Port=${Port} | Version=MLDV2 |␣
↪IPv6TrafficClassValue=0xff |
```

static edit_modifier(*Stream*, *Attribute*, *Level=0*, ***kwargs*)

修改测试仪表流量模板中指定报文字段的跳变域

参数

- **Stream** (StreamTemplate) --
- **Attribute** (*str*) -- 要修改的报文字段参数名称
- **Level** (*int*) -- 当 HeaderType=None 表示要修改的报文字段在流量模板中所有报文头部的序列号, 当 HeaderType!=None 表示要修改的报文字段在流量模板中所有相同类型报文头部的序列号, 默认值：0

Keyword Args: 跳变域参数参数, 支持以下参数:

Type (str): 跳变类型:

Increment

Decrement

Random

List

Start (str): 跳变起始数据

Count (int): 跳变数量

Step (int): 跳变步长

Range (int): 随机跳变范围

Seed (int): 随机跳变种子

StreamType (str): 流跳变类型

Offset (int): 跳变偏移位

Mask (str): 跳变掩码

List (list): list 跳变时, 跳变列表

HeaderType (str): 要跳变的报文头部名称, 默认是 None, 如果 HeaderType 为 None, 修改的报文头 Level 决定 (要修改的报文字段在流量模板中所有报文头部的序列号)，支持:

ethernetii

vlan

vxlan

arp

ipv4

ipv6

tcp

udp

l2tpv2data

ppp

icmpv4echorequest

destunreach

icmpv4echoreply

informationreply

informationrequest

icmpv4parameterproblem

icmpv4redirect

sourcequench

timeexceeded

timestampreply

timestamprequest

icmpmaskrequest

icmpmaskreply

destinationunreachable

icmpv6echoreply
icmpv6echorequest
packettoobig
icmpv6parameterproblem
timeexceed
routersolicit
routeradvertise
icmpv6redirect
neighborsolicit
neighboradvertise
mldv1query
mldv1report
mldv1done
mldv2query
mldv2report
igmpv1
igmpv1query
igmpv2
igmpv2query
igmpv3report
igmpv3query
custom
ospfv2linkstateupdate
ospfv2linkstaterequest
ospfv2databasedescription
ospfv2linkstateacknowledge
ospfv2unknown
ospfv2hello
mpls

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Streams} | add_stream | Port=${Port} |
| ${HeaderTypes} | Create List | EthernetII | IPv4 | TCP |
| Create Stream Header | Stream=${Stream} | HeaderTypes=${HeaderTypes} |
# 不指定HeaderType, Level=1选中IPv4头部
| Edit Modifier | Stream=${Stream} | Level=1 | Attribute=source |␣
↪Start=192.168.1.1 | Count=10 | Step=1 |
# 指定HeaderType=IPv4, Level=0选中IPv4头部
| Edit Modifier | Stream=${Stream} | Level=0 | HeaderType=IPv4 |␣
↪Attribute=destination | Start=192.168.1.1 | Count=10 | Step=1 |
```

static edit_mpls_customer_port(*Wizard*, *Port*, ***kwargs*)

配置 MPLS 向导客户侧端口

参数

- **Wizard** (WizardConfig) -- wizard config
- **Port** (list(*:obj:'Port*))) -- 测试仪表端口对象列表

关键字参数

- **PortIndex** (*int*) -- 默认值: 0,
- **EnableSubInterface** (*bool*) -- 默认值: False,
- **SubInterfaceCount** (*int*) -- 默认值: 1,
- **DutIpv4Address** (*str*) -- 默认值: '192.85.1.1',
- **DutIpv4AddressStep** (*str*) -- 默认值: '0.0.1.0',
- **Ipv4PrefixLength** (*int*) -- 默认值: 24,
- **DutIpv6Address** (*str*) -- 默认值: '::',
- **DutIpv6AddressStep** (*str*) -- 默认值: '0:0:0:1::',
- **Ipv6PrefixLength** (*int*) -- 默认值: 64,
- **VlanId** (*int*) -- 默认值: 1,
- **VlanIdStep** (*int*) -- 默认值: 1

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_fec128(*Wizard*, ***kwargs*)

配置 MPLS fec128 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **StartVcId** (*int*) -- Start vc id
- **StepVcId** (*int*) -- Step vc id

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_fec129(*Wizard*, ***kwargs*)

配置 MPLS fec129 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **Agi** (*str*) -- AGI
- **AgiIncrement** (*str*) -- AGI increment
- **Saii** (*str*) -- SAII
- **SaiiIncrement** (*str*) -- SAII increment
- **Taii** (*str*) -- TAII
- **TaiiIncrement** (*str*) -- TAII increment
- **EnableBgpAutoDiscovery** (*bool*) -- Enable BGP auto discovery
- **DutAsNumber** (*int*) -- DUT AS number
- **RdAssignment** (*str*) -- RD assignment: UseRT Manual
- **AgiAssignment** (*str*) -- AGI assignment UseRT Manual
- **Rt** (*str*) -- RT
- **RtIncrement** (*str*) -- RT increment
- **Rd** (*str*) -- RD
- **RdIncrement** (*str*) -- RD increment

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_host(*Wizard*, ***kwargs*)

配置 MPLS 向导 Host 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **HostMacStart** (*str*) -- Host MAC start
- **HostMacStep** (*str*) -- Host MAC step
- **EnableOverlapHosts** (*bool*) -- Enable overlap hosts
- **EnableHostVlan** (*bool*) -- Enable host VLAN
- **NumberOfCustomerSideVlanHeaders** (*int*) -- Number of customer side VLAN headers
- **NumberOfProviderSideVlanHeaders** (*int*) -- Number of provider side VLAN headers
- **VlanIdStart** (*int*) -- VLAN ID start
- **VlanIdStepPerVpls** (*int*) -- VLAN ID step per VPLS
- **VlanIdStepPerHost** (*int*) -- VLAN ID step per host
- **HostAssignment** (*int*) -- Host assignment: HostsOrMacsPerCe HostsOrMacsPerVpls TotalHostsOrMacs
- **HostsPerCustomerCe** (*int*) -- Host per customer CE
- **HostsPerProviderCe** (*int*) -- Host per provider CE

- **HostsPerVpls** (*int*) -- Host per VPLS
- **CustomerHostPercent** (*int*) -- Customer host percent
- **ProviderHostPercent** (*int*) -- Providier host percent

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_provider_port(*Wizard*, *Port*, ***kwargs*)

配置 MPLS 向导提供商侧端口

参数

- **Wizard** (WizardConfig) -- wizard config
- **Port** (list(*:obj:'Port*))) -- 测试仪表端口对象列表

关键字参数

- **PortIndex** (*int*) -- 默认值: 0,
- **EnableSubInterface** (*bool*) -- 默认值: False,
- **SubInterfaceCount** (*int*) -- 默认值: 1,
- **DutIpv4Address** (*str*) -- 默认值: '192.85.1.1',
- **DutIpv4AddressStep** (*str*) -- 默认值: '0.0.1.0',
- **Ipv4PrefixLength** (*int*) -- 默认值: 24,
- **VlanId** (*int*) -- 默认值: 1,
- **VlanIdStep** (*int*) -- 默认值: 1

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_provider_route_reflector(*Wizard*, ***kwargs*)

static edit_mpls_provider_router_basic_parameters(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **DutRouterId='10.0.0.1',** --
- **DutAsNumber=1,** --
- **Enable4ByteDutAs=False,** --
- **FourByteDutAsNumber='1** -- 1',
- **IgpProtocol=EnumMplsIgpProtocols.OSPF,** --
- **MplsProtocol=EnumMplsMplsProtocols.LDP,** --
- **EnablePRouter=True,** --
- **PRoutersPerInterface=1,** --
- **TopologyType=EnumMplsTopologyType.Tree,** --
- **PRouterStartIp='1.0.0.1',** --
- **PRouterPrefixLength=24,** --

- **PRouterIdStart='192.0.1.1',** --
- **PRouterIdStep='0.0.1.0',** --
- **PeRoutersPerInterface=1,** --
- **PeRouterIdStart='10.0.0.2',** --
- **PeRouterIdStep='0.0.0.1',** --
- **EnableRouteReflectors=None** --
- **Enable6Vpe=None** --

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_provider_router_isis(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由路由器 OSPF 协议

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **UseSrcMacAsSystemId** (*bool*) -- Use source MAC as system id
- **SystemId** (*str*) -- System ID
- **SystemIdStep** (*str*) -- System ID step
- **Level** (*str*) -- Level: L1 L2 L1L2
- **NetworkType** (*str*) -- Network type BROADCAST P2P
- **RouterPriority** (*int*) -- Router priority
- **MetricMode** (*str*) -- Metric mode NARROW WIDE NARROWWIDE
- **AuthenticationMode** (*str*) -- Authentication mode NONE SIMPLE MD5
- **Password** (*str*) -- Password
- **AreaId** (*list*) -- Area ID, hex int
- **EnableGracefulRestart** (*bool*) -- Enable graceful restart
- **MultiTopologyId** (*str*) -- Multi-topology ID NOSHOW IPV4 IPV6
- **EnableBfd** (*bool*) -- Enable BFD
- **HelloPadding** (*bool*) -- Enable hello padding
- **Algorithm** (*int*) -- SR algorithm
- **SidLabelBase** (*int*) -- SR SID/Label base
- **SidLabelRange** (*int*) -- SR SID/Label range
- **NodeSidIndex** (*int*) -- SR node SID index
- **NodeSidIdnexStep** (*int*) -- SR node SID index step

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_provider_router_ldp(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由路由器 LDP 协议

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **HelloType** (*str*) -- Hello type: DIRECT TARGETED DIRECT_TARGETED
- **TransportAddressTlvMode** (*str*) -- Transport address TLV mode TESTER_IP ROUTER_ID NONE
- **LabelAdvertisementMode** (*str*) -- Label advertisement mode: DU DOD
- **EgressLabelMode** (*str*) -- Egress label mode: NEXT_AVAILABLE IMPLICIT EXPLICIT
- **MinLabel** (*int*) -- Min Label
- **AuthenticationMode** (*str*) -- Authentication mode: NONE MD5
- **Password** (*str*) -- Password

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_provider_router_ospf(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由路由器 OSPF 协议

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **AreaId** (*str*) -- Area ID
- **NetworkType** (*str*) -- Network type: BROADCAST or P2P
- **RouterPriority** (*int*) -- Router priority
- **AuthenticationType** (*str*) -- Authentication type NONE SIMPLE MD5
- **Password** (*str*) -- Password
- **Md5Key** (*int*) -- MD5 key
- **Options** (*list*) -- Options: NONTBIT TOSBIT EBIT MCBIT NPBIT EABIT DCBIT OBIT DNBIT
- **EnableGracefulRestart** (*bool*) -- Enable graceful restart
- **GracefulRestartReason** (*str*) -- Gracefull restart reason: UNKNOWN SOFTWARE RELOADORUPGRADE SWITCH
- **EnableBfd** (*bool*) -- Enable BFD
- **Algorithm** (*int*) -- SR algorithm
- **SidLabelBase** (*int*) -- SR SID/Label base
- **SidLabelRange** (*int*) -- SR SID/Label range
- **NodeSidIndex** (*int*) -- SR node SID index
- **NodeSidIdnexStep** (*int*) -- SR node SID index step

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_provider_router_rip(*Wizard*, ***kwargs*)

配置 MPLS 提供商侧路由路由器 Rip 协议

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **RipVersion** (*str*) -- RIP version: RIPV1 RIPV2 RIPNG
- **UpdateType** (*str*) -- Update type: BROADCAST MULTICAST UNICAST
- **UpdateInterval** (*int*) -- Update interval(sec)
- **UpdateJitter** (*int*) -- Update jitter
- **MaxRouteNumPerUpdate** (*int*) -- Max route per update
- **AuthenticationMode** (*str*) -- Authentication mode: NONE SIMPLE MD5
- **Password** (*str*) -- Password
- **Md5KeyId** (*int*) -- MD5 key ID
- **SplitHorizon** (*bool*) -- Enable split horizon

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_pwe_basic_parameters(*Wizard*, ***kwargs*)

配置 MPLS pwe 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **NumberOfPseudoWire** (*int*) -- Number of pseudowire
- **Mtu** (*int*) -- MTU
- **GroupId** (*int*) -- Group ID
- **EnableCBit** (*bool*) -- Enable C-Bit
- **IncludeStatusTlv** (*bool*) -- Include status TLV
- **StatusCode** (*list*) -- Status code: PseudowireNotForwarding LocalAttachmentCircuitReceiveFault LocalAttachmentCircuitTransmitFault LocalPsnFacingPwIngressReceiveFault LocalPsnFacingPwEgressTransmitFault
- **EnableOverrideEncapsulation** (*bool*) -- Enable override encapsulation
- **Encapsulation** (*str*) -- Encapsulation EthernetVlan Ethernet EthernetVpls
- **EnableOverlapVcidsOnDifferentPes** (*bool*) -- Enable overlap VC IDs on different PEs
- **EnableCreateProviderHostsForUnusedVpls** (*bool*) -- Enable Create provider hosts for unsued VPLSs
- **FecType** (*str*) -- FEC type: FEC128 FEC129

返回 True

返回类型 (bool)

引发 exception.ContinuableFailure --

static edit_mpls_vpls_basic_parameters(*Wizard*, ***kwargs*)

配置 MPLS VPLS 参数

参数 Wizard (WizardConfig) -- wizard config

关键字参数

- **NumberOfVpls** (*int*) -- Number of VPLSs
- **Mtu** (*int*) -- MTU
- **GroupId** (*int*) -- Group ID
- **EnableCBit** (*bool*) -- Enable C-Bit
- **IncludeStatusTlv** (*bool*) -- Include status TLV
- **StatusCode** (*list*) -- Status code: PseudowireNotForwarding LocalAttachmentCircuitReceiveFault LocalAttachmentCircuitTransmitFault LocalPsnFacingPwIngressReceiveFault LocalPsnFacingPwEgressTransmitFault
- **EnableOverrideEncapsulation** (*bool*) -- Enable override encapsulation
- **Encapsulation** (*str*) -- Encapsulation EthernetVlan Ethernet EthernetVpls
- **VplsAssignment** (*str*) -- Vpls assginment: RoundRobin Sequential
- **CreateProviderHostsForUnusedVplss** (*bool*) -- Enable Create provider hosts for unsued VPLSs
- **FecType** (*str*) -- FEC type: FEC128 FEC129

返回 True

返回类型 (bool)

引发 exception.ContinuableFailure --

static edit_mpls_vpn_as_number(*Wizard*, ***kwargs*)

static edit_mpls_vpn_customer_parameters(*Wizard*, ***kwargs*)

配置 MPLS VPN 客户侧参数

参数 Wizard (WizardConfig) -- wizard config

关键字参数

- **VpnAssignment** (*int*) -- number of vpns
- **CeProtocol** (*str*) -- CE protocol type, params: BGP RIP ISIS OSPF Static Mixed
- **CeProAssignment** (*str*) -- CE protocol assignment (e.g. 'BGP=100%')
- **CustomerRdStart** (*str*) -- customer RD start (e.g. '1:0')
- **CustomerRdStepPerVpnEnabled** (*bool*) -- customer Rd step PerVpn enabled
- **CustomerRdStepPerVpn** (*str*) -- customer Rd step PerVpn (e.g. '1:0')
- **CustomerRdStepPerCeEnabled** (*bool*) -- customer Rd step PerCe enabled

- **CustomerRdStepPerCe** (*str*) -- customer Rd step PerCe (e.g. '0:0')

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_vpn_ipv4_route_customer_parameters(*Wizard*, ***kwargs*)

配置 MPLS VPN 路由客户端侧参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **CustomerStartRoute** (*str*) -- customer start route Ipv4Address
- **CustomerRouteStep** (*str*) -- customer route step Ipv4Address
- **CustomerPrefixLength** (*int*) -- customer prefix length
- **CustomerRoutesPerCe** -- (int) customer route PerCe
- **CustomerOverlapRoutes** (*bool*) -- customer overlap routes
- **CustomerRouteType** (*str*) -- customer route type, Internal or External

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_vpn_ipv4_route_provider_parameters(*Wizard*, ***kwargs*)

配置 MPLS VPN 路由提供商端侧参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **ProviderStartRoute** (*str*) -- provider start route, Ipv4Address
- **ProviderRouteStep** (*str*) -- provider route step, Ipv4Address
- **ProviderPrefixLength** (*int*) -- prefix length
- **ProviderRoutesPerCe** (*int*) -- provider route PerCe
- **ProviderOverlapRoutes** (*bool*) -- provider overlap routes
- **ProviderLabelType** (*str*) -- provider route type, LabelPerSite or LabelPerRoute
- **ProviderStartLabel** (*int*) -- start label

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_vpn_ipv6_route_customer_parameters(*Wizard*, ***kwargs*)

配置 MPLS VPN 路由客户端侧 IPv6 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **CustomerStartRoute** (*str*) -- customer start route, Ipv6Address
- **CustomerRouteStep** (*str*) -- customer route step, Ipv6Address
- **CustomerPrefixLength** (*int*) -- prefix length

- **CustomerRoutesPerCe** (*int*) -- customer route PerCe
- **CustomerOverlapRoutes** (*bool*) -- customer overlap routes
- **CustomerRouteType** (*str*) -- customer route type, Internal or External

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_vpn_ipv6_route_provider_parameters(*Wizard*, ***kwargs*)

配置 MPLS VPN 路由提供商端侧 IPv6 参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **ProviderStartRoute** (*str*) -- provider start route Ipv6Address
- **ProviderRouteStep** (*str*) -- provider route step Ipv6Address
- **ProviderPrefixLength** (*int*) -- prefix length
- **ProviderRoutesPerCe** (*int*) -- provider route PerCe
- **ProviderOverlapRoutes** (*bool*) -- provider overlap routes
- **ProviderLabelType** (*str*) -- provider route type, LabelPerSite or LabelPerRoute
- **ProviderStartLabel** (*int*) -- start label

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_vpn_parameters(*Wizard*, ***kwargs*)

配置 MPLS VPN 基本参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **NumberOfVpns** (*int*) -- number of vpns
- **RdAssignment** (*str*) -- Route Target Assignment, support: UseRT、Manual
- **RouteTargetStart** (*str*) -- route target start (e.g. '1:0')
- **RouteTargetStep** (*str*) -- route target step (e.g. '1:0')

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_mpls_vpn_provider_parameters(*Wizard*, ***kwargs*)

配置 MPLS VPN 提供商侧参数

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **ProviderDisSel** (*str*) -- provider distribution selector, support: VPNsPerPE、PEsPerVPN
- **ProviderDisSelCount** (*int*) -- provider distribution selector count

- **ProviderMeshed** (*bool*) -- provider meshed
- **ProviderRdStart** (*str*) -- provider RD start (e.g. '1:0')
- **ProviderRdStepPerVpnEnabled** (*bool*) -- provider Rd step PerVpn enabled
- **ProviderRdStepPerVpn** (*str*) -- route target start (e.g. '1:0')
- **ProviderRdStepPerCeEnabled** (*bool*) -- provider Rd step PerCe enabled
- **ProviderRdStepPerCe** (*str*) -- provider Rd step PerCe (e.g. '0:0')

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_ospf(*Session*, ***kwargs*)

编辑 OSPFv2 协议会话对象参数

参数 **Session** (OspfRouter) -- OSPFv2 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- OSPFv2 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 OSPFv2 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **AreaId** (*str*) -- 区域 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 取值范围：Broadcast 或 P2P, 默认值：Broadcast
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **Cost** (*int*) -- 接口开销, 类型为：number, 取值范围：1-65535, 默认值：10
- **AuthenticationType** (*str*) -- 类型为：string, 取值范围：None Simple 或 MD5, 默认值：None
- **Password** (*str*) -- 密码, 类型为：string, 默认值：Xinertel
- **Md5KeyId** (*int*) -- MD5 密钥, 类型为：number, 取值范围：0-255, 默认值：1
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'EBIT'], 支持选项有：

  NONTBIT

  TOSBIT

  EBIT

  MCBIT

  NPBIT

  EABIT

  DCBIT

  OBIT

DNBIT

- **EnableOspfv2Mtu** (*bool*) -- 使能 OSPF MTU, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GracefulRestartReason** (*str*) -- 平滑重启原因, 类型为：string, 默认值：UNKNOWN, 支持的原因：

  UNKNOWN

  SOFTWARE

  RELOADORUPGRADE

  SWITCH

- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **HelloInterval** (*int*) -- Hello 包间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **RouterDeadInterval** (*int*) -- 路由器失效间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：40
- **LsaRetransInterval** (*int*) -- LSA 重传间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：5
- **LsaRefreshTime** (*int*) -- LSA 刷新间隔 (秒), 类型为：number, 取值范围：1-1800, 默认值：1800
- **EnableSrManagement** (*bool*) -- 启用 SR, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Ospf | Session=${Session} |
```

static edit_ospf_port_config(*Ports*, ***kwargs*)

修改 Ospf 端口统计对象

参数 **Ports** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **TransmitRate** (*int*) -- OSPFv2 Message Tx Rate (messages/second), 取值范围：1-9000, 默认值：100
- **SessionOutstanding** (*int*) -- OSPFv2 Session Outstanding, 取值范围：1-1000, 默认值：20
- **UpdateMsgTransmitRate** (*int*) -- Deprecated. OSPFv2 Update Message Tx Rate (messages/second), 取值范围：1-9000, 默认值：10
- **EnableLoop** (*bool*) -- Enable Loop Back, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Ospf Port Config | Ports=${Ports} | TransmitRate=100 |
```

static edit_ospf_te_lsa_link(*TeLsa*, ***kwargs*)

编辑 OSPFv2 Te LSA Link 参数

参数 **TeLsa** (Ospfv2TeLsaConfig) -- OSPFv2 Te LSA 对象, 类型为：object

关键字参数

- **Name** (*str*) -- 创建的 OSPFv2 Te LSA Link 对象的名称, 类型为：string
- **Enable** (*bool*) -- 使能, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableLocalIp** (*bool*) -- 使能本端 IPv4 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LocalIp** (*str*) -- 本端 IPv4 地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableRemoteIp** (*bool*) -- 使能远端 IPv4 地址, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **RemoteIp** (*str*) -- 远端 IPv4 地址, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableGroup** (*bool*) -- 启动组, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **Group** (*int*) -- 组 ID, 类型为：number, 取值范围：0-4294967295, 默认值：1
- **EnableMaxBandwidth** (*bool*) -- 启动最大带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **MaximumBandwidth** (*int*) -- 最大带宽, 类型为：number, 取值范围：0-16777215, 默认值：1000
- **EnableReservedBandwidth** (*bool*) -- 启动预留带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ReservableBandwidth** (*int*) -- 预留带宽, 类型为：number, 取值范围：0-4294967295, 默认值：1000
- **EnableUnreservedBandwidth** (*bool*) -- 启动未预留带宽, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **UnreservedBandwidth0** (*int*) -- 未预留带宽优先级 0, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth1** (*int*) -- 未预留带宽优先级 1, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth2** (*int*) -- 未预留带宽优先级 2, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth3** (*int*) -- 未预留带宽优先级 3, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth4** (*int*) -- 未预留带宽优先级 4, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth5** (*int*) -- 未预留带宽优先级 5, 类型为：number, 取值范围：0-4294967295, 默认值：0
- **UnreservedBandwidth6** (*int*) -- 未预留带宽优先级 6, 类型为：number, 取值范围：0-4294967295, 默认值：0

- **UnreservedBandwidth7** (*int*) -- 未预留带宽优先级 7, 类型为：number, 取值范围：0-4294967295, 默认值：0

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${TeLsa} | Create Ospf Te Lsa | Session=${Session} | Age=20 |
| Edit Ospf Te Lsa Link| TeLsa=${TeLsa} | LocalIp=2.2.2.2 |
```

static edit_ospfv3(*Session*, ***kwargs*)

编辑 OSPFv3 协议会话对象参数

参数 **Session** (list(Ospfv3Router)) -- OSPFv3 协议会话对象列表

关键字参数

- **Name** (*str*) -- OSPFv3 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 OSPFv3 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **InstanceId** (*int*) -- 实例 ID, 类型为：number, 取值范围：0-255, 默认值：0
- **AreaId** (*str*) -- 区域 ID, 类型为：string, 取值范围：0.0.0.0-255.255.255.255, 默认值：0.0.0.0
- **EnableExtendedLsa** (*bool*) -- 使能扩展 LSA, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **ExtendedLsaMode** (*str*) -- 扩展 LSA 模式, 类型为：string, 默认值：Full, 取值范围：

  NONE

  MixedModeOriginateOnly

  MixedModeOriginateSPF

  Full

- **AreaExtendedLsaMode** (*str*) -- 扩展区域 LSA 模式, 类型为：string, 默认值：InheritGlobal, 取值范围：

  InheritGlobal

  NONE

  MixedModeOriginateOnly

  MixedModeOriginateSPF

  Full

- **EnableBfd** (*bool*) -- 使能 BFD, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **NetworkType** (*str*) -- 网络类型, 类型为：string, 取值范围：Broadcast 或 P2P, 默认值：Broadcast
- **Priority** (*int*) -- 路由器优先级, 类型为：number, 取值范围：0-255, 默认值：0
- **InterfaceId** (*int*) -- 接口 ID, 类型为：number, 取值范围：0-4294967295, 默认值：10

- **Cost** (*int*) -- 接口开销, 类型为：number, 取值范围：1-65535, 默认值：10
- **Options** (*list*) -- 选项, 类型为：list, 默认值：['NONTBIT', 'V6BIT', 'EBIT', 'RBIT'], 支持选项有：

  NONTBIT

  V6BIT

  EBIT

  MCBIT

  NBIT

  RBIT

  DCBIT

  Unused17

  Unused16

  Unused15

  Unused14

  Unused13

  Unused12

  Unused11

  Unused10

  Unused9

  Unused8

  Unused7

  Unused6

  Unused5

  Unused4

  Unused3

  Unused2

  Unused1

  Unused0
- **EnableOspfv3Mtu** (*bool*) -- 使能 OSPFv3 MTU, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **EnableGracefulRestart** (*bool*) -- 使能平滑重启, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **GracefulRestartReason** (*str*) -- 平滑重启原因, 类型为：string, 默认值：UNKNOWN, 取值范围：

  UNKNOWN

  SOFTWARE

  RELOADORUPGRADE

  SWITCH
- **EnableViewRoutes** (*bool*) -- 使能查看路由, 类型为：bool, 取值范围：True 或 False, 默认值：False

- **HelloInterval** (*int*) -- Hello 包间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：10
- **RouterDeadInterval** (*int*) -- 路由器失效间隔 (秒), 类型为：number, 取值范围：0-65535, 默认值：40
- **LsaRetransInterval** (*int*) -- LSA 重传间隔 (秒), 类型为：number, 取值范围：0-4294967295, 默认值：5
- **LsaRefreshTime** (*int*) -- LSA 刷新间隔 (秒), 类型为：number, 取值范围：1-1800, 默认值：1800

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Ospfv3 | Session=${Session} |
```

static edit_ospfv3_port_config(*Ports*, ***kwargs*)

修改 Ospfv3 端口统计对象

参数 **Ports** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **TransmitRate** (*int*) -- OSPFv3 Message Tx Rate (messages/second), 取值范围：1-9000, 默认值：100
- **SessionOutstanding** (*int*) -- OSPFv3 Session Outstanding, 取值范围：1-1000, 默认值：20
- **UpdateMsgTransmitRate** (*int*) -- Deprecated. OSPFv3 Update Message Tx Rate (messages/second), 取值范围：1-9000, 默认值：10
- **EnableLoop** (*bool*) -- Enable Loop Back, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Ospfv3 Port Config | Ports=${Ports} | TransmitRate=100 |
```

static edit_overall_setting(***kwargs*)

编辑测试仪表统全局参数

关键字参数

- 流全局配置，参数支持： -- PortSendMode: 端口发送模式

  SYNCHRONOUS ASYNCHRONOUS

  MeshCreationMode: 拓扑创建模式

  PortBased EndpointBased

- 二层学习，参数支持： -- Rate: 速率（帧/秒），类型：number，值范围：1-4294967295, 默认值：100

  RepeatCount: 重复次数, 类型：number，值范围：0-4294967295, 默认值：3

DelayTime: 学习前延迟时间, 类型：number，值范围：0-4294967295, 默认值：1

RxLearningEncapsulation: 封装类型

NO_ENCAPSULATION TX_ENCAPSULATION

- **ARP/ND** 选项，参数支持： -- EnableAutoArp: 使能自动 ARP/ND, 类型: bool, 默认值：True

  StopOnArpFail: ARP/ND 失败自动停止测试, 类型: bool, 默认值：False

  AutoArpWaitTime: 自动 ARP/ND 等待时间（秒）, 类型：number，值范围：0-4294967295, 默认值：30

- **LM** 全局配置，参数支持： (*Y.1731*) -- TestModeType: 测试模式, 类型: string, 默认值：TYPE_NORMAL

  TYPE_NORMAL TYPE_CC_SCALE_MODE TYPE_CC_SCALE_MODE_WITHOUT_RX

  LmrRxFCfStart: LMR 帧的 RxFCF 初始值, 类型：number，值范围：0-4294967295, 默认值：1

  LmrRxFCfStep: LMR 帧的 RxFCF 更新步长, 类型：number，值范围：0-65535, 默认值：1

  LmrTxFCbStart: LMR 帧的 TxFCF 初始值, 类型：number，值范围：0-4294967295, 默认值：1

  LmrTxFCbStep: LMR 帧的 TxFCF 更新步长, 类型：number，值范围：1-65535, 默认值：9

  LmmTxFCfOffset: LMM 帧的 TxFCF 的偏移值, 类型：number，值范围：0-32767, 默认值：0

  LmrRxFCfOffset: LMR 帧的 RxFCF 的偏移值, 类型：number，值范围：0-32767, 默认值：0

  LmrTxFCbOffset: LMR 帧的 TxFCF 的偏移值, 类型：number，值范围：0-32767, 默认值：0

  DmTimeUnit: DM 时间统计单位, 类型：string, 默认值：TYPE_NORMAL

  TIME_MS TIME_NS

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| edit_overall_setting | PortSendMode=ASYNCHRONOUS |␣
↪RxLearningEncapsulation=TX_ENCAPSULATION | EnableAutoArp=False |␣
↪TestModeType=TYPE_CC_SCALE_MODE |
```

static edit_pcep(*Sessions*, ***kwargs*)

编辑 PCEP 协议会话对象参数

参数 **Sessions** (Pcep) -- ：PCEP 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- PCEP 协会话名称, 类型为：string

- **Enable** (*bool*) -- 使能 PCEP 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Role** (*str*) -- PCEP 角色, 类型为：string, 默认值：PCE, 取值范围:

  PCE

  PCC
- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPv4, 取值范围:

  IPv4

  IPv6
- **UseGatewayAsDutIp** (*bool*) -- 使用网关地址作为 DUT 地址, 选中则使用接口上配置的网关 IP 地址作为 DUT 地址；未选中则自定义 DUT IP 地址, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SessionIpAddress** (*str*) -- 会话 IP 地址, 用于 PCEP 连接的 IP 类型, 类型为：string, 默认值：Interface_IP, 取值范围:

  Interface_IP

  Router_ID
- **PeerIpv4Address** (*str*) -- DUT IPv4 地址, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv4 时可见, 指定 DUT 的 IPv4 地址, 类型为：string, 默认值：192.85.1.1, 取值范围：有效的 Ipv4 地址
- **PeerIpv4AddressStep** (*str*) -- DUT IPv4 地址跳变, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv4 时可见, 指定 DUT IPv4 地址的增量步长, 类型为：string, 默认值：0.0.0.1, 取值范围：有效的 Ipv4 地址
- **PeerIpv6Address** (*str*) -- DUT IPv6 地址, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv6 时可见, 指定 DUT 的 IPv6 地址, 类型为：string, 默认值：2000::1, 取值范围：有效的 Ipv6 地址
- **PeerIpv6AddressStep** (*str*) -- DUT IPv6 地址跳变, 使用网关地址作为 DUT 地址未选中且 IP 版本为 IPv6 时可见, 指定 DUT 的 IPv6 地址的增量步长, 类型为：string, 默认值：::1, 取值范围：有效的 Ipv6 地址
- **SessionInitiator** (*bool*) -- 会话发起者, 选中则主动发起会话建立请求；未选中则监听对端的发起会话建立请求。双方均主动发起会话建立请求时，IP 地址大的一方优先级更高, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **Negotiation** (*bool*) -- 使能 Negotiation, 选中则对 Keepalive Timer 和 Dead Timer 的值进行协商, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **KeepAlive** (*str*) -- Keep Alive 间隔 (sec), KEEPALIVE 消息的发送间隔，以秒为单位, 类型为：string, 默认值：30, 0-65535
- **MinKeepAlive** (*int*) -- KEEPALIVE 消息发送间隔的最小值。以秒为单位, 类型为：number, 取值范围：0-255, 默认值：0
- **MaxKeepAlive** (*int*) -- KEEPALIVE 消息发送间隔的最大值，以秒为单位, 类型为：number, 取值范围：0-255, 默认值：255
- **Dead** (*str*) -- Dead 间隔 (sec), 从未收到对端消息到 PCEP 会话断开连接之间的时间间隔。类型为：string, 取值范围：0-65535, 默认值：120
- **MinDeadAlive** (*int*) -- 最小可接受 Dead 间隔 (sec), 类型为：number, 取值范围：0-255, 默认值：0
- **MaxDeadAlive** (*int*) -- 最大可接受 Dead 间隔 (sec), 类型为：number, 取值范围：0-255, 默认值：255

- **EnableStatefulCapability** (*bool*) -- 选中则 OPEN 消息中包含 Stateful PCE Capability TLV, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **StatefulCapability** (*list*) -- 使能 PCE Stateful Capability 选中时可见, 单击单元格并从下拉菜单中选择一个或多个能力, 类型为：list, 默认值：['LSP_UPDATE','LSP_INSTANTIATION'], 取值范围:

  LSP_UPDATE

  INCLUDE_DB_VERSION

  LSP_INSTANTIATION

  TRIGGERED_RESYNC

  DELTA_LSP_SYN

  TRIGGERED_INITIAL_SYNC
- **EnableSegmentRoutingCapability** (*list*) -- 选择段路由扩展，OPEN 消息中将包括该 Capability TLV, 类型为：list, 默认值：['SR'], 取值范围：

  SR

  SRv6
- **PathSetupTypeList** (*list*) -- 添加路径建立类型, 类型为：list, 默认值：[0,1]
- **SrCapabilityFlags** (*list*) -- 选择一个或多个 SR 能力标志, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SR 时可见, 类型为：list, 默认值：['NONTBIT','NFlag','XFlag'],

  NONTBIT

  NFlag

  XFlag
- **Srv6CapabilityFlags** (*list*) -- 选择一个或多个 SRv6 能力标志, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SRv6 时可见, 类型为：list, 默认值：['NONTBIT','NFlag','XFlag'],

  NONTBIT

  NFlag

  XFlag
- **MSDs** (*list*) -- 选择一个或多个 MSD 类型, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SRv6 时可见, 类型为：list, 默认值：['NONTBIT']

  NONTBIT

  MaxiSegmentLeft

  MaxiEndPop

  MaxiHEncaps

  MaxiEndD
- **MaximumSidDepth** (*int*) -- 指定 SID 的最大数量, PCEP 角色为 PCC，且使能 Segment Routing Capability 中选中 SR 时可见, 类型为：number, 默认值：0, 取值范围：0-255
- **MaxSegmentsLeft** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum Segments Left 时可见, 类型为：number, 默认值：8, 取值范围：0-255

- **MaxEndPop** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum End Pop 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxHencaps** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum H.Encaps 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **MaxEndD** (*int*) -- 指定 MSD 取值, PCEP 角色为 PCC，使能 Segment Routing Capability 中选中 SRv6，且 MSDs 中选中 Maximum End D 时可见, 类型为：number, 默认值：8, 取值范围：0-255
- **EnableDbVersionTlv** (*bool*) -- 选中则配置 DB version TLV, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **LspStateDbVersion** (*int*) -- 指定 LSP 状态数据库的初始版本号, 选中使能 DB Version TLV 时可见, 类型为：number, 默认值：1, 取值范围：1-18446744073709551614

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Pcep | Port=${Port} |
| Edit Pcep | Session=${Session} | Role=PCC |
```

static edit_pcep_port_config(*Ports*, ***kwargs*)

修改 PCEP 端口配置对象

Args：

Ports (*Port*): 测试仪表端口对象, 类型为：object

关键字参数

- **MaxOutstanding** (*int*) -- 最大会话负载数量, 取值范围：1-65535, 默认值：100
- **RetryCount** (*int*) -- 会话尝试建立次数, 取值范围：0-65535, 默认值：5
- **RetryInterval** (*int*) -- 会话尝试建立间隔（sec）, 取值范围：0-65535, 默认值：30
- **MaxLspPerMessage** (*int*) -- 消息中 LSP 的最大个数, 取值范围：1-2000, 默认值：100

返回 PCEP Client Custom Options 对象, 类型：object / list

返回类型 (PCEPPortRateConfig)

实际案例

```
| Edit PCEP Client Port Config | Ports=${Port} | TcpServerPort=10 |
```

static edit_pim(*Session*, ***kwargs*)

编辑 PIM 协议会话对象参数

参数 **Session** (PimRouter) -- PIM 协议会话对象列表, 类型为：object

关键字参数

- **Name** (*str*) -- PIM 协会话名称, 类型为：string

- **Enable** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：True
- **SessionMode** (*str*) -- 协议模式, 类型为：string, 默认值：SM, 支持版本：

  SM

  SSM
- **IpVersion** (*str*) -- IP 版本, 类型为：string, 默认值：IPV4, 支持版本：

  IPV4

  IPV6
- **DrPriority** (*int*) -- DR 优先级, 类型为：number, 取值范围：1-65535, 默认值：1
- **DrAddr** (*str*) -- DR 地址, 类型为：string, 取值范围：IPv4 地址, 默认值：0.0.0.0
- **DrIpv6Addr** (*str*) -- DR IPv6 地址, 类型为：string, 取值范围：IPv6 地址, 默认值：'::'
- **GenIdMode** (*str*) -- GenID 模式, 类型为：string, 默认值：FIXED, 支持参数：

  FIXED

  INCR

  RAND
- **RegisterEnable** (*bool*) -- Register 使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **BsrEnable** (*bool*) -- BSR 使能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **BsrPriority** (*int*) -- BSR 优先级, 类型为：number, 取值范围：0-255, 默认值：1
- **BsrInterval** (*int*) -- BSR 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-3600, 默认值：60
- **HelloInterval** (*int*) -- Hello 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-3600, 默认值：30
- **HelloHoldTime** (*int*) -- Hello 消息超时时间（秒）, 类型为：number, 取值范围：1-65535, 默认值：105
- **JoinPruneInterval** (*int*) -- Join/Prune 消息发送时间间隔（秒）, 类型为：number, 取值范围：1-65535, 默认值：60
- **JoinPruneHoldTime** (*int*) -- Join/Prune 消息超时时间（秒）, 类型为：number, 取值范围：1-65535, 默认值：210

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Session} | Create Pim | Port=${Port} |
| Edit Pim | Session=${Session} | HelloInterval=60 |
```

static edit_pim_port_config(*Ports*, ***kwargs*)

修改 PIM 协议会话的端口配置

参数 **Ports** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **MsgTransRate** (*int*) -- PIM Message Transmit Rate (messages/sec), 类型为：number, 取值范围：1-10000, 默认值：500
- **TriggerHelloDelay** (*int*) -- Trigger Hello Delay (sec), 类型为：number, 取值范围：0-60, 默认值：5
- **DisableHelloExpireTimer** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **DisableRecvHelloInNeighborState** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **DisableNonHelloRecv** (*bool*) -- 使能 PIM 协议会话, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pim Port Config | Ports=${Ports} | DisableNonHelloRecv=True |
```

static edit_port(*Ports*, ***kwargs*)

修改测试仪表端口参数

参数 **Ports** (list(*Port*)) -- 测试仪表端口列表

关键字参数

- **EnableLink** (*bool*) -- 设置端口 link Up 和 Down, 默认值: True
- **AutoNegotiation** (*bool*) -- 自协商, 默认值: False
- **Mtu** (*int*) -- 端口 MTU 值, 范围: 128-9600
- **FecType** (*str*) -- Fec 类型, 支持: TYPE_OFF TYPE_RS_FEC_CLAUSE91 TYPE_FEC_CLAUSE74 TYPE_RS_FEC_CLAUSE108 TYPE_RS_FEC_CONSORTIUM TYPE_RS_FEC_CLAUSE119
- **LineSpeed** (*str*) -- 端口速率切换, 支持: SPEED_UNKNOWN SPEED_10M SPEED_100M SPEED_1G SPEED_2_5G SPEED_5G SPEED_10G SPEED_25G SPEED_40G SPEED_50G SPEED_100G SPEED_200G SPEED_400G
- **Duplex** (*str*) -- 全双工半双工, 支持: HALF FULL
- **FlowControl** (*str*) -- 流控, 支持: DISABLE ENABLE AUTO
- **Media** (*str*) -- 媒介, 支持: COPPER FIBER FAKE
- **PhyMode** (*str*) -- Phy Mode, 支持: MODE_AUTO MODE_1000BASEX MODE_SGMII

- **PpmAdjust** (*int*) -- Ppm Adjust, 范围: -300-300
- **DataPathMode** (*str*) -- Data Path 模式, 支持: NORMAL LOOPBACK
- **RemoteFault** (*str*) -- 远端错误, 支持: NORMAL IGNORE
- **Master** (*str*) -- Master, 支持: ADVERTISE_SINGLE_PORT ADVERTISE_MULTI_PORT MANUAL_MASTER MANUAL_SLAVE
- **NoParam** (*bool*) -- 远端错误, 默认值: False

返回 字符串：string，返回保存的 DB 文件的绝对路径字符串

实际案例

robotframework:

```
| Edit Port | Ports=${Ports} | AutoNegotiation=True | FecType=TYPE_OFF |
```

static edit_port_load_profile(*Ports*, ***kwargs*)

编辑测试仪表负载配置文件参数

参数 **Ports** (list(IsisIpv4Router) -- 测试仪表端口对象 object 列表

关键字参数

- **TransmitMode** (*str*) -- 传输模式, 默认值：CONTINUOUS，取值范围：
  CONTINUOUS: 连续
  BURST: 突发
  TIME: 按时间突发
  STEP: 单步突发
  ONSTREAM: 基于流调速
- **BurstSize** (*int*) -- 突发报文数，默认值：1
- **InterFrameGap** (*int*) -- 突发间隔，默认值：12.0
- **InterFrameGapUnit** (*str*) -- 突发间隔单位，默认值：BYTES，取值范围：
  NS
  MS
  US
  SEC
  BYTES
- **BurstCount** (*int*) -- 突发次数，默认值：1
- **Seconds** (*int*) -- 发送时间，单位：sec，默认值：100
- **Frames** (*int*) -- 发送帧数，默认值：1
- **LoadProfileType** (*str*) -- 负载类型，默认值：PORT_BASE，取值范围：
  PORT_BASE:
  STREAM_BASE
  PRIORITY_BASE
  MANUAL_BASE
- **Rate** (*int*) -- 端口负载，默认值：10

- **Unit** (*str*) -- 端口负载单位，默认值：PERCENT，取值范围：

  PERCENT:

  FRAME_PER_SEC

  BYTE_PER_SEC

  LINEBIT_PER_SEC

  KLINEBIT_PER_SEC

  MLINEBIT_PER_SEC

  INTER_FRAME_GAP
- **GenerateError** (*str*) -- 报文造错，默认值：NO_ERROR，取值范围：

  NO_ERROR CRC
- **IgnoreLinkState** (*str*) -- 忽略连接状态，默认值：NO，取值范围：

  NO YES
- **TimeStampPosTx** (*str*) -- 发送时间戳位置，默认值：TIMESTAMP_HEAD，取值范围：

  TIMESTAMP_HEAD TIMESTAMP_TAIL
- **TimeStampPosRx** (*str*) -- 接收时间戳位置，默认值：TIMESTAMP_HEAD，取值范围：

  TIMESTAMP_HEAD TIMESTAMP_TAIL
- **LatencyCompensationTx** (*int*) -- 发送时延补偿，默认值：0
- **LatencyCompensationRx** (*int*) -- 接收时延补偿，默认值：0
- **LatencyCompensationOn** (*bool*) -- 时延补偿开启，默认值：True

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=CONTINUOUS |␣
↪Unit=PERCENT | Rate=100 |
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=BURST |␣
↪BurstSize=10 | InterFrameGap=20 | InterFrameGapUnit=MS | BurstCount=100␣
↪|
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=TIME |␣
↪Seconds=10 |
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=STEP | Frames=10␣
↪|
| Edit Port Load Profile | Ports=${Ports} | TransmitMode=ONSTREAM |␣
↪Rate=50 | Unit=FRAME_PER_SEC |
```

static edit_pppoe_clinet(*Session*, ***kwargs*)

创建 PPPoE 协议会话对象

参数 **Port** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数

- **Name** (*str*) -- PPPoE 协会话名称
- **Enable** (*bool*) -- 使能 PPPoE 协议会话, 默认值：True

- **EmulationMode** (*str*) -- PPPoE 角色, 默认值：CLIENT, 取值范围：

  CLIENT SERVER PPPoL2TP

- **AuthenticationType** (*str*) -- 认证方式, 默认值：NO_AUTHENTICATION, 取值范围：

  NO_AUTHENTICATION NEGOTIATION CHAP_MD5 PAP

- **Username** (*str*) -- 用户名, 默认值：xinertel, 取值范围：string length in [1,126]
- **Password** (*str*) -- 密码, 默认值：xinertel, 取值范围：string length in [1,126]
- **ServiceName** (*str*) -- 服务名, 默认值：””, 取值范围：string length in [0,255]
- **EnableMaxPayloadTag** (*bool*) -- 使能最大净荷标签, 默认值：False
- **MaxPayloadBytes** (*int*) -- 最大净荷 (字节), 取值范围：1-65535, 默认值：1500
- **LcpConfigReqTimeout** (*int*) -- LCP Configure-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **LcpConfigReqMaxAttempts** (*int*) -- LCP Configure-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **LcpTermReqTimeout** (*int*) -- LCP Terminate-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **LcpTermReqMaxAttempts** (*int*) -- LCP Terminate-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **NcpConfigReqTimeout** (*int*) -- NCP Configure-Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **NcpConfigReqMaxAttempts** (*int*) -- NCP Configure-Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **LcpNcpMaxNak** (*int*) -- LCP/NCP 最大 Nak 数量, 取值范围：1-65535, 默认值：5
- **EnableMruNegotiation** (*bool*) -- 使能 MRU 协商, 默认值：True
- **MruSize** (*int*) -- MRU(字节), 取值范围：128-65535, 默认值：1492
- **EnableEchoRequest** (*bool*) -- 使能 Echo-Request 报文, 默认值：False
- **EchoRequestInterval** (*int*) -- Echo-Request 间隔 (sec), 取值范围：1-65535, 默认值：10
- **EchoRequestMaxAttempts** (*int*) -- Echo-Request 最大尝试次数, 取值范围：1-65535, 默认值：3
- **EnableMagicNumber** (*bool*) -- 使能 Magic Number, 默认值：True
- **PadiTimeout** (*int*) -- Client 参数, PADI 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PadiMaxAttempts** (*int*) -- Client 参数, PADI 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PadrTimeout** (*int*) -- Client 参数, PADR 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PadrMaxAttempts** (*int*) -- Client 参数, PADR 最大尝试次数, 取值范围：1-65535, 默认值：10
- **EnableRelayAgent** (*bool*) -- Client 参数, 启用中继代理, 默认值：False

- **RelayAgentDestMac** (*str*) -- Client 参数, 中继代理 MAC 地址, 取值范围：有效的 mac 地址, 默认值：00:00:00:00:00:00
- **RelayAgentDestMacStep** (*str*) -- Client 参数, 中继代理 MAC 地址跳变, 取值范围：有效的 mac 地址, 默认值：00:00:00:00:00:01
- **UseRelayAgentPadi** (*bool*) -- Client 参数, PADI 中包含中继代理信息, 默认值：True
- **UseRelayAgentPadr** (*bool*) -- Client 参数, PADR 中包含中继代理信息, 默认值：True
- **RelayAgentType** (*str*) -- Client 参数, 中继代理类型, 默认值：RFC2516, 取值范围：

  RFC2516 DSL_FORUM
- **RelaySessionId** (*str*) -- Client 参数, 中继会话 ID, 取值范围：string length in [0,12], 默认值：""
- **CircuitId** (*str*) -- Client 参数, 环路 ID, 取值范围：string length in [0,63], 默认值：@s
- **RemoteId** (*str*) -- Client 参数, 远程 ID, 取值范围：string length in [0,63], 默认值：@m-@p
- **ChapChalReqTimeout** (*int*) -- Client 参数, CHAP Challenge Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapAckTimeout** (*int*) -- Client 参数, CHAP Ack 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapMaxReplyAttempts** (*int*) -- Client 参数, CHAP Reply 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PapReqTimeout** (*int*) -- Client 参数, PAP Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **PapReqMaxAttempts** (*int*) -- Client 参数, PAP Request 最大尝试次数, 取值范围：1-65535, 默认值：10
- **EnableAutoRetry** (*bool*) -- Client 参数, 使能 PPPoE 协议会话, 默认值：False
- **AutoRetryCount** (*int*) -- Client 参数, 重连次数, 取值范围：1-65535, 默认值：65535
- **LcpDelay** (*int*) -- Client 参数, LCP 推迟时间 (ms), 取值范围：1-65535, 默认值：0
- **EnableAutoFillIpv6** (*bool*) -- Client 参数, 启用获取 Global IPv6 地址, 默认值：True
- **AcName** (*str*) -- Server 参数, 访问集中器名称, 默认值：Xinertel
- **ChapReplyTimeout** (*int*) -- Server 参数, CHAP Reply 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **ChapMaxChalAttempts** (*int*) -- Server 参数, CHAP Challenge 最大尝试次数, 取值范围：1-65535, 默认值：10
- **PapPeerReqTimeout** (*int*) -- Server 参数, 等待 PAP Request 超时时间 (sec), 取值范围：1-65535, 默认值：3
- **Ipv4Start** (*str*) -- Server 参数, IPv4 起始地址, 默认值：192.0.1.0
- **Ipv4Step** (*str*) -- Server 参数, IPv4 地址步长, 默认值：0.0.0。1
- **Ipv4Count** (*int*) -- Server 参数, IPv4 地址数量, 取值范围：1-65535, 默认值：3

- **Ipv6InterfaceId** (*str*) -- Server 参数, 起始 Interface ID, 默认值: ”::2”
- **Ipv6InterfaceIdStep** (*str*) -- Server 参数, Interface ID 跳变步长, 默认值: ”::1”
- **Ipv6PrefixStart** (*str*) -- Server 参数, IPv6 起始前缀, 默认值: ”2002::”
- **Ipv6PrefixStep** (*str*) -- Server 参数, IPv6 前缀跳变步长, 默认值: ”0:0:0:1::”
- **Ipv6Count** (*int*) -- Server 参数, IPv6 前缀数量, 取值范围: 1-65535, 默认值: 1
- **EnableForceConnectMode** (*bool*) -- 强制重连模式, 默认值: False
- **UnconnectedSessionThreshold** (*int*) -- 未连接会话门限值, 取值范围: 1-65535, 默认值: 1
- **MAndOFlag** (*str*) -- Server 参数, M 与 O 标志位, 默认值: M0_O0, 支持

  M0_O0 M0_O1 M1

返回 PPPoE 协议会话对象, 类型: object

返回类型 (PppoeClent)

实际案例

```
| Create Pppoe | Port=${Port} |
```

static edit_rip(*Session*, ***kwargs*)

编辑 Rip 协议会话对象参数

参数 **Session** (RipRouter) -- Rip 协议会话对象列表, 类型为: object

关键字参数

- **Name** (*str*) -- RIP 协会话名称, 类型为: string
- **Enable** (*bool*) -- 使能 RIP 协议会话, 类型为: bool, 取值范围: True 或 False, 默认值: True
- **Version** (*str*) -- RIP 版本, 类型为: string, 默认值: RIPV2, 支持版本:

  RIPV1

  RIPV2

  RIPNG
- **UpdateType** (*str*) -- 仿真路由器指定发送 RIP 消息的通信方式, 类型为: string, 默认值: MULTICAST, 支持方式:

  BROADCAST

  MULTICAST

  UNICAST
- **DutIpv4Address** (*str*) -- 指定接收 RIP 消息的路由器的 IP 地址, 当 RIP 版本为 RIPv1 或者 RIPv2 时, 该选项可配。类型为: string, 默认值: 224.0.0.9
- **DutIpv6Address** (*str*) -- 指定接收 RIP 消息的路由器的 IP 地址, 当 RIP 版本为 RIPng 并且更新类型指定为 Unicast 时, 该选项可配。类型为: string, 默认值: ff02::9

- **AuthMethod** (*str*) -- 认证方式, 当 RIP 版本为 RIPv2 时配置该选项。类型为：string, 默认值：NONE, 支持方式：

  NONE

  SIMPLE

  MD5

- **Password** (*str*) -- 当认证方式为 Simple/MD5 时，输入的认证密码, 类型为：string, 默认值：Xinetel
- **Md5KeyId** (*int*) -- 当认证方式为 MD5 时，输入的 MD5 密钥, 类型为：number, 取值范围：0-255, 默认值：1
- **UpdateInterval** (*int*) -- 发送 RIP 更新消息的时间间隔，单位为秒, 类型为：number, 取值范围：1-65535, 默认值：30
- **UpdateJitter** (*int*) -- 发送 RIP 更新消息的时间抖动, 类型为：number, 取值范围：0-5, 默认值：0
- **MaxRoutePerUpdate** (*int*) -- 更新消息中可携带的最大路由数, 类型为：number, 取值范围：1-70, 默认值：25
- **SplitHorizon** (*bool*) -- 是否开启水平分割功能, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableViewRoutes** (*bool*) -- 是否需要查看学到的路由信息, 类型为：bool, 取值范围：True 或 False, 默认值：False
- **EnableIpAddrValidation** (*bool*) -- 验证收到的 IP 地址是否和本地地址在同一网段, 类型为：bool, 取值范围：True 或 False, 默认值：False

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Rip | Session=${Session} | EnableViewRoutes=True |
```

static edit_rip_port_config(*Ports*, ***kwargs*)

修改 RIP 端口统计对象

参数 **Ports** (*Port*) -- 测试仪表端口对象, 类型为：object

关键字参数 **UpdateRoutesTransmitRate** (*int*) -- RIP Tx Rate (messages/sec) , 取值范围：1-1000000000, 默认值：1000

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Rip Port Config | Ports=${Ports} | UpdateRoutesTransmitRate=100 |
```

static edit_stream(*Stream*, ***kwargs*)

修改测试仪表流量模板参数

参数 **Stream** (StreamTemplate) --

关键字参数

- **RepeatCount** (*int*) -- 流模板发送重复次数，默认值：1，支持值：1-4294967295
- **EnableSignature** (*bool*) -- 启用签名，默认值：True
- **FrameLengthType** (*str*) -- 流模板帧长度类型类型为：string，默认值：FIXED，支持值：

  FIXED INCREMENT RANDOM AUTO DECREMENT IMIX
- **RandomLengthSeed** (*int*) -- 随机种子，类型为：number，默认值：10900842，支持值：0-4294967295
- **FixedLength** (*int*) -- 固定帧长，默认值：128，支持值：12-16383
- **MinLength** (*int*) -- 最小帧长，默认值：128，支持值：12-16383
- **MaxLength** (*int*) -- 最大帧长，默认值：256，支持值：12-16383
- **StepLength** (*int*) -- 帧长跳变步长，默认值：1，支持值：1-8192
- **PayloadType** (*str*) -- 净荷类型，默认值：CYCLE，支持值：

  CYCLE INCREMENT RANDOM
- **PayloadValue** (*str*) -- 帧长跳变步长，默认值：0x0
- **PayloadValueType** (*str*) -- 净荷类型，默认值：CYCLE，支持值：

  SINGLE_BYTE DOUBLE_BYTE
- **EnableNDResponse** (*bool*) -- 使用 ARP ND 自动回复，默认值：False
- **TopLayerType** (*str*) -- 流模板报文模板类型，默认值：IPV4，支持值：

  ETHERNETII VLAN GRE IPV4 IPV6
- **RxPorts** (list (*Port*)) -- 指定流量收端口
- **TrafficMeshMode** (*str*) -- binding 流参数，默认值：MANY_TO_MANY，支持值：

  ONE_TO_ONE MANY_TO_MANY FULL_MESH CONGESTION LEARNING BACK_BONE PAIR
- **HostsMesh** (*str*) -- binding 流参数，默认值：ROUND_ROBIN，支持值：

  ROUND_ROBIN MANY_TO_MANY

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Edit Stream | Stream=${Stream} | TopLayerType=ETHERNETII |␣
↪TrafficMeshMode=FULL_MESH |
```

static edit_stream_load_profile(*Streams*, ***kwargs*)

编辑测试仪表负载配置文件参数

参数 **Streams** (list(SreamTemplate)) -- 测试仪表流量对象列表，测试仪表流量对象 object 列表

:keyword : param Rate (int): 流量负载，默认值：10 :keyword : param Unit (int): 流量负载单位，默认值：PERCENT，取值范围：

PERCENT

FRAME_PER_SEC

BYTE_PER_SEC

LINEBIT_PER_SEC

KLINEBIT_PER_SEC

MLINEBIT_PER_SEC

INTER_FRAME_GAP

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Port Load Profile | Ports=${Ports} | LoadProfileType=STREAM_BASE |
| Edit Stream Load Profile | Streams=${Streams} | Rate=50 | Unit=FRAME_
↪PER_SEC |
```

static edit_traffic_parameters(*Wizard*, ***kwargs*)

配置 MPLS 流量

参数 **Wizard** (WizardConfig) -- wizard config

关键字参数

- **TrafficFlow** (*str*) -- traffic flow type, support: None FullyMeshedInVpn FullyMeshedInVpls Customer2Provider Provider2Customer CustomerProviderBoth
- **StreamBlockGrouping** (*str*) -- streamblock grouping type, support: Aggregate VPNAggregate NotAggregate
- **UseSingleStreamNumber** (*bool*) -- use single stream number
- **TrafficLoadPercentProvider** (*int*) -- traffic load percent provider
- **TrafficLoadPercentCustomer** (*bool*) -- traffic load percent customer

返回 True

返回类型 (bool)

引发 **exception.ContinuableFailure** --

static edit_vxlan(*Session*, ***kwargs*)

创建 Vxlan 协议会话对象

参数 **Session** (Vxlan) -- Vxlan 协议会话对象, 类型为：object

关键字参数

- **Name** (*str*) -- Vxlan 协会话名称, 类型为：string
- **Enable** (*bool*) -- 使能 Vxlan 协议会话, 默认值：True
- **AutoUdpSourcePort** (*bool*) -- 自动计算 UDP 源端口, 默认值：True
- **UdpSourcePort** (*int*) -- 配置 UDP 源端口, 取值范围：3-4095, 默认值：1025
- **EnableUdpChecksum** (*bool*) -- 使能计算 UDP 校验和, 默认值：False

- **EvpnLearning** (*bool*) -- 使能 EVPN 学习, 默认值：False
- **OvsdbLearning** (*bool*) -- 使能 OVSDB 学习, 默认值：False
- **MulticastType** (*str*) -- 组播类型, 默认值：IGMP, 取值范围：

  IGMP

  PIM

  MLD
- **VtepTunnelIp** (*str*) -- VTEP 隧道 IP 地址, 默认值：INTERFACEIP, 取值范围：

  INTERFACEIP

  ROUTERID
- **EnableIrb** (*bool*) -- 默认值：False
- **RPAddress** (*str*) -- 选择 PIM 的 RP 地址, 取值范围：IPv4 地址, 默认值：192.0.0.1
- **RPIpv6Address** (*str*) -- 选择 PIM 的 RP 地址, 取值范围：IPv6 地址, 默认值：2000::1
- **IrbMode** (*str*) -- 默认值：Symmetric, 取值范围：

  Symmetric

返回 Vxlan 协议会话对, 类型：object

返回类型 (Vxlan)

实际案例

```
| Create Vxlan | Port=${Port} |
```

static establish_bgp(*Sessions*)

建立 BGP 协议会话

参数 **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Establish Bgp | Sessions=${Sessions} |
```

static establish_ldp(*Sessions*)

建立 LDP 协议会话

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Establish Ldp | Sessions=${Sessions} |
```

static establish_ospf(*Sessions*)

建立 OSPFv2 协议会话

参数 **Sessions** (list(OspfRouter)) -- OSPFv2 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Establish Ospf | Sessions=${Sessions} |
```

static establish_ospfv3(*Sessions*)

建立 OSPFv3 协议会话

参数 **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Establish Ospfv3 | Sessions=${Sessions} |
```

static expand_benchmark(*Config*)

测试仪表生成测试仪表测试套件

参数 **Config** (wizard_config) -- 仪表测试测试套件对象 object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Items} | Create List | throughput | frameloss |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items}
↪ |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} |␣
↪Mode=meshed | Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=${L2_TestTime} |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{L2_
↪FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

static expand_mpls_wizard(*Wizard*)

生成测试仪表 MPLS 向导配置

参数 **Wizard** (WizardConfig) -- wizard config

返回 bool：布尔值 (范围：True / False)

实际案例

```
| Expand Mpls Wizard | Wizard=${Wizard} |
```

static export_benchmark_result(*Result*, *Path*, *Sheet*)

将测试套件的结果导出到 excel 表格

参数

- **Result** (*list*) -- 测试套件直接结果 DB 文件中获取指定测试结果数据列表
- **Path** (*str*) -- 导出文件的路径, (例如：”C:/Report.xls”)
- **Sheet** -- (str) 导入的 excel 文件的 sheet 页名称, (例如：”ipv4 natp”)

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${result} | Export Benchmark Result | ${Result} | ${Path} | ${Sheet} |
```

static format_benchmark_result(*Result*)

格式化列表为二维表格形式

参数 **Result** (*list*) -- 测试套件直接结果 DB 文件中获取指定测试结果数据列表

返回 PrettyTable 对象

返回类型 (PrettyTable)

实际案例

robotframework:

```
| ${result} | Format Benchmark Result | ${Result} |
```

static get_benchmark_result(*DB*, *Type*, *Item*, *FrameSize=None*, *Mbps=None*, *All=False*)

从测试套件执行结果 DB 文件中获取指定测试结果数据

参数

- **DB** (*str*) -- 测试结果 DB 文件的绝对路径, (例如: ”C:/TestSuite/Benchmark/2021_07_29_21_10_36/Asymmetric_throughput_summary2021-07-29_21-11-08/Asymmetric_throughput_summary_2021-07-29_21-11-08.db”)
- **Type** (*str*) -- 测试套件类型, (取值范围：RFC2544 / Asymmetric / RFC2889 / RFC3918)

- **Item** (*str*) -- 测试套件中的测试项目, (取值范围：Throughput / Latency / FrameLoss)
- **FrameSize** (*list*) -- 测试套件测试帧长, (取值范围：[64, 128, 256, 512, 1024. 1280, 1518])

返回 测试套件直接结果 DB 文件中获取指定测试结果数据列表

返回类型 list

实际案例

robotframework:

```
| ${DB}  == "C:/TestSuite/Benchmark/2021_07_29_21_10_36/Asymmetric_
↪throughput_summary2021-07-29_21-11-08/Asymmetric_throughput_summary_
↪2021-07-29_21-11-08.db") |
| ${Type} == "Asymmetric" |
| ${Item} == "Throughput" |
| ${FrameSize} == [64, 128, 256, 512, 1024. 1280, 1518] |
| ${result} | Get Benchmark Result | ${DB} | ${Type} | ${Item} | $
↪{FrameSize} |
```

static get_bfd_ipv4_session_result(*Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 BFD IPV4 会话统计结果

参数

- **Session** (BfdIpv4SessionConfig) -- BFD IPV4 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  Ipv4SessionKeyID

  Ipv4SessionID

  SessionID

  SessionIndex

  Ipv4SourceAddress

  Ipv4DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

  BfdControlBits_PFCADM

  ReceiveCount

  TransmitCount

  TransmitInterval

  ReceivedRequiredMinRXInterval

  ReceivedRequiredMinEchoRXInterval

FlapsDetected

TimeoutsDetected

RXAvgRate

RXMaxRate

RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BfdIpv4SessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Ipv4 Session Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_bfd_ipv6_session_result(*Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 BFD IPV6 会话统计结果

参数

- **Session** (BfdIpv6SessionConfig) -- BFD IPV6 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  Ipv6SessionKeyID

  Ipv6SessionID

  SessionID

  SessionIndex

  Ipv6SourceAddress

  Ipv6DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

BfdDiagnostic

LastBfdDiagnosticErrorRx

BfdControlBits_PFCADM

ReceiveCount

TransmitCount

TransmitInterval

ReceivedRequiredMinRXInterval

ReceivedRequiredMinEchoRXInterval

FlapsDetected

TimeoutsDetected

RXAvgRate

RXMaxRate

RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BfdIpv6SessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Ipv6 Session Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_bfd_isis_ipv6_session_result(*BfdSession*, *IsisSession*, *SessionId*, *StaItems: Optional[list] = None*)

获取 ISIS BFD IPV6 会话统计结果

参数

- **BfdSession** (BfdRouter) -- BFD 会话对象, 类型为：Object
- **IsisSession** (IsisRouter) -- ISIS 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  IpSessionKeyID

IpSessionID

SessionID

SessionIndex

Ipv6SourceAddress

Ipv6DestinationAddress

BfdSessionState

MyDiscriminator

YourDiscriminator

BfdDiagnostic

LastBfdDiagnosticErrorRx

BfdControlBits_PFCADM

ReceiveCount

TransmitCount

TransmitInterval

ReceivedRequiredMinRXInterval

ReceivedRequiredMinEchoRXInterval

FlapsDetected

TimeoutsDetected

RXAvgRate

RXMaxRate

RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=IsisBfdIpv6SessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Isis Ipv6 Session Statistic | BfdSession=$
↪{BfdSession} | IsisSession=${IsisSession} | SessionId=${SessionId} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_bfd_isis_session_result(*BfdSession*, *IsisSession*, *SessionId*, *StaItems: Optional[list] = None*)

获取 ISIS BFD 会话统计结果

参数

- **BfdSession** (BfdRouter) -- BFD 会话对象, 类型为：Object
- **IsisSession** (IsisRouter) -- ISIS 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  IpSessionKeyID

  IpSessionID

  SessionID

  SessionIndex

  Ipv4SourceAddress

  Ipv4DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

  BfdControlBits_PFCADM

  ReceiveCount

  TransmitCount

  TransmitInterval

  ReceivedRequiredMinRXInterval

  ReceivedRequiredMinEchoRXInterval

  FlapsDetected

  TimeoutsDetected

  RXAvgRate

  RXMaxRate

  RXMinRate

  TXAvgRate

  TXMaxRate

  TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=IsisBfdSessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Isis Session Statistic | BfdSession=${BfdSession} |␣
↪IsisSession=${IsisSession} | SessionId=${SessionId} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_bfd_ospfv2_session_result(*BfdSession*, *Ospfv2Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 OSPFV2 BFD 会话统计结果

参数

- **Ospfv2Session** (BfdRouter) -- OSPFv2 会话对象, 类型为：Object
- **BfdSession** (OspfRouter) -- BFD 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  Ipv4SessionKeyID

  Ipv4SessionID

  SessionID

  SessionIndex

  Ipv4SourceAddress

  Ipv4DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

  BfdControlBits_PFCADM

  ReceiveCount

  TransmitCount

  TransmitInterval

  ReceivedRequiredMinRXInterval

  ReceivedRequiredMinEchoRXInterval

  FlapsDetected

  TimeoutsDetected

  RXAvgRate

  RXMaxRate

  RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv2BfdSessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Ospfv2 Session Statistic | BfdSession=${BfdSession}
↪| Ospfv2Session=${Ospfv2Session} | SessionId=${SessionId} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_bfd_ospfv3_session_result(*BfdSession*, *Ospfv3Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 OSPFV3 BFD 会话统计结果

参数

- **BfdSession** (BfdRouter) -- BFD 会话对象, 类型为：Object
- **Ospfv3Session** (Ospfv3Router) -- OSPFv3 会话对象, 类型为：Object
- **SessionId** (*str*) -- BFD 会话的索引号, 类型为：string
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  Ipv6SessionKeyID

  Ipv6SessionID

  SessionID

  SessionIndex

  Ipv6SourceAddress

  Ipv6DestinationAddress

  BfdSessionState

  MyDiscriminator

  YourDiscriminator

  BfdDiagnostic

  LastBfdDiagnosticErrorRx

  BfdControlBits_PFCADM

  ReceiveCount

TransmitCount

TransmitInterval

ReceivedRequiredMinRXInterval

ReceivedRequiredMinEchoRXInterval

FlapsDetected

TimeoutsDetected

RXAvgRate

RXMaxRate

RXMinRate

TXAvgRate

TXMaxRate

TXMinRate

返回

eg:

```
{
    'TXAvgRate': 10,
    'RXAvgRate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv3BfdSessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Ospfv3 Session Statistic | BfdSession=${BfdSession}
↪| Ospfv3Session=${Ospfv3Session} | SessionId=${SessionId} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_bfd_session_result(*Session*, *StaItems: Optional[list] = None*)

获取 BFD 协议会话统计结果

参数

- **Session** (BfdRouter) -- BFD 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项：

  SessionID

  SessionState

  BfdSessionUpCount

  BfdSessionDownCount

  TXBfdPackets

  RXBfdPackets

  TimeoutsDetected

FlapsDetected

返回

eg:

```
{
    'TXBfdPackets': 10,
    'RXBfdPackets': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BfdSessionResult |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bfd Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_bgp_evpn_routes_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 Bgp Evpn Routes 统计结果

参数

- **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxAdRouteCount

  RxAdRouteCount

  TxMacIpRouteCount

  RxMacIpRouteCount

  TxInclusiveMcastRouteCount

  RxInclusiveMcastRouteCount

  TxEthernetSegmentRouteCount

  RxEthernetSegmentRouteCount

  TxIpPrefixRouteCount

  RxIpPrefixRouteCount

  TxWithdrawnAdRouteCount

  RxWithdrawnAdRouteCount

  TxWithdrawnMacIpRouteCount

  RxWithdrawnMacIpRouteCount

  TxWithdrawnInclusiveMcastRouteCount

  RxWithdrawnInclusiveMcastRouteCount

  TxWithdrawnEthernetSegmentRouteCount

  RxWithdrawnEthernetSegmentRouteCount

TxWithdrawnIpPrefixRouteCount

RxWithdrawnIpPrefixRouteCount

返回

eg:

```
{
    'TxAdRouteCount': 10,
    'RxAdRouteCount': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=EvpnRoutesStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bgp Evpn Routes Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_bgp_link_state_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 Bgp Evpn Routes 统计结果

参数

- **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxAdvertisedNodeCount

  RxAdvertisedNodeCount

  TxWithdrawnNodeCount

  RxWithdrawnNodeCount

  TxAdvertisedLinkCount

  RxAdvertisedLinkCount

  TxWithdrawnLinkCount

  RxWithdrawnLinkCount

  TxAdvertisedIpv4PrefixCount

  RxAdvertisedIpv4PrefixCount

  TxWithdrawnIpv4PrefixCount

  RxWithdrawnIpv4PrefixCount

  TxAdvertisedIpv6PrefixCount

  RxAdvertisedIpv6PrefixCount

  TxWithdrawnIpv6PrefixCount

  RxWithdrawnIpv6PrefixCount

  TxAdvertisedSrv6SidCount

RxAdvertisedSrv6SidCount

TxWithdrawnSrv6SidCount

RxWithdrawnSrv6SidCount

返回

eg:

```
{
    'TxAdRouteCount': 10,
    'RxAdRouteCount': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BgpLinkStateStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bgp Link State Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_bgp_router_from_route_pool(*Configs*, *Type='ipv4'*)

获取 BGP Route Pool 对应的绑定流源或目的端点对象

参数

- **Configs** (list(BgpIpv4RoutepoolConfig)) -- 测试仪表 BGP Route Pool 对象列表
- **Type** (*str*) -- Route Pool 类型支持 ipv4 和 ipv6

返回 BGP Route Pool 对应的绑定流源或目的端点对象列表

返回类型 (list(BgpIpv4RoutepoolConfig))

实际案例

```
| ${Session} | Create Bgp | Port=${Port} |
| ${RouterPool} | Create Bgp Ipv4 Route Pool | Session=${Session} |
| ${Point} | Get Router From Route Pool | Configs=${RouterPool} |
```

static get_bgp_session_block_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 Bgp Session Block 统计结果

参数

- **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  PeerState

  TxOpen

  RxOpen

TxKeepalive

RxKeepalive

TxUpdate

RxUpdate

TxAdvertisedUpdate

RxAdvertisedUpdate

TxWithdrawnUpdate

RxWithdrawnUpdate

TxAdvertisedRoutes

RxAdvertisedRoutes

TxWithdrawnRoutes

RxWithdrawnRoutes

LastTxUpdateRoutes

LastRxUpdateRoutes

TxNotification

RxNotification

TxRefresh

RxRefresh

返回

eg:

```
{
    'TxAdRouteCount': 10,
    'RxAdRouteCount': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BgpSessionBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bgp Session Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_bgp_session_statistic(*Session*, *Id*, *StaItems=None*)

获取 Bgp Session 统计结果

参数

- **Session** (BgpRouter) -- BGP 协议会话对象, 类型为：Object
- **Id** (*int*) -- Bgp Peer Id, 类型为：number
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  PeerState

TxOpen

RxOpen

TxKeepalive

RxKeepalive

TxUpdate

RxUpdate

TxAdvertisedUpdate

RxAdvertisedUpdate

TxWithdrawnUpdate

RxWithdrawnUpdate

TxAdvertisedRoutes

RxAdvertisedRoutes

TxWithdrawnRoutes

RxWithdrawnRoutes

LastTxUpdateRoutes

LastRxUpdateRoutes

TxNotification

RxNotification

TxRefresh

RxRefresh

返回

eg:

```
{
    'TxAdRouteCount': 10,
    'RxAdRouteCount': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=BgpSessionStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Bgp Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_capture_info(*Port*, *Items=None*)

在指定端口报文捕获信息

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为: object

- **Items** (*list*) -- 端口报文捕获信息, 支持参数:CaptureState ElapsedTime CapturedPacketCount BufferFull DownloadedPacketCount CurrentDataFile

返回类型 dict

实际案例

robotframework:

```
| Get Capture Info | Port=${Port} |
```

static get_config_children(*Configs*, *Children*)

static get_configs(*Configs=None*, *KeyType='handle'*, *Upper=None*)

获取测试仪表指定对象

参数

- **Configs** -- 测试仪表端口对象类型列表, 类型为：list
- **KeyType** -- 返回字典使用指定类型作为字典的 key，支持：handle、name
- **Upper** -- 指定上层节点获取对象

返回 object} 或者 {'name': object}

返回类型 字典 {'handle'

实际案例

robotframework:

```
| ${Result} | Get Configs | KeyType=name |
| ${Result} | Get Configs | Configs=StreamTemplate | KeyType=handle |
| ${Result} | Get Configs | Configs=BgpProtocolConfig | KeyType=name |
| ${Result} | Get Configs | Configs=StreamTemplate | KeyType=handle |␣
↪Upper=${Port_1} |
| ${Result} | Get Configs | Configs=BgpProtocolConfig | KeyType=name |␣
↪Upper=${Port_1} |
```

static get_dhcp_client_block_statistic(*Session*, *StaItems=None*)

获取 Dhcp Client Block Statistic 统计结果

参数

- **Session** (DhcpClient) -- Dhcp 客户端会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  BlockState

  AttemptRate

  SetupRate

  CurrentAttempt

  CurrentBound

  TotalAttempt

  TotalBound

  TotalFailed

TotalReboot

TotalRenew

TotalRebind

TotalRetry

TxDiscover

RxOffer

TxRequest

RxAck

RxNak

TxRenew

TxRebind

TxReboot

TxRelease

TxDecline

RxForceRenew

返回

eg:

```
{
    'CurrentAttempt': 10,
    'CurrentBound': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | CurrentAttempt | CurrentBound |
| Subscribe Result | Types=Dhcpv4ClientBlockStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Client Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_dhcp_client_statistic(*Session*, *Id=1*, *StaItems=None*)

获取 Dhcp Client Statistic 统计结果

参数

- **Session** (DhcpClient) -- Dhcp 客户端会话对象, 类型为：Object
- **Id** (*int*) -- Dhcp 客户端会话 Index, 默认值: 1
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  ClientState

  IpAddress

  LeaseTime

  LeaseLeft

ErrorStatus

DiscoverResponseTime

RequestResponseTime

返回

eg:

```
{
    'LeaseTime': 10,
    'DiscoverResponseTime': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | LeaseTime | DiscoverResponseTime |
| Subscribe Result | Types=Dhcpv4ClientStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Client Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_dhcp_port_statistic(*Port*, *StaItems=None*)

获取 Dhcp Port Statistic 统计结果

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  CurrentAttempt

  CurrentBound

  TotalAttempt

  TotalBound

  TotalFailed

  TotalReboot

  TotalRenew

  TotalRebind

  TotalRetry

  TxDiscover

  RxOffer

  TxRequest

  RxAck

  RxNak

  TxRenew

  TxRebind

  TxReboot

TxRelease

RxForceRenew

返回

eg:

```
{
    'CurrentAttempt': 10,
    'CurrentBound': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | CurrentAttempt | CurrentBound |
| Subscribe Result | Types=Dhcpv4PortStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Port Statistic | Port=${Port} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_dhcp_server_lease_statistic(*Session*, *ClientId*, *StaItems=None*)

获取 Dhcp Server Lease Statistic 统计结果

参数

- **Session** (DhcpServer) -- DHCP 服务端会话对象, 类型为：object
- **ClientId** (*str*) -- DHCP Client Mac Address
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  ClientIp

  LeaseTime

  LeaseLeft

返回

eg:

```
{
    'LeaseTime': 100,
    'LeaseLeft': 50,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | LeaseTime | LeaseLeft |
| Subscribe Result | Types=Dhcpv4ServerStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Server Lease Statistic | Session=${Session} |␣
↪ClientId=00:00:12:01:01:03 | StaItems=@{StaItems} |
| Clear Result |
```

static get_dhcp_server_statistic(*Session*, *StaItems=None*)

获取 Dhcp Server Statistic 统计结果

参数

- **Session** (DhcpServer) -- DHCP 服务端会话对象, 类型为：object / list
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  CurrentBound

  TotalBound

  TotalExpire

  TotalReboot

  TotalRenew

  TotalRebind

  TotalRelease

  RxDiscover

  TxOffer

  RxRequest

  TxAck

  TxNak

  RxDecline

  RxRelease

  TxForceRenew

返回

eg:

```
{
    'CurrentBound': 10,
    'TotalBound': 20,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | CurrentBound | TotalBound |
| Subscribe Result | Types=Dhcpv4ServerStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcp Server Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_dhcpv6_client_block_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Client Block Statistic 统计结果

参数

- **Session** (Dhcpv6Client) -- Dhcpv6 客户端会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  BlockSessionState

  CurrentlyAttempting

  CurrentlyIdl

  CurrentlyBound

  AttemptRate

  BindRate

  RebindRate

  ReleaseRate

  RenewRate

  AverageRebindToReplyTime

  AverageReleaseToReplyTime

  AverageRenewToReplyTime

  AverageRequestToReplyTime

  AverageSolicitToAdvertiseTime

  AverageSolicitToReplyTime

  MaxRebindToReplyTime

  MaxReleaseToReplyTime

  MaxRenewToReplyTime

  MaxRequestToReplyTime

  MaxSolicitToAdvertiseTime

  MaxSolicitToReplyTime

  MinRebindToReplyTime

  MinReleaseToReplyTime

  MinRenewToReplyTime

  MinRequestToReplyTime

  MinSolicitToAdvertiseTime

MinSolicitToReplyTime

AdvertiseRxCount

ReplyRxCount

ReconfigureRxCount

SolicitTxCount

RequestTxCount

ReleaseTxCount

RenewTxCount

RebindTxCount

ConfirmTxCount

InfoRequestTxCount

TotalAttempted

TotalBound

TotalFailed

TotalRebound

TotalReleased

TotalReleaseRetried

TotalRenewed

TotalRenewedRetried

TotalRetired

返回

eg:

```
{
    'TotalRenewedRetried': 10,
    'TotalRetired': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ClientBlockStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Client Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_dhcpv6_client_statistic(*Session*, *Id*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Client Statistic 统计结果

参数

- **Session** (Dhcpv6Client) -- Dhcpv6 客户端会话对象, 类型为：Object
- **Id** (*int*) -- Dhcpv6 客户端会话 Index

- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  IaidValue

  MacAddr

  LeaseRx

  AddressType

  SessionState

  StateCode

  IpAddress

  LeaseRemaining

  PrefixLength

  RequestResponseTime

  SolicitResponseTime

返回

eg:

```
{
    'RequestResponseTime': 10,
    'SolicitResponseTime': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ClientStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Client Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_dhcpv6_pd_client_statistic(*Session*, *Id*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 pd Client Statistic 统计结果

参数

- **Session** (Dhcpv6Client) -- Dhcpv6 客户端会话对象, 类型为：Object
- **Id** (*int*) -- Dhcpv6 客户端会话 Index
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  Dhcpv6PdClientId

  IaidValue

  SessionIndex

  MacAddr

  VlanId

LeaseRx

AddressType

SessionState

StateCode

IpAddress

LeaseRemaining

PrefixLength

RequestResponseTime

SolicitResponseTime

返回

eg:

```
{
    'RequestResponseTime': 10,
    'SolicitResponseTime': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ClientStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Client Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_dhcpv6_port_statistic(*Port*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Port Statistic 统计结果

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  CurrentlyAttempting

  CurrentlyIdl

  CurrentlyBound

  AverageSetupTime

  MaxSetupTime

  MinSetupTime

  SolicitTxCount

  RequestTxCount

  ReleaseTxCount

  RenewTxCount

  RebindTxCount

ConfirmTxCount

InfoRequestTxCount

AdvertiseRxCount

ReconfigureRxCount

ReplyRxCount

SuccessPercentage

TotalAttempted

TotalBound

TotalBoundFailed

TotalRebound

TotalReleased

TotalReleaseRetried

TotalRenewed

TotalRenewedRetried

TotalRetired

返回

eg:

```
{
    'TotalRenewedRetried': 10,
    'TotalRetired': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6PortStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Port Statistic | Port=${Port} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_dhcpv6_server_lease_statistic(*Session*, *Pool*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Server Lease Statistic 统计结果

参数

- **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list
- **Pool** (Dhcpv6AddressPoolsConfig) -- DHCPv6 Server Address Pool 对象
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  ServerState

  CurrentlyBound

ReconfigureRebindTxCount

ReconfigureRenewTxCount

ReconfigureTxCount

AdvertiseTxCount

ReplyTxCount

SolicitRxCount

RequestRxCount

ReleaseRxCount

RenewRxCount

RebindRxCount

TotalBound

TotalExpired

TotalReleased

TotalRenewed

返回

eg:

```
{
    'TotalReleased': 10,
    'TotalRenewed': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ServerStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Server Lease Statistic | Session=${Session} |␣
↪Pool=${Pool} | StaItems=@{StaItems} |
| Clear Result |
```

static get_dhcpv6_server_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Server Statistic 统计结果

参数

- **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  ServerState

  CurrentlyBound

  ReconfigureRebindTxCount

  ReconfigureRenewTxCount

  ReconfigureTxCount

AdvertiseTxCount

ReplyTxCount

SolicitRxCount

RequestRxCount

ReleaseRxCount

RenewRxCount

RebindRxCount

TotalBound

TotalExpired

TotalReleased

TotalRenewed

返回

eg:

```
{
    'TotalReleased': 10,
    'TotalRenewed': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dhcpv6ServerStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dhcpv6 Server Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_dot1x_block_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 802.1x session block 统计结果

参数

- **Session** (Dot1x) -- 802.1x 会话对象, 类型为：object / list
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  BlockState

  CurrentAuthenticatedAttempt

  CurrentAuthenticated

  CurrentFailed

  CurrentLogoff

  AuthenticatedAttemptRate

  AuthenticatedRate

  LogoffRate

  TotalAttempt

TotalAuthenticated

TotalFailed

TotalLogoff

TotalRetry

TotalRetransmit

RxEapFailure

RxEapRequest

RxEapSucess

TxEapResponse

MaxAuthenticatedTime

MaxLogoffTime

返回

eg:

```
{
    'MaxAuthenticatedTime': 10,
    'MaxLogoffTime': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dot1xBlockStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dot1x Block Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_dot1x_port_statistic(*Port*, *StaItems: Optional[list] = None*)

获取 802.1x port block 统计结果

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  CurrentAuthenticatedAttempt

  CurrentAuthenticated

  CurrentFailed

  CurrentLogoff

  TotalAttempt

  TotalAuthenticated

  TotalFailed

  TotalLogoff

  TotalRetry

TotalRetransmit

返回

eg:

```
{
    'TotalRetry': 10,
    'TotalRetransmit': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dot1xPortStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dot1x Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_dot1x_statistic(*Session*, *Index*, *StaItems: Optional[list] = None*)

获取 802.1x 统计结果

参数

- **Session** (Dot1x) -- 802.1x 会话对象, 类型为：object / list
- **Index** (*int*) -- Session Index
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  State

  ReqIdentity

  RespIdentity

  ReqChallenge

  RespChallenge

  TLSEstablish

  ReceiveOK

  ReceiveFail

返回

eg:

```
{
    'ReceiveOK': 10,
    'ReceiveFail': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Dot1xStatistics |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Dot1x Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_gateway_mac(*Interface*)

获取测试仪表学习到的网关 Mac 地址

参数 **Interface** (Interface) -- 测试仪表接口对象

返回 Mac 地址列表 List

返回类型 list

实际案例

robotframework:

```
| Get Gateway Mac | Interface=${Interface} |
```

static get_igmp_host_statistic(*Session*, *StaItems=None*)

获取 Igmp 协议会话统计结果

参数

- **Session** (Igmp) -- Igmp 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  IgmpTxFrames

  IgmpRxFrames

  IgmpRxUnknownTypes

  IgmpRxChecksumErrors

  IgmpRxLengthErrors

返回

eg:

```
{
    'IgmpTxFrames': 8,
    'IgmpRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | IgmpTxFrames | IgmpRxFrames |
| Subscribe Result | Types=IgmpHostResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Igmp Host Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_igmp_port_statistic(*Port*, *StaItems=None*)

获取 Igmp Port 统计结果

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  IgmpTxFrames

  IgmpRxFrames

  IgmpTxV1Reports

  IgmpTxV2Reports

  IgmpTxLeaveGroups

  IgmpTxV3Reports

  IgmpTxV3ModeInclude

  IgmpTxV3ModeExclude

  IgmpTxV3ModeChangeToInclude

  IgmpTxV3ModeChangeToExclude

  IgmpTxV3ModeAllowNewSources

  IgmpTxV3ModeBlockOldSources

  IgmpRxV1Queries

  IgmpRxV2Queries

  IgmpRxV3Queries

  IgmpRxGeneralQueries

  IgmpRxGroupSpecificQueries

  IgmpRxGroupAndSourceSpecificQueries

  IgmpRxUnknownTypes

  IgmpRxChecksumErrors

  IgmpRxLengthErrors

返回

eg:

```
{
    'IgmpTxFrames': 8,
    'IgmpRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | IgmpTxFrames | IgmpRxFrames |
| Subscribe Result | Types=IgmpPortAggregatedResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Igmp Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_igmp_querier_statistic(*Session*, *StaItems=None*)

获取 Igmp Querier 协议会话统计结果

参数

- **Session** (IgmpQuerier) -- Igmp 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  QuerierTxFrames

  QuerierRxFrames

  QuerierRxUnknownTypes

  QuerierRxChecksumErrors

  QuerierRxLengthErrors

返回

eg:

```
{
    'QuerierTxFrames': 8,
    'QuerierRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | QuerierTxFrames | QuerierRxFrames |
| Subscribe Result | Types=IgmpQuerierResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Igmp Querier Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_imix_from_name(*Name*)

通过 Imix 模板名称获取流量 Imix 模板对象

参数 **Name** (*str*) -- 创建的 Imix 模板名称

返回 Imix 模板对象

返回类型 (Imix)

实际案例

```
| ${Imix_TCPv4} | Get Imix From Name | Name=TCPv4 |
```

static get_interfaces(*Ports=None*, *Types=None*)

获取测试仪表学习到的网关 Mac 地址

参数 **Ports** -- 测试仪表接口对象 object

Returns: 测试仪表接口对象列表 List

Examples: robotframework:

```
| Get Gateway Mac | Interface=${Interface} |
```

static get_isis_mt_params(*Session*, *Index=0*)

获取 ISIS 协议会话 MT 参数

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Index** (*int*) -- ISIS 协议会话 MT 参数对象序号，类型为：number, 取值范围：0-1, 默认值：0

返回

eg:

```
{
    'MtId': 'IPV4',
    'MtFlags': '['ABIT', 'OBIT'],
}
```

返回类型 dict

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True | MtParams=2 |
| Edit Isis Mt Params | Session=${Session} | Index=0 | MtId=IPv4 |␣
↪MtFlags=${{MtFlags} |
| Edit Isis Mt Params | Session=${Session} | Index=1 | MtId=IPv6 |␣
↪MtFlags=${{MtFlags} |
| Get Isis Mt Params | Session=${Session} | Index=0 |
| Get Isis Mt Params | Session=${Session} | Index=1 |
```

static get_isis_per_pdu(*Session*, *Index=0*)

获取 ISIS 协议会话 Per Pdu Authentication 参数

参数

- **Session** (IsisRouter) -- ISIS 协议会话对象列表, 类型为：object
- **Index** (*int*) -- ISIS 协议会话 Per Pdu Authentication 参数对象序号，类型为：number, 取值范围：0-4, 默认值：0

返回

eg:

```
{
    'PdusType': 'L1_HELLO',
    'AuthMethod': 'SIMPLE',
    'Password': 'Xinertel',
}
```

返回类型 dict

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| ${Session} | Create Isis | Port=${Port} |
| ${{MtFlags} | Create List |  ABIT | OBIT |
| Edit Isis | Session=${Session} | EnableViewRoutes=True |␣
↪PerPduAuthentication=1 |
| Edit Isis Per Pdu Authentication | Session=${Session} | PdusType=L2_
↪HELLO | AuthMethod=SIMPLE | Password=Test |
| Get Isis Per Pdu Authentication | Session=${Session} | Index=0 |
```

static get_isis_router_from_tlv(*Configs*)

获取 ISIS TLV 对应的绑定流源或目的端点对象

参数 **Configs** (list(IsisIpv4TlvConfig, *IsisIpv6TlvConfig*)) -- 测试仪表 ISIS TLV 对象列表, 类型为：list

返回 ISIS TLV 对应的绑定流源或目的端点对象列表

返回类型 (list(IsisIpv4Router)

实际案例

```
| ${Session} | Create Isis | Port=${Port} |
| ${LSP} | Create Isis Lsp | Session=${Session} |␣
↪SystemId=00:00:00:00:00:02 |
| ${TLV} | Create Isis Ipv4 Tlv | Lsp=${LSP} | SystemId=00:00:00:00:00:02␣
↪|
| Get Isis Router From Tlv | Configs=${TLV} |
```

static get_isis_session_stats(*Session*, *StaItems=None*)

获取 Isis Session 统计结果

参数

- **Session** (IsisRouter) -- Isis 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxP2pHello

  RxP2pHello

  TxLanL1Hello

  RxLanL1Hello

  TxLanL2Hello

  RxLanL2Hello

  TxL1Lsp

RxL1Lsp

TxL2Lsp

RxL2Lsp

TxL1Csnp

RxL1Csnp

TxL2Csnp

RxL2Csnp

TxL1Psnp

RxL1Psnp

TxL2Psnp

RxL2Psnp

返回

eg:

```
{
    'TxL1Lsp': 10,
    'RxL1Lsp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=IsisSessionStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Isis Session Stats | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_isis_tlv_stats(*Session*, *StaItems=None*)

获取 Isis Session 统计结果

参数

- **Session** (IsisRouter) -- Isis 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxPrefixSid

  RxPrefixSid

  TxAdjSid

  RxAdjSid

  TxLanAdjSid

  RxLanAdjSid

  TxSidBinding

  RxSidBinding

  TxSrv6Loc

RxSrv6Loc

TxSrv6EndX

RxSrv6EndX

TxSrv6LanEndX

RxSrv6LanEndX

返回

eg:

```
{
    'TxPrefixSid': 10,
    'RxPrefixSid': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=IsisTlvStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Isis Tlv Stats | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_l2tp_block_statistic(*Session*, *StaItems=None*)

获取 L2tp Block Statistic 统计结果

参数

- **Session** (L2tp) -- L2tp 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TunnelCount

  SessionCount

  TunnelUp

  TunnelDown

  SessionUp

  SessionDown

  TunnelSetupRate

  SessionSetupRate

  AverageTunnelSetupTime

  MaxTunnelSetupTime

  MinTunnelSetupTime

  AverageSessionSetupTime

  MaxSessionSetupTime

  MinSessionSetupTime

  TxPackets

RxPackets

TxSccrq

RxSccrq

TxSccrp

RxSccrp

TxScccn

RxScccn

TxIcrq

RxIcrq

TxIcrp

RxIcrp

TxIccn

RxIccn

TxSli

RxSli

TxStopCcn

RxStopCcn

TxWen

RxWen

TxHello

RxHello

TxCdn

RxCdn

TxZlb

RxZlb

返回

eg:

```
{
    'TxZlb': 10,
    'RxZlb': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=L2tpBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get L2tp Block Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_l2tp_port_statistic(*Port*, *StaItems=None*)

获取 L2tp Session Statistic 统计结果

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  LacCount

  LnsCount

  TunnelCount

  SessionCount

  TunnelUp

  TunnelDown

  SessionUp

  SessionDown

返回

eg:

```
{
    'TunnelUp': 10,
    'SessionUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=L2tpPortStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get L2tp Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_l2tp_session_statistic(*Session*, *NodeIndexInBlock*, *StaItems=None*)

获取 L2tp Session Statistic 统计结果

参数

- **Session** (L2tp) -- L2tp 协议会话对象, 类型为：Object

- **NodeIndexInBlock** (*int*) -- Session Index, 类型为：int
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  LocalTunnelId

  RemoteTunnelId

  LocalSessionId

  RemoteSessionId

  SessionState

  LocalTunnelIpAddress

  RemoteTunnelIpAddress

  LocalTunnelIpv6Address

  RemoteTunnelIpv6Address

  TxIcrq

  RxIcrq

  TxIcrp

  RxIcrp

  TxIccn

  RxIccn

  TxCdn

  RxCdn

  ResultCode

  ErrorCode

  ErrorMessage

返回

eg:

```
{
    'TxIcrq': 10,
    'RxIcrq': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=L2tpSessionStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get L2tp Session Statistic | Session=${Session} |␣
↪NodeIndexInBlock=1 | StaItems=@{StaItems} |
| Clear Result |
```

static get_l2tp_tunnel_statistic(*Session*, *NodeIndexInBlock*, *StaItems=None*)

获取 L2tp Tunnel Statistic 统计结果

参数

- **Session** (L2tp) -- L2tp 协议会话对象, 类型为：Object
- **NodeIndexInBlock** (*int*) -- Session Index, 类型为：int
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  LocalTunnelId

  RemoteTunnelId

  TunnelState

  UdpSourcePort

  UdpDestinationPort

  LocalIpAddress

  RemoteIpAddress

  LocalIpv6Address

  RemoteIpv6Address

  SessionCount

  SessionUp

  SessionDown

  TxPackets

  RxPackets

  TxSccrq

  RxSccrq

  TxSccrp

  RxSccrp

  TxScccn

  RxScccn

  TxSli

  RxSli

  TxStopCcn

  RxStopCcn

  TxWen

  RxWen

  TxHello

  RxHello

返回

eg:

```
{
    'TxHello': 10,
    'RxHello': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=L2tpTunnelStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get L2tp Tunnel Statistic | Session=${Session} |␣
↪NodeIndexInBlock=1 | StaItems=@{StaItems} |
| Clear Result |
```

static get_layer_from_interfaces(*Interfaces*, *Layer='ipv4'*)

获取测试仪表接口的封装层对象

:param：param Interfaces: 测试仪表接口对象列表:param：type Interfaces: 类型为: list

Returns: 测试仪表接口的封装层对象列表 List

Examples: robotframework:

```
| Get Layer From Interfaces | Interfaces=${Interface} | Layer=ipv4 |
```

static get_ldp_lsp_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 Ldp Lsp 统计结果

参数

- **Session** (Ldp) -- LDP 会话对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  FECInfo

  FECType

  LabelValue

  LspMode

  LspState

  LspType

返回

eg:

```
{
    'LabelValue': 16,
    'LspMode': DU,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LdpLspStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Ldp Lsp Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_ldp_point_from_lsp(*Configs*)

获取 LDP LSP 对应的绑定流源或目的端点对象

参数 **Configs** (*list*) -- 测试仪表 LDP LSP 对象列表, 类型为：list

返回 LDP LSP 对应的绑定流源或目的端点对象列表, 类型：list

返回类型 (LdpIpv4EgressLspConfig, *LdpIpv4IngressLspConfig*, *LdpFec128LspConfig*, *LdpFec129LspConfig*)

实际案例

```
| Get Ldp Point From Lsp | Configs=${IPv4EgressLsp} |
```

static get_ldp_session_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 Ldp Session 统计结果

参数

- **Session** (Ldp) -- LDP 会话对象, 类型为：object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxAddressWithdraw

  RxAddressWithdraw

  TxAddress

  RxAddress

  TxDirectHello

  RxDirectHello

  TxInitialization

  RxInitialization

  TxKeepAlive

  RxKeepAlive

  TxLabelAbort

  RxLabelAbort

  TxLabelMapping

  RxLabelMapping

  TxLabelRelease

  RxLabelRelease

  TxLabelRequest

RxLabelRequest

TxLabelWithdraw

RxLabelWithdraw

TxNotification

RxNotification

TxTargetHello

RxTargetHello

TxIPv6DirectHello

RxIPv6DirectHello

TxIPv6TargetHello

RxIPv6TargetHello

返回

eg:

```
{
    'TxIPv6TargetHello': 10,
    'RxIPv6TargetHello': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LdpSessionStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Ldp Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_lsp_ping_echo_request_statistic(*Session*, *EchoRequest*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Port Statistic 统计结果

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object
- **EchoRequest** (LspPingEchoRequestConfig) -- Lsp Ping Echo Request 对象, 类型为：object
- **StaItems** (*list*) -- 需要获取的统计项目，类型为：list，目前支持的统计项

  FailCount

  SuccessCount

  FecInfo

  MaxPingLatency

  AvgPingLatency

  MinPingLatency

  RxReturnCode

返回

eg:

```
{
    'MinPingLatency': 10,
    'RxReturnCode': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LspPingEchoRequestStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Lsp Ping Echo Request Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_lsp_ping_session_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Port Statistic 统计结果

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object / list
- **StaItems** (*list*) -- 需要获取的统计项目，类型为：list，目前支持的统计项

  TxEchoRequest

  RxEchoRequest

  TxEchoReply

  RxEchoReply

返回

eg:

```
{
    'TxEchoReply': 10,
    'RxEchoReply': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LspPingSessionStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Lsp Ping Session Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_lsp_trace_echo_request_statistic(*Session*, *EchoRequest*, *StaItems: Optional[list] = None*)

获取 Dhcpv6 Port Statistic 统计结果

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object
- **EchoRequest** (LspPingEchoRequestConfig) -- Lsp Ping Echo Request 对象, 类型为：object
- **StaItems** (*list*) -- 需要获取的统计项目，类型为：list，目前支持的统计项

  FailCount

  SuccessCount

  FecInfo

  MaxPingLatency

  AvgPingLatency

  MinPingLatency

  RxReturnCode

返回

eg:

```
{
    'MinPingLatency': 10,
    'RxReturnCode': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=LspPingEchoRequestStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Lsp Trace Echo Request Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_mld_host_statistic(*Session*, *StaItems=None*)

获取 Mld 协议会话统计结果

参数

- **Session** (Mld) -- Mld 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  MldTxFrames

  MldRxFrames

  MldRxUnknownTypes

  MldRxChecksumErrors

  MldRxLengthErrors

返回

eg:

```
{
    'MldTxFrames': 8,
    'MldRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | MldTxFrames | MldRxFrames |
| Subscribe Result | Types=MldHostResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Mld Host Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_mld_port_statistic(*Port*, *StaItems=None*)

获取 Mld Port 统计结果

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  MldTxFrames

  MldRxFrames

  MldTxV1Reports

  MldStopListenGroups

  MldTxV2Reports

  MldTxV2ModeInclude

  MldTxV2ModeExclude

  MldTxV2ModeChangeToInclude

  MldTxV2ModeChangeToExclude

  MldTxV2ModeAllowNewSources

  MldTxV2ModeBlockOldSources

  MldRxV1Queries

  MldRxV2Queries

  MldRxGeneralQueries

  MldRxGroupSpecificQueries

  MldRxGroupAndSourceSpecificQueries

  MldRxUnknownTypes

  MldRxChecksumErrors

  MldRxLengthErrors

返回

eg:

```
{
    'MldTxFrames': 8,
    'MldRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | MldTxFrames | MldRxFrames |
| Subscribe Result | Types=MldPortAggregatedResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Mld Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_mld_querier_statistic(*Session*, *StaItems=None*)

获取 Mld Querier 协议会话统计结果

参数

- **Session** (MldQuerier) -- Mld 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  QuerierTxFrames

  QuerierRxFrames

  QuerierRxUnknownTypes

  QuerierRxChecksumErrors

  QuerierRxLengthErrors

返回

eg:

```
{
    'QuerierTxFrames': 8,
    'QuerierRxFrames': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | QuerierTxFrames | QuerierRxFrames |
| Subscribe Result | Types=MldQuerierResults |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Mld Querier Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_ospf_router_from_lsa(*Lsa*)

获取 OSPF LSA 对应的绑定流源或目的端点对象

参数 **Lsa** (*Port*) -- 测试仪表 OSPFv2 或 OSPFv3 LSA 对象, 类型为：object

返回 OSPFv2 或 OSPFv3 LSA 对应的绑定流源或目的端点对象, 类型：object

实际案例

```
| ${Session} | Create Ospf | Port=${Port} |
| ${RouterLsa} | Create Ospf Router Lsa | Session=${Session} | Age=20 |
| ${Point} | Get Ospf Router From Lsa | Lsa=${RouterLsa} |
```

static get_ospf_statistic(*Session*, *StaItems=None*)

获取 OSPFv2 协议会话统计结果

参数

- **Session** (OspfRouter) -- OSPFv2 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  RouterState

  AdjacencyState

  TxHello

  RxHello

  TxDd

  RxDd

  TxRouterLsa

  RxRouterLsa

  TxNetworkLsa

  RxNetworkLsa

  TxSummaryLsa

  RxSummaryLsa

  TxAsbrSummaryLsa

  RxAsbrSummaryLsa

  TxAsExternalLsa

  RxAsExternalLsa

  TxNssaLsa

  RxNssaLsa

  TxTeLsa

  RxTeLsa

  TxOpaqueRouterInfoLsa

  RxOpaqueRouterInfoLsa

  TxOpaqueExtendedPrefixLsa

  RxOpaqueExtendedPrefixLsa

  TxOpaqueExtendedLinkLsa

RxOpaqueExtendedLinkLsa

TxRequest

RxRequest

TxUpdate

RxRequest

TxAck

RxAck

返回

eg:

```
{
    'AdjacencyState': 'Full',
    'TxUpdate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv2SessionResultPropertySet |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Ospf Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_ospfv3_statistic(*Session*, *StaItems=None*)

获取 OSPFv3 协议会话统计结果

参数

- **Session** (Ospfv3Router) -- OSPFv3 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，默认为:None 表示获取所有统计项, 类型为：list，目前支持的统计项

  RouterState

  AdjacencyState

  TxHello

  RxHello

  TxDd

  RxDd

  TxRouterLsa

  RxRouterLsa

  TxNetworkLsa

  RxNetworkLsa

  TxInterAreaPrefixLsa

  RxInterAreaPrefixLsa

  TxInterAreaRouterLsa

RxInterAreaRouterLsa

TxAsExternalLsa

RxAsExternalLsa

TxNssaLsa

RxNssaLsa

TxLinkLsa

RxLinkLsa

TxIntraAreaPrefixLsa

RxIntraAreaPrefixLsa

TxOpaqueRouterInfoLsa

RxOpaqueRouterInfoLsa

TxSrv6LocatorLsa

RxSrv6LocatorLsa

TxRequest

RxRequest

TxUpdate

RxUpdate

TxAck

RxAck

返回

eg:

```
{
    'AdjacencyState': 'Full',
    'TxUpdate': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv3SessionResultPropertySet |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Ospfv3 Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_pcep_lsp_block_statistic(*Session*, *SessionId*, *Lsp*, *StaItems: Optional[list] = None*)

获取 PCEP LSP BLOCK 统计结果

参数

- **Session** (Pcep) -- : PCEP 协议会话对象, 类型为: Object
- **SessionId** (*int*) -- PCEP 协议会话 ID, 类型为: Number
- **Lsp** (PccLspConfig) -- : LSP 对象, 类型为: Object

- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  LspIdentify

  SessionIndex

  SessionLocalIP

  SessionPeerIP

  Role

  LspCount

  RequestedLsps

  RepliedLsps

  DelegatedLsps

  UpdatedLsps

  RevokedLsps

  ReturnedLsps

  InitiatedLsps

  StateDownLsps

  StateUpLsps

  StateActiveLsps

  StateGoingDownLsps

  StateGoingUpLsps

  StateOtherLsps

返回

eg:

```
{
    'LspCount': 1,
    'RequestedLsps': 0,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepLspBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Lsp Block Statistic | Session=@{Session} |␣
↪SessionId=1 | Lsp=@{Lsp} | StaItems=@{StaItems} |
| Clear Result |
```

static get_pcep_lsp_statistic(*Session*, *SessionId*, *Lsp*, *LspId*, *StaItems: Optional[list] = None*)

获取 PCEP LSP 统计结果

参数

- **Session** (Pcep) -- : PCEP 协议会话对象, 类型为：Object
- **SessionId** (*int*) -- PCEP 协议会话 ID, 类型为：Number
- **Lsp** (PccLspConfig) -- : LSP 对象, 类型为：Object
- **LspId** (*int*) -- LSP 对象 ID, 类型为：Number
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  LspIdentify

  SessionIndex

  LspIndex

  SessionLocalIP

  SessionPeerIP

  Role

  SymbolicName

  LspSourceIP

  LspDestinationIP

  LspState

  PLSPId

  LSPId

  SRPId

  RPId

返回

eg:

```
{
    'PLSPId': 1,
    'LSPId': 1,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepLspStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Lsp Statistic | Session=@{Session} | SessionId=1 |␣
↪Lsp=@{Lsp} | LspId=1 | StaItems=@{StaItems} |
| Clear Result |
```

static get_pcep_port_statistic(*Port*, *StaItems: Optional[list] = None*)

获取 PCEP Port 统计结果

参数

- **Port** (*Port*) -- PCEP 协议会话所在的端口对象, 类型为：Object

- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockCount

  SessionBlockIdleCount

  SessionBlockPendingCount

  SessionBlockUpCount

返回

eg:

```
{
    'SessionBlockCount': 1,
    'SessionBlockIdleCount': 0,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepPortStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Port Statistic | Port=@{Port} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_pcep_session_block_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 PCEP session block 统计结果

参数

- **Session** (Pcep) -- ：PCEP 协议会话对象，类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  SessionCount

  IdleCount

  PendingCount

  UpCount

  LspCount

  StateDownLsps

  StateUpLsps

  StateActiveLsps

  StateGoingDownLsps

  StateGoingUpLsps

  StateOtherLsps

  TxOpenCount

RxOpenCount

TxKeepaliveCount

RxKeepaliveCount

TxReportCount

RxReportCount

TxUpdateCount

RxUpdateCount

TxRequestCount

RxRequestCount

TxReplyCount

RxReplyCount

TxInitiateCount

RxInitiateCount

TxCloseCount

RxCloseCount

TxErrorCount

RxErrorCount

返回

eg:

```
{
    'TxErrorCount': 1,
    'RxErrorCount': 0,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepSessionBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Session Block Statistic | Session=@{Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_pcep_session_statistic(*Session*, *SessionId*, *StaItems: Optional[list] = None*)

获取 PCEP session 统计结果

参数

- **Session** (Pcep) -- ：PCEP 协议会话对象, 类型为：Object
- **SessionId** (*int*) -- PCEP 协议会话 ID, 类型为：Number
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  Role

LocalIP

PeerIP

State

LspCount

StateDownLsps

StateUpLsps

StateActiveLsps

StateGoingDownLsps

StateGoingUpLsps

StateOtherLsps

TxOpenCount

RxOpenCount

TxKeepaliveCount

RxKeepaliveCount

TxReportCount

RxReportCount

TxUpdateCount

RxUpdateCount

TxRequestCount

RxRequestCount

TxReplyCount

RxReplyCount

TxInitiateCount

RxInitiateCount

TxCloseCount

RxCloseCount

TxErrorCount

RxErrorCount

返回

eg:

```
{
    'TxErrorCount': 1,
    'RxErrorCount': 0,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=PcepSessionStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pcep Session Statistic | Session=@{Session} |␣
↪SessionId=@{SessionId} | StaItems=@{StaItems} |
| Clear Result |
```

static get_pim_group_stats(*Session*, *Group*, *StaItems: Optional[list] = None*)

获取 Pim Group Stats 统计结果

参数

- **Session** (PimRouter) -- BGP 协议会话对象, 类型为：Object
- **Group** (PimGroupConfig) -- Pim 协议组对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  TxAnyG

  TxSG

  TxRP

  TxRpt

返回

eg:

```
{
    'TxRpt': 10,
    'RxRpt': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | TxHello | RxHello |
| Subscribe Result | Types=PimGroupStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pim Group Stats | Session=${Session} | Group=${Group} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_pim_session_stats(*Session*, *StaItems: Optional[list] = None*)

获取 Pim Session Stats 统计结果

参数

- **Session** (PimRouter) -- BGP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  NeighborNum

  TxHello

  RxHello

TxJoin

RxJoin

TxAnyG

RxAnyG

TxSG

RxSG

TxRP

RxRP

TxRpt

RxRpt

TxBsr

TxRegister

RxRegisterStop

返回

eg:

```
{
    'TxRpt': 10,
    'RxRpt': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | TxHello | RxHello |
| Subscribe Result | Types=PimSessionStats |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pim Session Stats | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_port_latency_statistic(*Port*, *StaItems=None*, *Mode=True*)

获取测试仪表端口时延统计结果

参数

- **Port** (*Port*) -- 指定需要获取结果的端口对象 object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取端口统计项目, 目前支持的统计项

  PortID：端口名称

  MinLatency：最小时延

  MaxLatency：最大时延

  AvaLatency：平均时延

返回

eg:

```
{
    'MinLatency': 1.2311,
    'MaxLatency': 5.123,
}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | MaxLatency |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Port} | Get Ports |
| &{Result} | Get Port Latency Statistic | Port=${Port} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_port_speed(*Ports*)

修改测试仪表端口参数

参数 **Ports** (list(*Port*)) -- 测试仪表端口列表

返回 端口速率列表

返回类型 list

实际案例

robotframework:

```
| Get Port Speed | Ports=${Ports} |
```

static get_port_statistic(*Port*, *StaItems=None*, *Mode=True*)

获取测试仪表端口统计结果

参数

- **Port** (*Port*) -- 指定需要获取结果的端口对象 object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取端口统计项目, 目前支持的统计项

  TxTotalFrames：发送报文总数

  RxTotalFrames：接收报文总数

  TxStreamFrames：发送流报文总数

  TxSignatureStreamFrames：发送带标签流报文总数

  RxSignatureStreamFrames：接收带标签流报文总数

  TxFrameRate：发送报文速率 (fps)

  RxFrameRate：接收报文速率 (fps)

  TxL1Rate：发送线速 (bps)

  RxL1Rate：接收线速 (bps)

TxUtil：发送百分比 (%)
RxUtil：接收百分比 (%)
TxByteRate：发送字节速率 (Bps)
RxByteRate：接收字节速率 (Bps)
TxBitRate：发送比特速率 (bps)
RxBitRate：接收比特速率 (bps)
TxTotalBytes：发送字节总数
RxTotalBytes：接收字节总数
RxFCSErr：接收 FCS 错误数
RxIpv4ChecksumError：接收 IPv4 Checksum Error 报文数
RxTcpChecksumError：接收 TCP Checksum Error 报文数
RxUdpChecksumError：接收 UDP Checksum Error 报文数
RxPrbsFillBytes：接收 PRBS 填充字节数
RxPrbsErrorBits：接收 PRBS 错误位数
RxPrbsErrorFrames：接收 PRBS 错误报文数
RxIpv4Frames：接收 IPv4 帧数
RxIpv6Frames：接收 IPv6 帧数
RxTcpFrames：接收 TCP 帧数
RxUdpFrames：接收 UDP 帧数
RxMplsCount：接收 MPLS 帧数
RxIcmpFrames：接收 ICMP 帧数
RxVlanFrames：接收 VLAN 帧数
RxFCoEFrames：接收 FCoE 帧数
RxPauseFrames：接收 Pause 帧数
RxUndersizeFrames：接收超短帧数
RxOversizeFrames：接收超长帧数
RxJumboFrames：接收巨型帧数
RxOutofSequenceCount：接收乱序帧数
RxFilter0Count：接收过滤帧数 _0
RxFilter1Count：接收过滤帧数 _1
RxFilter2Count：接收过滤帧数 _2
RxFilter3Count：接收过滤帧数 _3
RxFilter4Count：接收过滤帧数 _4
RxFilter5Count：接收过滤帧数 _5
RxFilter6Count：接收过滤帧数 _6
RxFilter7Count：接收过滤帧数 _7
RxPktLossCount：接收丢包帧数
RxInOrderCount：接收有序帧数
RxReorderCount：接收重排序帧数

RxRepeatFrameCount：接收重复帧数

RxPortLateCount：接收端口延迟帧数

RxCorrectedRSFECErrorsFramesCodewords：接收 Corrected RS FEC Error 帧数（codewords）

RxUncorrectedRSFECErrorsFramesCodewords：接收 Uncorrected RS FEC Error 帧数（codewords）

RxCorrectedBaseRFECErrorsFramesCodewords：接收 Corrected BaseR FEC Error 帧数（codewords）

RxUncorrectedBaseRFECErrorsFramesCodewords：接收 Uncorrected BaseR FEC Error 帧数（codewords）

TxCrcFrameCount：发送 CRC 帧数

TxErr3CheckFrameCount：发送 IP Checksum Error 报文数

TxErr4CheckFrameCount：发送 L4 Checksum Error 报文数

TxIpv4Count：发送 IPv4 帧数

TxIpv6Count：发送 IPv6 帧数

TxMplsCount：发送 MPLS 帧数

TxIpv4FrameCount：发送 IPv4 流帧数

TxIpv6FrameCount：发送 IPv6 流帧数

TxVlanFrameCount：发送 VLAN 流帧数

TxMplsFrameCount：发送 MPLS 流帧数

TxOversizeFrames：发送超长帧数

TxUndersizeFrames：发送超短帧数

TxJumboFrames：发送巨型帧数

RxPFCFrames：接收 PFC 帧数

RxPFCRate：接收 PFC 速率

RxPFCPriority0Frames：接收 PFC 优先级是 0 的帧数

RxPFCPriority1Frames：接收 PFC 优先级是 1 的帧数

RxPFCPriority2Frames：接收 PFC 优先级是 2 的帧数

RxPFCPriority3Frames：接收 PFC 优先级是 3 的帧数

RxPFCPriority4Frames：接收 PFC 优先级是 4 的帧数

RxPFCPriority5Frames：接收 PFC 优先级是 5 的帧数

RxPFCPriority6Frames：接收 PFC 优先级是 6 的帧数

RxPFCPriority7Frames：接收 PFC 优先级是 7 的帧数

TxPFCFrames：发送 PFC 帧数

TxPFCRate：发送 PFC 速率

TxPFCPriority0Frames：发送 PFC 优先级是 0 的帧数

TxPFCPriority1Frames：发送 PFC 优先级是 1 的帧数

TxPFCPriority2Frames：发送 PFC 优先级是 2 的帧数

TxPFCPriority3Frames：发送 PFC 优先级是 3 的帧数

TxPFCPriority4Frames：发送 PFC 优先级是 4 的帧数

TxPFCPriority5Frames：发送 PFC 优先级是 5 的帧数

TxPFCPriority6Frames：发送 PFC 优先级是 6 的帧数

TxPFCPriority7Frames：发送 PFC 优先级是 7 的帧数

RxARPFrames：接收 ARP 报文数

TxARPFrames：发送 ARP 报文数

RxBroadcastFrames：接收广播报文数

TxBroadcastFrames：发送广播报文数

RxIpv4LengthErrorFrames：接收 IPv4 长度错误帧数

RxUserDefinedCapture0Frames：接收自定义统计 0 报文数

RxUserDefinedCapture0Rate：接收自定义统计 0 报文速率 (fps)

RxUserDefinedCapture1Frames：接收自定义统计 1 报文数

RxUserDefinedCapture1Rate：接收自定义统计 1 报文速率 (fps)

RxUserDefinedCapture2Frames：接收自定义统计 2 报文数

RxUserDefinedCapture2Rate：接收自定义统计 2 报文速率 (fps)

RxUserDefinedCapture3Frames：接收自定义统计 3 报文数

RxUserDefinedCapture3Rate：接收自定义统计 3 报文速率 (fps)

RxUserDefinedCapture4Frames：接收自定义统计 4 报文数

RxUserDefinedCapture4Rate：接收自定义统计 4 报文速率 (fps)

RxUserDefinedCapture5Frames：接收自定义统计 5 报文数

RxUserDefinedCapture5Rate：接收自定义统计 5 报文速率 (fps)

RxFirstFrameArrivalTime：接收第一个帧的时间

RxLastFrameArrivalTime：接收最后一个帧的时间

返回

eg:

```
{
    'RxFirstFrameArrivalTime': 1000,
    'RxLastFrameArrivalTime': 1000,
}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | TxTotalFrames | RxTotalFrames |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Port} | Get Ports |
| &{Result} | Get Port Statistic | Port=${Port} | StaItems=@{StaItems} |
| Clear Result |
```

static get_ports()

获取当前测试仪表配置中所有的端口对象

返回 Port 对象列表

返回类型 list

实际案例

robotframework:

```
| ${result} | Get Ports |
```

static get_pppoe_client_block_statistic(*Session*, *StaItems=None*)

获取 PPPoE Server Block Statistic 统计结果

参数

- **Session** (PppoeClient) -- 测试仪表端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取 PPPoE Client Block Statistic 统计项目，类型为：list，目前支持的统计项

  IpcpState

  Ipv6cpState

  SessionCount

  SessionsUp

  SessionsRetried

  AttemptedConnects

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

  MaxSetupTime

  MinSetupTime

  AverageSetupTime

  SuccessfulSetupRate

  TxPadi

  RxPado

  TxPadr

  RxPads

  TxPadt

  RxPadt

  TxLcpConfigRequest

  RxLcpConfigRequest

  TxLcpConfigAck

  RxLcpConfigAck

  TxLcpConfigNak

RxLcpConfigNak

TxLcpConfigReject

RxLcpConfigReject

TxLcpEchoRequest

RxLcpEchoRequest

TxLcpEchoReply

RxLcpEchoReply

TxLcpTerminateRequest

RxLcpTerminateRequest

TxLcpTerminateAck

RxLcpTerminateAck

TxChap

RxChap

TxPap

RxPap

TxIpcp

RxIpcp

TxIpv6cp

RxIpv6cp

TxIpv4

RxIpv4

TxIpv6

RxIpv6

返回

eg:

```
{
    'SessionCount': 10,
    'SessionsUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionCount | SessionsUp |
| Subscribe Result | Types=PppoeClientBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Client Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_pppoe_client_statistic(*Session*, *Index=1*, *StaItems=None*)

获取 PPPoE Client Statistic 统计结果

参数

- **Session** (PppoeClient) -- 测试仪表端口对象, 类型为：Object
- **Index** (*int*) -- PppoeClient Block 里会话的 index, 默认值为: 1
- **StaItems** (*list*) -- 需要获取 PPPoE Client Block Statistic 统计项目，类型为：list，目前支持的统计项

  IpcpState

  Ipv6cpState

  MacAddress

  PeerMacAddress

  PppoeSessionId

  VlanId

  InnerVlanId

  Ipv4Address

  PeerIpv4Address

  Ipv6LinklocalAddress

  PeerIpv6LinklocalAddress

  Ipv6GlobalAddress

  SessionsRetried

  AttemptedConnects

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

  SetupTime

  TxPadi

  RxPado

  TxPadr

  RxPads

  TxPadt

  RxPadt

  TxLcpConfigRequest

  RxLcpConfigRequest

  TxLcpConfigAck

  RxLcpConfigAck

  TxLcpConfigNak

  RxLcpConfigNak

  TxLcpConfigReject

RxLcpConfigReject

TxLcpEchoRequest

RxLcpEchoRequest

TxLcpEchoReply

RxLcpEchoReply

TxLcpTerminateRequest

RxLcpTerminateRequest

TxLcpTerminateAck

RxLcpTerminateAck

TxChap

RxChap

TxPap

RxPap

TxIpcp

RxIpcp

TxIpv6cp

RxIpv6cp

TxIpv4

RxIpv4

TxIpv6

RxIpv6

返回

eg:

```
{
    'SessionCount': 10,
    'SessionsUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionCount | SessionsUp |
| Subscribe Result | Types=PppoeClientStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Client Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_pppoe_port_statistic(*Port*, *StaItems=None*)

获取 PPPoE Port Statistic 统计结果

参数

- **Port** (*Port*) -- 测试仪表端口对象, 类型为：Object

- **StaItems** (*list*) -- 需要获取 PPPoE Port Statistic 统计项目，类型为：list，目前支持的统计项

  SessionBlockCount

  SessionCount

  SessionsUp

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

返回

eg:

```
{
    'SessionBlockCount': 10,
    'SessionCount': 100,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionBlockCount | SessionCount |
| Subscribe Result | Types=PppoePortStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Port Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_pppoe_server_block_statistic(*Session*, *StaItems=None*)

获取 PPPoE Server Block Statistic 统计结果

参数

- **Session** (PppoeServer) -- 测试仪表端口对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取 PPPoE Server Block Statistic 统计项目，类型为：list，目前支持的统计项

  IpcpState

  Ipv6cpState

  SessionCount

  SessionsUp

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

  MaxSetupTime

  MinSetupTime

  AverageSetupTime

SuccessfulSetupRate
RxPadi
TxPado
RxPadr
TxPads
TxPadt
RxPadt
TxLcpConfigRequest
RxLcpConfigRequest
TxLcpConfigAck
RxLcpConfigAck
TxLcpConfigNak
RxLcpConfigNak
TxLcpConfigReject
RxLcpConfigReject
TxLcpEchoRequest
RxLcpEchoRequest
TxLcpEchoReply
RxLcpEchoReply
TxLcpTerminateRequest
RxLcpTerminateRequest
TxLcpTerminateAck
RxLcpTerminateAck
TxChap
RxChap
TxPap
RxPap
TxIpcp
RxIpcp
TxIpv6cp
RxIpv6cp
TxIpv4
RxIpv4
TxIpv6
RxIpv6

返回

eg:

```
{
    'SessionCount': 10,
    'SessionsUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionCount | SessionsUp |
| Subscribe Result | Types=PppoeServerBlockStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Server Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_pppoe_server_statistic(*Session*, *Index=1*, *StaItems=None*)

获取 PPPoE Server Statistic 统计结果

参数

- **Session** (PppoeServer) -- 测试仪表端口对象, 类型为：Object
- **Index** (*int*) -- PppoeServer Block 里会话的 index, 默认值为: 1
- **StaItems** (*list*) -- 需要获取 PPPoE Server Statistic 统计项目，类型为: list，目前支持的统计项

  IpcpState

  Ipv6cpState

  MacAddress

  PeerMacAddress

  PppoeSessionId

  VlanId

  InnerVlanId

  Ipv4Address

  PeerIpv4Address

  Ipv6LinklocalAddress

  PeerIpv6LinklocalAddress

  SuccessfulConnects

  FailedConnects

  SucessfulDisconnects

  FailedDisconnects

  SetupTime

  RxPadi

  TxPado

  RxPadr

  TxPads

TxPadt

RxPadt

TxLcpConfigRequest

RxLcpConfigRequest

TxLcpConfigAck

RxLcpConfigAck

TxLcpConfigNak

RxLcpConfigNak

TxLcpConfigReject

RxLcpConfigReject

TxLcpEchoRequest

RxLcpEchoRequest

TxLcpEchoReply

RxLcpEchoReply

TxLcpTerminateRequest

RxLcpTerminateRequest

TxLcpTerminateAck

RxLcpTerminateAck

TxChap

RxChap

TxPap

RxPap

TxIpcp

RxIpcp

TxIpv6cp

RxIpv6cp

TxIpv4

RxIpv4

TxIpv6

RxIpv6

返回

eg:

```
{
    'SessionCount': 10,
    'SessionsUp': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | SessionCount | SessionsUp |
| Subscribe Result | Types=PppoeServerStatistic |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Pppoe Server Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_rip_router_from_route(*Route*)

获取 OSPF LSA 对应的绑定流源或目的端点对象

参数 **Route** (*Port*) -- Rip Ipv4 / Ipv6 Route 对象, 类型为：object

返回 Rip Route 对应的绑定流源或目的端点对象, 类型：object

实际案例

```
| ${Session} | Create Rip | Port=${Port} |
| ${RouterLsa} | Create Rip Ipv4 Router | Session=${Session} | Age=20 |
| ${Point} | Get Rip Router From Route | Route=${RouterLsa} |
```

static get_rip_session_block_statistic(*Session*, *StaItems: Optional[list] = None*)

获取 RIP 协议会话统计结果

参数

- **Session** (RipRouter) -- RIP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  SessionCount

  TxAdvertised

  RxAdvertised

  TxWithdrawn

  RxWithdrawn

返回

eg:

```
{
    'TxAdvertised': 10,
    'RxAdvertised': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv2SessionResultPropertySet |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Rip Session Block Statistic | Session=${Session} |␣
↪StaItems=@{StaItems} |
| Clear Result |
```

static get_rip_session_statistic(*Session*, *SessionId*, *StaItems=None*)

获取 RIP 协议会话统计结果

参数

- **Session** (RipRouter) -- RIP 协议会话对象, 类型为：Object
- **SessionId** (*int*) -- RIP 协议会话对象, 类型为：Object
- **StaItems** (*list*) -- 需要获取流模板统计项目，类型为：list，目前支持的统计项

  SessionBlockId

  SessionId

  SessionState

  TxAdvertised

  RxAdvertised

  TxWithdrawn

  RxWithdrawn

返回

eg:

```
{
    'TxAdvertised': 10,
    'RxAdvertised': 10,
}
```

返回类型 dict

实际案例

```
| @{StaItems} | Create List | AdjacencyState | TxUpdate |
| Subscribe Result | Types=Ospfv2SessionResultPropertySet |
| Start Protocol |
| Sleep | 60 |
| &{Result} | Get Rip Session Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_sessions(*Ports=None*, *Protocols=None*)

获取当前测试仪表配置中所有的协议对象

参数

- **Ports** (*list*) -- 端口对象的列表

- **Protocols** (*list*) -- 指定需要启动的协议类型列表，目前支持的协议类型：
  bgp
  bfd
  isis
  ospfv2
  ospfv3
  pim
  rip
  dot1x
  dhcpv4server
  dhcpv4
  dhcpv6server
  dhcpv6
  vxlan
  saa
  IGMP
  igmpquery
  mld
  mldquery
  l2tp
  pppoe
  ldp
  lspping
  pcep

返回 协议会话对象列表

实际案例

```
| ${result} | Get Sessions |
```

static get_stream_rx_statistic(*Stream*, *Port*, *StreamID=1*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板统计结果

参数

- **Stream** (StreamTemplate) --
- **Port** (*Port*) -- 接收端口对象 object, 类型为：object
- **StreamID** (*int*) -- 指定需要获取结果的流模板中流的 Id, 默认值：1
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认：True

- **StaItems** (*list*) -- 需要获取流模板统计项目，目前支持的统计项：需要获取流模板统计项目

  StreamID

  StreamBlockID

  ChannelId

  PortID

  LoadBalance

  RxStreamFrames：接收报文数

  RxFrameRate：接收报文速率 (fps)

  RxByteRate：接收字节速率 (Bps)

  RxSeqErr

  RxPayloadErr

  MinLatency：最小时延 (us)

  MaxLatency：最大时延 (us)

  AvaLatency：平均时延 (us)

  ShortTermAvgLatency：短期平均时延 (us)

  RxBitRate：接收比特速率 (bps)

  RxUtil：接收百分比 (%)

  MinJitter：最小延迟抖动 (us)

  MaxJitter：最大延迟抖动 (us)

  AvaJitter：平均延迟抖动 (us)

  ShortTermAvgJitter：短期平均延迟抖动 (us)

  RxLossStreamFrames：实时丢包数

  RxIpLengthErrorCount

  RxIpv4ChecksumErrorFrames：接收 IPv4 Checksum Error 报文数

  PrbsFillBytes：接收 PRBS 填充字节

  DuplicateFrames：接收重复帧

  InOrderFrames：接收有序帧

  ReOrderFrames：接收重排序帧

  PrbsErrorBits：接收 PRBS 错误位数

  PrbsErrorFrames：接收 PRBS 错误帧数

  RxFcsErrorFrames：接收 FCS 错误帧

  RxFcsErrorFrameRate：接收 FCS 错误帧速率 (fps)

  TcpChecksumErrorFrames：接收 TCP/UDP 校验错误帧

  RxL1Rate：接收线速 (bps)

  RxTotalBytes：接收总字节数

  RxLateCount：接收延迟计数

  RxInSequenceCount：接收按顺序计数

  RxOutofSequenceCount：接收未按顺序计数

RxMinInterArrivalTime：接收最小到达时间 (us)

RxMaxInterArrivalTime：接收最大到达时间 (us)

RxAvgInterArrivalTime：接收平均到达时间 (us)

RxShortTermAvgInterArrivalTime：接收短期平均到达时间 (us)

返回

eg:

```
{"RxAvgInterArrivalTime": 1000, "RxShortTermAvgInterArrivalTime
↪": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | RxAvgInterArrivalTime |␣
↪RxShortTermAvgInterArrivalTime |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Stream Rx Statistic | Stream=${Stream} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_stream_statistic(*Stream*, *StreamID=1*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板统计结果

参数

- **Stream** (StreamTemplate) --
- **StreamID** (*int*) -- 指定需要获取结果的流模板中流的 Id, 默认值：1
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项: 需要获取流模板统计项目

  StreamBlockID：流量模板名称

  StreamID：流量 ID

  TxPortID：发送端口

  RxPortID：接收端口

  LoadBalance：接收端口

  TxStreamFrames：发送报文数

  RxStreamFrames：接收报文数

  TxFrameRate：发送报文速率 (fps)

  RxFrameRate：接收报文速率 (fps)

  TxL1Rate：发送线速 (bps)

  RxL1Rate：接收线速 (bps)

RxLossStreamFrames：实时丢包数

RealtimeLossRate：实时丢包率 (%)

ResumeTime：恢复时间 (s)

StartTime：流启动时间

TxUtil：发送百分比 (%)

RxUtil：接收百分比 (%)

RxPayloadErr：接收 Payload Error 报文数

RxSeqErr：接收 Sequence Error 报文数

RxIpLengthErrorCount：接收 IP 长度错误计数

TxByteRate：发送字节速率 (Bps)

RxByteRate：接收字节速率 (Bps)

TxBitRate：发送比特速率 (bps)

RxBitRate：接收比特速率 (bps)

MinLatency：最小延迟 (us)

MaxLatency：最大延迟 (us)

AvaLatency：平均延迟抖动 (us)

MinJitter：最小延迟抖动 (us)

MaxJitter：最大延迟抖动 (us)

AvaJitter：平均延迟抖动 (us)

RxIpv4ChecksumErrorFrames：接收 Ipv4 Checksum 错误

PrbsFillBytes：接收端口

DuplicateFrames：接收端口

InOrderFrames：接收端口

ReOrderFrames：接收端口

PrbsErrorBits：接收端口

PrbsErrorFrames：接收端口

RxFcsErrorFrames：接收 FCS Checksum 错误

RxFcsErrorFrameRate：接收 FCS Checksum 错误速率

TcpChecksumErrorFrames：接收 TCP Checksum 错误

LostStreamFrames：丢包数

返回

eg:

```
{"RxFcsErrorFrames": 1000, "LostStreamFrames": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | RxIpv4ChecksumError |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Stream Statistic | Stream=${Stream} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_stream_tx_statistic(*Stream*, *Port=None*, *StreamID=1*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板统计结果

参数

- **Stream** (StreamTemplate) --
- **Port** (*Port*) -- 发送端口对象 object, 类型为：object
- **StreamID** (*int*) -- 指定需要获取结果的流模板中流的 Id, 默认值：1
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项: 需要获取流模板统计项目

  StreamID

  StreamBlockID

  ChannelId

  PortID

  TxStreamFrames：发送报文数

  TxFrameRate：发送报文速率 (fps)

  TxByteRate：发送字节速率 (Bps)

  TxBitRate：发送比特速率 (bps)

  TxL1Rate：发送线速 (bps)

  TxUtil：发送百分比 (%)

  TxTotalBytes：发送总字节数

返回

eg:

```
{"TxL1Rate": 1000, "TxTotalBytes": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | TxUtil | TxTotalBytes |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Stream Tx Statistic | Stream=${Stream} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_streamblock_rx_statistic(*Stream*, *Port*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板块统计结果

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Port** (*Port*) -- 接收端口对象 object, 类型为：object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项: 需要获取流模板统计项目

  StreamBlockID

  PortID

  LoadBalance

  RxStreamFrames：接收报文数

  RxFrameRate：接收报文速率 (fps)

  RxByteRate：接收字节速率 (Bps)

  RxSeqErr

  RxPayloadErr

  MinLatency：最小时延 (us)

  MaxLatency：最大时延 (us)

  AvaLatency：平均时延 (us)

  ShortTermAvgLatency：短期平均时延 (us)

  RxBitRate：接收比特速率 (bps)

  RxUtil：接收线速 (bps)

  MinJitter：最小延迟抖动 (us)

  MaxJitter：最大延迟抖动 (us)

  AvaJitter：平均延迟抖动 (us)

  ShortTermAvgJitter：短期平均延迟抖动 (us)

  RxLossStreamFrames

  RxIpLengthErrorCount

  RxL1Rate

RxIpv4ChecksumErrorFrames：接收 IPv4 Checksum Error 报文数

PrbsFillBytes：接收 PRBS 填充字节

DuplicateFrames：接收重复帧

InOrderFrames：接收有序帧

ReOrderFrames：接收重排序帧

PrbsErrorBits：接收 PRBS 错误位数

PrbsErrorFrames：接收 PRBS 错误帧数

RxFcsErrorFrames：接收 FCS 错误帧

RxFcsErrorFrameRate：接收 FCS 错误帧速率 (fps)

TcpChecksumErrorFrames：接收 TCP/UDP 校验错误帧

RxAvgRate

RxAvgFps

RxMaxRate

RxMaxFps

RxTotalBytes：接收总字节数

RxLateCount：接收延迟计数

RxInSequenceCount：接收按顺序计数

RxOutofSequenceCount：接收未按顺序计数

RxMinInterArrivalTime：接收最小到达时间 (us)

RxMaxInterArrivalTime：接收最大到达时间 (us)

RxAvgInterArrivalTime：接收平均到达时间 (us)

RxShortTermAvgInterArrivalTime：接收短期平均到达时间 (us)

返回

eg:

```
{"TxUtil": 1000, "TxTotalBytes": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | RxIpv4ChecksumError |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Streamblock Rx Statistic | Stream=${Stream} | Port=$
↪{Port} | StaItems=@{StaItems} |
| Clear Result |
```

static get_streamblock_statistic(*Stream*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板块统计结果

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项: 需要获取流模板统计项目

  TxPortID：发送端口

  RxPortID：接收端口

  TxStreamFrames：发送带标签流报文总数

  RxStreamFrames：接收带标签流报文总数

  TxFrameRate：发送报文速率 (fps)

  RxFrameRate：接收报文速率 (fps)

  TxL1Rate：发送线速 (bps)

  RxL1Rate：接收线速 (bps)

  TxUtil：发送百分比 (%)

  RxUtil：接收百分比 (%)

  RxLossStreamFrames：实时丢包数

  RealtimeLossRate：实时丢包率 (%)

  ResumeTime：恢复时间 (s)

  StartTime：流启动时间

  MinLatency：最小延迟 (us)

  MaxLatency：最大延迟 (us)

  AvaLatency：平均延迟 (us)

  MinJitter：最小延迟抖动 (us)

  MaxJitter：最大延迟抖动 (us)

  AvaJitter：平均延迟抖动 (us)

  TxByteRate：发送字节速率 (Bps)

  RxByteRate：接收字节速率 (Bps)

  TxBitRate：发送比特速率 (bps)

  RxBitRate：接收比特速率 (bps)

  TxTotalBytes：发送字节总数

  RxTotalBytes：接收字节总数

  RxPayloadErr：接收 Payload Error 报文数

  RxInSequenceCount：接收 Sequence Error 报文数

  RxFCSErr：接收 FCS 错误数

  RxIpv4ChecksumError：接收 IPv4 Checksum Error 报文数

  RxTcpChecksumError：接收 TCP Checksum Error 报文数

返回

eg:

```
{"TxTotalFrames": 1000, "RxTotalFrames": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | RxIpv4ChecksumError |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Streamblock Statistic | Stream=${Stream} | StaItems=@
↪{StaItems} |
| Clear Result |
```

static get_streamblock_tx_statistic(*Stream*, *Port=None*, *StaItems=None*, *Mode=True*)

获取测试仪表流模板块统计结果

参数

- **Stream** (StreamTemplate) -- 测试仪表流量对象 object, 类型为：object
- **Port** (*Port*) -- 发送端口对象 object, 类型为：object
- **Mode** (*bool*) -- 是否从仪表后台读取统计, 默认: True
- **StaItems** (*list*) -- 需要获取流模板统计项目, 目前支持的统计项: 需要获取流模板统计项目

  StreamBlockID

  PortID

  ChannelCount

  TxStreamFrames：发送报文数

  TxFrameRate：发送报文速率 (fps)

  TxByteRate：发送字节速率 (Bps)

  TxBitRate：发送比特速率 (bps)

  TxL1Rate：发送线速 (bps)

  TxUtil：发送百分比 (%)

  TxTotalBytes：发送字节数

返回

eg:

```
{"TxUtil": 1000, "TxTotalBytes": 1000}
```

返回类型 dict

实际案例

robotframework:

```
| @{StaItems} | Create List | MinLatency | RxIpv4ChecksumError |
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${Stream} | Get Streams |
| &{Result} | Get Streamblock Tx Statistic | Stream=${Stream} | Port=$
↪{Port} | StaItems=@{StaItems} |
| Clear Result |
```

static get_streams(*Ports=None*)

获取测试仪表流量对象

参数 **Ports** (list (*Port*)) -- 测试仪表端口对象列表

返回 测试仪表流量对象列表

返回类型 Streams (list (StreamTemplate))

实际案例

```
| Get Streams | Ports=${Ports} |
```

static get_vxlan_statistic(*Session*, *StaItems=None*)

获取测试仪表 vxlan 统计

参数

- **Session** (Vxlan) -- Vxlan 协议会话对象, 类型为：object
- **StaItems** (*list*) -- 需要获取 Vxlan 统计项目, 类型为：list，目前支持的统计项

  VtepId: VXLAN 会话的名称

  VtepState: VXLAN 会话的状态

  TotalVmCount：VM 总数

  ResolvedVmCount: 已解析 VM

  UnresolvedVmCount: 未解析 VM

返回

```
{"TotalVmCount": 100, "ResolvedVmCount": 100}
```

返回类型 dict

实际案例

```
| Subscribe Result |
| Start Protocol |
| Sleep | 10 |
| Stop Protocol |
| Sleep | 3 |
| ${Port} | Get Ports |
| ${Session} | Get Session | Ports=@{Port} | Protocols=vxlan |
| ${StaItems} | Create List | TotalVmCount | ResolvedVmCount |
| &{Result} | Get Vxlan Statistic | Session=${Session} | StaItems=@
↪{StaItems} |
```

static get_vxlan_vm_property(*interface*)

获取 Vxlan Vm Property 对象

参数 **Interface** (Interface) -- Interface 对象, 类型为：object

返回 Vxlan Vm Property 对象, 类型：object

返回类型 (VxlanVmProperty)

实际案例

```
| Get Vxlan Vm Property | Interface=${Interface} |
```

static grace_restart_ospf(*Sessions*)

平滑重启 OSPFv2 协议会话

参数 **Sessions** (list(OspfRouter)) -- OSPFv2 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Grace Restart Ospf | Sessions=${Sessions} |
```

static grace_restart_ospfv3(*Sessions*)

平滑重启 OSPFv3 协议会话

参数 **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Grace Restart Ospfv3 | Sessions=${Sessions} |
```

static graceful_restart_isis(*Sessions*)

平滑重启 Isis 协议会话

参数 **Sessions** (list (IsisRouter)) -- ISIS 会话对象列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Graceful Restart Isis | Session=${Session} |
```

static init_tester(*Product='BIGTAO', Mode='performance', Log=True*)

初始化测试仪表

参数

- **Product** (*str*) -- 测试产品类型, 支持 BIGTAO 和 DARYU
- **Mode** (*str*) -- 统计模式, 支持 Performance 和 DB
- **Log** (*bool*) -- 是否使能机箱日志

返回 sys_entry 测试仪表根节点对象

返回类型 (sys_entry)

实际案例

robotframework:

```
| ${result} | init tester | Product=DARYU | Mode=Performance |
```

static load_case(*Path*)

测试仪表加载配置文件

参数 **Path** (*str*) -- 配置文件路径, 类型 string (例如："C:/test.xcfg")

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${result} | Load Case | Path=='C:/test.xcfg' |
```

static pause_lsp_ping(*Sessions*)

暂停发送 LSP Ping 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pause Lsp Ping | Sessions=${Sessions} |
```

static pause_lsp_trace(*Sessions*)

暂停发送 LSP Trace 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pause Lsp Trace | Sessions=${Sessions} |
```

static pcep_establish(*Sessions*)

建立 PCEP 会话

参数 **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Establish | Sessions=${Sessions} |
```

static pcep_pcc_delegate_lsp(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送托管 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Delegate Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pcc_end_sync(*Sessions*, *Lsps=None*)

停止 PCC 向 PCE 发送初始状态同步报文

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc End Sync | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pcc_initial_sync(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送初始状态同步报文

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Initial Sync | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pcc_remove_delegate_lsp(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送删除托管 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Remove Delegate Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pcc_report_lsp(*Sessions*, *Lsps=None*)

PCC 向 PCE 报告 LSP 状态

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Report Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pcc_request_lsp(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送路径计算请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Request Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pcc_revoke_lsp(*Sessions*, *Lsps=None*)

PCC 向 PCE 发送取消托管 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (BgpRouter) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Revoke Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pcc_synchronize_lsp(*Sessions*, *Lsps=None*)

PCC 向 PCE 报告 LSP 状态

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pcc Synchronize Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pce_initiate_lsp(*Sessions*, *Lsps=None*)

PCE 向 PCC 发送初始化 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pce Initiate Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pce_remove_initiated_lsp(*Sessions*, *Lsps=None*)

PCE 向 PCC 发送删除指定 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pce Remove Initiate Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pce_return_lsp(*Sessions*, *Lsps=None*)

使 Stateful PCE 向 PCC 归还 LSP 托管权限

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pce Return Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_pce_update_lsp(*Sessions*, *Lsps=None*)

使 Stateful PCE 向 PCC 发送更新 LSP 请求

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list
- **Lsps** (PccLspConfig) -- : PCEP Lsp 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Pce Update Lsp | Sessions=${Sessions} | Lsps=&{Lsps} |
```

static pcep_resume_keep_alive(*Sessions*)

指定 PCEP 会话恢复发送 Keepalive 报文

参数 **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Resume Keep Alive | Sessions=${Sessions} |
```

static pcep_stop_keep_alive(*Sessions*)

指定 PCEP 会话暂停发送 Keepalive 报文

参数 **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pcep Stop Keep Alive | Sessions=${Sessions} |
```

static pim_change_gen_id(*Sessions*)

修改 PIM 协议会话的 GenId

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Join Group | Sessions=${Sessions} |
```

static pim_join_group(*Sessions*)

PIM 协议会话发送加入组数据包

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Join Group | Sessions=${Sessions} |
```

static pim_leave_group(*Sessions*)

PIM 协议会话发送离开组数据包

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Leave Group | Sessions=${Sessions} |
```

static pim_start_boot_strap(*Sessions*)

启动 PIM 协议会话 BootStrap

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Start Boot Strap | Sessions=${Sessions} |
```

static pim_start_register(*Sessions*)

PIM 协议会话开始发送 Register 消息

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Start Register | Sessions=${Sessions} |
```

static pim_stop_boot_strap(*Sessions*)

停止 PIM 协议会话 BootStrap

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Pim Stop Boot Strap | Sessions=${Sessions} |
```

static pim_stop_register(*Sessions*)

PIM 协议会话停止发送 Register 消息

参数 **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Pim Stop Register | Sessions=${Sessions} |
```

static relate_benchmark_ports(*Config*, *Ports*)

指定测试仪表测试套件使用的端口

参数

- **Config** (wizard_config) -- 仪表测试测试套件对象 object
- **Ports** (list (*Port*)) -- 测试仪表端口对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Items} | Create List | throughput | frameloss |
| ${Wizard} | ${Config} | Create Benchmark | Type=rfc2544 | Items=${Items}
↪ |
| Relate Benchmark Ports | Config=${Wizard} | Ports=${Ports} |
| Create Benchmark Streams | Config=${Wizard} | Items=@{RFC2544Items} |␣
↪Type=eth | SrcPoints=@{SrcPoints} | DstPoints=@{SrcPoints} |␣
↪Mode=meshed | Mapping=roundrobin |
| Edit Benchmark Learning | Configs=${Config} | Frequency=once |
| Edit Benchmark Duration | Config=${Config} | Count=${L2_TestTime} |
| Edit Benchmark Frame | Config=${Config} | Type=custom | Custom=@{L2_
↪FrameSize} |
| Edit Benchmark Search | Config=${Config} | Init=100 |
| Expand Benchmark | Config=${Wizard} |
```

static release_port(*Locations=None*, *Ports=None*)

释放测试仪表的端口

参数 **Locations** (*list*) -- 端口在测试仪表机箱硬件中的位置, //$(机箱 IP 地址)/$(板卡序号)/$(端口序号) (例如:[//192.168.0.1/1/1, //192.168.0.1/1/2])

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} == [//192.168.0.1/1/1, //192.168.0.1/1/2] |
| ${result} | Release Port | ${Locations} |
```

static relocate_ports(*Ports*, *Locations*)

逻辑端口迁移到执行的测试仪表的物理端口.

参数

- **Ports** (*list*) -- 端口对象的列表
- **Locations** (*list*) -- 端口在测试仪表机箱硬件中的位置, //$(机箱 IP 地址)/$(板卡序号)/$(端口序号) (例如: [//192.168.0.1/1/1, //192.168.0.1/1/2])

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} == [//192.168.0.1/1/1, //192.168.0.1/1/2] |
| ${result} | Relocate Ports | ${Ports} | ${Locations} |
```

static request_ldp_label(*Configs*)

LDP 协议会话 LSP 请求标签

参数 **Configs** (*list*) -- LDP LSP 对象列表, 类型为: list

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

```
| Request Ldp Label | Configs=${Configs} |
```

static reserve_port(*Locations*, *Force=False*, *Debug=False*)

预约测试仪表的端口

参数

- **Locations** (*list*) -- 端口在测试仪表机箱硬件中的位置, //$(机箱 IP 地址)/$(板卡序号)/$(端口序号) (例如: [//192.168.0.1/1/1, //192.168.0.1/1/2])
- **Force** (*bool*) -- 强制占用测试仪表端口, 默认值: False
- **Debug** (*bool*) -- 调试模式, 只创建离线端口, 默认值: False

返回 端口对象列表

返回类型 list

实际案例

robotframework:

```
| ${Locations} == [//192.168.0.1/1/1, //192.168.0.1/1/2] |
| ${result} | Reserve Port | ${Locations} |
```

static reset_statistic()

重置测试仪表已缓存统计

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Reset Statistic |
```

static reset_tester()

清空测试仪表所有配置

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Reset Tester |
```

static restart_ldp(*Sessions*)

重启 LDP 协议会话

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Restart Ldp | Sessions=${Sessions} |
```

static resume_ldp_hello(*Sessions*)

恢复 LDP 协议会话 Hello 发送

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Ldp Hello | Sessions=${Sessions} |
```

static resume_ldp_keepalive(*Sessions*)

恢复 LDP 协议会话 Keepalive 发送

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Ldp Keepalive | Sessions=${Sessions} |
```

static resume_lsp_ping(*Sessions*)

继续发送 LSP Ping 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Lsp Ping | Sessions=${Sessions} |
```

static resume_lsp_trace(*Sessions*)

继续发送 LSP Trace 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Lsp Trace | Sessions=${Sessions} |
```

static resume_rip(*Sessions*)

恢复 RIP 协议

参数 **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Resume Rip | Sessions=${Sessions} |
```

static run_benchmark(*Mode=0*, *Timer=1800*, *Analyzer=False*)

执行测试仪表配置中的测试套件

参数

- **Mode** (*int*) -- 智能脚本的执行模式, (取值, 0 / 1) 0: 连续执行, 1: 单步执行
- **Timer** (*int*) -- 测试套件执行超时时间, (单位, 秒), 默认值: 1800 秒
- **Analyzer** (*bool*) -- 结果分析器开关, 布尔值 Bool (范围: True / False), 默认值: False

返回 测试结果 DB 文件的绝对路径

返回类型 str

实际案例

robotframework:

```
| ${result} | Run Benchmark |
| ${result} | Run Benchmark | Mode==1 |
| ${result} | Run Benchmark | Timer==3600 |
| ${result} | Run Benchmark | Analyzer==True |
| ${result} | Run Benchmark | Mode==1 | Timer==3600 | Analyzer==True |
```

static save_case(*Path*)

测试仪表保存配置文件

参数 **Path** (*str*) -- 配置文件路径, 例如: "C:/test.xcfg"

返回 布尔值 Bool (范围: True / False)

返回类型 bool

实际案例

robotframework:

```
| ${result} | Save Case | Path=='C:/test.xcfg' |
```

static save_result(*Path*, *FileName*)

保存测试仪表统计结果到 DB 文件

参数

- **Path** (*str*) -- 保存文件的路径
- **FileName** (*str*) -- 保存文件名称

返回 返回保存的 DB 文件的绝对路径字符串

返回类型 str

实际案例

robotframework:

```
| @{Subscribe} | Create List | PortStats | StreamBlockStats |
| Subscribe Result | Types=@{Subscribe} |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| ${DB} | Save Result | Path=D: est | FileName=test |
```

static select_interface(*Session*, *Interface*)

协议绑定测试仪表接口

参数

- **Session** (*object*) -- 测试仪表协议对象
- **Interface** (*object*) -- 测试仪表接口对象

返回 布尔值 Bool (范围：True / False)

实际案例

```
| Select Interface | Session=${Session} | Interface=${Interface} |
```

static select_rx_port(*Streams*, *RxPorts*, *Mode=1*, *ExcludeTxPort=True*)

选择流量的收端口

参数

- **Streams** (list (StreamTemplate)) -- 测试仪表流量对象 object 列表
- **RxPorts** (list (*Port*)) -- 指定流量收端口对象列表
- **Mode** -- 模式, 默认值：ONE_TO_ONE，支持类型

  ONE_TO_ONE

  ONE_TO_MANY

  MANY_TO_ONE

  PAIR
- **ExcludeTxPort** (*bool*) -- 是否包括流量发送端口

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Select Rx Port | Streams=${Streams} | RxPorts=${Ports} | Mode=ONE_TO_
↪MANY | ExcludeTxPort=True |
```

static select_source_interface(*Session*, *Memberships*, *Interface*)

将协议会话组播组过滤源地址绑定到指定接口

参数

- **Session** (Igmp) -- IGMP/MLD 协会话对象, 类型为：object

- **Memberships** (MldMembershipsConfig) -- 组播协议和组播组绑定关系对象, 类型为：object
- **Interface** (Interface) -- 测试仪表接口对象, 类型为：object

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| ${Interface} |  create_interface | Port=${Port} |
| ${Group} | Create Multicast Group | Version=IPV4 | Start=225.0.1.1 |␣
↪Number=20 |
| ${Session} | Create Igmp | Port=${Port} | Version=IGMPV3 |
| ${Memberships} | Create Memberships | Session=${Session} | Start=225.0.
↪1.1 | DeviceGroupMapping=ONETOONE |
| binding_multicast_group | Session=${Session} | Memberships=$
↪{Memberships} | MulticastGroup=${Group} |
| Select Source Interface | Session=${Session} | Memberships=$
↪{Memberships} | Interface=${Interface} |
```

static start_arp(*Ports=None*, *Interfaces=None*)

测试仪表启动接口 ARP 功能

:param： param Ports: 端口对象的列表:param： type Ports: 类型为：list :param： param Interfaces: 接口对象的列表:param： type Interfaces: 类型为：list，当 Ports 和 Interfaces 都为 None 时，表示启动所有接口的 ARP

Returns: 布尔值 Bool (范围：True / False)

Examples: robotframework:

```
| Start Arp |
```

static start_capture(*Ports=None*)

启动测试仪表端口数据抓包

参数 **Ports** (*list*) -- 测试仪仪表端口 Port 对象列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 | //192.168.0.1/1/2 |
| ${Ports} | Reserve Ports | ${Ports} | ${Locations} |
| Start Capture | Ports=${Ports} |
```

static start_l2_learning(*Type=None*, *Ports=None*, *Streams=None*, *WaitLearningDone=True*, *WaitTime=30*)

启动测试仪表流量二层学习

参数

- **Type** (*str*) -- 启动流二层学习的的类型，类型为：string，支持：Tx 或 Rx
- **Ports** (list (*Port*)) -- 测试仪表端口对象 object 列表，类型为：list，端口对象 object 列表

- **Streams** (list (StreamTemplate)) -- 测试仪表流量模板对象 object 列表，类型为：list，目流量模板对象 object 列表
- **WaitLearningDone** (*bool*) -- 是否等待二层学习完成，类型为：Bool (范围：True / False)
- **WaitTime** (*int*) -- 等待二层学习完成时间，类型为：number，默认值：30 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start L2 Learning |
| Sleep | 10 |
| Stop L2 Learning |
| Sleep | 3 |
```

static start_l3_learning(*Ports=None*, *Streams=None*)

启动测试仪表流量三层 ARP ND 学习

参数

- **Ports** (list (*Port*)) -- 测试仪表端口对象 object 列表，类型为：list，端口对象 object 列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量模板对象 object 列表，类型为：list，目流量模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start L3 Learning |
| Sleep | 10 |
| Stop L3 Learning |
| Sleep | 3 |
```

static start_ldp(*Sessions*)

启动 LDP 协议会话

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Start Ldp | Sessions=${Sessions} |
```

static start_lsp_ping(*Sessions*)

开始发送 LSP Ping 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Start Lsp Ping | Sessions=${Sessions} |
```

static start_protocol(*Ports=None*, *Protocols=None*, *Objects=None*)

测试仪表启动协议

参数

- **Ports** (*list*) -- 端口对象的列表
- **Protocols** (*list*) -- 指定需要启动的协议类型列表, 目前支持的协议类型：

  ospfv2

  ospfv3

  dhcpv4

  dhcpv6

  vxlan

- **Objects** (*list*) -- 当 Ports 和 Protocols 都为 None 时，可以指定协议会话对象列表

返回 布尔值 Bool (范围：True / False)

实际案例

```
| Start Protocol |
```

static start_stream(*Type=None*, *Objects=None*)

测试仪表开始发送数据流

参数

- **Type** (*str*) -- 当值为 None 时发送所有流量，按指定端口发送数据流或者按指定流模板发送数据流, 类型 string(”port” 或”stream”),
- **Objects** (*list*) -- 按指定端口发送数据流或者按指定流模板发送数据流时需要指定端口对象或流模板对象列表

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| @{Objects} | Get Ports |
| Start Stream | Type=port | Objects=@{Objects} |
| Sleep | 10 |
| Stop Stream | Type=port | Objects=@{Objects} |
```

static start_vxlan_ping(*Interfaces*, ***kwargs*)

启动 Vxlan Ping

参数 **Interfaces** (Interface) -- Interface 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Start Vxlan Ping | Interfaces=${Interface} |
```

static stop_arp(*Ports=None*, *Interfaces=None*)

测试仪表停止接口 ARP 功能

:param： param Ports: 端口对象的列表:param： type Ports: 类型为：list :param： param Interfaces: 接口对象的列表:param： type Interfaces: 类型为：list，当 Ports 和 Interfaces 都为 None 时，表示启动所有接口的 ARP

Returns: 布尔值 Bool (范围：True / False)

Examples: robotframework:

```
| Stop Arp |
```

static stop_capture(*Ports=None*)

停止测试仪表端口数据抓包

参数 **Ports** (*list*) -- 测试仪仪表端口 Port 对象列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| ${Locations} | Create List | //192.168.0.1/1/1 | //192.168.0.1/1/2 |
| ${Ports} | Reserve Ports | ${Ports} | ${Locations} |
| Start Capture | Ports=${Ports} |
| Sleep | 30 |
| Stop Capture | Ports=${Ports} |
```

static stop_l2_learning(*Ports=None*, *Streams=None*)

停止测试仪表流量二层学习

参数

- **Ports** (list (*Port*)) -- 测试仪表端口对象 object 列表，类型为：list，端口对象 object 列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量模板对象 object 列表，类型为：list，目流量模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start L2 Learning |
| Sleep | 10 |
| Stop L2 Learning |
| Sleep | 3 |
```

static stop_l3_learning(*self*, *Ports=None*, *Streams=None*)

停止测试仪表流量三层 ARP ND 学习

参数

- **Ports** (list (*Port*)) -- 测试仪表端口对象 object 列表，类型为：list，端口对象 object 列表
- **Streams** (list (StreamTemplate)) -- 测试仪表流量模板对象 object 列表，类型为：list，目流量模板对象 object 列表

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start L3 Learning |
| Sleep | 10 |
| Stop L3 Learning |
| Sleep | 3 |
```

static stop_ldp(*Sessions*)

停止 LDP 协议会话

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Ldp | Sessions=${Sessions} |
```

static stop_ldp_hello(*Sessions*)

停止 LDP 协议会话 Hello 发送

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Ldp Hello | Sessions=${Sessions} |
```

static stop_ldp_keepalive(*Sessions*)

停止 LDP 协议会话 Keepalive 发送

参数 **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Ldp Keepalive | Sessions=${Sessions} |
```

static stop_lsp_ping(*Sessions*)

停止发送 LSP Ping 消息

参数 **Sessions** (LspPing) -- LspPing 会话对象, 类型为：object / list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Lsp Ping | Sessions=${Sessions} |
```

static stop_protocol(*Ports=None*, *Protocols=None*, *Objects=None*)

测试仪表停止协议

参数

- **Ports** (*list*) -- 端口对象的列表
- **Protocols** (*list*) -- 指定需要停止的协议类型列表，目前支持的协议类型：

  ospfv2

  ospfv3

  dhcpv4

  dhcpv6

  vxlan
- **Objects** (*list*) -- 当 Ports 和 Protocols 都为 None 时，可以指定协议会话对象列表

返回 布尔值 Bool (范围：True / False)

实际案例

```
| Stop Protocol |
```

static stop_stream(*Type=None*, *Objects=None*)

测试仪表停止发送数据流

参数

- **Type** (*str*) -- 当值为 None 时发送所有流量，按指定端口发送数据流或者按指定流模板停止数据流, 类型 string(”port” 或”stream”),
- **Objects** (*list*) -- 按指定端口停止数据流或者按指定流模板发送数据流时需要指定端口对象或流模板对象列表

返回 (范围：True / False)

返回类型 bool

实际案例

```
| Subscribe Result |
| Start Stream |
| Sleep | 10 |
| Stop Stream |
| Sleep | 3 |
| @{Objects} | Get Ports |
| Start Stream | Types=port | Objects=@{Objects} |
| Sleep | 10 |
| Stop Stream | Types=port | Objects=@{Objects} |
```

static stop_vxlan_ping(*Interfaces*, ***kwargs*)

停止 Vxlan Ping

参数 **Interfaces** (Interface) -- Interface 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Stop Vxlan Ping | Interfaces=${Interface} |
```

static subscribe_result(*Types=None*)

订阅测试仪表统计视图

参数 **Types** (*list*) -- 需要订阅测试仪表统计视图列表，当传入为 None 时，订阅当前配置的所有视图，目前支持的统计视图：

PortStats

PortAvgLatencyStats

StreamStats

StreamTxStats

StreamRxStats

StreamBlockStats

StreamBlockTxStats

StreamBlockRxStats

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| @{Types} | Create List | PortStats | StreamBlockStats |
| Subscribe Result | Types=${Types} |
```

static suspend_rip(*Sessions*)

暂停 RIP 协议

参数 **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Suspend Rip | Sessions=${Sessions} |
```

static wait_bfd_state(*Sessions*, *State='RUNNING'*, *Interval=1*, *TimeOut=60*)

等待 BFD 协议会话达到指定状态

参数

- **Sessions** (list (BfdRouter)) -- BFD 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BFD 协议会话达到的状态, 类型为：string, 默认值：达到 RUNNING, 支持下列状态：

  DISABLED

  NOT_STARTED

  IDLE

  RUNNING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bfd State | Sessions=${Sessions} | State=RUNNING | Interval=2 |␣
↪TimeOut=120 |
```

static wait_bgp_ipv4_router_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 BGP IPv4 会话组达到指定状态

参数

- **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BGP IPv4 会话组达到的状态, 类型为：string, 默认值：达到 ESTABLISHED, 支持下列状态：

  NOT_START

  IDLE

  CONNECT

  ACTIVE

  OPEN_SENT

  OPEN_CONFIRM

  ESTABLISHED
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bgp IPv4 Router State | Sessions=${Sessions} | State=ESTABLISHED |␣
↪Interval=2 | TimeOut=120 |
```

static wait_bgp_ipv6_router_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 BGP IPv6 会话组达到指定状态

参数

- **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BGP IPv6 会话组达到的状态, 类型为：string, 默认值：达到 ESTABLISHED, 支持下列状态：

  NOT_START

  IDLE

  CONNECT

  ACTIVE

  OPEN_SENT

  OPEN_CONFIRM

  ESTABLISHED

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bgp IPv6 Router State | Sessions=${Sessions} | State=ESTABLISHED |␣
↪Interval=2 | TimeOut=120 |
```

static wait_bgp_router_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 BGP 会话组达到指定状态

参数

- **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BGP IPv4 会话组达到的状态, 类型为：string, 默认值：达到 ESTABLISHED, 支持下列状态：

  NOT_START

  IDLE

  CONNECT

  ACTIVE

  OPEN_SENT

  OPEN_CONFIRM

  ESTABLISHED
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bgp Router State | Sessions=${Sessions} | State=ESTABLISHED |␣
↪Interval=2 | TimeOut=120 |
```

static wait_bgp_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 BGP 协议会话达到指定状态

参数

- **Sessions** (BgpRouter) -- BGP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 BGP 协议会话达到的状态, 类型为：string, 默认值：达到 RUNNING, 支持下列状态：

  DISABLED

  NOT_START

  RUNNING

STARTING

STOPPING

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Bgp State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

static wait_dhcp_client_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 DHCP 协议会话达到指定状态

参数

- **Sessions** (*list*) -- DHCP 协议会话对象列表
- **State** (*str*) -- 等待 DHCP 协议会话达到的状态，默认值：BOUND，支持下列状态：

  NOT_READY

  IDLE

  BINDING

  BOUND

  RELEASING

  RENEWING

  REBINDING

  REBOOTING

- **Interval** (*number*) -- 查询 DHCP 协议会话的间隔，默认值：1 sec
- **TimeOut** (*number*) -- 等待 DHCP 协议会话状态的超时时间，默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcp Client State | Sessions=${Sessions} | State=${State} |␣
↪Interval=${Interval} | TimeOut=${TimeOut} |
```

static wait_dhcp_server_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 DHCP 服务器协议会话达到指定状态

参数

- **Sessions** (*list*) -- DHCP Server 协议会话对象列表

- **State** (*str*) -- 等待 DHCP Server 协议会话达到的状态，默认值：UP，支持下列状态：

  NOT_READY

  DOWN

  UP

- **Interval** (*number*) -- 查询 DHCP 协议会话的间隔，默认值：1 sec
- **TimeOut** (*number*) -- 等待 DHCP 协议会话状态的超时时间，默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcp Server State | Sessions=${Sessions} | State=${State} |␣
↪Interval=${Interval} | TimeOut=${TimeOut} |
```

static wait_dhcpv6_client_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Dhcpv6 客户端会话达到指定状态

参数

- **Session** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list
- **State** (*str*) -- 等待 DHCPv6 客户端会话组达到的状态, 类型为：string, 默认值：达到 BOUND, 支持下列状态：

  DISABLED

  IDLE

  BOUND

  SOLICITING

  REQUESTING

  RELEASING

  RENEWING

  REBINDING

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcpv6 Client State | Sessions=${Sessions} | State=BOUND |␣
↪Interval=2 | TimeOut=120 |
```

static wait_dhcpv6_pd_client_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Dhcpv6 PD 客户端会话达到指定状态

参数

- **Session** (Dhcpv6Client) -- DHCPv6 客户端会话对象, 类型为：object / list
- **State** (*str*) -- 等待 DHCPv6 客户端会话组达到的状态, 类型为：string, 默认值：达到 BOUND, 支持下列状态：

  DISABLED

  IDLE

  BOUND

  SOLICITING

  REQUESTING

  RELEASING

  RENEWING

  REBINDING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcpv6 Pd Client State | Sessions=${Sessions} | State=BOUND |␣
↪Interval=2 | TimeOut=120 |
```

static wait_dhcpv6_server_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Dhcpv6 服务端会话达到指定状态

参数

- **Session** (Dhcpv6Server) -- DHCPv6 服务端会话对象, 类型为：object / list
- **State** (*str*) -- 等待 DHCPv6 客户端会话组达到的状态, 类型为：string, 默认值：达到 UP, 支持下列状态：

  NOTSTART

  UP

  DISABLED
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec

- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dhcpv6 Server State | Sessions=${Sessions} | State=UP | Interval=2␣
↪| TimeOut=120 |
```

static wait_dot1x_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 802.1x 会话达到指定状态

参数

- **Session** (Dot1x) -- 802.1x 会话对象, 类型为：object / list
- **State** (*str*) -- 等待 802.1x 会话组达到的状态, 类型为：string, 默认值：达到 AUTHENTICATED, 支持下列状态：

  DISABLED

  DOWN

  UNAUTHORIZED

  AUTHENTICATING

  AUTHENTICATED

  FAILED

  LOGGING_OFF
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Dot1x State | Sessions=${Sessions} | State=AUTHENTICATED |␣
↪Interval=2 | TimeOut=120 |
```

static wait_igmp_querier_state(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 Igmp Querier 协议会话达到指定状态

参数

- **Sessions** (list (IgmpQuerier)) -- Igmp Querier 协议会话对象列表
- **State** (*str*) -- 等待 Igmp Querier 协议会话达到的状态, 默认值：达到 UP, 支持下列状态：

  NOTSTARTED

  UP
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec

- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Igmp Querier State | Sessions=${Sessions} | State=UP | Interval=2␣
↪| TimeOut=120 |
```

static wait_igmp_state(*Sessions*, *State='MEMBER'*, *Interval=1*, *TimeOut=60*)

等待 Igmp 协议会话达到指定状态

参数

- **Sessions** (list (Igmp)) -- Igmp 协议会话对象列表
- **State** (*str*) -- 等待 Igmp 协议会话达到的状态, 默认值：达到 MEMBER, 支持下列状态：

  NONMEMBER JOINING MEMBER LEAVING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Igmp State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

static wait_isis_l1_broadcast_adj_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 ISIS 协议会话 l1_broadcast_adj_state 达到指定状态

参数

- **Sessions** (list (IsisRouter)) -- ISIS 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 ISIS 协议会话 l1_broadcast_adj_state 达到的状态, 类型为：string, 默认值：达到 DISOTHER 或 DIS, 支持下列状态：

  NOTSTART

  IDLE

  INIT

  DISOTHER

  DIS

  GR

  GRHELPER

  NA
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec

- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Isis L1 Broadcast Adj State | Sessions=${Sessions} | State=RUNNING
↪| Interval=2 | TimeOut=120 |
```

static wait_isis_l2_broadcast_adj_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 ISIS 协议会话 l2_broadcast_adj_state 达到指定状态

参数

- **Sessions** (list (IsisRouter)) -- ISIS 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 ISIS 协议会话 l2_broadcast_adj_state 达到的状态, 类型为：string, 默认值：达到 DISOTHER 或 DIS, 支持下列状态：

  NOTSTART

  IDLE

  INIT

  DISOTHER

  DIS

  GR

  GRHELPER

  NA

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Isis L2 Broadcast Adj State | Sessions=${Sessions} | State=RUNNING
↪| Interval=2 | TimeOut=120 |
```

static wait_isis_state(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 ISIS 协议会话达到指定状态

参数

- **Sessions** (list (IsisRouter)) -- ISIS 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 ISIS 协议会话达到的状态, 类型为：string, 默认值：达到 UP, 支持下列状态：

  NOTSTART

  IDLE

INIT

UP

GR

GRHELPER

DISABLE

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Isis State | Sessions=${Sessions} | State=GR | Interval=2 |␣
↪TimeOut=120 |
```

static wait_isis_three_way_p2p_adj_state(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 ISIS 协议会话 three_way_p2p_adj 达到指定状态

参数

- **Sessions** (list (IsisRouter)) -- ISIS 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 ISIS 协议会话 three_way_p2p_adj 达到的状态, 类型为：string, 默认值：达到 UP, 支持下列状态：

  UP

  INIT

  DOWN

  NOTSTART

  NA

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Isis Three Way P2p Adj State | Sessions=${Sessions} | State=INIT |␣
↪Interval=2 | TimeOut=120 |
```

static wait_l2tp_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 L2tp 协议会话达到指定状态

参数

- **Sessions** (L2tp) -- L2tp 协议会话对象列表, 类型为：list

- **State** (*str*) -- 等待 L2tp 协议会话达到的状态, 类型为：string, 默认值：达到 CONNECTED, 支持下列状态：

  NONE IDLE CONNECTING CONNECTED DISCONNECTING

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait L2tp State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

static wait_ldp_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 LDP 协议会话达到指定状态

参数

- **Sessions** (*list*) -- LDP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 LDP 协议会话达到的状态, 类型为：string, 默认值：达到 OPERATIONAL, 支持下列状态：

  DISABLED

  NOT_STARTED

  NON_EXISTENT

  INITIAL

  OPEN_SENT

  OPEN_REC

  OPERATIONAL

  RESTARTING

  HELPING

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ldp State | Sessions=${Sessions} | State=RESTARTING | Interval=2 |␣
↪TimeOut=120 |
```

static wait_lsp_ping_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Lsp Ping 会话的 Ping 消息达到指定状态

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object / list
- **State** (*str*) -- 等待 Lsp Ping 会话组的 Ping 消息达到的状态, 类型为：string, 默认值：PAUSE_SEND, 支持下列状态：

  IDLE

  PAUSE_SEND

  RESUME_SEND
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Lsp Ping State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

static wait_lsp_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Lsp Ping 会话达到指定状态

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object / list
- **State** (*str*) -- 等待 Lsp Ping 会话组达到的状态, 类型为：string, 默认值：UP, 支持下列状态：

  DISABLE

  NOTSTART

  UP

  DOWN
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Lsp State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

static wait_lsp_trace_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 Lsp Ping 会话的 Trace 消息达到指定状态

参数

- **Session** (LspPing) -- Lsp Ping 会话对象, 类型为：object / list
- **State** (*str*) -- 等待 Lsp Ping 会话组的 Trace 消息达到的状态, 类型为：string, 默认值：PAUSE_SEND, 支持下列状态：

  IDLE

  PAUSE_SEND

  RESUME_SEND
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Lsp Trace State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

static wait_mld_querier_state(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 Mld Querier 协议会话达到指定状态

参数

- **Sessions** (list (MldQuerier)) -- Mld Querier 协议会话对象列表
- **State** (*str*) -- 等待 Mld Querier 协议会话达到的状态, 默认值：达到 UP, 支持下列状态：

  NOTSTARTED

  UP
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Mld Querier State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

static wait_mld_state(*Sessions*, *State='MEMBER'*, *Interval=1*, *TimeOut=60*)

等待 Mld 协议会话达到指定状态

参数

- **Sessions** (list (Mld)) -- Mld 协议会话对象列表
- **State** (*str*) -- 等待 Mld 协议会话达到的状态, 默认值：达到 MEMBER, 支持下列状态：

  NONMEMBER JOINING MEMBER LEAVING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Mld State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

static wait_ospf_adjacency_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 OSPFv2 或 OSPFv3 协议会话达到指定邻接状态

参数

- **Sessions** (list(OspfRouter)) or (list(Ospfv3Router)) -- OSPFv2 或 OSPFv3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 OSPFv2 协议会话达到的邻接状态, 类型为：string, 默认值：FULL, 支持下列状态：

  DOWN

  INIT

  TWOWAY

  EXSTART

  EXCHANGE

  LOADING

  FULL
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话邻接状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ospf Adjacency State | Sessions=${Sessions} | State=FULL |␣
↪Interval=2 | TimeOut=120 |
```

static wait_ospf_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 OSPFv2 或 OSPFv3 协议会话达到指定状态

参数

- **Sessions** (list(OspfRouter)) or (list(Ospfv3Router)) -- OSPFv2 或 OSPFv3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 OSPFv2 协议会话达到的状态, 类型为：string, 默认值：达到 DR 或 BACKUP 或 DROTHER, 支持下列状态：

  NOTSTART

  P2P

  WAITING

  DR

  BACKUP

  DROTHER

  DISABLE

  DOWN
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ospf State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

static wait_ospfv3_adjacency_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 OSPFv3 协议会话达到指定邻接状态

参数

- **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 OSPFv3 协议会话达到的邻接状态, 类型为：string, 默认值：FULL, 支持下列状态：

  DOWN

  INIT

  TWOWAY

  EXSTART

  EXCHANGE

  LOADING

FULL

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话邻接状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ospfv3 Adjacency State | Sessions=${Sessions} | State=FULL |␣
↪Interval=2 | TimeOut=120 |
```

static wait_ospfv3_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 OSPFv3 协议会话达到指定状态

参数

- **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 OSPFv3 协议会话达到的状态, 类型为：string, 默认值：达到 DR 或 BACKUP 或 DROTHER, 支持下列状态：

  NOTSTART

  P2P

  WAITING

  DR

  BACKUP

  DROTHER

  DISABLE

  DOWN

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Ospfv3 State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

static wait_pcep_state(*Sessions*, *State='UP'*, *Interval=1*, *TimeOut=60*)

等待 PCEP 协议会话达到指定状态

参数

- **Sessions** (Pcep) -- : PCEP 协议会话对象列表, 类型为：list

- **State** (*str*) -- 等待 PCEP 协议会话达到的状态, 类型为：string, 默认值：达到 UP, 支持下列状态：

  DISABLED

  IDLE

  PENDING

  UP

  CLOSING

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Pcep State | Sessions=${Sessions} | State=UP | Interval=2 |␣
↪TimeOut=120 |
```

static wait_pim_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 PIM 协议会话达到指定状态

参数

- **Sessions** (list (PimRouter)) -- PIM 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 PIM 协议会话达到的状态, 类型为：string, 默认值：达到 NEIGHBOR, 支持下列状态：

  DISABLED

  HELLO

  NEIGHBOR

  IDLE

  NOTSTARTED

- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Pim State | Sessions=${Sessions} | State=NEIGHBOR | Interval=2 |␣
↪TimeOut=120 |
```

static wait_port_state(*Ports=None*, *State=None*, *Interval=1*, *TimeOut=60*)

等待测试仪表端口链路达到指定状态

参数

- **Ports** (list(*Port*)) -- 测试仪表端口对象列表
- **State** (*str*) -- 测试仪表连接端口状态, 默认值 UP: DOWN UP
- **Interval** (*int*) -- 状态查询间隔, 默认值:1
- **TimeOut** (*int*) -- 超时时间, 默认值:60

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

robotframework:

```
| Edit Port | Ports=${Ports} | EnableLink=False |
| Wait Port State | Ports=${Ports} | State=DOWN |
```

static wait_pppoe_ipcp_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 PPPoE IPCP 达到指定状态

参数

- **Sessions** (list (PppoeClent)) -- PPPoE 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 PPPoE IPCP 达到的状态, 类型为：string, 默认值：达到 CONNECTED, 支持下列状态：

  NONE

  IDLE

  CONNECTED

  CONNECTING

  DISCONNECTING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Pppoe State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

static wait_pppoe_ipv6cp_state(*Sessions*, *State=None*, *Interval=1*, *TimeOut=60*)

等待 PPPoE IPv6CP 达到指定状态

参数

- **Sessions** (list (PppoeClent)) or (list (Pppoev3Router)) -- PPPoE 或 Pppoev3 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 PPPoE IPv6CP 达到的状态, 类型为：string, 默认值：达到 CONNECTED, 支持下列状态：

  NONE

  IDLE

  CONNECTED

  CONNECTING

  DISCONNECTING
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Pppoe State | Sessions=${Sessions} | State=DR | Interval=2 |␣
↪TimeOut=120 |
```

static wait_rip_state(*Sessions*, *State='OPEN'*, *Interval=1*, *TimeOut=60*)

等待 RIP 协议会话达到指定状态

参数

- **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list
- **State** (*str*) -- 等待 RIP 协议会话达到的状态, 类型为：string, 默认值：达到 OPEN, 支持下列状态：

  DISABLED

  NOTSTART

  CLOSED

  OPEN

  SUSPENDED
- **Interval** (*int*) -- 查询协议会话的间隔, 类型为：number, 默认值：1 sec
- **TimeOut** (*int*) -- 等待协议会话状态的超时时间, 类型为：number, 默认值：60 sec

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Wait Rip State | Sessions=${Sessions} | State=OPEN | Interval=2 |␣
↪TimeOut=120 |
```

static wait_stream_state(*Stream=None*, *State=['READY']*, *TimeOut=60*)

测试仪表停止发送数据流

参数

- **Stream** (list (StreamTemplate)) -- 测试仪表流模板对象列表
- **State** (*list*) -- 流模板状态, 默认值 READY, 支持:
  DISABLED NOTREADY READY RUNNING STOPPED PAUSED
- **TimeOut** (*int*) -- 超时时间, 默认值:60

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Start Stream |
| Sleep | 10 |
| Wait Stream State |
| Sleep | 3 |
```

static withdraw_bgp(*Sessions*)

撤销 BGP 协议会话 lsa

参数 **Sessions** (BgpRouter) -- OSPFv3 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Bgp | Sessions=${Sessions} |
```

static withdraw_bgp_route(*Routes*)

撤销 BGP 协议指定 lsa

参数 **Routes** (*list*) -- BGP 协议路由对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Bgp Route | Routes=${Routes} |
```

static withdraw_isis(*Lsps*)

通告 Isis 协议会话 lsp

参数 **Lsps** (IsisLspConfig) -- ISIS LSP 对象, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Isis | Lsp=${Lsp} |
```

static withdraw_ldb_label(*Configs*)

撤销 LDP 协议会话 LSP 标签

参数 **Configs** (*list*) -- LDP LSP 对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Ldp Label | Configs=${Configs} |
```

static withdraw_ospf_lsa(*Sessions=None*, *Type=None*, *Lsa=None*)

撤销 OSPFv2 协议会话 lsa

参数

- **Sessions** (list(OspfRouter)) -- OSPFv2 协议会话对象列表, 类型为：list
- **Type** (*str*) -- OSPFv2 lsa 类型, 类型为：string, 支持的 lsa 类型：

  Router

  Network

  Summary

  AsbrSummary

  External
- **Lsa** (*list*) -- OSPFv2 lsa 列表, 类型为：list, 当 Type=None 时参数生效

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Ospf Lsa | Sessions=${Sessions} | Type=router |
| Withdraw Ospf Lsa | Sessions=${Sessions} | Lsa=${Lsas} |
```

static withdraw_ospfv3_lsa(*Sessions=None*, *Type=None*, *Lsa=None*)

撤销 OSPFv3 协议会话 lsa

参数

- **Sessions** (Ospfv3Router) -- OSPFv3 协议会话对象列表, 类型为：list
- **Type** (*str*) -- OSPFv3 lsa 类型, 类型为：string, 支持的 lsa 类型：

  router

  network

  InterAreaPrefix

  InterAreaRouter

  AsExternal
- **Lsa** (*list*) -- OSPFv3 lsa 列表, 类型为：list, 当 Type=None 时参数生效

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Ospfv3 Lsa | Sessions=${Sessions} | Type=router |
| Withdraw Ospfv3 Lsa | Sessions=${Sessions} | Lsa=${Lsas} |
```

static withdraw_rip(*Sessions*)

撤销 RIP 协议通告路由

参数 **Sessions** (RipRouter) -- RIP 协议会话对象列表, 类型为：list

返回 布尔值 Bool (范围：True / False)

返回类型 bool

实际案例

```
| Withdraw Rip | Sessions=${Sessions} |
```

1.4 TesterLibrary.data module

1.5 Module contents

2

Indices and tables

- genindex
- modindex
- search

Python 模块索引

t

TesterLibrary, 984
TesterLibrary.base, 499
TesterLibrary.data, 984
TesterLibrary.Overall, 5
TesterLibrary.Overall.common, 1
TesterLibrary.Port, 21
TesterLibrary.Port.capture, 5
TesterLibrary.Port.common, 11
TesterLibrary.Port.interface, 17
TesterLibrary.Protocol, 374
TesterLibrary.Protocol.bfd, 21
TesterLibrary.Protocol.bgp, 36
TesterLibrary.Protocol.common, 102
TesterLibrary.Protocol.dhcpv4, 104
TesterLibrary.Protocol.dhcpv6, 117
TesterLibrary.Protocol.dot1x, 139
TesterLibrary.Protocol.igmp, 144
TesterLibrary.Protocol.isis, 151
TesterLibrary.Protocol.l2tp, 186
TesterLibrary.Protocol.ldp, 197
TesterLibrary.Protocol.lsp_ping, 210
TesterLibrary.Protocol.mld, 221
TesterLibrary.Protocol.multicast, 227
TesterLibrary.Protocol.ospfv2, 229
TesterLibrary.Protocol.ospfv3, 260
TesterLibrary.Protocol.pcep, 294
TesterLibrary.Protocol.pim, 334
TesterLibrary.Protocol.pppoe, 346
TesterLibrary.Protocol.rip, 361
TesterLibrary.Protocol.vxlan, 369
TesterLibrary.Statistic, 388
TesterLibrary.Statistic.common, 374
TesterLibrary.Stream, 468
TesterLibrary.Stream.common, 456
TesterLibrary.Stream.Header, 456
TesterLibrary.Stream.Header.Access, 390
TesterLibrary.Stream.Header.Access.common, 388
TesterLibrary.Stream.Header.Access.l2tpv2, 389
TesterLibrary.Stream.Header.Basic, 390
TesterLibrary.Stream.Header.Basic.common, 390
TesterLibrary.Stream.Header.Gre, 391
TesterLibrary.Stream.Header.Gre.common, 390
TesterLibrary.Stream.Header.L2, 409
TesterLibrary.Stream.Header.L2.common, 391
TesterLibrary.Stream.Header.L2.isis, 393
TesterLibrary.Stream.Header.L3, 442
TesterLibrary.Stream.Header.L3.common, 409
TesterLibrary.Stream.Header.L3.icmpv4, 413
TesterLibrary.Stream.Header.L3.icmpv6, 424
TesterLibrary.Stream.Header.L3.igmp, 437
TesterLibrary.Stream.Header.L4, 443
TesterLibrary.Stream.Header.L4.common, 442
TesterLibrary.Stream.Header.Routing, 456
TesterLibrary.Stream.Header.Routing.ospfv2, 443
TesterLibrary.Stream.imix, 466
TesterLibrary.Wizard, 499
TesterLibrary.Wizard.benchmark, 468
TesterLibrary.Wizard.mpls, 489

索引

A

abort_dot1x() (*TesterLibrary.base.TesterLibrary* 静态方法), 499
abort_dot1x() (在 *TesterLibrary.Protocol.dot1x* 模块中), 139
abort_l2tp() (*TesterLibrary.base.TesterLibrary* 静态方法), 500
abort_l2tp() (在 *TesterLibrary.Protocol.l2tp* 模块中), 186
abort_pppoe() (*TesterLibrary.base.TesterLibrary* 静态方法), 500
abort_pppoe() (在 *TesterLibrary.Protocol.pppoe* 模块中), 346
abort_request_ldp_label() (*TesterLibrary.base.TesterLibrary* 静态方法), 500
abort_request_ldp_label() (在 *TesterLibrary.Protocol.ldp* 模块中), 197
add_imix_distribution_frame() (*TesterLibrary.base.TesterLibrary* 静态方法), 500
add_imix_distribution_frame() (在 *TesterLibrary.Stream.imix* 模块中), 466
add_stream() (*TesterLibrary.base.TesterLibrary* 静态方法), 501
add_stream() (在 *TesterLibrary.Stream.common* 模块中), 456
advertise_bgp() (*TesterLibrary.base.TesterLibrary* 静态方法), 502
advertise_bgp() (在 *TesterLibrary.Protocol.bgp* 模块中), 36
advertise_bgp_route() (*TesterLibrary.base.TesterLibrary* 静态方法), 502
advertise_bgp_route() (在 *TesterLibrary.Protocol.bgp* 模块中), 36
advertise_isis() (*TesterLibrary.base.TesterLibrary* 静态方法), 503
advertise_isis() (在 *TesterLibrary.Protocol.isis* 模块中), 151
advertise_ldp_label() (*TesterLibrary.base.TesterLibrary* 静态方法), 503
advertise_ldp_label() (在 *TesterLibrary.Protocol.ldp* 模块中), 197
advertise_ospf_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 503
advertise_ospf_lsa() (在 *TesterLibrary.Protocol.ospfv2* 模块中), 229
advertise_ospfv3_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 504
advertise_ospfv3_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 260
advertise_rip() (*TesterLibrary.base.TesterLibrary* 静态方法), 504
advertise_rip() (在 *TesterLibrary.Protocol.rip* 模块中), 361
API (*TesterLibrary.base.TesterLibrary property*), 499
apply_igmp_querier() (*TesterLibrary.base.TesterLibrary* 静态方法), 504
apply_igmp_querier() (在 *TesterLibrary.Protocol.igmp* 模块中), 144
apply_mld_querier() (*TesterLibrary.base.TesterLibrary* 静态方法), 505
apply_mld_querier() (在 *TesterLibrary.Protocol.mld* 模块中), 221

B

benchmark_stream_use_exist() (*TesterLibrary.base.TesterLibrary* 静态方法), 505
benchmark_stream_use_exist() (在 *TesterLibrary.Wizard.benchmark* 模块中), 468
bfd_admin_down() (*TesterLibrary.base.TesterLibrary* 静态方法), 505
bfd_admin_down() (在 *TesterLibrary.Protocol.bfd* 模块中), 21
bfd_admin_up() (*TesterLibrary.base.TesterLibrary* 静态方法), 506
bfd_admin_up() (在 *TesterLibrary.Protocol.bfd* 模块中), 21
bfd_disable_demand_mode() (*TesterLibrary.base.TesterLibrary* 静态方法), 506
bfd_disable_demand_mode() (在 *TesterLibrary.Protocol.bfd* 模块中), 21
bfd_enable_demand_mode() (*TesterLibrary.base.TesterLibrary* 静态方法), 506
bfd_enable_demand_mode() (在 *TesterLibrary.Protocol.bfd* 模块中), 22
bfd_initiate_poll() (*TesterLibrary.base.TesterLibrary* 静态方法), 506
bfd_initiate_poll() (在 *TesterLibrary.Protocol.bfd* 模块中), 22

bfd_resume_pdus() (*TesterLibrary.base.TesterLibrary* 静态方法), 507
bfd_resume_pdus() (在 *TesterLibrary.Protocol.bfd* 模块中), 22
bfd_set_diagnostic_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 507
bfd_set_diagnostic_state() (在 *TesterLibrary.Protocol.bfd* 模块中), 22
bfd_stop_pdus() (*TesterLibrary.base.TesterLibrary* 静态方法), 507
bfd_stop_pdus() (在 *TesterLibrary.Protocol.bfd* 模块中), 23
bind_stream_imix() (*TesterLibrary.base.TesterLibrary* 静态方法), 508
bind_stream_imix() (在 *TesterLibrary.Stream.imix* 模块中), 467
binding_multicast_group() (*TesterLibrary.base.TesterLibrary* 静态方法), 508
binding_multicast_group() (在 *TesterLibrary.Protocol.multicast* 模块中), 227
binding_vxlan_multicast_group() (*TesterLibrary.base.TesterLibrary* 静态方法), 508
binding_vxlan_multicast_group() (在 *TesterLibrary.Protocol.vxlan* 模块中), 369
binding_vxlan_vm() (*TesterLibrary.base.TesterLibrary* 静态方法), 509
binding_vxlan_vm() (在 *TesterLibrary.Protocol.vxlan* 模块中), 369
binding_vxlan_vtep() (*TesterLibrary.base.TesterLibrary* 静态方法), 509
binding_vxlan_vtep() (在 *TesterLibrary.Protocol.vxlan* 模块中), 369

C

clear_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 509
clear_result() (在 *TesterLibrary.Statistic.common* 模块中), 374
connect_bgp() (*TesterLibrary.base.TesterLibrary* 静态方法), 510
connect_bgp() (在 *TesterLibrary.Protocol.bgp* 模块中), 36
connect_chassis() (*TesterLibrary.base.TesterLibrary* 静态方法), 510
connect_chassis() (在 *TesterLibrary.Overall.common* 模块中), 1
connect_l2tp() (*TesterLibrary.base.TesterLibrary* 静态方法), 510
connect_l2tp() (在 *TesterLibrary.Protocol.l2tp* 模块中), 186
connect_pppoe() (*TesterLibrary.base.TesterLibrary* 静态方法), 510
connect_pppoe() (在 *TesterLibrary.Protocol.pppoe* 模块中), 346
create_benchmark() (*TesterLibrary.base.TesterLibrary* 静态方法), 511
create_benchmark() (在 *TesterLibrary.Wizard.benchmark* 模块中), 469
create_benchmark_streams() (*TesterLibrary.base.TesterLibrary* 静态方法), 512
create_benchmark_streams() (在 *TesterLibrary.Wizard.benchmark* 模块中), 470
create_bfd() (*TesterLibrary.base.TesterLibrary* 静态方法), 513
create_bfd() (在 *TesterLibrary.Protocol.bfd* 模块中), 23
create_bfd_ipv4_session() (*TesterLibrary.base.TesterLibrary* 静态方法), 514
create_bfd_ipv4_session() (在 *TesterLibrary.Protocol.bfd* 模块中), 24
create_bfd_ipv6_session() (*TesterLibrary.base.TesterLibrary* 静态方法), 515
create_bfd_ipv6_session() (在 *TesterLibrary.Protocol.bfd* 模块中), 25
create_bgp() (*TesterLibrary.base.TesterLibrary* 静态方法), 516
create_bgp() (在 *TesterLibrary.Protocol.bgp* 模块中), 36
create_bgp_capability() (*TesterLibrary.base.TesterLibrary* 静态方法), 518
create_bgp_capability() (在 *TesterLibrary.Protocol.bgp* 模块中), 38
create_bgp_evpn_ethernet_segment_routes() (*TesterLibrary.base.TesterLibrary* 静态方法), 518
create_bgp_evpn_ethernet_segment_routes() (在 *TesterLibrary.Protocol.bgp* 模块中), 39
create_bgp_evpn_inclusive_multicast_routes() (*TesterLibrary.base.TesterLibrary* 静态方法), 519
create_bgp_evpn_inclusive_multicast_routes() (在 *TesterLibrary.Protocol.bgp* 模块中), 40
create_bgp_evpn_ip_prefix_routes() (*TesterLibrary.base.TesterLibrary* 静态方法), 521
create_bgp_evpn_ip_prefix_routes() (在 *TesterLibrary.Protocol.bgp* 模块中), 42
create_bgp_evpn_mac_ip_routes() (*TesterLibrary.base.TesterLibrary* 静态方法), 523
create_bgp_evpn_mac_ip_routes() (在 *TesterLibrary.Protocol.bgp* 模块中), 44
create_bgp_evpn_route_ad() (*TesterLibrary.base.TesterLibrary* 静态方法), 526
create_bgp_evpn_route_ad() (在 *TesterLibrary.Protocol.bgp* 模块中), 47
create_bgp_flow_spec_conponent_type() (*TesterLibrary.base.TesterLibrary* 静态方法), 528
create_bgp_flow_spec_conponent_type() (在 *TesterLibrary.Protocol.bgp* 模块中), 49
create_bgp_flow_spec_custom_path_attribute() (*TesterLibrary.base.TesterLibrary* 静态方法), 529
create_bgp_flow_spec_custom_path_attribute() (在 *TesterLibrary.Protocol.bgp* 模块中), 50
create_bgp_flow_specs_actions() (*TesterLibrary.base.TesterLibrary* 静态方法), 530
create_bgp_flow_specs_actions() (在 *TesterLibrary.Protocol.bgp* 模块中), 51
create_bgp_ipv4_flow_specs() (*TesterLibrary.base.TesterLibrary* 静态方法), 531
create_bgp_ipv4_flow_specs() (在 *TesterLibrary.Protocol.bgp* 模块中), 52
create_bgp_ipv4_flowspec_performance() (*TesterLibrary.base.TesterLibrary* 静态方法), 532
create_bgp_ipv4_flowspec_performance() (在 *TesterLibrary.Overall.common* 模块中), 1

create_bgp_ipv4_route_pool() (*TesterLibrary.base.TesterLibrary* 静态方法), 533
create_bgp_ipv4_route_pool() (在 *TesterLibrary.Protocol.bgp* 模块中), 53
create_bgp_ipv4_vpls() (*TesterLibrary.base.TesterLibrary* 静态方法), 537
create_bgp_ipv4_vpls() (在 *TesterLibrary.Protocol.bgp* 模块中), 57
create_bgp_ipv6_flow_spec() (*TesterLibrary.base.TesterLibrary* 静态方法), 538
create_bgp_ipv6_flow_spec() (在 *TesterLibrary.Protocol.bgp* 模块中), 58
create_bgp_ipv6_flow_spec_action() (*TesterLibrary.base.TesterLibrary* 静态方法), 540
create_bgp_ipv6_flow_spec_action() (在 *TesterLibrary.Protocol.bgp* 模块中), 60
create_bgp_ipv6_route_pool() (*TesterLibrary.base.TesterLibrary* 静态方法), 541
create_bgp_ipv6_route_pool() (在 *TesterLibrary.Protocol.bgp* 模块中), 61
create_bgp_ipv6_vpls() (*TesterLibrary.base.TesterLibrary* 静态方法), 544
create_bgp_ipv6_vpls() (在 *TesterLibrary.Protocol.bgp* 模块中), 65
create_bgp_link_states() (*TesterLibrary.base.TesterLibrary* 静态方法), 546
create_bgp_link_states() (在 *TesterLibrary.Protocol.bgp* 模块中), 66
create_bgp_link_states_link() (*TesterLibrary.base.TesterLibrary* 静态方法), 550
create_bgp_link_states_link() (在 *TesterLibrary.Protocol.bgp* 模块中), 70
create_bgp_link_states_prefix() (*TesterLibrary.base.TesterLibrary* 静态方法), 557
create_bgp_link_states_prefix() (在 *TesterLibrary.Protocol.bgp* 模块中), 78
create_bgp_link_states_prefix_sr_range_sub_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 561
create_bgp_link_states_prefix_sr_range_sub_tlv() (在 *TesterLibrary.Protocol.bgp* 模块中), 82
create_bgp_link_states_srv6_sid() (*TesterLibrary.base.TesterLibrary* 静态方法), 562
create_bgp_link_states_srv6_sid() (在 *TesterLibrary.Protocol.bgp* 模块中), 83
create_bgp_random_route() (*TesterLibrary.base.TesterLibrary* 静态方法), 564
create_bgp_random_route() (在 *TesterLibrary.Protocol.bgp* 模块中), 85
create_bgp_route_pool_custom_path_attribute() (*TesterLibrary.base.TesterLibrary* 静态方法), 564
create_bgp_route_pool_custom_path_attribute() (在 *TesterLibrary.Protocol.bgp* 模块中), 85
create_bgp_segement_sub_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 565
create_bgp_segement_sub_tlv() (在 *TesterLibrary.Protocol.bgp* 模块中), 86
create_bgp_sr_te_policy() (*TesterLibrary.base.TesterLibrary* 静态方法), 567
create_bgp_sr_te_policy() (在 *TesterLibrary.Protocol.bgp* 模块中), 88
create_bgp_sr_te_policy_Segement_list() (*TesterLibrary.base.TesterLibrary* 静态方法), 571
create_bgp_sr_te_policy_Segement_list() (在 *TesterLibrary.Protocol.bgp* 模块中), 91
create_capture_byte_pattern() (*TesterLibrary.base.TesterLibrary* 静态方法), 571
create_capture_byte_pattern() (在 *TesterLibrary.Port.capture* 模块中), 5
create_capture_pdu_pattern() (*TesterLibrary.base.TesterLibrary* 静态方法), 572
create_capture_pdu_pattern() (在 *TesterLibrary.Port.capture* 模块中), 6
create_dhcp_client() (*TesterLibrary.base.TesterLibrary* 静态方法), 572
create_dhcp_client() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 104
create_dhcp_client_custom_option() (*TesterLibrary.base.TesterLibrary* 静态方法), 574
create_dhcp_client_custom_option() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 105
create_dhcp_server() (*TesterLibrary.base.TesterLibrary* 静态方法), 574
create_dhcp_server() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 106
create_dhcp_server_address_pool() (*TesterLibrary.base.TesterLibrary* 静态方法), 575
create_dhcp_server_address_pool() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 106
create_dhcp_server_custom_option() (*TesterLibrary.base.TesterLibrary* 静态方法), 575
create_dhcp_server_custom_option() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 107
create_dhcpv6_client() (*TesterLibrary.base.TesterLibrary* 静态方法), 576
create_dhcpv6_client() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 117
create_dhcpv6_client_custom_options() (*TesterLibrary.base.TesterLibrary* 静态方法), 578
create_dhcpv6_client_custom_options() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 119
create_dhcpv6_server() (*TesterLibrary.base.TesterLibrary* 静态方法), 579
create_dhcpv6_server() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 120
create_dhcpv6_server_address_pool() (*TesterLibrary.base.TesterLibrary* 静态方法), 581
create_dhcpv6_server_address_pool() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 122
create_dhcpv6_server_custom_options() (*TesterLibrary.base.TesterLibrary* 静态方法), 581
create_dhcpv6_server_custom_options() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 122
create_dhcpv6_server_prefix_pool() (*TesterLibrary.base.TesterLibrary* 静态方法), 582
create_dhcpv6_server_prefix_pool() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 123
create_dot1x() (*TesterLibrary.base.TesterLibrary* 静

态方法), 583
create_dot1x() (在 TesterLibrary.Protocol.dot1x 模块中), 139
create_igmp() (TesterLibrary.base.TesterLibrary 静态方法), 584
create_igmp() (在 TesterLibrary.Protocol.igmp 模块中), 144
create_igmp_querier() (TesterLibrary.base.TesterLibrary 静态方法), 585
create_igmp_querier() (在 TesterLibrary.Protocol.igmp 模块中), 145
create_imix() (TesterLibrary.base.TesterLibrary 静态方法), 585
create_imix() (在 TesterLibrary.Stream.imix 模块中), 467
create_interface() (TesterLibrary.base.TesterLibrary 静态方法), 586
create_interface() (在 TesterLibrary.Port.interface 模块中), 17
create_isis() (TesterLibrary.base.TesterLibrary 静态方法), 586
create_isis() (在 TesterLibrary.Protocol.isis 模块中), 152
create_isis_binding_sr_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 588
create_isis_binding_sr_sid_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 154
create_isis_capability_sr_algorithm_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 589
create_isis_capability_sr_algorithm_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 154
create_isis_capability_sr_capability_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 589
create_isis_capability_sr_capability_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 154
create_isis_capability_sr_fad_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 590
create_isis_capability_sr_fad_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 155
create_isis_capability_sr_node_msd_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 591
create_isis_capability_sr_node_msd_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 156
create_isis_capability_srms_preference_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 592
create_isis_capability_srms_preference_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 157
create_isis_capability_srv6_capability_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 592
create_isis_capability_srv6_capability_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 157
create_isis_capability_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 593
create_isis_capability_tlv() (在 TesterLibrary.Protocol.isis 模块中), 158
create_isis_ipv4_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 593
create_isis_ipv4_tlv() (在 TesterLibrary.Protocol.isis 模块中), 159
create_isis_ipv6_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 594
create_isis_ipv6_tlv() (在 TesterLibrary.Protocol.isis 模块中), 160
create_isis_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 595
create_isis_lsp() (在 TesterLibrary.Protocol.isis 模块中), 160
create_isis_neighbor_custom_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 596
create_isis_neighbor_custom_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 161
create_isis_neighbor_sr_adj_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 596
create_isis_neighbor_sr_adj_sid_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 162
create_isis_neighbor_sr_lan_adj_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 597
create_isis_neighbor_sr_lan_adj_sid_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 162
create_isis_neighbor_sr_link_msd_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 598
create_isis_neighbor_sr_link_msd_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 163
create_isis_neighbor_srv6_endx_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 599
create_isis_neighbor_srv6_endx_sid_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 164
create_isis_neighbor_srv6_lan_endx_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 601
create_isis_neighbor_srv6_lan_endx_sid_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 166
create_isis_neighbor_te_config() (TesterLibrary.base.TesterLibrary 静态方法), 603
create_isis_neighbor_te_config() (在 TesterLibrary.Protocol.isis 模块中), 168
create_isis_neighbor_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 604
create_isis_neighbor_tlv() (在 TesterLibrary.Protocol.isis 模块中), 169
create_isis_sr_binding_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 604
create_isis_sr_binding_tlv() (在 TesterLibrary.Protocol.isis 模块中), 170
create_isis_srv6_end_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 605
create_isis_srv6_end_sid_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 171
create_isis_srv6_location_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 607
create_isis_srv6_location_tlv() (在 TesterLibrary.Protocol.isis 模块中), 172
create_isis_tlv_bier_Mpls_sub_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 608
create_isis_tlv_bier_Mpls_sub_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 173
create_isis_tlv_bier_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 608
create_isis_tlv_bier_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 174
create_isis_tlv_bierv6_bift_id_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法),

609
create_isis_tlv_bierv6_bift_id_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 174
create_isis_tlv_bierv6_sub_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 609
create_isis_tlv_bierv6_sub_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 174
create_isis_tlv_end_bier_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 609
create_isis_tlv_end_bier_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 175
create_isis_tlv_flex_algorithm_prefix_metric_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 610
create_isis_tlv_flex_algorithm_prefix_metric_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 175
create_isis_tlv_prefix_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 610
create_isis_tlv_prefix_sid_sub_tlv() (在 TesterLibrary.Protocol.isis 模块中), 176
create_l2tp() (TesterLibrary.base.TesterLibrary 静态方法), 611
create_l2tp() (在 TesterLibrary.Protocol.l2tp 模块中), 187
create_ldp() (TesterLibrary.base.TesterLibrary 静态方法), 613
create_ldp() (在 TesterLibrary.Protocol.ldp 模块中), 197
create_ldp_fec_128() (TesterLibrary.base.TesterLibrary 静态方法), 615
create_ldp_fec_128() (在 TesterLibrary.Protocol.ldp 模块中), 199
create_ldp_fec_129() (TesterLibrary.base.TesterLibrary 静态方法), 616
create_ldp_fec_129() (在 TesterLibrary.Protocol.ldp 模块中), 200
create_ldp_ipv4_egress() (TesterLibrary.base.TesterLibrary 静态方法), 617
create_ldp_ipv4_egress() (在 TesterLibrary.Protocol.ldp 模块中), 201
create_ldp_ipv4_ingress() (TesterLibrary.base.TesterLibrary 静态方法), 618
create_ldp_ipv4_ingress() (在 TesterLibrary.Protocol.ldp 模块中), 202
create_lsp_ping() (TesterLibrary.base.TesterLibrary 静态方法), 619
create_lsp_ping() (在 TesterLibrary.Protocol.lsp_ping 模块中), 210
create_lsp_ping_echo_request() (TesterLibrary.base.TesterLibrary 静态方法), 619
create_lsp_ping_echo_request() (在 TesterLibrary.Protocol.lsp_ping 模块中), 210
create_lsp_ping_fec_ldp_ipv4() (TesterLibrary.base.TesterLibrary 静态方法), 620
create_lsp_ping_fec_ldp_ipv4() (在 TesterLibrary.Protocol.lsp_ping 模块中), 212
create_lsp_ping_fec_segment_routing() (TesterLibrary.base.TesterLibrary 静态方法), 621
create_lsp_ping_fec_segment_routing() (在 TesterLibrary.Protocol.lsp_ping 模块中), 212
create_lsp_ping_fec_sr_adjacency() (TesterLibrary.base.TesterLibrary 静态方法), 621
create_lsp_ping_fec_sr_adjacency() (在 TesterLibrary.Protocol.lsp_ping 模块中), 213
create_lsp_ping_fec_sr_prefix() (TesterLibrary.base.TesterLibrary 静态方法), 622
create_lsp_ping_fec_sr_prefix() (在 TesterLibrary.Protocol.lsp_ping 模块中), 214
create_lsp_ping_fec_vpn_ipv4() (TesterLibrary.base.TesterLibrary 静态方法), 622
create_lsp_ping_fec_vpn_ipv4() (在 TesterLibrary.Protocol.lsp_ping 模块中), 214
create_memberships() (TesterLibrary.base.TesterLibrary 静态方法), 623
create_memberships() (在 TesterLibrary.Protocol.multicast 模块中), 227
create_mld() (TesterLibrary.base.TesterLibrary 静态方法), 624
create_mld() (在 TesterLibrary.Protocol.mld 模块中), 221
create_mld_querier() (TesterLibrary.base.TesterLibrary 静态方法), 625
create_mld_querier() (在 TesterLibrary.Protocol.mld 模块中), 222
create_mpls_wizard() (TesterLibrary.base.TesterLibrary 静态方法), 625
create_mpls_wizard() (在 TesterLibrary.Wizard.mpls 模块中), 489
create_multicast_group() (TesterLibrary.base.TesterLibrary 静态方法), 626
create_multicast_group() (在 TesterLibrary.Protocol.multicast 模块中), 228
create_ospf() (TesterLibrary.base.TesterLibrary 静态方法), 626
create_ospf() (在 TesterLibrary.Protocol.ospfv2 模块中), 229
create_ospf_adj_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 628
create_ospf_adj_sid_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 231
create_ospf_asbr_summary_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 628
create_ospf_asbr_summary_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 231
create_ospf_bier_mpls_encap_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 629
create_ospf_bier_mpls_encap_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 232
create_ospf_bier_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 630
create_ospf_bier_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 233
create_ospf_custom_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 630
create_ospf_custom_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 233
create_ospf_ext_prefix_range_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 631
create_ospf_ext_prefix_range_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 234
create_ospf_ext_prefix_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 632

create_ospf_ext_prefix_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 235
create_ospf_extended_link_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 633
create_ospf_extended_link_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 236
create_ospf_external_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 633
create_ospf_external_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 236
create_ospf_lan_adj_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 635
create_ospf_lan_adj_sid_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 238
create_ospf_network_atch_router() (TesterLibrary.base.TesterLibrary 静态方法), 636
create_ospf_network_atch_router() (在 TesterLibrary.Protocol.ospfv2 模块中), 239
create_ospf_network_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 636
create_ospf_network_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 239
create_ospf_opaque_extended_link_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 637
create_ospf_opaque_extended_link_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 240
create_ospf_opaque_extended_prefix_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 638
create_ospf_opaque_extended_prefix_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 241
create_ospf_opaque_router_info_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 639
create_ospf_opaque_router_info_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 242
create_ospf_prefix_sid_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 640
create_ospf_prefix_sid_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 243
create_ospf_router_info_capability_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 640
create_ospf_router_info_capability_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 244
create_ospf_router_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 641
create_ospf_router_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 244
create_ospf_router_lsa_link() (TesterLibrary.base.TesterLibrary 静态方法), 642
create_ospf_router_lsa_link() (在 TesterLibrary.Protocol.ospfv2 模块中), 245
create_ospf_sid_label_binding_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 642
create_ospf_sid_label_binding_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 246
create_ospf_sr_algorithm_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 643
create_ospf_sr_algorithm_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 246
create_ospf_sr_fad_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 644
create_ospf_sr_fad_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 247
create_ospf_sr_fapm_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 645
create_ospf_sr_fapm_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 248
create_ospf_sr_link_msd_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 645
create_ospf_sr_link_msd_sub_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 248
create_ospf_sr_node_msd_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 646
create_ospf_sr_node_msd_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 249
create_ospf_sr_sid_label_range_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 647
create_ospf_sr_sid_label_range_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 250
create_ospf_sr_srms_preference_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 648
create_ospf_sr_srms_preference_tlv() (在 TesterLibrary.Protocol.ospfv2 模块中), 251
create_ospf_summary_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 648
create_ospf_summary_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 251
create_ospf_te_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 649
create_ospf_te_lsa() (在 TesterLibrary.Protocol.ospfv2 模块中), 252
create_ospfv3() (TesterLibrary.base.TesterLibrary 静态方法), 650
create_ospfv3() (在 TesterLibrary.Protocol.ospfv3 模块中), 261
create_ospfv3_as_external_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 652
create_ospfv3_as_external_lsa() (在 TesterLibrary.Protocol.ospfv3 模块中), 263
create_ospfv3_bier_mpls_encap_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 654
create_ospfv3_bier_mpls_encap_sub_tlv() (在 TesterLibrary.Protocol.ospfv3 模块中), 264
create_ospfv3_bier_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 654
create_ospfv3_bier_sub_tlv() (在 TesterLibrary.Protocol.ospfv3 模块中), 265
create_ospfv3_endx_sid_structure_sub_tlv() (TesterLibrary.base.TesterLibrary 静态方法), 655
create_ospfv3_endx_sid_structure_sub_tlv() (在 TesterLibrary.Protocol.ospfv3 模块中), 265
create_ospfv3_inter_area_prefix_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 655
create_ospfv3_inter_area_prefix_lsa() (在 TesterLibrary.Protocol.ospfv3 模块中), 266
create_ospfv3_inter_area_router_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 657
create_ospfv3_inter_area_router_lsa() (在 TesterLibrary.Protocol.ospfv3 模块中), 267

create_ospfv3_intra_area_prefix_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 658
create_ospfv3_intra_area_prefix_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 268
create_ospfv3_lan_endx_sid_structure_sub_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 659
create_ospfv3_lan_endx_sid_structure_sub_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 270
create_ospfv3_link_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 660
create_ospfv3_link_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 270
create_ospfv3_network_atch_router() (*TesterLibrary.base.TesterLibrary* 静态方法), 662
create_ospfv3_network_atch_router() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 272
create_ospfv3_network_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 662
create_ospfv3_network_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 273
create_ospfv3_nssa_external_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 664
create_ospfv3_nssa_external_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 274
create_ospfv3_opaque_router_info_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 665
create_ospfv3_opaque_router_info_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 276
create_ospfv3_router_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 668
create_ospfv3_router_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 278
create_ospfv3_router_lsa_link() (*TesterLibrary.base.TesterLibrary* 静态方法), 669
create_ospfv3_router_lsa_link() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 280
create_ospfv3_sr_algorithm_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 670
create_ospfv3_sr_algorithm_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 280
create_ospfv3_sr_fad_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 670
create_ospfv3_sr_fad_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 281
create_ospfv3_sr_fapm_sub_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 671
create_ospfv3_sr_fapm_sub_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 282
create_ospfv3_srv6_capabilities_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 672
create_ospfv3_srv6_capabilities_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 282
create_ospfv3_srv6_endx_sid_sub_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 673
create_ospfv3_srv6_endx_sid_sub_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 283
create_ospfv3_srv6_lan_endx_sid_sub_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 674
create_ospfv3_srv6_lan_endx_sid_sub_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 284
create_ospfv3_srv6_link_msd_sub_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 675
create_ospfv3_srv6_link_msd_sub_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 285
create_ospfv3_srv6_location_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 676
create_ospfv3_srv6_location_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 286
create_ospfv3_srv6_location_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 676
create_ospfv3_srv6_location_tlv() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 287
create_pcep() (*TesterLibrary.base.TesterLibrary* 静态方法), 677
create_pcep() (在 *TesterLibrary.Protocol.pcep* 模块中), 294
create_pcep_bw_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 680
create_pcep_bw_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 297
create_pcep_endpoint_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 680
create_pcep_endpoint_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 297
create_pcep_lsp_auto_tx_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 681
create_pcep_lsp_auto_tx_parameters() (在 *TesterLibrary.Protocol.pcep* 模块中), 298
create_pcep_lspa_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 682
create_pcep_lspa_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 298
create_pcep_metric_list() (*TesterLibrary.base.TesterLibrary* 静态方法), 683
create_pcep_metric_list() (在 *TesterLibrary.Protocol.pcep* 模块中), 299
create_pcep_metric_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 683
create_pcep_metric_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 300
create_pcep_no_path_reason() (*TesterLibrary.base.TesterLibrary* 静态方法), 684
create_pcep_no_path_reason() (在 *TesterLibrary.Protocol.pcep* 模块中), 300
create_pcep_pcc_auto_delegation_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 684
create_pcep_pcc_auto_delegation_parameters() (在 *TesterLibrary.Protocol.pcep* 模块中), 301
create_pcep_pcc_auto_request_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 685
create_pcep_pcc_auto_request_parameters() (在 *TesterLibrary.Protocol.pcep* 模块中), 301
create_pcep_pcc_auto_sync_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 685
create_pcep_pcc_auto_sync_parameters() (在 *TesterLibrary.Protocol.pcep* 模块中), 302
create_pcep_pcc_lsp()

(*TesterLibrary.base.TesterLibrary* 静态方法), 686
create_pcep_pcc_lsp() (在 *TesterLibrary.Protocol.pcep* 模块中), 302
create_pcep_pcc_lsp_info() (*TesterLibrary.base.TesterLibrary* 静态方法), 687
create_pcep_pcc_lsp_info() (在 *TesterLibrary.Protocol.pcep* 模块中), 304
create_pcep_pce_auto_initiate_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 688
create_pcep_pce_auto_initiate_parameters() (在 *TesterLibrary.Protocol.pcep* 模块中), 305
create_pcep_pce_auto_reply_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 689
create_pcep_pce_auto_reply_parameters() (在 *TesterLibrary.Protocol.pcep* 模块中), 305
create_pcep_pce_auto_update_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 689
create_pcep_pce_auto_update_parameters() (在 *TesterLibrary.Protocol.pcep* 模块中), 306
create_pcep_pce_lsp() (*TesterLibrary.base.TesterLibrary* 静态方法), 690
create_pcep_pce_lsp() (在 *TesterLibrary.Protocol.pcep* 模块中), 306
create_pcep_pce_lsp_info() (*TesterLibrary.base.TesterLibrary* 静态方法), 691
create_pcep_pce_lsp_info() (在 *TesterLibrary.Protocol.pcep* 模块中), 308
create_pcep_rp_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 692
create_pcep_rp_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 308
create_pcep_sr_ero_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 692
create_pcep_sr_ero_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 309
create_pcep_sr_ero_sub_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 693
create_pcep_sr_ero_sub_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 310
create_pcep_sr_rro_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 695
create_pcep_sr_rro_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 311
create_pcep_sr_rro_sub_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 695
create_pcep_sr_rro_sub_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 312
create_pcep_srp_info() (*TesterLibrary.base.TesterLibrary* 静态方法), 697
create_pcep_srp_info() (在 *TesterLibrary.Protocol.pcep* 模块中), 313
create_pcep_srv6_ero_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 697
create_pcep_srv6_ero_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 314
create_pcep_srv6_ero_sub_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 698
create_pcep_srv6_ero_sub_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 314
create_pcep_srv6_rro_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 700
create_pcep_srv6_rro_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 316
create_pcep_srv6_rro_sub_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 700
create_pcep_srv6_rro_sub_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 317
create_pcep_xro_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 702
create_pcep_xro_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 319
create_pcep_xro_sub_object() (*TesterLibrary.base.TesterLibrary* 静态方法), 703
create_pcep_xro_sub_object() (在 *TesterLibrary.Protocol.pcep* 模块中), 319
create_peclsp_for_srte() (*TesterLibrary.base.TesterLibrary* 静态方法), 704
create_peclsp_for_srte() (在 *TesterLibrary.Overall.common* 模块中), 2
create_pim() (*TesterLibrary.base.TesterLibrary* 静态方法), 704
create_pim() (在 *TesterLibrary.Protocol.pim* 模块中), 334
create_pim_group() (*TesterLibrary.base.TesterLibrary* 静态方法), 705
create_pim_group() (在 *TesterLibrary.Protocol.pim* 模块中), 335
create_pim_ipv6_group() (*TesterLibrary.base.TesterLibrary* 静态方法), 706
create_pim_ipv6_group() (在 *TesterLibrary.Protocol.pim* 模块中), 336
create_pim_ipv6_register_group() (*TesterLibrary.base.TesterLibrary* 静态方法), 707
create_pim_ipv6_register_group() (在 *TesterLibrary.Protocol.pim* 模块中), 337
create_pim_ipv6_rp_map() (*TesterLibrary.base.TesterLibrary* 静态方法), 708
create_pim_ipv6_rp_map() (在 *TesterLibrary.Protocol.pim* 模块中), 338
create_pim_register_group() (*TesterLibrary.base.TesterLibrary* 静态方法), 708
create_pim_register_group() (在 *TesterLibrary.Protocol.pim* 模块中), 338
create_pim_rp_map() (*TesterLibrary.base.TesterLibrary* 静态方法), 709
create_pim_rp_map() (在 *TesterLibrary.Protocol.pim* 模块中), 339
create_pppoe() (*TesterLibrary.base.TesterLibrary* 静态方法), 710
create_pppoe() (在 *TesterLibrary.Protocol.pppoe* 模块中), 346
create_pppoe_custom_option() (*TesterLibrary.base.TesterLibrary* 静态方法), 713
create_pppoe_custom_option() (在 *TesterLibrary.Protocol.pppoe* 模块中), 349
create_rip() (*TesterLibrary.base.TesterLibrary* 静态方法), 713
create_rip() (在 *TesterLibrary.Protocol.rip* 模块中), 362

create_rip_ipv4_route() (*TesterLibrary.base.TesterLibrary* 静态方法), 714
create_rip_ipv4_route() (在 *TesterLibrary.Protocol.rip* 模块中), 363
create_rip_ipv6_route() (*TesterLibrary.base.TesterLibrary* 静态方法), 715
create_rip_ipv6_route() (在 *TesterLibrary.Protocol.rip* 模块中), 363
create_stream_header() (*TesterLibrary.base.TesterLibrary* 静态方法), 716
create_stream_header() (在 *TesterLibrary.Stream.common* 模块中), 457
create_vxlan() (*TesterLibrary.base.TesterLibrary* 静态方法), 717
create_vxlan() (在 *TesterLibrary.Protocol.vxlan* 模块中), 370
create_vxlan_segment() (*TesterLibrary.base.TesterLibrary* 静态方法), 718
create_vxlan_segment() (在 *TesterLibrary.Protocol.vxlan* 模块中), 370

D

del_benchmark() (*TesterLibrary.base.TesterLibrary* 静态方法), 719
del_benchmark() (在 *TesterLibrary.Wizard.benchmark* 模块中), 471
del_imix_distribution_frame() (*TesterLibrary.base.TesterLibrary* 静态方法), 719
del_imix_distribution_frame() (在 *TesterLibrary.Stream.imix* 模块中), 468
del_objects() (*TesterLibrary.base.TesterLibrary* 静态方法), 719
del_objects() (在 *TesterLibrary.Overall.common* 模块中), 2
del_port() (*TesterLibrary.base.TesterLibrary* 静态方法), 720
del_port() (在 *TesterLibrary.Port.common* 模块中), 11
del_stream() (*TesterLibrary.base.TesterLibrary* 静态方法), 720
del_stream() (在 *TesterLibrary.Stream.common* 模块中), 459
dhcp_abort() (*TesterLibrary.base.TesterLibrary* 静态方法), 720
dhcp_abort() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 108
dhcp_bind() (*TesterLibrary.base.TesterLibrary* 静态方法), 721
dhcp_bind() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 108
dhcp_rebind() (*TesterLibrary.base.TesterLibrary* 静态方法), 721
dhcp_rebind() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 108
dhcp_reboot() (*TesterLibrary.base.TesterLibrary* 静态方法), 721
dhcp_reboot() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 108
dhcp_release() (*TesterLibrary.base.TesterLibrary* 静态方法), 721
dhcp_release() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 109
dhcp_renew() (*TesterLibrary.base.TesterLibrary* 静态方法), 722
dhcp_renew() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 109
dhcpv6_client_abort() (*TesterLibrary.base.TesterLibrary* 静态方法), 722
dhcpv6_client_abort() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 124
dhcpv6_client_active_lease_query() (*TesterLibrary.base.TesterLibrary* 静态方法), 722
dhcpv6_client_active_lease_query() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 124
dhcpv6_client_bind() (*TesterLibrary.base.TesterLibrary* 静态方法), 722
dhcpv6_client_bind() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 124
dhcpv6_client_bulk_lease_query() (*TesterLibrary.base.TesterLibrary* 静态方法), 723
dhcpv6_client_bulk_lease_query() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 124
dhcpv6_client_confirm() (*TesterLibrary.base.TesterLibrary* 静态方法), 723
dhcpv6_client_confirm() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 125
dhcpv6_client_info_request() (*TesterLibrary.base.TesterLibrary* 静态方法), 724
dhcpv6_client_info_request() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 125
dhcpv6_client_lease_query() (*TesterLibrary.base.TesterLibrary* 静态方法), 724
dhcpv6_client_lease_query() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 125
dhcpv6_client_rebind() (*TesterLibrary.base.TesterLibrary* 静态方法), 724
dhcpv6_client_rebind() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 126
dhcpv6_client_release() (*TesterLibrary.base.TesterLibrary* 静态方法), 725
dhcpv6_client_release() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 126
dhcpv6_client_renew() (*TesterLibrary.base.TesterLibrary* 静态方法), 725
dhcpv6_client_renew() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 126
dhcpv6_client_start_tls() (*TesterLibrary.base.TesterLibrary* 静态方法), 725
dhcpv6_client_start_tls() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 127
dhcpv6_server_abort() (*TesterLibrary.base.TesterLibrary* 静态方法), 725
dhcpv6_server_abort() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 127
dhcpv6_server_reconfigure_rebind() (*TesterLibrary.base.TesterLibrary* 静态方法), 726
dhcpv6_server_reconfigure_rebind() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 127
dhcpv6_server_reconfigure_renew() (*TesterLibrary.base.TesterLibrary* 静态方法), 726
dhcpv6_server_reconfigure_renew() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 127
dhcpv6_server_start() (*TesterLibrary.base.TesterLibrary* 静态方法), 726
dhcpv6_server_start() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 128
dhcpv6_server_stop()

(*TesterLibrary.base.TesterLibrary* 静态方法), 726
dhcpv6_server_stop() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 128
disconnect_bgp() (*TesterLibrary.base.TesterLibrary* 静态方法), 727
disconnect_bgp() (在 *TesterLibrary.Protocol.bgp* 模块中), 92
disconnect_l2tp() (*TesterLibrary.base.TesterLibrary* 静态方法), 727
disconnect_l2tp() (在 *TesterLibrary.Protocol.l2tp* 模块中), 189
disconnect_pppoe() (*TesterLibrary.base.TesterLibrary* 静态方法), 727
disconnect_pppoe() (在 *TesterLibrary.Protocol.pppoe* 模块中), 349
dot1x_delete_certificate() (*TesterLibrary.base.TesterLibrary* 静态方法), 727
dot1x_delete_certificate() (在 *TesterLibrary.Protocol.dot1x* 模块中), 140
dot1x_upload_certificate() (*TesterLibrary.base.TesterLibrary* 静态方法), 728
dot1x_upload_certificate() (在 *TesterLibrary.Protocol.dot1x* 模块中), 140
download_packages() (*TesterLibrary.base.TesterLibrary* 静态方法), 728
download_packages() (在 *TesterLibrary.Port.capture* 模块中), 6

E

edit_benchmark_address_learning_capacity() (*TesterLibrary.base.TesterLibrary* 静态方法), 728
edit_benchmark_address_learning_capacity() (在 *TesterLibrary.Wizard.benchmark* 模块中), 471
edit_benchmark_address_learning_rate() (*TesterLibrary.base.TesterLibrary* 静态方法), 729
edit_benchmark_address_learning_rate() (在 *TesterLibrary.Wizard.benchmark* 模块中), 473
edit_benchmark_burst_count_loop() (*TesterLibrary.base.TesterLibrary* 静态方法), 729
edit_benchmark_burst_count_loop() (在 *TesterLibrary.Wizard.benchmark* 模块中), 473
edit_benchmark_duration() (*TesterLibrary.base.TesterLibrary* 静态方法), 730
edit_benchmark_duration() (在 *TesterLibrary.Wizard.benchmark* 模块中), 474
edit_benchmark_errored_frame_filtering() (*TesterLibrary.base.TesterLibrary* 静态方法), 730
edit_benchmark_errored_frame_filtering() (在 *TesterLibrary.Wizard.benchmark* 模块中), 475
edit_benchmark_frame() (*TesterLibrary.base.TesterLibrary* 静态方法), 731
edit_benchmark_frame() (在 *TesterLibrary.Wizard.benchmark* 模块中), 476
edit_benchmark_latency() (*TesterLibrary.base.TesterLibrary* 静态方法), 732
edit_benchmark_latency() (在 *TesterLibrary.Wizard.benchmark* 模块中), 477
edit_benchmark_learning() (*TesterLibrary.base.TesterLibrary* 静态方法), 732
edit_benchmark_learning() (在 *TesterLibrary.Wizard.benchmark* 模块中), 477
edit_benchmark_multicast_base_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 733
edit_benchmark_multicast_base_parameters() (在 *TesterLibrary.Wizard.benchmark* 模块中), 478
edit_benchmark_multicast_group_count_loop() (*TesterLibrary.base.TesterLibrary* 静态方法), 734
edit_benchmark_multicast_group_count_loop() (在 *TesterLibrary.Wizard.benchmark* 模块中), 479
edit_benchmark_multicast_join_leave_delay() (*TesterLibrary.base.TesterLibrary* 静态方法), 734
edit_benchmark_multicast_join_leave_delay() (在 *TesterLibrary.Wizard.benchmark* 模块中), 480
edit_benchmark_multicast_mixed_throughput_unicast_streams() (*TesterLibrary.base.TesterLibrary* 静态方法), 735
edit_benchmark_multicast_mixed_throughput_unicast_streams() (在 *TesterLibrary.Wizard.benchmark* 模块中), 480
edit_benchmark_multicast_other() (*TesterLibrary.base.TesterLibrary* 静态方法), 735
edit_benchmark_multicast_other() (在 *TesterLibrary.Wizard.benchmark* 模块中), 481
edit_benchmark_multicast_stream_tos() (*TesterLibrary.base.TesterLibrary* 静态方法), 736
edit_benchmark_multicast_stream_tos() (在 *TesterLibrary.Wizard.benchmark* 模块中), 481
edit_benchmark_multicast_traffic_ratio_loop() (*TesterLibrary.base.TesterLibrary* 静态方法), 736
edit_benchmark_multicast_traffic_ratio_loop() (在 *TesterLibrary.Wizard.benchmark* 模块中), 482
edit_benchmark_path() (*TesterLibrary.base.TesterLibrary* 静态方法), 737
edit_benchmark_path() (在 *TesterLibrary.Wizard.benchmark* 模块中), 483
edit_benchmark_result_file_name() (*TesterLibrary.base.TesterLibrary* 静态方法), 737
edit_benchmark_result_file_name() (在 *TesterLibrary.Wizard.benchmark* 模块中), 483
edit_benchmark_search() (*TesterLibrary.base.TesterLibrary* 静态方法), 737
edit_benchmark_search() (在 *TesterLibrary.Wizard.benchmark* 模块中), 483
edit_benchmark_traffic_load_loop() (*TesterLibrary.base.TesterLibrary* 静态方法), 738
edit_benchmark_traffic_load_loop() (在

TesterLibrary.Wizard.benchmark 模块中), 484
edit_benchmark_transport_layer() (*TesterLibrary.base.TesterLibrary* 静态方法), 739
edit_benchmark_transport_layer() (在 *TesterLibrary.Wizard.benchmark* 模块中), 485
edit_bfd() (*TesterLibrary.base.TesterLibrary* 静态方法), 739
edit_bfd() (在 *TesterLibrary.Protocol.bfd* 模块中), 26
edit_bgp() (*TesterLibrary.base.TesterLibrary* 静态方法), 740
edit_bgp() (在 *TesterLibrary.Protocol.bgp* 模块中), 92
edit_bgp_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 742
edit_bgp_port_config() (在 *TesterLibrary.Protocol.bgp* 模块中), 94
edit_capture() (*TesterLibrary.base.TesterLibrary* 静态方法), 743
edit_capture() (在 *TesterLibrary.Port.capture* 模块中), 7
edit_capture_event() (*TesterLibrary.base.TesterLibrary* 静态方法), 744
edit_capture_event() (在 *TesterLibrary.Port.capture* 模块中), 8
edit_capture_filter() (*TesterLibrary.base.TesterLibrary* 静态方法), 745
edit_capture_filter() (在 *TesterLibrary.Port.capture* 模块中), 9
edit_capture_pattern() (*TesterLibrary.base.TesterLibrary* 静态方法), 745
edit_capture_pattern() (在 *TesterLibrary.Port.capture* 模块中), 9
edit_configs() (*TesterLibrary.base.TesterLibrary* 静态方法), 746
edit_configs() (在 *TesterLibrary.Overall.common* 模块中), 2
edit_dhcp_client() (*TesterLibrary.base.TesterLibrary* 静态方法), 746
edit_dhcp_client() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 109
edit_dhcp_client_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 747
edit_dhcp_client_port_config() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 110
edit_dhcp_server() (*TesterLibrary.base.TesterLibrary* 静态方法), 748
edit_dhcp_server() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 111
edit_dhcp_server_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 748
edit_dhcp_server_port_config() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 111
edit_dhcpv6_client_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 749
edit_dhcpv6_client_port_config() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 128
edit_dot1x_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 751
edit_dot1x_port_config() (在 *TesterLibrary.Protocol.dot1x* 模块中), 141
edit_header_arp() (*TesterLibrary.base.TesterLibrary* 静态方法), 751
edit_header_arp() (在 *TesterLibrary.Stream.Header.L2.common* 模块中), 391
edit_header_custom() (*TesterLibrary.base.TesterLibrary* 静态方法), 752
edit_header_custom() (在 *TesterLibrary.Stream.Header.Basic.common* 模块中), 390
edit_header_ethernet() (*TesterLibrary.base.TesterLibrary* 静态方法), 752
edit_header_ethernet() (在 *TesterLibrary.Stream.Header.L2.common* 模块中), 392
edit_header_gre() (*TesterLibrary.base.TesterLibrary* 静态方法), 753
edit_header_gre() (在 *TesterLibrary.Stream.Header.Gre.common* 模块中), 390
edit_header_icmp_dest_unreach() (*TesterLibrary.base.TesterLibrary* 静态方法), 753
edit_header_icmp_dest_unreach() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 413
edit_header_icmp_echo_reply() (*TesterLibrary.base.TesterLibrary* 静态方法), 755
edit_header_icmp_echo_reply() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 414
edit_header_icmp_echo_request() (*TesterLibrary.base.TesterLibrary* 静态方法), 755
edit_header_icmp_echo_request() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 414
edit_header_icmp_information_reply() (*TesterLibrary.base.TesterLibrary* 静态方法), 756
edit_header_icmp_information_reply() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 415
edit_header_icmp_information_request() (*TesterLibrary.base.TesterLibrary* 静态方法), 756
edit_header_icmp_information_request() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 416
edit_header_icmp_mask_reply() (*TesterLibrary.base.TesterLibrary* 静态方法), 757
edit_header_icmp_mask_reply() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 416
edit_header_icmp_mask_request() (*TesterLibrary.base.TesterLibrary* 静态方法), 757
edit_header_icmp_mask_request() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 417
edit_header_icmp_parameter_problem() (*TesterLibrary.base.TesterLibrary* 静态方法), 758
edit_header_icmp_parameter_problem() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 417
edit_header_icmp_redirect() (*TesterLibrary.base.TesterLibrary* 静态方法), 759
edit_header_icmp_redirect() (在

TesterLibrary.Stream.Header.L3.icmpv4 模块中), 419
edit_header_icmp_source_quench() (*TesterLibrary.base.TesterLibrary* 静态方法), 760
edit_header_icmp_source_quench() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 420
edit_header_icmp_time_exceeded() (*TesterLibrary.base.TesterLibrary* 静态方法), 761
edit_header_icmp_time_exceeded() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 421
edit_header_icmp_time_stamp_reply() (*TesterLibrary.base.TesterLibrary* 静态方法), 763
edit_header_icmp_time_stamp_reply() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 422
edit_header_icmp_time_stamp_request() (*TesterLibrary.base.TesterLibrary* 静态方法), 763
edit_header_icmp_time_stamp_request() (在 *TesterLibrary.Stream.Header.L3.icmpv4* 模块中), 423
edit_header_icmpv6_destination_unreachable() (*TesterLibrary.base.TesterLibrary* 静态方法), 764
edit_header_icmpv6_destination_unreachable() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 424
edit_header_icmpv6_echo_reply() (*TesterLibrary.base.TesterLibrary* 静态方法), 765
edit_header_icmpv6_echo_reply() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 425
edit_header_icmpv6_echo_request() (*TesterLibrary.base.TesterLibrary* 静态方法), 765
edit_header_icmpv6_echo_request() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 425
edit_header_icmpv6_group_records() (*TesterLibrary.base.TesterLibrary* 静态方法), 766
edit_header_icmpv6_group_records() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 426
edit_header_icmpv6_header_option() (*TesterLibrary.base.TesterLibrary* 静态方法), 766
edit_header_icmpv6_header_option() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 426
edit_header_icmpv6_mldv1_done() (*TesterLibrary.base.TesterLibrary* 静态方法), 768
edit_header_icmpv6_mldv1_done() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 428
edit_header_icmpv6_mldv1_query() (*TesterLibrary.base.TesterLibrary* 静态方法), 768
edit_header_icmpv6_mldv1_query() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 429
edit_header_icmpv6_mldv1_report() (*TesterLibrary.base.TesterLibrary* 静态方法), 769
edit_header_icmpv6_mldv1_report() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 429
edit_header_icmpv6_mldv2_query() (*TesterLibrary.base.TesterLibrary* 静态方法), 769
edit_header_icmpv6_mldv2_query() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 430
edit_header_icmpv6_mldv2_report() (*TesterLibrary.base.TesterLibrary* 静态方法), 770
edit_header_icmpv6_mldv2_report() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 430
edit_header_icmpv6_neighbor_advertise() (*TesterLibrary.base.TesterLibrary* 静态方法), 771
edit_header_icmpv6_neighbor_advertise() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 431
edit_header_icmpv6_neighbor_solicitation() (*TesterLibrary.base.TesterLibrary* 静态方法), 771
edit_header_icmpv6_neighbor_solicitation() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 432
edit_header_icmpv6_packet_too_big() (*TesterLibrary.base.TesterLibrary* 静态方法), 772
edit_header_icmpv6_packet_too_big() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 432
edit_header_icmpv6_parameter_problem() (*TesterLibrary.base.TesterLibrary* 静态方法), 773
edit_header_icmpv6_parameter_problem() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 433
edit_header_icmpv6_redirect() (*TesterLibrary.base.TesterLibrary* 静态方法), 773
edit_header_icmpv6_redirect() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 434
edit_header_icmpv6_redirected_header() (*TesterLibrary.base.TesterLibrary* 静态方法), 774
edit_header_icmpv6_redirected_header() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 434
edit_header_icmpv6_router_advertise() (*TesterLibrary.base.TesterLibrary* 静态方法), 775
edit_header_icmpv6_router_advertise() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 435
edit_header_icmpv6_router_solicitation() (*TesterLibrary.base.TesterLibrary* 静态方法), 776
edit_header_icmpv6_router_solicitation() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 436
edit_header_icmpv6_time_exceed() (*TesterLibrary.base.TesterLibrary* 静态方法), 776
edit_header_icmpv6_time_exceed() (在 *TesterLibrary.Stream.Header.L3.icmpv6* 模块中), 437
edit_header_igmpv1_query() (*TesterLibrary.base.TesterLibrary* 静态方法), 777
edit_header_igmpv1_query() (在 *TesterLibrary.Stream.Header.L3.igmp* 模块中), 437

edit_header_igmpv1_report()
(TesterLibrary.base.TesterLibrary 静态方法), 777
edit_header_igmpv1_report() (在 TesterLibrary.Stream.Header.L3.igmp 模块中), 438
edit_header_igmpv2_query()
(TesterLibrary.base.TesterLibrary 静态方法), 778
edit_header_igmpv2_query() (在 TesterLibrary.Stream.Header.L3.igmp 模块中), 438
edit_header_igmpv2_report()
(TesterLibrary.base.TesterLibrary 静态方法), 778
edit_header_igmpv2_report() (在 TesterLibrary.Stream.Header.L3.igmp 模块中), 439
edit_header_igmpv3_group_records()
(TesterLibrary.base.TesterLibrary 静态方法), 779
edit_header_igmpv3_group_records() (在 TesterLibrary.Stream.Header.L3.igmp 模块中), 439
edit_header_igmpv3_query()
(TesterLibrary.base.TesterLibrary 静态方法), 779
edit_header_igmpv3_query() (在 TesterLibrary.Stream.Header.L3.igmp 模块中), 440
edit_header_igmpv3_report()
(TesterLibrary.base.TesterLibrary 静态方法), 780
edit_header_igmpv3_report() (在 TesterLibrary.Stream.Header.L3.igmp 模块中), 441
edit_header_ipv4()
(TesterLibrary.base.TesterLibrary 静态方法), 781
edit_header_ipv4() (在 TesterLibrary.Stream.Header.L3.common 模块中), 409
edit_header_ipv4_option()
(TesterLibrary.base.TesterLibrary 静态方法), 782
edit_header_ipv4_option() (在 TesterLibrary.Stream.Header.L3.common 模块中), 410
edit_header_ipv6()
(TesterLibrary.base.TesterLibrary 静态方法), 784
edit_header_ipv6() (在 TesterLibrary.Stream.Header.L3.common 模块中), 412
edit_header_isis_area_address_entry()
(TesterLibrary.base.TesterLibrary 静态方法), 784
edit_header_isis_area_address_entry() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 393
edit_header_isis_csnp()
(TesterLibrary.base.TesterLibrary 静态方法), 785
edit_header_isis_csnp() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 394
edit_header_isis_external_metric_entry()
(TesterLibrary.base.TesterLibrary 静态方法), 786
edit_header_isis_external_metric_entry() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 395
edit_header_isis_hello()
(TesterLibrary.base.TesterLibrary 静态方法), 787
edit_header_isis_hello() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 396
edit_header_isis_internal_metric_entry()
(TesterLibrary.base.TesterLibrary 静态方法), 788
edit_header_isis_internal_metric_entry() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 397
edit_header_isis_lsp()
(TesterLibrary.base.TesterLibrary 静态方法), 789
edit_header_isis_lsp() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 398
edit_header_isis_lsp_entry()
(TesterLibrary.base.TesterLibrary 静态方法), 790
edit_header_isis_lsp_entry() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 400
edit_header_isis_metric_entry()
(TesterLibrary.base.TesterLibrary 静态方法), 791
edit_header_isis_metric_entry() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 400
edit_header_isis_nlpid_entry()
(TesterLibrary.base.TesterLibrary 静态方法), 792
edit_header_isis_nlpid_entry() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 401
edit_header_isis_psnp()
(TesterLibrary.base.TesterLibrary 静态方法), 792
edit_header_isis_psnp() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 402
edit_header_isis_sub_tlv()
(TesterLibrary.base.TesterLibrary 静态方法), 793
edit_header_isis_sub_tlv() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 403
edit_header_isis_tlv_header()
(TesterLibrary.base.TesterLibrary 静态方法), 795
edit_header_isis_tlv_header() (在 TesterLibrary.Stream.Header.L2.isis 模块中), 405
edit_header_l2tpv2_data()
(TesterLibrary.base.TesterLibrary 静态方法), 799
edit_header_l2tpv2_data() (在 TesterLibrary.Stream.Header.Access.l2tpv2 模块中), 389
edit_header_mpls()
(TesterLibrary.base.TesterLibrary 静态方法), 799
edit_header_mpls() (在 TesterLibrary.Stream.Header.L2.common 模块中), 392
edit_header_ospfv2_ack()
(TesterLibrary.base.TesterLibrary 静态方法), 800
edit_header_ospfv2_ack() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 443
edit_header_ospfv2_dd()
(TesterLibrary.base.TesterLibrary 静态方法),

801
edit_header_ospfv2_dd() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 444
edit_header_ospfv2_hello() (TesterLibrary.base.TesterLibrary 静态方法), 802
edit_header_ospfv2_hello() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 445
edit_header_ospfv2_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 803
edit_header_ospfv2_lsa() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 446
edit_header_ospfv2_request() (TesterLibrary.base.TesterLibrary 静态方法), 804
edit_header_ospfv2_request() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 448
edit_header_ospfv2_unknown() (TesterLibrary.base.TesterLibrary 静态方法), 805
edit_header_ospfv2_unknown() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 448
edit_header_ospfv2_update() (TesterLibrary.base.TesterLibrary 静态方法), 805
edit_header_ospfv2_update() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 449
edit_header_ospfv2_update_lsa() (TesterLibrary.base.TesterLibrary 静态方法), 806
edit_header_ospfv2_update_lsa() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 450
edit_header_ospfv2_update_nework_attached_route() (TesterLibrary.base.TesterLibrary 静态方法), 808
edit_header_ospfv2_update_nework_attached_route() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 452
edit_header_ospfv2_update_route_link_tos_metric() (TesterLibrary.base.TesterLibrary 静态方法), 809
edit_header_ospfv2_update_route_link_tos_metric() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 453
edit_header_ospfv2_update_route_lsa_link() (TesterLibrary.base.TesterLibrary 静态方法), 810
edit_header_ospfv2_update_route_lsa_link() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 453
edit_header_ospfv2_update_tos_metric() (TesterLibrary.base.TesterLibrary 静态方法), 810
edit_header_ospfv2_update_tos_metric() (在 TesterLibrary.Stream.Header.Routing.ospfv2 模块中), 454
edit_header_ppp() (TesterLibrary.base.TesterLibrary 静态方法), 811
edit_header_ppp() (在 TesterLibrary.Stream.Header.Access.common 模块中), 388
edit_header_pppoe() (TesterLibrary.base.TesterLibrary 静态方法), 812
edit_header_pppoe() (在 TesterLibrary.Stream.Header.Access.common 模块中), 388
edit_header_tcp() (TesterLibrary.base.TesterLibrary 静态方法), 812
edit_header_tcp() (在 TesterLibrary.Stream.Header.L4.common 模块中), 442
edit_header_udp() (TesterLibrary.base.TesterLibrary 静态方法), 813
edit_header_udp() (在 TesterLibrary.Stream.Header.L4.common 模块中), 443
edit_header_vlan() (TesterLibrary.base.TesterLibrary 静态方法), 813
edit_header_vlan() (在 TesterLibrary.Stream.Header.L2.common 模块中), 393
edit_igmp() (TesterLibrary.base.TesterLibrary 静态方法), 814
edit_igmp() (在 TesterLibrary.Protocol.igmp 模块中), 146
edit_igmp_querier() (TesterLibrary.base.TesterLibrary 静态方法), 814
edit_igmp_querier() (在 TesterLibrary.Protocol.igmp 模块中), 147
edit_interface() (TesterLibrary.base.TesterLibrary 静态方法), 815
edit_interface() (在 TesterLibrary.Port.interface 模块中), 17
edit_interface_stack() (TesterLibrary.base.TesterLibrary 静态方法), 817
edit_interface_stack() (在 TesterLibrary.Port.interface 模块中), 19
edit_isis() (TesterLibrary.base.TesterLibrary 静态方法), 818
edit_isis() (在 TesterLibrary.Protocol.isis 模块中), 176
edit_isis_mt_params() (TesterLibrary.base.TesterLibrary 静态方法), 820
edit_isis_mt_params() (在 TesterLibrary.Protocol.isis 模块中), 178
edit_isis_per_pdu() (TesterLibrary.base.TesterLibrary 静态方法), 820
edit_isis_per_pdu() (在 TesterLibrary.Protocol.isis 模块中), 179
edit_isis_port_config() (TesterLibrary.base.TesterLibrary 静态方法), 821
edit_isis_port_config() (在 TesterLibrary.Protocol.isis 模块中), 180
edit_l2tp() (TesterLibrary.base.TesterLibrary 静态方法), 821
edit_l2tp() (在 TesterLibrary.Protocol.l2tp 模块中), 189
edit_l2tp_port_config() (TesterLibrary.base.TesterLibrary 静态方法), 824
edit_l2tp_port_config() (在 TesterLibrary.Protocol.l2tp 模块中), 191
edit_ldp() (TesterLibrary.base.TesterLibrary 静态方法), 824
edit_ldp() (在 TesterLibrary.Protocol.ldp 模块中), 202
edit_ldp_port_config() (TesterLibrary.base.TesterLibrary 静态方法),

826
edit_ldp_port_config() (在 TesterLibrary.Protocol.ldp 模块中), 204
edit_lsp_ping() (TesterLibrary.base.TesterLibrary 静态方法), 826
edit_lsp_ping() (在 TesterLibrary.Wizard.mpls 模块中), 489
edit_lsp_ping_port_config() (TesterLibrary.base.TesterLibrary 静态方法), 826
edit_lsp_ping_port_config() (在 TesterLibrary.Protocol.lsp_ping 模块中), 215
edit_mld() (TesterLibrary.base.TesterLibrary 静态方法), 827
edit_mld() (在 TesterLibrary.Protocol.mld 模块中), 222
edit_mld_querier() (TesterLibrary.base.TesterLibrary 静态方法), 828
edit_mld_querier() (在 TesterLibrary.Protocol.mld 模块中), 223
edit_modifier() (TesterLibrary.base.TesterLibrary 静态方法), 828
edit_modifier() (在 TesterLibrary.Stream.common 模块中), 459
edit_mpls_customer_port() (TesterLibrary.base.TesterLibrary 静态方法), 831
edit_mpls_customer_port() (在 TesterLibrary.Wizard.mpls 模块中), 489
edit_mpls_fec128() (TesterLibrary.base.TesterLibrary 静态方法), 831
edit_mpls_fec128() (在 TesterLibrary.Wizard.mpls 模块中), 490
edit_mpls_fec129() (TesterLibrary.base.TesterLibrary 静态方法), 831
edit_mpls_fec129() (在 TesterLibrary.Wizard.mpls 模块中), 490
edit_mpls_host() (TesterLibrary.base.TesterLibrary 静态方法), 832
edit_mpls_host() (在 TesterLibrary.Wizard.mpls 模块中), 490
edit_mpls_provider_port() (TesterLibrary.base.TesterLibrary 静态方法), 833
edit_mpls_provider_port() (在 TesterLibrary.Wizard.mpls 模块中), 491
edit_mpls_provider_route_reflector() (TesterLibrary.base.TesterLibrary 静态方法), 833
edit_mpls_provider_route_reflector() (在 TesterLibrary.Wizard.mpls 模块中), 492
edit_mpls_provider_router_basic_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 833
edit_mpls_provider_router_basic_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 492
edit_mpls_provider_router_isis() (TesterLibrary.base.TesterLibrary 静态方法), 834
edit_mpls_provider_router_isis() (在 TesterLibrary.Wizard.mpls 模块中), 492
edit_mpls_provider_router_ldp() (TesterLibrary.base.TesterLibrary 静态方法), 834
edit_mpls_provider_router_ldp() (在 TesterLibrary.Wizard.mpls 模块中), 493
edit_mpls_provider_router_ospf() (TesterLibrary.base.TesterLibrary 静态方法), 835
edit_mpls_provider_router_ospf() (在 TesterLibrary.Wizard.mpls 模块中), 493
edit_mpls_provider_router_rip() (TesterLibrary.base.TesterLibrary 静态方法), 836
edit_mpls_provider_router_rip() (在 TesterLibrary.Wizard.mpls 模块中), 494
edit_mpls_pwe_basic_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 836
edit_mpls_pwe_basic_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 494
edit_mpls_vpls_basic_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 837
edit_mpls_vpls_basic_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 495
edit_mpls_vpn_as_number() (TesterLibrary.base.TesterLibrary 静态方法), 837
edit_mpls_vpn_as_number() (在 TesterLibrary.Wizard.mpls 模块中), 496
edit_mpls_vpn_customer_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 837
edit_mpls_vpn_customer_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 496
edit_mpls_vpn_ipv4_route_customer_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 838
edit_mpls_vpn_ipv4_route_customer_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 496
edit_mpls_vpn_ipv4_route_provider_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 838
edit_mpls_vpn_ipv4_route_provider_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 496
edit_mpls_vpn_ipv6_route_customer_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 838
edit_mpls_vpn_ipv6_route_customer_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 497
edit_mpls_vpn_ipv6_route_provider_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 839
edit_mpls_vpn_ipv6_route_provider_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 497
edit_mpls_vpn_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 839
edit_mpls_vpn_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 498
edit_mpls_vpn_provider_parameters() (TesterLibrary.base.TesterLibrary 静态方法), 839
edit_mpls_vpn_provider_parameters() (在 TesterLibrary.Wizard.mpls 模块中), 498
edit_ospf() (TesterLibrary.base.TesterLibrary 静态方法), 840
edit_ospf() (在 TesterLibrary.Protocol.ospfv2 模块中), 253
edit_ospf_port_config() (TesterLibrary.base.TesterLibrary 静态方法), 841
edit_ospf_port_config() (在 TesterLibrary.Protocol.ospfv2 模块中), 255
edit_ospf_te_lsa_link() (TesterLibrary.base.TesterLibrary 静态方法), 842
edit_ospf_te_lsa_link() (在 TesterLibrary.Protocol.ospfv2 模块中), 255
edit_ospfv3() (TesterLibrary.base.TesterLibrary 静态方法), 843
edit_ospfv3() (在 TesterLibrary.Protocol.ospfv3 模块

中), 288
edit_ospfv3_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 845
edit_ospfv3_port_config() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 290
edit_overall_setting() (*TesterLibrary.base.TesterLibrary* 静态方法), 845
edit_overall_setting() (在 *TesterLibrary.Overall.common* 模块中), 2
edit_pcep() (*TesterLibrary.base.TesterLibrary* 静态方法), 846
edit_pcep() (在 *TesterLibrary.Protocol.pcep* 模块中), 320
edit_pcep_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 849
edit_pcep_port_config() (在 *TesterLibrary.Protocol.pcep* 模块中), 323
edit_pim() (*TesterLibrary.base.TesterLibrary* 静态方法), 849
edit_pim() (在 *TesterLibrary.Protocol.pim* 模块中), 340
edit_pim_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 851
edit_pim_port_config() (在 *TesterLibrary.Protocol.pim* 模块中), 341
edit_port() (*TesterLibrary.base.TesterLibrary* 静态方法), 851
edit_port() (在 *TesterLibrary.Port.common* 模块中), 12
edit_port_load_profile() (*TesterLibrary.base.TesterLibrary* 静态方法), 852
edit_port_load_profile() (在 *TesterLibrary.Port.common* 模块中), 12
edit_pppoe_clinet() (*TesterLibrary.base.TesterLibrary* 静态方法), 853
edit_pppoe_clinet() (在 *TesterLibrary.Protocol.pppoe* 模块中), 349
edit_rip() (*TesterLibrary.base.TesterLibrary* 静态方法), 856
edit_rip() (在 *TesterLibrary.Protocol.rip* 模块中), 364
edit_rip_port_config() (*TesterLibrary.base.TesterLibrary* 静态方法), 857
edit_rip_port_config() (在 *TesterLibrary.Protocol.rip* 模块中), 365
edit_stream() (*TesterLibrary.base.TesterLibrary* 静态方法), 857
edit_stream() (在 *TesterLibrary.Stream.common* 模块中), 461
edit_stream_load_profile() (*TesterLibrary.base.TesterLibrary* 静态方法), 858
edit_stream_load_profile() (在 *TesterLibrary.Port.common* 模块中), 14
edit_traffic_parameters() (*TesterLibrary.base.TesterLibrary* 静态方法), 859
edit_traffic_parameters() (在 *TesterLibrary.Wizard.mpls* 模块中), 498
edit_vxlan() (*TesterLibrary.base.TesterLibrary* 静态方法), 859
edit_vxlan() (在 *TesterLibrary.Protocol.vxlan* 模块中), 371
establish_bgp() (*TesterLibrary.base.TesterLibrary* 静态方法), 860
establish_bgp() (在 *TesterLibrary.Protocol.bgp* 模块中), 94
establish_ldp() (*TesterLibrary.base.TesterLibrary* 静态方法), 860
establish_ldp() (在 *TesterLibrary.Protocol.ldp* 模块中), 205
establish_ospf() (*TesterLibrary.base.TesterLibrary* 静态方法), 861
establish_ospf() (在 *TesterLibrary.Protocol.ospfv2* 模块中), 256
establish_ospfv3() (*TesterLibrary.base.TesterLibrary* 静态方法), 861
establish_ospfv3() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 290
expand_benchmark() (*TesterLibrary.base.TesterLibrary* 静态方法), 861
expand_benchmark() (在 *TesterLibrary.Wizard.benchmark* 模块中), 486
expand_mpls_wizard() (*TesterLibrary.base.TesterLibrary* 静态方法), 861
expand_mpls_wizard() (在 *TesterLibrary.Wizard.mpls* 模块中), 499
export_benchmark_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 862
export_benchmark_result() (在 *TesterLibrary.Wizard.benchmark* 模块中), 486

F

format_benchmark_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 862
format_benchmark_result() (在 *TesterLibrary.Wizard.benchmark* 模块中), 487

G

get_benchmark_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 862
get_benchmark_result() (在 *TesterLibrary.Wizard.benchmark* 模块中), 487
get_bfd_ipv4_session_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 863
get_bfd_ipv4_session_result() (在 *TesterLibrary.Protocol.bfd* 模块中), 27
get_bfd_ipv6_session_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 864
get_bfd_ipv6_session_result() (在 *TesterLibrary.Protocol.bfd* 模块中), 28
get_bfd_isis_ipv6_session_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 865
get_bfd_isis_ipv6_session_result() (在 *TesterLibrary.Protocol.bfd* 模块中), 29
get_bfd_isis_session_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 866
get_bfd_isis_session_result() (在 *TesterLibrary.Protocol.bfd* 模块中), 30
get_bfd_ospfv2_session_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 868
get_bfd_ospfv2_session_result() (在 *TesterLibrary.Protocol.bfd* 模块中), 32

get_bfd_ospfv3_session_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 869
get_bfd_ospfv3_session_result() (在 *TesterLibrary.Protocol.bfd* 模块中), 33
get_bfd_session_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 870
get_bfd_session_result() (在 *TesterLibrary.Protocol.bfd* 模块中), 34
get_bgp_evpn_routes_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 871
get_bgp_evpn_routes_statistic() (在 *TesterLibrary.Protocol.bgp* 模块中), 95
get_bgp_link_state_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 872
get_bgp_link_state_statistic() (在 *TesterLibrary.Protocol.bgp* 模块中), 96
get_bgp_router_from_route_pool() (*TesterLibrary.base.TesterLibrary* 静态方法), 873
get_bgp_router_from_route_pool() (在 *TesterLibrary.Protocol.bgp* 模块中), 97
get_bgp_session_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 873
get_bgp_session_block_statistic() (在 *TesterLibrary.Protocol.bgp* 模块中), 97
get_bgp_session_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 874
get_bgp_session_statistic() (在 *TesterLibrary.Protocol.bgp* 模块中), 98
get_capture_info() (*TesterLibrary.base.TesterLibrary* 静态方法), 875
get_capture_info() (在 *TesterLibrary.Port.capture* 模块中), 10
get_config_children() (*TesterLibrary.base.TesterLibrary* 静态方法), 876
get_config_children() (在 *TesterLibrary.Overall.common* 模块中), 3
get_configs() (*TesterLibrary.base.TesterLibrary* 静态方法), 876
get_configs() (在 *TesterLibrary.Overall.common* 模块中), 3
get_dhcp_client_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 876
get_dhcp_client_block_statistic() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 112
get_dhcp_client_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 877
get_dhcp_client_statistic() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 113
get_dhcp_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 878
get_dhcp_port_statistic() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 113
get_dhcp_server_lease_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 879
get_dhcp_server_lease_statistic() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 114
get_dhcp_server_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 880
get_dhcp_server_statistic() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 115
get_dhcpv6_client_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 881
get_dhcpv6_client_block_statistic() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 130
get_dhcpv6_client_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 882
get_dhcpv6_client_statistic() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 132
get_dhcpv6_pd_client_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 883
get_dhcpv6_pd_client_statistic() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 133
get_dhcpv6_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 884
get_dhcpv6_port_statistic() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 134
get_dhcpv6_server_lease_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 885
get_dhcpv6_server_lease_statistic() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 135
get_dhcpv6_server_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 886
get_dhcpv6_server_statistic() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 136
get_dot1x_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 887
get_dot1x_block_statistic() (在 *TesterLibrary.Protocol.dot1x* 模块中), 141
get_dot1x_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 888
get_dot1x_port_statistic() (在 *TesterLibrary.Protocol.dot1x* 模块中), 142
get_dot1x_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 889
get_dot1x_statistic() (在 *TesterLibrary.Protocol.dot1x* 模块中), 143
get_gateway_mac() (*TesterLibrary.base.TesterLibrary* 静态方法), 890
get_gateway_mac() (在 *TesterLibrary.Port.interface* 模块中), 20
get_igmp_host_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 890
get_igmp_host_statistic() (在 *TesterLibrary.Protocol.igmp* 模块中), 148
get_igmp_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 891
get_igmp_port_statistic() (在 *TesterLibrary.Protocol.igmp* 模块中), 148
get_igmp_querier_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 892
get_igmp_querier_statistic() (在 *TesterLibrary.Protocol.igmp* 模块中), 149
get_imix_from_name() (*TesterLibrary.base.TesterLibrary* 静态方法), 892
get_imix_from_name() (在 *TesterLibrary.Stream.imix* 模块中), 468
get_interfaces() (*TesterLibrary.base.TesterLibrary* 静态方法), 893
get_interfaces() (在 *TesterLibrary.Port.interface* 模

块中), 20
get_isis_mt_params() (*TesterLibrary.base.TesterLibrary* 静态方法), 893
get_isis_mt_params() (在 *TesterLibrary.Protocol.isis* 模块中), 180
get_isis_per_pdu() (*TesterLibrary.base.TesterLibrary* 静态方法), 893
get_isis_per_pdu() (在 *TesterLibrary.Protocol.isis* 模块中), 180
get_isis_router_from_tlv() (*TesterLibrary.base.TesterLibrary* 静态方法), 894
get_isis_router_from_tlv() (在 *TesterLibrary.Protocol.isis* 模块中), 181
get_isis_session_stats() (*TesterLibrary.base.TesterLibrary* 静态方法), 894
get_isis_session_stats() (在 *TesterLibrary.Protocol.isis* 模块中), 181
get_isis_tlv_stats() (*TesterLibrary.base.TesterLibrary* 静态方法), 895
get_isis_tlv_stats() (在 *TesterLibrary.Protocol.isis* 模块中), 182
get_l2tp_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 896
get_l2tp_block_statistic() (在 *TesterLibrary.Protocol.l2tp* 模块中), 192
get_l2tp_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 898
get_l2tp_port_statistic() (在 *TesterLibrary.Protocol.l2tp* 模块中), 193
get_l2tp_session_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 898
get_l2tp_session_statistic() (在 *TesterLibrary.Protocol.l2tp* 模块中), 194
get_l2tp_tunnel_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 899
get_l2tp_tunnel_statistic() (在 *TesterLibrary.Protocol.l2tp* 模块中), 195
get_layer_from_interfaces() (*TesterLibrary.base.TesterLibrary* 静态方法), 901
get_layer_from_interfaces() (在 *TesterLibrary.Port.interface* 模块中), 20
get_ldp_lsp_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 901
get_ldp_lsp_statistic() (在 *TesterLibrary.Protocol.ldp* 模块中), 205
get_ldp_point_from_lsp() (*TesterLibrary.base.TesterLibrary* 静态方法), 902
get_ldp_point_from_lsp() (在 *TesterLibrary.Protocol.ldp* 模块中), 205
get_ldp_session_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 902
get_ldp_session_statistic() (在 *TesterLibrary.Protocol.ldp* 模块中), 206
get_lsp_ping_echo_request_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 903
get_lsp_ping_echo_request_statistic() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 215
get_lsp_ping_session_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 904
get_lsp_ping_session_statistic() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 216
get_lsp_trace_echo_request_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 904
get_lsp_trace_echo_request_statistic() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 217
get_mld_host_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 905
get_mld_host_statistic() (在 *TesterLibrary.Protocol.mld* 模块中), 224
get_mld_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 906
get_mld_port_statistic() (在 *TesterLibrary.Protocol.mld* 模块中), 224
get_mld_querier_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 907
get_mld_querier_statistic() (在 *TesterLibrary.Protocol.mld* 模块中), 225
get_ospf_router_from_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 907
get_ospf_router_from_lsa() (在 *TesterLibrary.Protocol.ospfv2* 模块中), 257
get_ospf_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 908
get_ospf_statistic() (在 *TesterLibrary.Protocol.ospfv2* 模块中), 257
get_ospfv3_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 909
get_ospfv3_statistic() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 291
get_pcep_lsp_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 910
get_pcep_lsp_block_statistic() (在 *TesterLibrary.Protocol.pcep* 模块中), 323
get_pcep_lsp_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 911
get_pcep_lsp_statistic() (在 *TesterLibrary.Protocol.pcep* 模块中), 324
get_pcep_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 912
get_pcep_port_statistic() (在 *TesterLibrary.Protocol.pcep* 模块中), 325
get_pcep_session_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 913
get_pcep_session_block_statistic() (在 *TesterLibrary.Protocol.pcep* 模块中), 326
get_pcep_session_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 914
get_pcep_session_statistic() (在 *TesterLibrary.Protocol.pcep* 模块中), 327
get_pim_group_stats() (*TesterLibrary.base.TesterLibrary* 静态方法), 916
get_pim_group_stats() (在 *TesterLibrary.Protocol.pim* 模块中), 342
get_pim_session_stats() (*TesterLibrary.base.TesterLibrary* 静态方法), 916
get_pim_session_stats() (在 *TesterLibrary.Protocol.pim* 模块中), 342

get_port_latency_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 917
get_port_latency_statistic() (在 *TesterLibrary.Statistic.common* 模块中), 374
get_port_speed() (*TesterLibrary.base.TesterLibrary* 静态方法), 918
get_port_speed() (在 *TesterLibrary.Port.common* 模块中), 15
get_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 918
get_port_statistic() (在 *TesterLibrary.Statistic.common* 模块中), 375
get_ports() (*TesterLibrary.base.TesterLibrary* 静态方法), 921
get_ports() (在 *TesterLibrary.Port.common* 模块中), 15
get_pppoe_client_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 922
get_pppoe_client_block_statistic() (在 *TesterLibrary.Protocol.pppoe* 模块中), 352
get_pppoe_client_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 923
get_pppoe_client_statistic() (在 *TesterLibrary.Protocol.pppoe* 模块中), 354
get_pppoe_port_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 925
get_pppoe_port_statistic() (在 *TesterLibrary.Protocol.pppoe* 模块中), 356
get_pppoe_server_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 926
get_pppoe_server_block_statistic() (在 *TesterLibrary.Protocol.pppoe* 模块中), 356
get_pppoe_server_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 928
get_pppoe_server_statistic() (在 *TesterLibrary.Protocol.pppoe* 模块中), 358
get_rip_router_from_route() (*TesterLibrary.base.TesterLibrary* 静态方法), 930
get_rip_router_from_route() (在 *TesterLibrary.Protocol.rip* 模块中), 365
get_rip_session_block_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 930
get_rip_session_block_statistic() (在 *TesterLibrary.Protocol.rip* 模块中), 366
get_rip_session_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 931
get_rip_session_statistic() (在 *TesterLibrary.Protocol.rip* 模块中), 366
get_sessions() (*TesterLibrary.base.TesterLibrary* 静态方法), 931
get_sessions() (在 *TesterLibrary.Protocol.common* 模块中), 102
get_stream_rx_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 932
get_stream_rx_statistic() (在 *TesterLibrary.Statistic.common* 模块中), 378
get_stream_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 934
get_stream_statistic() (在 *TesterLibrary.Statistic.common* 模块中), 380
get_stream_tx_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 936
get_stream_tx_statistic() (在 *TesterLibrary.Statistic.common* 模块中), 381
get_streamblock_rx_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 937
get_streamblock_rx_statistic() (在 *TesterLibrary.Statistic.common* 模块中), 382
get_streamblock_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 938
get_streamblock_statistic() (在 *TesterLibrary.Statistic.common* 模块中), 384
get_streamblock_tx_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 940
get_streamblock_tx_statistic() (在 *TesterLibrary.Statistic.common* 模块中), 386
get_streams() (*TesterLibrary.base.TesterLibrary* 静态方法), 941
get_streams() (在 *TesterLibrary.Stream.common* 模块中), 463
get_vxlan_statistic() (*TesterLibrary.base.TesterLibrary* 静态方法), 941
get_vxlan_statistic() (在 *TesterLibrary.Protocol.vxlan* 模块中), 372
get_vxlan_vm_property() (*TesterLibrary.base.TesterLibrary* 静态方法), 942
get_vxlan_vm_property() (在 *TesterLibrary.Protocol.vxlan* 模块中), 373
grace_restart_ospf() (*TesterLibrary.base.TesterLibrary* 静态方法), 942
grace_restart_ospf() (在 *TesterLibrary.Protocol.ospfv2* 模块中), 258
grace_restart_ospfv3() (*TesterLibrary.base.TesterLibrary* 静态方法), 942
grace_restart_ospfv3() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 292
graceful_restart_isis() (*TesterLibrary.base.TesterLibrary* 静态方法), 942
graceful_restart_isis() (在 *TesterLibrary.Protocol.isis* 模块中), 183

I

init_tester() (*TesterLibrary.base.TesterLibrary* 静态方法), 943
init_tester() (在 *TesterLibrary.Overall.common* 模块中), 4

L

load_case() (*TesterLibrary.base.TesterLibrary* 静态方法), 943
load_case() (在 *TesterLibrary.Overall.common* 模块中), 4

M

Meta (*TesterLibrary.base* 中的类), 499

P

pause_lsp_ping() (*TesterLibrary.base.TesterLibrary* 静态方法), 943
pause_lsp_ping() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 218
pause_lsp_trace() (*TesterLibrary.base.TesterLibrary* 静态方法), 943

pause_lsp_trace() (在 TesterLibrary.Protocol.lsp_ping 模块中), 218
pcep_establish() (TesterLibrary.base.TesterLibrary 静态方法), 944
pcep_establish() (在 TesterLibrary.Protocol.pcep 模块中), 329
pcep_pcc_delegate_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 944
pcep_pcc_delegate_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 329
pcep_pcc_end_sync() (TesterLibrary.base.TesterLibrary 静态方法), 944
pcep_pcc_end_sync() (在 TesterLibrary.Protocol.pcep 模块中), 329
pcep_pcc_initial_sync() (TesterLibrary.base.TesterLibrary 静态方法), 944
pcep_pcc_initial_sync() (在 TesterLibrary.Protocol.pcep 模块中), 330
pcep_pcc_remove_delegate_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 945
pcep_pcc_remove_delegate_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 330
pcep_pcc_report_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 945
pcep_pcc_report_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 330
pcep_pcc_request_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 945
pcep_pcc_request_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 331
pcep_pcc_revoke_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 946
pcep_pcc_revoke_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 331
pcep_pcc_synchronize_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 946
pcep_pcc_synchronize_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 331
pcep_pce_initiate_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 946
pcep_pce_initiate_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 332
pcep_pce_remove_initiated_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 947
pcep_pce_remove_initiated_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 332
pcep_pce_return_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 947
pcep_pce_return_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 332
pcep_pce_update_lsp() (TesterLibrary.base.TesterLibrary 静态方法), 947
pcep_pce_update_lsp() (在 TesterLibrary.Protocol.pcep 模块中), 333
pcep_resume_keep_alive() (TesterLibrary.base.TesterLibrary 静态方法), 948
pcep_resume_keep_alive() (在 TesterLibrary.Protocol.pcep 模块中), 333
pcep_stop_keep_alive() (TesterLibrary.base.TesterLibrary 静态方法), 948
pcep_stop_keep_alive() (在 TesterLibrary.Protocol.pcep 模块中), 333
pim_change_gen_id() (TesterLibrary.base.TesterLibrary 静态方法), 948
pim_change_gen_id() (在 TesterLibrary.Protocol.pim 模块中), 343
pim_join_group() (TesterLibrary.base.TesterLibrary 静态方法), 948
pim_join_group() (在 TesterLibrary.Protocol.pim 模块中), 343
pim_leave_group() (TesterLibrary.base.TesterLibrary 静态方法), 949
pim_leave_group() (在 TesterLibrary.Protocol.pim 模块中), 344
pim_start_boot_strap() (TesterLibrary.base.TesterLibrary 静态方法), 949
pim_start_boot_strap() (在 TesterLibrary.Protocol.pim 模块中), 344
pim_start_register() (TesterLibrary.base.TesterLibrary 静态方法), 949
pim_start_register() (在 TesterLibrary.Protocol.pim 模块中), 344
pim_stop_boot_strap() (TesterLibrary.base.TesterLibrary 静态方法), 949
pim_stop_boot_strap() (在 TesterLibrary.Protocol.pim 模块中), 344
pim_stop_register() (TesterLibrary.base.TesterLibrary 静态方法), 950
pim_stop_register() (在 TesterLibrary.Protocol.pim 模块中), 345

R

relate_benchmark_ports() (TesterLibrary.base.TesterLibrary 静态方法), 950
relate_benchmark_ports() (在 TesterLibrary.Wizard.benchmark 模块中), 487
release_port() (TesterLibrary.base.TesterLibrary 静态方法), 950
release_port() (在 TesterLibrary.Port.common 模块中), 15
relocate_ports() (TesterLibrary.base.TesterLibrary 静态方法), 951
relocate_ports() (在 TesterLibrary.Port.common 模块中), 15
request_ldp_label() (TesterLibrary.base.TesterLibrary 静态方法), 951
request_ldp_label() (在 TesterLibrary.Protocol.ldp 模块中), 207
reserve_port() (TesterLibrary.base.TesterLibrary 静态方法), 951
reserve_port() (在 TesterLibrary.Port.common 模块中), 16
reset_statistic() (TesterLibrary.base.TesterLibrary 静态方法), 952
reset_statistic() (在 TesterLibrary.Statistic.common 模块中), 386
reset_tester() (TesterLibrary.base.TesterLibrary 静态方法), 952
reset_tester() (在 TesterLibrary.Overall.common 模块中), 4
restart_ldp() (TesterLibrary.base.TesterLibrary 静态方法), 952
restart_ldp() (在 TesterLibrary.Protocol.ldp 模块中), 207

resume_ldp_hello() (*TesterLibrary.base.TesterLibrary* 静态方法), 952
resume_ldp_hello() (在 *TesterLibrary.Protocol.ldp* 模块中), 208
resume_ldp_keepalive() (*TesterLibrary.base.TesterLibrary* 静态方法), 953
resume_ldp_keepalive() (在 *TesterLibrary.Protocol.ldp* 模块中), 208
resume_lsp_ping() (*TesterLibrary.base.TesterLibrary* 静态方法), 953
resume_lsp_ping() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 218
resume_lsp_trace() (*TesterLibrary.base.TesterLibrary* 静态方法), 953
resume_lsp_trace() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 218
resume_rip() (*TesterLibrary.base.TesterLibrary* 静态方法), 953
resume_rip() (在 *TesterLibrary.Protocol.rip* 模块中), 367
ROBOT_AUTO_KEYWORDS (*TesterLibrary.base.TesterLibrary* 属性), 499
ROBOT_LIBRARY_SCOPE (*TesterLibrary.base.TesterLibrary* 属性), 499
run_benchmark() (*TesterLibrary.base.TesterLibrary* 静态方法), 954
run_benchmark() (在 *TesterLibrary.Wizard.benchmark* 模块中), 488

S

save_case() (*TesterLibrary.base.TesterLibrary* 静态方法), 954
save_case() (在 *TesterLibrary.Overall.common* 模块中), 5
save_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 954
save_result() (在 *TesterLibrary.Statistic.common* 模块中), 387
select_interface() (*TesterLibrary.base.TesterLibrary* 静态方法), 955
select_interface() (在 *TesterLibrary.Protocol.common* 模块中), 103
select_rx_port() (*TesterLibrary.base.TesterLibrary* 静态方法), 955
select_rx_port() (在 *TesterLibrary.Stream.common* 模块中), 463
select_source_interface() (*TesterLibrary.base.TesterLibrary* 静态方法), 955
select_source_interface() (在 *TesterLibrary.Protocol.igmp* 模块中), 150
start_arp() (*TesterLibrary.base.TesterLibrary* 静态方法), 956
start_arp() (在 *TesterLibrary.Port.interface* 模块中), 20
start_capture() (*TesterLibrary.base.TesterLibrary* 静态方法), 956
start_capture() (在 *TesterLibrary.Port.capture* 模块中), 11
start_l2_learning() (*TesterLibrary.base.TesterLibrary* 静态方法), 956
start_l2_learning() (在 *TesterLibrary.Stream.common* 模块中), 463
start_l3_learning() (*TesterLibrary.base.TesterLibrary* 静态方法), 957
start_l3_learning() (在 *TesterLibrary.Stream.common* 模块中), 464
start_ldp() (*TesterLibrary.base.TesterLibrary* 静态方法), 957
start_ldp() (在 *TesterLibrary.Protocol.ldp* 模块中), 208
start_lsp_ping() (*TesterLibrary.base.TesterLibrary* 静态方法), 958
start_lsp_ping() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 219
start_protocol() (*TesterLibrary.base.TesterLibrary* 静态方法), 958
start_protocol() (在 *TesterLibrary.Protocol.common* 模块中), 103
start_stream() (*TesterLibrary.base.TesterLibrary* 静态方法), 958
start_stream() (在 *TesterLibrary.Stream.common* 模块中), 464
start_vxlan_ping() (*TesterLibrary.base.TesterLibrary* 静态方法), 959
start_vxlan_ping() (在 *TesterLibrary.Protocol.vxlan* 模块中), 373
stop_arp() (*TesterLibrary.base.TesterLibrary* 静态方法), 959
stop_arp() (在 *TesterLibrary.Port.interface* 模块中), 21
stop_capture() (*TesterLibrary.base.TesterLibrary* 静态方法), 959
stop_capture() (在 *TesterLibrary.Port.capture* 模块中), 11
stop_l2_learning() (*TesterLibrary.base.TesterLibrary* 静态方法), 959
stop_l2_learning() (在 *TesterLibrary.Stream.common* 模块中), 465
stop_l3_learning() (*TesterLibrary.base.TesterLibrary* 静态方法), 960
stop_l3_learning() (在 *TesterLibrary.Stream.common* 模块中), 465
stop_ldp() (*TesterLibrary.base.TesterLibrary* 静态方法), 960
stop_ldp() (在 *TesterLibrary.Protocol.ldp* 模块中), 208
stop_ldp_hello() (*TesterLibrary.base.TesterLibrary* 静态方法), 960
stop_ldp_hello() (在 *TesterLibrary.Protocol.ldp* 模块中), 209
stop_ldp_keepalive() (*TesterLibrary.base.TesterLibrary* 静态方法), 961
stop_ldp_keepalive() (在 *TesterLibrary.Protocol.ldp* 模块中), 209
stop_lsp_ping() (*TesterLibrary.base.TesterLibrary* 静态方法), 961
stop_lsp_ping() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 219
stop_protocol() (*TesterLibrary.base.TesterLibrary* 静态方法), 961
stop_protocol() (在 *TesterLibrary.Protocol.common* 模块中), 104
stop_stream() (*TesterLibrary.base.TesterLibrary* 静态方法), 962
stop_stream() (在 *TesterLibrary.Stream.common* 模块中), 465
stop_vxlan_ping() (*TesterLibrary.base.TesterLibrary* 静态方法), 962
stop_vxlan_ping() (在 *TesterLibrary.Protocol.vxlan* 模块中), 373
subscribe_result() (*TesterLibrary.base.TesterLibrary* 静态方法), 962
subscribe_result() (在 *TesterLibrary.Statistic.common* 模块中), 387
suspend_rip() (*TesterLibrary.base.TesterLibrary* 静态方法), 963

suspend_rip() (在 TesterLibrary.Protocol.rip 模块中), 367

T

TesterLibrary
    模块, 984
TesterLibrary (TesterLibrary.base 中的类), 499
TesterLibrary.base
    模块, 499
TesterLibrary.data
    模块, 984
TesterLibrary.Overall
    模块, 5
TesterLibrary.Overall.common
    模块, 1
TesterLibrary.Port
    模块, 21
TesterLibrary.Port.capture
    模块, 5
TesterLibrary.Port.common
    模块, 11
TesterLibrary.Port.interface
    模块, 17
TesterLibrary.Protocol
    模块, 374
TesterLibrary.Protocol.bfd
    模块, 21
TesterLibrary.Protocol.bgp
    模块, 36
TesterLibrary.Protocol.common
    模块, 102
TesterLibrary.Protocol.dhcpv4
    模块, 104
TesterLibrary.Protocol.dhcpv6
    模块, 117
TesterLibrary.Protocol.dot1x
    模块, 139
TesterLibrary.Protocol.igmp
    模块, 144
TesterLibrary.Protocol.isis
    模块, 151
TesterLibrary.Protocol.l2tp
    模块, 186
TesterLibrary.Protocol.ldp
    模块, 197
TesterLibrary.Protocol.lsp_ping
    模块, 210
TesterLibrary.Protocol.mld
    模块, 221
TesterLibrary.Protocol.multicast
    模块, 227
TesterLibrary.Protocol.ospfv2
    模块, 229
TesterLibrary.Protocol.ospfv3
    模块, 260
TesterLibrary.Protocol.pcep
    模块, 294
TesterLibrary.Protocol.pim
    模块, 334
TesterLibrary.Protocol.pppoe
    模块, 346
TesterLibrary.Protocol.rip
    模块, 361
TesterLibrary.Protocol.vxlan
    模块, 369
TesterLibrary.Statistic
    模块, 388
TesterLibrary.Statistic.common
    模块, 374
TesterLibrary.Stream
    模块, 468
TesterLibrary.Stream.common
    模块, 456
TesterLibrary.Stream.Header
    模块, 456
TesterLibrary.Stream.Header.Access
    模块, 390
TesterLibrary.Stream.Header.Access.common
    模块, 388
TesterLibrary.Stream.Header.Access.l2tpv2
    模块, 389
TesterLibrary.Stream.Header.Basic
    模块, 390
TesterLibrary.Stream.Header.Basic.common
    模块, 390
TesterLibrary.Stream.Header.Gre
    模块, 391
TesterLibrary.Stream.Header.Gre.common
    模块, 390
TesterLibrary.Stream.Header.L2
    模块, 409
TesterLibrary.Stream.Header.L2.common
    模块, 391
TesterLibrary.Stream.Header.L2.isis
    模块, 393
TesterLibrary.Stream.Header.L3
    模块, 442
TesterLibrary.Stream.Header.L3.common
    模块, 409
TesterLibrary.Stream.Header.L3.icmpv4
    模块, 413
TesterLibrary.Stream.Header.L3.icmpv6
    模块, 424
TesterLibrary.Stream.Header.L3.igmp
    模块, 437
TesterLibrary.Stream.Header.L4
    模块, 443
TesterLibrary.Stream.Header.L4.common
    模块, 442
TesterLibrary.Stream.Header.Routing
    模块, 456
TesterLibrary.Stream.Header.Routing.ospfv2
    模块, 443
TesterLibrary.Stream.imix
    模块, 466
TesterLibrary.Wizard
    模块, 499
TesterLibrary.Wizard.benchmark
    模块, 468
TesterLibrary.Wizard.mpls
    模块, 489

W

wait_bfd_state() (TesterLibrary.base.TesterLibrary 静态方法), 963
wait_bfd_state() (在 TesterLibrary.Protocol.bfd 模块中), 35
wait_bgp_ipv4_router_state() (TesterLibrary.base.TesterLibrary 静态方法), 964
wait_bgp_ipv4_router_state() (在 TesterLibrary.Protocol.bgp 模块中), 99
wait_bgp_ipv6_router_state() (TesterLibrary.base.TesterLibrary 静态方法), 964
wait_bgp_ipv6_router_state() (在 TesterLibrary.Protocol.bgp 模块中), 100
wait_bgp_router_state() (TesterLibrary.base.TesterLibrary 静态方法), 965
wait_bgp_router_state() (在 TesterLibrary.Protocol.bgp 模块中), 100
wait_bgp_state() (TesterLibrary.base.TesterLibrary 静态方法), 965
wait_bgp_state() (在 TesterLibrary.Protocol.bgp 模块中), 101

wait_dhcp_client_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 966
wait_dhcp_client_state() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 116
wait_dhcp_server_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 966
wait_dhcp_server_state() (在 *TesterLibrary.Protocol.dhcpv4* 模块中), 117
wait_dhcpv6_client_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 967
wait_dhcpv6_client_state() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 137
wait_dhcpv6_pd_client_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 968
wait_dhcpv6_pd_client_state() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 138
wait_dhcpv6_server_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 968
wait_dhcpv6_server_state() (在 *TesterLibrary.Protocol.dhcpv6* 模块中), 138
wait_dot1x_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 969
wait_dot1x_state() (在 *TesterLibrary.Protocol.dot1x* 模块中), 143
wait_igmp_querier_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 969
wait_igmp_querier_state() (在 *TesterLibrary.Protocol.igmp* 模块中), 150
wait_igmp_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 970
wait_igmp_state() (在 *TesterLibrary.Protocol.igmp* 模块中), 151
wait_isis_l1_broadcast_adj_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 970
wait_isis_l1_broadcast_adj_state() (在 *TesterLibrary.Protocol.isis* 模块中), 183
wait_isis_l2_broadcast_adj_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 971
wait_isis_l2_broadcast_adj_state() (在 *TesterLibrary.Protocol.isis* 模块中), 184
wait_isis_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 971
wait_isis_state() (在 *TesterLibrary.Protocol.isis* 模块中), 185
wait_isis_three_way_p2p_adj_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 972
wait_isis_three_way_p2p_adj_state() (在 *TesterLibrary.Protocol.isis* 模块中), 185
wait_l2tp_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 972
wait_l2tp_state() (在 *TesterLibrary.Protocol.l2tp* 模块中), 196
wait_ldp_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 973
wait_ldp_state() (在 *TesterLibrary.Protocol.ldp* 模块中), 209
wait_lsp_ping_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 974
wait_lsp_ping_state() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 219
wait_lsp_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 974
wait_lsp_state() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 220
wait_lsp_trace_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 975
wait_lsp_trace_state() (在 *TesterLibrary.Protocol.lsp_ping* 模块中), 220
wait_mld_querier_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 975
wait_mld_querier_state() (在 *TesterLibrary.Protocol.mld* 模块中), 226
wait_mld_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 976
wait_mld_state() (在 *TesterLibrary.Protocol.mld* 模块中), 226
wait_ospf_adjacency_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 976
wait_ospf_adjacency_state() (在 *TesterLibrary.Protocol.ospfv2* 模块中), 258
wait_ospf_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 977
wait_ospf_state() (在 *TesterLibrary.Protocol.ospfv2* 模块中), 259
wait_ospfv3_adjacency_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 977
wait_ospfv3_adjacency_state() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 292
wait_ospfv3_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 978
wait_ospfv3_state() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 293
wait_pcep_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 978
wait_pcep_state() (在 *TesterLibrary.Protocol.pcep* 模块中), 333
wait_pim_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 979
wait_pim_state() (在 *TesterLibrary.Protocol.pim* 模块中), 345
wait_port_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 980
wait_port_state() (在 *TesterLibrary.Port.common* 模块中), 16
wait_pppoe_ipcp_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 980
wait_pppoe_ipcp_state() (在 *TesterLibrary.Protocol.pppoe* 模块中), 360
wait_pppoe_ipv6cp_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 981
wait_pppoe_ipv6cp_state() (在 *TesterLibrary.Protocol.pppoe* 模块中), 361
wait_rip_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 981
wait_rip_state() (在 *TesterLibrary.Protocol.rip* 模块中), 368
wait_stream_state() (*TesterLibrary.base.TesterLibrary* 静态方法), 982
wait_stream_state() (在 *TesterLibrary.Stream.common* 模块中), 466
模块
    TesterLibrary, 984
    TesterLibrary.base, 499
    TesterLibrary.data, 984
    TesterLibrary.Overall, 5
    TesterLibrary.Overall.common, 1
    TesterLibrary.Port, 21

TesterLibrary.Port.capture, 5
TesterLibrary.Port.common, 11
TesterLibrary.Port.interface, 17
TesterLibrary.Protocol, 374
TesterLibrary.Protocol.bfd, 21
TesterLibrary.Protocol.bgp, 36
TesterLibrary.Protocol.common, 102
TesterLibrary.Protocol.dhcpv4, 104
TesterLibrary.Protocol.dhcpv6, 117
TesterLibrary.Protocol.dot1x, 139
TesterLibrary.Protocol.igmp, 144
TesterLibrary.Protocol.isis, 151
TesterLibrary.Protocol.l2tp, 186
TesterLibrary.Protocol.ldp, 197
TesterLibrary.Protocol.lsp_ping, 210
TesterLibrary.Protocol.mld, 221
TesterLibrary.Protocol.multicast, 227
TesterLibrary.Protocol.ospfv2, 229
TesterLibrary.Protocol.ospfv3, 260
TesterLibrary.Protocol.pcep, 294
TesterLibrary.Protocol.pim, 334
TesterLibrary.Protocol.pppoe, 346
TesterLibrary.Protocol.rip, 361
TesterLibrary.Protocol.vxlan, 369
TesterLibrary.Statistic, 388
TesterLibrary.Statistic.common, 374
TesterLibrary.Stream, 468
TesterLibrary.Stream.common, 456
TesterLibrary.Stream.Header, 456
TesterLibrary.Stream.Header.Access, 390
TesterLibrary.Stream.Header.Access.common, 388
TesterLibrary.Stream.Header.Access.l2tpv2, 389
TesterLibrary.Stream.Header.Basic, 390
TesterLibrary.Stream.Header.Basic.common, 390
TesterLibrary.Stream.Header.Gre, 391
TesterLibrary.Stream.Header.Gre.common, 390
TesterLibrary.Stream.Header.L2, 409
TesterLibrary.Stream.Header.L2.common, 391
TesterLibrary.Stream.Header.L2.isis, 393
TesterLibrary.Stream.Header.L3, 442
TesterLibrary.Stream.Header.L3.common, 409
TesterLibrary.Stream.Header.L3.icmpv4, 413
TesterLibrary.Stream.Header.L3.icmpv6, 424
TesterLibrary.Stream.Header.L3.igmp, 437
TesterLibrary.Stream.Header.L4, 443
TesterLibrary.Stream.Header.L4.common, 442
TesterLibrary.Stream.Header.Routing, 456
TesterLibrary.Stream.Header.Routing.ospfv2, 443
TesterLibrary.Stream.imix, 466
TesterLibrary.Wizard, 499
TesterLibrary.Wizard.benchmark, 468
TesterLibrary.Wizard.mpls, 489
withdraw_bgp() (*TesterLibrary.base.TesterLibrary* 静态方法), 982
withdraw_bgp() (在 *TesterLibrary.Protocol.bgp* 模块中), 102
withdraw_bgp_route() (*TesterLibrary.base.TesterLibrary* 静态方法), 982
withdraw_bgp_route() (在 *TesterLibrary.Protocol.bgp* 模块中), 102
withdraw_isis() (*TesterLibrary.base.TesterLibrary* 静态方法), 983
withdraw_isis() (在 *TesterLibrary.Protocol.isis* 模块中), 186
withdraw_ldb_label() (*TesterLibrary.base.TesterLibrary* 静态方法), 983
withdraw_ldb_label() (在 *TesterLibrary.Protocol.ldp* 模块中), 210
withdraw_ospf_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 983
withdraw_ospf_lsa() (在 *TesterLibrary.Protocol.ospfv2* 模块中), 260
withdraw_ospfv3_lsa() (*TesterLibrary.base.TesterLibrary* 静态方法), 984
withdraw_ospfv3_lsa() (在 *TesterLibrary.Protocol.ospfv3* 模块中), 293
withdraw_rip() (*TesterLibrary.base.TesterLibrary* 静态方法), 984
withdraw_rip() (在 *TesterLibrary.Protocol.rip* 模块中), 368