TS-009
NTAPI Emulated Devices

Abstract: Emulated Devices for Next-Gen High-Level API

Authors: Dan Diolosa (Spirent), Mickael Graham (Cisco), Nana He (Spirent), Todd Law (Spirent), Eric Miller (Spirent), Calvin Weng (Spirent), Sean Wu (Juniper)

Status: Release

Revision: 1

Revision date December 2015

Submission: ntaf ts-009

Revision History

Version	Date	By	Changes
1	2015-DEC-01	Technical Committee	First Release

Table of Contents

Works Cited

NTAF TS-005 Automation API Framework

1. Motivation

The motivation behind this specification is to define emulated devices and associated command(s) as part of the next-generation high-level API.

2. Introduction

The NTAF TS-005 specification laid down the framework for next-generation APIs at a high-level. TS-005 requires that NTAF APIs be language-agnostic, operating system agnostic, hardware-environment agnostic. In addition, TS-005 requires that APIs be object-oriented in nature. This specification provides the next logical step in defining high-level APIs by defining emulated devices, which are used for protocol emulations in traffic generators.

3. Concept

Next-generation high-level APIs will be object-oriented in nature. From a command perspective, high-level APIs reduce the number of commands to a reasonably small, consistent, and manageable set. A similar constraint applies to the objects defined in the high-level API. In other words, an object-oriented high-level API must define a reasonably small consistent and manageable set of objects. These objects are, in effect, aggregations of smaller objects in the data models of the underlying traffic generator.

4. Data Model

This specification follows the data model shown in Figure 1 below. The objects shown in the diagram are related to each other with parent-child relationships. For example, the "ProtocolCconfig" objects are children of the "EmulatedDevice" object.

Some objects can be both a parent and a child. For example, the "ProtocolConfig" object is simultaneously a child of the "EmulatedDevice" object, and a parent of the "ProtocolResults" object. Parent objects may also have multiple child objects. For example, the "EmulatedDevice" object can have multiple "ProtocolConfig" child objects under it.

This specification only defines the EmulatedDevice object. Objects to represent ProtocolConfigs and ProtocolResults are expected to be defined in follow-on specifications.

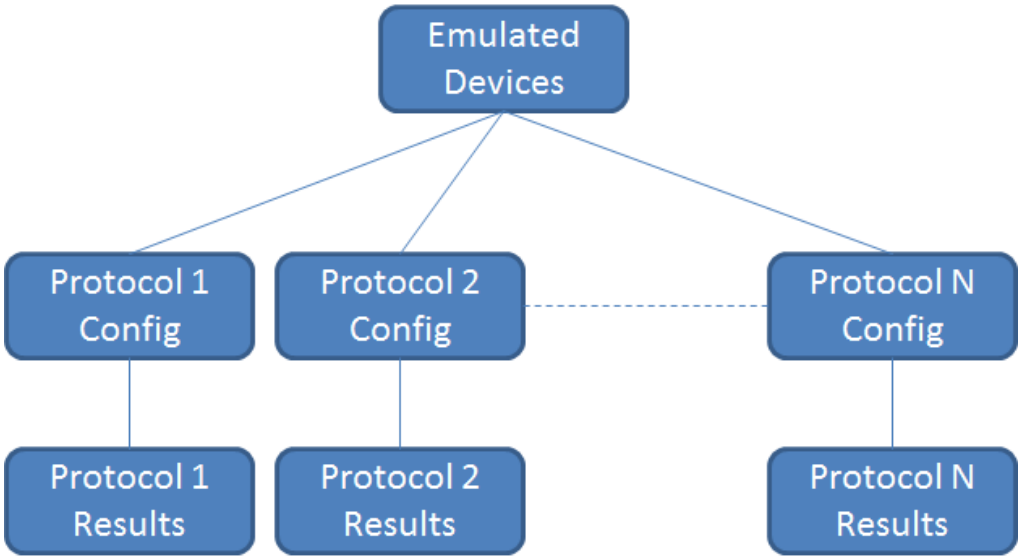

Figure 1 - NTAF Emulated Devices Data Model

5. URI

The base URI for this NTAPI is defined as:

- The NTAPI name is TS-009
- The NTAPI version is v1

- The supported objects are:
  - EmulatedDevices

The following is the URI for this specification's collection of devices:

/ntaf/ntapi/TS-009/v1/EmulatedDevices.

The following is the URI to refer to a single device:

/ntaf/ntapi/TS-009/v1/EmulatedDevices/<object handle>.

6. Object Definitions

EmulatedDevice

Attributes

The emulation_device object's attributes are described in the table below. All of attributes in the table below are readable/writable, except for "handle" which is automatically generated upon creation.

Name	Data Type/ Format	Description	Possible Values	Default Value
portHandle	**string**	**The port handle used for identification of the port.**	**Alpha-numeric**	
handle	**string**	**The device handle (instance number), used for identification.**	**Alpha-numeric**	
count	**integer**	**The number of devices.**		
encapsulation	**string**	**The type of Layer 2 encapsulation for the emulated device.**	**ethernet_ii (Ethernet II), ethernet_ii_qinq (Ethernet II with two VLAN tags), ethernet_ii_vlan (Ethernet II with a single VLAN tag)**	**ethernet_ii**
enablePingResponse	**boolean**	**Enables or disables the emulated device to respond to ping.**	**false (disable) and true (enable).**	**false**
ipVersion	**string**	**The IP version of the emulated device.**	**ipv4, ipv6 or ipv46**	**ipv4**
intfIpAddr	**IPv4**	**The IPv4 address of the**		**192.85.1.3**

	address	emulated device.		
intfIpAddrStep	IPv4 address	The difference between IPv4 interface addresses of consecutive devices when multiple emulated devices are created.		0.0.0.1
gatewayIpAddr	IPv4 address	The IPv4 gateway address for the emulated device.		
gatewayIpAddrStep	IPv4 address	The difference between IPv4 gateway addresses of consecutive devices when multiple emulated devices are created.		0.0.0.1
gatewayIpv6Addr	IPv6 address	The IPv6 gateway address for the emulated device.		
gatewayIpv6AddrStep	IPv6 address	The difference between IPv6 gateway addresses of consecutive devices when multiple emulated devices are created.		
intfPrefixLen	integer	The prefix length for the IPv4 address of the emulated device.	1 to 32	24
intfIpv6Addr	IPv6 address	The IPv6 address of the emulated device.		
intfIpv6AddrStep	IPv6 address	The difference between interface IPv6 addresses of consecutive devices when multiple emulated devices are created.		
intfIpv6PrefixLen	integer	The prefix length for the IPv6 address of the emulated device.	0 to 128	64
linkLocalIpv6Addr	IPv6 address	The starting link local IPv6 address for emulated devices.		FE80::0
linkLocalIpv6AddrStep	IPv6 address	The difference between link local IPv6 addresses of consecutive devices when multiple emulated devices		::1

		are created.		
linkLocalIpv6PrefixLen	**integer**	**The prefix length for the link local IPv6 address of the emulated device.**	**0 to 128**	**64**
macAddr	**MAC address**	**The MAC address of the emulated device.**		
macAddrStep	**MAC address**	**The difference between MAC addresses of consecutive devices when multiple emulated devices are created.**		00:00:00:00:00:01
qinqIncrMode	**string**	**Determines which VLAN ID to increment first.**	**inner (Increments the inner VLAN ID before the outer VLAN ID), outer (Increments the outer VLAN ID before the inner VLAN ID), both (Increment both the inner and outer VLAN ID at the same time)**	**inner**
routerId	**IPv4 address**	**The router ID of the emulated device.**		
routerIdIpv6	**IPv6 address**	**The IPv6 router ID of the emulated device.**		
vlanId	**integer**	**The starting VLAN ID for the ethernet_ii_vlan encapsulation or the ethernet_ii_qinq encapsulation. This attribute is available when -encapsulation is set to ethernet_ii_qinq or ethernet_ii_vlan.**	**0 to 4095**	**100**
vlanIdStep	**integer**	**The step size by which the VLAN ID is incremented.**	**0 to 4095**	**1**
vlanUserPri	**integer**	**The VLAN user priority assigned to emulated device.**	**0 to 7**	**0**
vlanOuterId	**integer**	**The starting outer VLAN ID for the QinQ encapsulation.**	**0 to 4095**	**100**

		This argument is available when -encapsulation is set to ethernet_ii_qinq.		
vlanOuterIdStep	**integer**	**The step size by which the outer VLAN ID is incremented.**	**0 to 4095**	**1**
vlanOuterTpid	**integer**	**The 16-bit Tag Protocol ID for outer VLAN tag in hex format**	**0x9100, 0x88a8, 0x8100**	**0x8100**
vlanOuterUserPri	**integer**	**The VLAN priority to assign to the outer VLAN header.**	**0 to 7**	**0**

7. Actions

The following actions are supported by the Emulated Devices NTAPI:

- Create – Used to create a new instance of an Object
- Read – Used to retrieve the details (attribute or names) of an existing Object or Objects
- Update – Used to modify an existing instance of an Object
- Delete – Used to remove an existing instance of an Object

The response for Create and Update is the details of the affected Object (including the handle which is used as the instance identifier in the URI).

There is only a status response for Delete.

8. Command: emulation_device_config

This section introduces the command emulation_device_config, which can be used to create and modify instances of emulated devices. Since Tcl is currently widely used in existing test automation scripts, the command is defined below using Tcl syntax. However, since other languages and automation paradigms are growing in popularity, and to demonstrate language-agnosticity, examples in Perl, Python, and JSON over HTTP are provided in subsequent sections.

Purpose:

Creates, modifies or deletes emulated devices.

Description:

The emulation_device_config function creates, modifies and deletes one or more emulated devices on the specified port. Use the -mode argument to specify the action to perform. (See the -mode argument description for information about the actions.)

When you create an emulated device, use the -port_handle argument to specify the port that the emulated device will use (the port handle value is contained in the keyed list returned by the connect function). The create mode returns the handle of the device.

Use this function when you want to create a device without configuring a protocol. You can enable a protocol on the created device if you need it later. You can also use the created device as the source or destination handle when you create a bound stream.

Synopsis:

```
emulation_device_config
        -mode create -port_handle <port_handle> |
        -mode {modify|delete} -handle < device_handle>
                [-count <integer>]
                [-encapsulation {ethernet_ii | ethernet_ii_vlan | ethernet_ii_qinq}]
                [-enable_ping_response {1 | 0}]
                [-ip_version {4 | 6}]
                [-intf_ip_addr <a.b.c.d>]
                [-intf_ip_addr_step <a.b.c.d>]
                [-gateway_ip_addr <a.b.c.d>]
                [-gateway_ip_addr_step <a.b.c.d>]
                [-gateway_ipv6_addr <aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh>]
                [-gateway_ipv6_addr_step <aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh> ]
                [-intf_prefix_len <1-32>]
                [-intf_ipv6_addr <aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh> ]
                [-intf_ipv6_addr_step <aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh>]
                [-intf_ipv6_prefix_len <1-128>]
                [-link_local_ipv6_addr <aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh>]
                [-link_local_ipv6_addr_step <aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh>]
                [-link_local_ipv6_prefix_len <0-128>]
                [-mac_addr <aa:bb:cc:dd:ee:ff>]
                [-mac_addr_step <aa:bb:cc:dd:ee:ff>]
                [-qinq_incr_mode {inner | outer | both}]
                [-router_id <a.b.c.d>]
                [-router_id_ipv6 <aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh>]
                [-vlan_id_step <0-4095>]
                [-vlan_user_pri <0-7>]
                [-vlan_outer_id <0-4095>]
                [-vlan_outer_id_step <0-4095>]
                [-vlan_outer_tpid {0x8100 | 0x88a8 | 0x9100}]
                [-vlan_outer_user_pri <0-7>]
```

Arguments:

```
-port_handle
        Specifies the port on which to create the emulated device. This handle is returned by the
        sth::connect function. It is mandatory for the "create" mode.

-handle
        Specifies the device handle. This argument is mandatory for -mode
        modify and delete.

-mode
        Specifies the action to perform on the test port. This argument is
        mandatory. Possible values are:
                create - Creates the device on the specified port. You must specify -port_handle.
                modify - Modifies the configured device identified by -handle.
                delete - Deletes the emulated device identified by -handle.


-count
        Specifies the number of emulated devices to be created. The default value is 1.

-encapsulation
        Specifies the type of Layer 2 encapsulation for the emulated device. Possible values are:
                ethernet_ii - Ethernet II
                ethernet_ii_vlan - Ethernet II with a single VLAN tag
                ethernet_ii_qinq - Ethernet II with two VLAN tags
        The default value is ethernet_ii.
```



```
-enable_ping_response
       Enables or disables the emulated device to respond to ping. Possible values are 0
       (disable) and 1 (enable). The default is 0.

-ip_version
       Defines the IP version of the emulated device. Possible values are ipv4, ipv6 and ipv46.
       The default value is ipv4.

-intf_ip_addr
       Specifies the IPv4 address of the emulated device. The default value is 192.85.1.3.

-intf_ip_addr_step
       Specifies the difference between IPv4 interface addresses of consecutive devices when
       multiple emulated devices are created. The value must be in IPv4 format. The default is
       0.0.0.1.

-gateway_ip_addr
       Specifies the IPv4 gateway address for the emulated device.

-gateway_ip_addr_step
       Specifies the difference between IPv4 gateway addresses of consecutive devices when
       multiple emulated devices are created.
       The default value is 0.0.0.1.
-intf_prefix_len
       Specifies the prefix length for the IPv6 address of the emulated device. Possible values
       range from 1 to 32. The default is 24.

-intf_ipv6_addr
       Specifies the IPv6 address of the emulated device.

-intf_ipv6_addr_step
       Specifies the difference between interface IPv6 addresses of consecutive devices when
       multiple emulated devices are created.

-intf_ipv6_prefix_len
       Specifies the prefix length for the IPv6 address of the emulated device. Possible values
       range from 0 to 128. The default is 64.

-gateway_ipv6_addr
       Specifies the IPv6 gateway address for the emulated device.

-gateway_ipv6_addr_step
       Specifies the difference between IPv6 gateway addresses of consecutive devices when
       multiple emulated devices are created.

-link_local_ipv6_addr
       Specifies the starting link local IPv6 address for emulated devices. The value must be in
       IPv6 format. The default is FE80::0.

-link_local_ipv6_addr_step
       Specifies the difference between link local IPv6 addresses of consecutive devices when
       multiple emulated devices are created. The value must be in IPv6 format. The default is
       ::1.

-link_local_ipv6_prefix_len
       Specifies the prefix length for the link local IPv6 address of the emulated device.
       Possible values range from 0 to 128. The default is 64.

-mac_addr
       Specifies the MAC address of the emulated device.

-mac_addr_step
       Specifies the difference between MAC addresses of consecutive devices when multiple
       emulated devices are created.

-qinq_incr_mode
       Determines which VLAN ID to increment first. Possible values are:
              inner - Increments the inner VLAN ID before the outer VLAN ID
              outer - Increments the outer VLAN ID before the inner VLAN ID
              both - Increment both the inner and outer VLAN ID at the same time
```



```
        The default value is inner.

-router_id
        Specifies the router ID of the emulated device. The value must be in IPv4 format.

-router_id_ipv6
        Specifies the IPv6 router ID of the emulated device. The value must be in IPv6 format.

-vlan_id
        Specifies the starting VLAN ID for the ethernet_ii_vlan encapsulation or the
        ethernet_ii_qinq encapsulation. Possible values range from 0 to 4095. The default value
        is 100. This argument is available when -encapsulation is set to ethernet_ii_qinq or
        ethernet_ii_vlan.

-vlan_id_step
        Specifies the step size by which the VLAN ID is incremented. Possible values range from 0
        to 4095. The default value is 1.

-vlan_user_pri
        Specifies the VLAN user priority assigned to emulated device. Possible values range from
        0 to 7. The default value is 0.

-vlan_outer_id
        Specifies the starting outer VLAN ID for the QinQ encapsulation. Possible values range
        from 0 to 4095. The default value is 100. This argument is available when -encapsulation
        is set to
        ethernet_ii_qinq.

-vlan_outer_id_step
        Specifies the step size by which the outer VLAN ID is incremented. Possible values range
        from 0 to 4095. The default
        value is 1.

-vlan_outer_tpid
        Specifies the Tag Protocol ID (TPID) for the outer VLAN header. Possible values are
        0x8100, 0x88a8 and 0x9100. The default value is 0x8100.

-vlan_outer_user_pri
        Specifies the VLAN priority to assign to the outer VLAN header. Possible values range
        from 0 to 7. The default value is 0.
```

Return Values:

Depending on the specific language used, the function returns a keyed list/dictionary/hash using the following keys (with corresponding data):

```
        status          $SUCCESS (1) or $FAILURE (0)
        log             Error message if command returns {status 0}
        handle          The device handle
```

9. Examples

The following examples create an emulated device:

Tcl

```
set device_ret1 [ntapi::emulation_device_config\
        -mode create\
        -port_handle $port1 \
        -ip_version ipv4 \
]
```

Sample Output:


```
{handle device1} {status 1}
```

Perl

```
my $device_ret1 = ntapi->ntapi::emulation_device_config (
       mode           => "config",
       port_handle    => $hport[2],
       ip_version     => "ipv4");

Sample Output (hash):
```

'status' => '1'
'handle' => 'emulateddevice1'

Python

```
device_ret1 = ntapi.emulation_device_config(mode = 'config',
       port_handle = port1[0], ip_version = 'ipv4')

Sample Output (Python dictionary):
```

{'status': '1', 'handle': 'emulateddevice1'}

JSON over HTTP

Request (Create)

```
POST /ntaf/ntapi/TS-009/v1/EmulatedDevices HTTP/1.1
Content-Length: 79
Content-Type: application/json

{
    "count": 2,
    "ipVersion": "ipv4",
    "macAddr": "aa:bb:cc:00:11:00"
}
```

Response

```
HTTP/1.1 200 OK
Content-Length: 468
Content-Type: application/json

{
    "handle": "3232",
    "count": 2,
    "encapsulation": "ethernet ii",
    "enablePingResponse": false,
    "ipVersion": "ipv4",
    "intfIpAddr": "192.85.1.3",
    "intfIpAddrStep": "0.0.0.1",
    "intfPrefixLen": 24,
    "macAddr": "aa:bb:cc:00:11:00",
    "macAddrStep": "00:00:00:00:00:01,
    "qinqIncrMode": "inner",
    "vlanId": 100,
    "vlanIdStep": 1,
    "vlanUserPri": 0,
    "vlanOuterId": 100,
    "vlanOuterIdStep": 1,
    "vlanOuterUserPri": 0
}
```


Request (Update)

```
PUT /ntaf/ntapi/TS-009/v1/EmulatedDevices/3232 HTTP/1.1
Content-Length: 429
Content-Type: application/json

{
    "count": 20,
    "encapsulation": "ethernet_ii",
    "enablePingResponse": false,
    "ipVersion": "ipv4",
    "intfIpAddr": "192.85.1.3",
    "intfIpAddrStep": "0.0.0.1",
    "intfPrefixLen": 24,
    "macAddr": "aa:bb:cc:00:11:00",
    "macAddrStep": "00:00:00:00:00:01,
    "qinqIncrMode": "inner",
    "vlanId": 100,
    "vlanIdStep": 1,
    "vlanUserPri": 0,
    "vlanOuterId": 100,
    "vlanOuterIdStep": 1,
    "vlanOuterUserPri": 0
}
```

Response

```
HTTP/1.1 200 OK
Content-Length: 469
Content-Type: application/json

{
    "handle": "3232",
    "count": 20,
    "encapsulation": "ethernet ii",
    "enablePingResponse": false,
    "ipVersion": "ipv4",
    "intfIpAddr": "192.85.1.3",
    "intfIpAddrStep": "0.0.0.1",
    "intfPrefixLen": 24,
    "macAddr": "aa:bb:cc:00:11:00",
    "macAddrStep": "00:00:00:00:00:01,
    "qinqIncrMode": "inner",
    "vlanId": 100,
    "vlanIdStep": 1,
    "vlanUserPri": 0,
    "vlanOuterId": 100,
    "vlanOuterIdStep": 1,
    "vlanOuterUserPri": 0
}
```

Request (Get)

```
GET /ntaf/ntapi/TS-009/v1/EmulatedDevices/3232 HTTP/1.1
```

Response

```
HTTP/1.1 200 OK
Content-Length: 469
Content-Type: application/json
```



```
{
    "handle": "3232",
    "count": 20,
    "encapsulation": "ethernet_ii",
    "enablePingResponse": false,
    "ipVersion": "ipv4",
    "intfIpAddr": "192.85.1.3",
    "intfIpAddrStep": "0.0.0.1",
    "intfPrefixLen": 24,
    "macAddr": "aa:bb:cc:00:11:00",
    "macAddrStep": "00:00:00:00:00:01,
    "qinqIncrMode": "inner",
    "vlanId": 100,
    "vlanIdStep": 1,
    "vlanUserPri": 0,
    "vlanOuterId": 100,
    "vlanOuterIdStep": 1,
    "vlanOuterUserPri": 0
}
```

Request (Get All)

```
GET /ntaf/ntapi/TS-009/v1/EmulatedDevices HTTP/1.1
```

Response

```
HTTP/1.1 200 OK
Content-Length: 1059
Content-Type: application/json

[
    {
        "handle": "3232",
        "count": 20,
        "encapsulation": "ethernet_ii",
        "enablePingResponse": false,
        "ipVersion": "ipv4",
        "intfIpAddr": "192.85.1.3",
        "intfIpAddrStep": "0.0.0.1",
        "intfPrefixLen": 24,
        "macAddr": "aa:bb:cc:00:11:00",
        "macAddrStep": "00:00:00:00:00:01,
        "qinqIncrMode": "inner",
        "vlanId": 100,
        "vlanIdStep": 1,
        "vlanUserPri": 0,
        "vlanOuterId": 100,
        "vlanOuterIdStep": 1,
        "vlanOuterUserPri": 0
    },
    {
        "handle": "56554534",
        "count": 1,
        "encapsulation": "ethernet_ii",
        "enablePingResponse": false,
        "ipVersion": "ipv4",
        "intfIpAddr": "192.185.1.3",
        "intfIpAddrStep": "0.0.0.1",
        "intfPrefixLen": 24,
        "macAddr": "cc:bb:cc:00:11:00",
        "macAddrStep": "00:00:00:00:00:01,
        "qinqIncrMode": "inner",
        "vlanId": 100,
        "vlanIdStep": 1,
        "vlanUserPri": 0,
        "vlanOuterId": 100,
```



```
        "vlanOuterIdStep": 1,
        "vlanOuterUserPri": 0
    }
]
```

Request (Delete)

```
DELETE /ntaf/ntapi/TS-009/v1/EmulatedDevices/3232 HTTP/1.1
```

Response

```
HTTP/1.1 200 OK
```

10. Compliance

NTAF's Automation API Framework document (TS-005) section 9, indicates that each NTAPI specification will outline its own requirements. For this specification, a compliant implementation must abide by all of the following points:

- Support for the command emulated_device_config for the use cases to create, read, update, and delete instances of EmulatedDevice objects
- Support for all default values for the attributes listed in section 6 above
- Support for at least one of the following:
    - A scripting or programming language, OR
    - A REST implementation
- In the case of REST, support JSON over HTTP encoding
- User documentation, where the documentation contains
    - List of all supported attributes
    - List of attributes which are in the specification, but not supported in the implementation
    - List of attributes which are not in the specification, but are supported in the implementation
    - Default values for all attributes
    - Examples
- If an attribute is implemented, the implementation must use the attribute using the name and meaning specified in section 6 of this document

Implementations which claim compliance must include a full report of the supported objects, attributes and associated values.

