**Broadcom Corporation Technical Publication**

**VRF**

**Test Plan**

**Arlo Release with SONIC 2.0**

Document Status: Draft

Document Issue: 00.03  
Issue Date: 02/01/2018  
Security Status: Broadcom Confidential  
Author: SQA Broadcom Corporation

© 2011 Broadcom Corporation  
All rights reserved.

UNCONTROLLED COPY: The master of this document is stored on an electronic database and is “write protected”; it may be altered only by authorized persons. While copies may be printed, it is not recommended. Viewing of the master electronically ensures access to the current issue. Any hardcopies taken must be regarded as uncontrolled copies.

**CONFIDENTIAL:** The information contained in this document is the property of Broadcom Corporation Except as expressly authorized in writing by Broadcom Corporation. The holder shall keep all information contained herein confidential, shall not disclose to anyone outside Broadcom Corporation, including customers and partners, regardless of any non-disclosure agreement (NDA) that maybe in place, without prior written consent from the Quality Assurance Manager or Author. In the absence of written consent, the holder shall only disclose the information herein to authorize Broadcom Corporation employees with a need to know basis, and shall protect the information from disclosure and dissemination to third parties. Except as expressly authorized in writing by Broadcom Corporation, the holder is granted no rights to use the information contained herein.

**Broadcom Corporation**   
{enter site address of Author}

1030 Swabia Court

Suite 400

Durham, NC 27703

Broadcom®, the pulse logo, Connecting everything®, and the Connecting everything logo are among the trademarks of Broadcom Corporation and/or its affiliates in the United States, certain other countries and/or the EU. Any other trademarks or trade names mentioned are the property of their respective owners.



Proprietary and Confidential Information

# Test Strategy Plan Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Change Description** |
| 0.01 | 04/05/2019 | Sonic Dev Test | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# List of Approvers

This section identifies the participants required to approve this document. Enter function title, name and once approved at the meeting or by email trail, update the date field with the date approved.

|  |  |  |
| --- | --- | --- |
| **Function** | **Name** | **Date Approved** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# List of Reviewers

Section identifies optional reviewers.

|  |  |
| --- | --- |
| **Function** | **Name** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Abbreviations and Glossary

This section describes any words that the reader may not be familiar with or abbreviations used that the reader may not be familiar with. Do not include definitions or abbreviations that should be generally known by your audience.

**AT** Acceptance Testing

**CLI** Command Line Interface

**DOR** Dead Office Recovery

**FCT** Feature Confidence Test

**FT** Functional Testing

**GUI** Graphical User Interface

**IEEE** Institute of Electrical and Electronics Engineers

**IT** Integration test or Component Integration testing (CIT)

**IP**  Internet Protocol (implies version 4)

**IPv4** Internet Protocol version 4

**IPv6**  Internet Protocol version 6

**MIB**  Management Information Base

**NMS** Network Management System

**QA** Quality Assurance

**RFT** Regression Functional Test

**RIT** Release Integration Testing

**RoR** Release over Release

**SCT** System Confidence Test

**ST** System Test

**TA** Test Automation

**TAM** Test Automation Manager

**TFP** Test First Pass

**UI** User Interface

**UT** Unit Testing

**VLAN** Virtual Local Area Network

Table of Contents

[Test Strategy Plan Revision History 2](#_3znysh7)

[List of Approvers 2](#_2et92p0)

[List of Reviewers 2](#_tyjcwt)

[Abbreviations and Glossary 3](#_3dy6vkm)

[1.](#_1t3h5sf) DOCUMENT REVISION HISTORY 5

[2.](#_4d34og8) TEST PLAN REVIEW HISTORY 5

[3.](#_17dp8vu) INTRODUCTION 5

[3.1.](#_3rdcrjn) OBJECTIVE 5

[3.2.](#_26in1rg) LIST OF REQUIREMENT IDS 5

[4.](#_lnxbz9) FEATURE DESCRIPTIONS 5

[4.1.](#_35nkun2) LEGACY SOFTWARE SUPPORT 5

[4.2.](#_1ksv4uv) NEW SOFTWARE FUNCTIONALITY SUPPORTED 5

[4.3.](#_44sinio) OTHER FEATURES ENABLED IN THIS TEST PLAN 5

[4.4.](#_2jxsxqh) QUALIFIED BROADCOM PLATFORMS 5

[4.5.](#_z337ya) UPGRADE-DOWNGRADE SUPPORT DETAILS 5

[5.](#_3j2qqm3) Test Environment 6

[5.1.](#_1y810tw) OTHER VENDOR EQUIPMENT 6

[5.2.](#_4i7ojhp) MONITORING 6

[5.3.](#_2xcytpi) NETWORK TEST TOPOLOGY 6

[6.](#_3whwml4) Detailed Description 6

[6.1.](#_2bn6wsx) VRF Tests 7

[6.1.1.](#_3as4poj) CLI test cases 7

[6.1.2.](#_1pxezwc) Functional test cases 13

[6.1.3.](#_147n2zr) Reload cases 37

[6.1.4.](#_3o7alnk) Scaling cases 38

[6.1.5.](#_23ckvvd) Negative cases 41

[1.](#_32hioqz) Reference and Appendix 42

# DOCUMENT REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Name** | **Comments** |
| 01/05/2019 | 0.1 | Sonic Devtest | Initial Version |

# TEST PLAN REVIEW HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Name** | **Comments** |
| 10/05/2019 | 0.1 | Sonic Devtest | Internal Review |
| 14/05/2019 | 0.1 | Sonic Devtest | External Review |

# INTRODUCTION

## OBJECTIVE

The main objective of this document is to cover the test cases that will be executed for VRF feature in Arlo release using Sonic 2.0. Topologies and test cases for testing the feature will be discussed as part of this document.

## LIST OF REQUIREMENT IDS

VRF test cases will be covered in this test plan. Will be updated after receiving FS

# FEATURE DESCRIPTIONS

## LEGACY SOFTWARE SUPPORT

VRF is not supported in SONIC 1.0

## NEW SOFTWARE FUNCTIONALITY SUPPORTED

VRF is supported from the Arlo release in Sonic 2.0

## OTHER FEATURES ENABLED IN THIS TEST PLAN

Various L2/L3 features will be enabled while testing the feature.

## QUALIFIED BROADCOM PLATFORMS

|  |  |
| --- | --- |
| **Platform** | **Features under Test enabled** |
| AS7786/TH2 | VRF |
| AS7712/TH1 | VRF |
| vSONIC | VRF |

## UPGRADE-DOWNGRADE SUPPORT DETAILS

# Test Environment

## OTHER VENDOR EQUIPMENT

The following vendor equipment will be used as part of various tests in this test design:

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Vendor** | **Model** | **Note** |
| STC Traffic generator | SPIRENT |  |  |
| IXIA Traffic generator | IXIA |  |  |

## MONITORING

The following monitoring will be conducted in the background:

* SNMP MIB walks in the background: Need to check if supported
* SNMP MIB get and polling for statistics: Need to check if supported
* REST: Need to check if supported

## NETWORK TEST TOPOLOGY

The topology depict the typical connections that will be used during test execution.

Topology 1: Most of the test case planned to use this topology

















# Detailed Description

This section covers all actual test cases. Please dedicate Section 4.x for each sub-feature and under each 4.x please have all following sections. If any sub-section is not relevant to the area, please still leave it there and mark content as “N/A” (Example: Some area won’t have RFC Conformance).

Each Sub-feature will have standard following sub-sections.

1. Functional Tests – Objective is to validate Sub-features in a recommended use-case based topology.
2. Scalability – This section validates Uni-dimensional and Multi-dimensional Scale testing.
3. Performance (RFC Conformance and Other metric collection for Prod mgmt)
4. Config Management (CLI)
5. Monitoring

Almost all these test cases executed with the following feature enabled unless otherwise mentioned

* L3 protocol config like BGP
* IPv4 and IPv6 config on L3 interfaces

All types of interfaces supported and 4X10G Breakout interfaces are used as edge port either directly connected to Traffic Generator or one Layer 2 Switche. And most test cases are executed on dual stack interfaces

Almost all these Test cases will have these standard triggers unless otherwise mentioned.

* Docker restart
* Uplink and Downlink interface flaps

## VRF Tests

All following functional tests will be executed on above topologies with following configurations enabled. Please note that even basic feature verification may be done with other variables in place such IP traffic, background monitoring etc as listed below.

**Sample configuration:**

### CLI test cases

#### Verify address family IPv4 and IPv6 in VRF instance

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli001 |
| **Title / Test Name** | Verify address family IPv4 and IPv6 in VRF instance |
| **Description** | Verify address family IPv4 and IPv6 in VRF instance |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Verify that address family IPv4 unicast vrf can be configured for BGP * Configure a non-default vrf with RD * Configure the address family IPv4 unicast under the vrf * Verify that the address family can be configured successfully. * Repeat the steps for IPv6 address family | |

#### Bind/unbind/rebind VRF to a physical interface

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli002 |
| **Title / Test Name** | Bind/unbind/rebind VRF to a physical interface |
| **Description** | Bind/unbind/rebind VRF to a physical interface |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a non-default vrf with RD * Configure the address family IPv4 unicast under the vrf * Map a physical interface to the vrf and add ip address * Verify show run int for the interface which is mapped to the vrf * Verify vrf forwarding is applied. * Unbind the physical interface, remove the address and verify vrf forwarding is removed. * Rebind the interface and repeat the steps. * Delete the vrf and reverify the steps * Repeat the steps for IPv6 address. | |

#### Bind/unbind/rebind VRF to a virtual interface

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli003 |
| **Title / Test Name** | Bind/unbind/rebind VRF to a virtual interface |
| **Description** | Bind/unbind/rebind VRF to a virtual interface |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a non-default vrf with RD * Configure the address family IPv4 unicast under the vrf * Map a virtual interface to the vrf and add ip address * Verify show run int for the interface which is mapped to the vrf * Verify vrf forwarding is applied. * Unbind the virtual interface, remove the address and verify vrf forwarding is removed. * Rebind the interface and repeat the steps. * Delete the vrf and reverify the steps * Repeat the steps for IPv6 address. | |

#### Bind/unbind/rebind VRF to a port-channel

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli004 |
| **Title / Test Name** | Bind/unbind/rebind VRF to a port-channel |
| **Description** | Bind/unbind/rebind VRF to a port-channel |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a non-default vrf with RD * Configure a port channel * Configure the address family IPv4 unicast under the vrf * Map a virtual interface to the vrf and add ip address * Verify show run int for the interface which is mapped to the vrf * Verify vrf forwarding is applied. * Unbind the port channel, remove the address and verify vrf forwarding is removed. * Rebind the port channel and repeat the steps. * Delete the vrf and reverify the steps * Repeat the steps for IPv6 address. | |

#### Add/Remove route targets from VRF

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli005 |
| **Title / Test Name** | Add/remove route targets from VRF |
| **Description** | Add/remove route targets from VRF |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few vrfs and enable vrf forwarding on few interfaces. Configure a RD for each vrf. * Add route-target under IPv4 address-family for the vrf. * Verify that it shows up in the configuration. * Configure the options present in route-target: both, import, export * Verify that the configuration is accepted and the commands show up in the configuration. * Delete the route-targets. * Verify that the configuration is removed from the running config. * Repeat the steps for IPv6 address family | |

#### Configure overlapping IP addresses belonging to different VRFs

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli006 |
| **Title / Test Name** | Configure overlapping IP address belonging to different VRFs |
| **Description** | Configure overlapping IP address belonging to different VRFs |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure two non-default vrfs, each with a RD defined. * To each vrf, map an interface. * Assign an overlapping IPv4 address to both the interfaces. * The CLI should accept the IPv6 address without complaining about the duplicate address * Repeat the step for IPv6 address | |

#### Configure multiple interfaces to a VRF and configure same interface to multiple VRFs

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli007 |
| **Title / Test Name** | Configure multiple interfaces to a VRF and configure same interface to multiple VRFs |
| **Description** | Configure multiple interfaces to a VRF and configure same interface to multiple VRFs |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a non-default vrf with a RD. * Map multiple interfaces (a mix of physical, virtual, lag) to the vrf. * Verify that all the interfaces are mapped to the vrf and the following message appear. * Add same interface to mutliple VRF * Verify it is not supported * Repeat the steps with IPv6 address configured | |

#### Dynamically change port membership from one non default VRF to another

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli008 |
| **Title / Test Name** | Dynamically change port membership from one non default vrf to another |
| **Description** | Dynamically change port membership from one non default vrf to another |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure two non-default vrfs, each with a RD defined. * Map two different interfaces to each vrf. * Map one of the interfaces to a different vrf. * Verify that the interface get mapped to the new vrf without any error. * Create a vrf with single physical port untagged in the vlan. * Send traffic between CE learnt routes in the vrf and verify there is no loss | |

#### Dynamically modify port channel interfaces in a vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli009 |
| **Title / Test Name** | Dynamically modify port channel interfaces in a vrf |
| **Description** | Dynamically modify port channel interfaces in a vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a non-default vrf with RD * Configure a port channel * Configure the address family IPv4 unicast under the vrf * Map a virtual interface to the vrf and add ip address * Verify show run int for the interface which is mapped to the vrf * Verify vrf forwarding is applied. * Unbind the port channel, remove the address and verify vrf forwarding is removed. * Modify the ports in the port channel and rebind it to the vrf. * Repeat the steps for IPv6 address. | |

#### Configure multiple loopback interfaces per VRF

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfCli010 |
| **Title / Test Name** | Configure multiple loopback interfaces per VRF |
| **Description** | Configure multiple loopback interfaces per VRF |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure three vrfs and assign interfaces. * Configure multiple loopback addresses in each vrf. * Verify if the lowest loopback address is chosen for router-id. * Remove the chosen loop back interface verify if the id moves to the next available address. * Configure a lower ip address than the currently chosen one, Verify if the current one is still maintained unless it is deleted or clear ip router-id is issued. * After the last available loopback address chosen as router-id, delete it. * Verify if the router-id selection moves to one of the regular interface ip address. * Change the router-id in one vrf. * Verify it shouldn't recalculate the ids in others including all the protocols. * With loopback interfaces configured, the router-id should be selected as the ip address of the lowest numbered loopback interface. | |

### Functional test cases

#### Add/delete static route under vrf with next hop as physical interface

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun011 |
| **Title / Test Name** | Add/delete static route under vrf with next hop as a physical interface |
| **Description** | Add/delete static route under vrf with next hop as a physical interface |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a vrf and enable vrf forwarding on few interfaces. * Configure a static IPv4 route under the vrf with next hop as a physical port. * Verify that the route is configured without any error. * Verify that the route is displayed in the output of “show ip route vrf <vrf-name”. * Verify that the traffic destined for the destination network specified in static route is forwarded to the corresponding physical interface specified in the static route command. * Repeat the steps for IPv6 address | |

#### Add/delete static route under vrf with next hop as virtual interface

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun012 |
| **Title / Test Name** | Add/delete static route under vrf with next hop as a virtual interface |
| **Description** | Add/delete static route under vrf with next hop as a virtual interface |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a vrf and enable vrf forwarding on few interfaces. * Configure a static IPv4 route under the vrf with next hop as a virtual interface. * Verify that the route is configured without any error. * Verify that the route is displayed in the output of “show ip route vrf <vrf-name”. * Verify that the traffic destined for the destination network specified in static route is forwarded to the corresponding virtual interface specified in the static route command. * Repeat the steps for IPv6 address | |

#### Add/delete static route under vrf with next hop as port channel

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun013 |
| **Title / Test Name** | Add/delete static route under vrf with next hop as a port channel |
| **Description** | Add/delete static route under vrf with next hop as a port channel |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a vrf and enable vrf forwarding on few interfaces. * Configure a port channel between DUT1 and DUT2 * Configure a static IPv4 route under the vrf with next hop as a port channel * Verify that the route is configured without any error. * Verify that the route is displayed in the output of “show ip route vrf <vrf-name”. * Verify that the traffic destined for the destination network specified in static route is forwarded to the corresponding port channel specified in the static route command. * Repeat the steps for IPv6 address | |

#### Verify IPv4 Ping/Traceroute between end points on the same vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun014 |
| **Title / Test Name** | IPv4 Ping/Traceroute between end points on the same vrf |
| **Description** | Ping/Traceroute between end points on the same vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few non-default vrfs and enable vrf forwarding on the interface. * Configure a RD for each vrf. * Perform an IPv4 Ping between the end points belonging to the same vrf. * Verify that the ping goes through if a route exists between the end points * Repeat the steps for traceroute | |

#### Verify IPv4 Ping/Traceroute from non-default vrf to default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun015 |
| **Title / Test Name** | Ping/Traceroute from non-default vrf to default vrf |
| **Description** | Ping/Traceroute from non-default vrf to default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few non-default vrfs and enable vrf forwarding on the interface. * Configure a RD for each vrf. * Perform an IPv4 Ping between a non-default vrf to default vrf * Verify that the ping goes through if a route exists between the end points * Repeat the steps for traceroute | |

#### Verify IPv4 Ping/Traceroute from default vrf to non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun016 |
| **Title / Test Name** | Ping/Traceroute from default vrf to non-default vrf |
| **Description** | Ping/Traceroute from default vrf to non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few non-default vrfs and enable vrf forwarding on the interface. * Configure a RD for each vrf. * Perform an IPv4 Ping between a default vrf to non-default vrf * Verify that the ping goes through if a route exists between the end points * Repeat the steps for traceroute | |

#### Verify IPv4 Ping/Traceroute from non-default vrf to non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun017 |
| **Title / Test Name** | Ping/Traceroute from non-default vrf to non-default vrf |
| **Description** | Ping/Traceroute from non-default vrf to non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few non-default vrfs and enable vrf forwarding on the interface. * Configure a RD for each vrf. * Perform an IPv4 Ping between a non-default vrf to non-default vrf * Verify that the ping goes through if a route exists between the end points * Repeat the steps for traceroute | |

#### IPv4 static route leak from default vrf to non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun018 |
| **Title / Test Name** | IPv4 static route leak from default vrf to non-default vrf |
| **Description** | IPv4 static route leak from default vrf to non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a static IPv4 route under the default vrf. * Verify that the route is configured without any error. * Configure a vrf, assign rd and map to an interface * Verify traffic. * Verify route leak from a default vrf to non-default vrf * Verify 'show ip route' displays the route with the right suffix for protocol through which the route was learnt | |

#### IPv4 static route leak from non-default vrf to default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun019 |
| **Title / Test Name** | IPv4 static route leak from non-default vrf to default vrf |
| **Description** | IPv4 static route leak from non-default vrf to default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a static IPv4 route under the default vrf. * Verify that the route is configured without any error. * Configure a vrf, assign rd and map to an interface * Verify traffic * Verify route leak from a non-default vrf to default vrf * verify 'show ip route' displays the route with the right suffix for protocol through which the route was learnt | |

#### IPv4 static route leak from non-default vrf to another non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun020 |
| **Title / Test Name** | IPv4 static route leak from non-default vrf to another non-default vrf |
| **Description** | IPv4 static route leak from non-default vrf to another non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure multiple vrf, assign rds and map them to different interfaces * Configure a static IPv4 route under one of the non-default vrf. * Verify that the route is configured without any error. * Leak the static route from that non-default vrf to another non-default vrf * Verify traffic * verify 'show ip route' displays the route with the right suffix for protocol through which the route was learnt | |

#### Verify IPv6 Ping/Traceroute between end points on the same vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun021 |
| **Title / Test Name** | IPv6 Ping/Traceroute between end points on the same vrf |
| **Description** | Ping/Traceroute between end points on the same vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few non-default vrfs and enable vrf forwarding on the interface. * Configure a RD for each vrf. * Perform an IPv6 Ping between the end points belonging to the same vrf. * Verify that the ping goes through if a route exists between the end points * Repeat the steps for traceroute | |

#### Verify IPv6 Ping/Traceroute from non-default vrf to default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun022 |
| **Title / Test Name** | Ping/Traceroute from non-default vrf to default vrf |
| **Description** | Ping/Traceroute from non-default vrf to default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few non-default vrfs and enable vrf forwarding on the interface. * Configure a RD for each vrf. * Perform an IPv6 Ping between a non-default vrf to default vrf * Verify that the ping goes through if a route exists between the end points * Repeat the steps for traceroute | |

#### Verify IPv6 Ping/Traceroute from default vrf to non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun023 |
| **Title / Test Name** | Ping/Traceroute from default vrf to non-default vrf |
| **Description** | Ping/Traceroute from default vrf to non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few non-default vrfs and enable vrf forwarding on the interface. * Configure a RD for each vrf. * Perform an IPv6 Ping between a default vrf to non-default vrf * Verify that the ping goes through if a route exists between the end points * Repeat the steps for traceroute | |

#### Verify IPv6 Ping/Traceroute from default vrf to non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun024 |
| **Title / Test Name** | IPv6 Ping/Traceroute from default vrf to non-default vrf |
| **Description** | IPv6 Ping/Traceroute from default vrf to non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure few non-default vrfs and enable vrf forwarding on the interface. * Configure a RD for each vrf. * Perform an IPv6 Ping between a non-default vrf to non-default vrf * Verify that the ping goes through if a route exists between the end points * Repeat the steps for traceroute | |

#### IPv6 static route leak from default vrf to non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun025 |
| **Title / Test Name** | IPv6 static route leak from default vrf to non-default vrf |
| **Description** | IPv6 static route leak from default vrf to non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a static IPv4 route under the default vrf. * Verify that the route is configured without any error. * Configure a vrf, assign rd and map to an interface * Leak the static route from default vrf to non-default vrf * Verify traffic * verify 'show ip route' displays the route with the right suffix for protocol through which the route was learnt | |

#### IPv6 static route leak from non-default vrf to default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun026 |
| **Title / Test Name** | IPv6 static route leak from non-default vrf to default vrf |
| **Description** | IPv6 static route leak from non-default vrf to default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure a static IPv4 route under the default vrf. * Verify that the route is configured without any error. * Configure a vrf, assign rd and map to an interface * Leak the static route from non-default vrf to default vrf * Verify traffic * verify 'show ip route' displays the route with the right suffix for protocol through which the route was learnt | |

#### IPv6 static route leak from non-default vrf to another non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun027 |
| **Title / Test Name** | IPv4 static route leak from non-default vrf to another non-default vrf |
| **Description** | IPv4 static route leak from non-default vrf to another non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure multiple vrf, assign rds and map them to different interfaces * Configure a static IPv4 route under one of the non-default vrf. * Verify that the route is configured without any error. * Leak the static route from that non-default vrf to another non-default vrf * Verify traffic * verify 'show ip route' displays the route with the right suffix for protocol through which the route was learnt | |

#### Import same route from VRF A to VRF B, C and D

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun028 |
| **Title / Test Name** | Import same route from VRF A to VRF B, C, and D |
| **Description** | Import same route from VRF A to VRF B, C, and D |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Modify nexthop for that route in the source vrf (VRF A) and verify the VRFs B,C and D. * Delete the route in VRF A and verify route is removed from routing tables of the VRFs B,C and D. Also verify no crash * Delete VRF A and verify no crash and verify route is removed from routing tables of the other VRFs. * Clear routes in VRF B, and verify no impact on the source VRF, VRF A. No impact on VRFs C and D. * Clear route in VRF A and verify the other VRFs * Verify the above steps for IPv6 routes as well | |

#### Verify IBGP neighbor for BGPv4 in vrf for ipv4 address family

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun029 |
| **Title / Test Name** | Verify IBGP neighbor for BGPv4 in vrf for ipv4 address family |
| **Description** | Verify IBGP neighbor for BGPv4 in vrf for ipv4 address family |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. | |

#### Verify EBGP neighbor for BGPv4 in vrf for ipv4 address family

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun030 |
| **Title / Test Name** | Verify EBGP neighbor for BGPv4 in vrf for ipv4 address family |
| **Description** | Verify EBGP neighbor for BGPv4 in vrf for ipv4 address family |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. | |

#### Verify the EBGP peer connection and route advertisement under IPV4 address-family in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun031 |
| **Title / Test Name** | Verify EBGP peer connection and route advertisement under IPv4 address family in non-default vrf |
| **Description** | Verify EBGP peer connection and route advertisement under IPv4 address family in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor relationship between two DUTs and have multiple route advertisements. * Verify the IBGP neighbor relationships come up and the advertised routes are seen on the neighbor * Send bidirectional IPv4 traffic for these BGP learned routes and verify traffic * Clear off some of the neighbor relationship between the peers, verify that once they come back the traffic resumes properly. | |

#### Verify multihop EBGP under IPv4 address-family in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun032 |
| **Title / Test Name** | Verify multihop EBGP neighbor for BGPv4 in vrf for ipv4 address family |
| **Description** | Verify multihop EBGP neighbor for BGPv4 in vrf for ipv4 address family |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure multihop EBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. | |

#### Add/delete non default vrf with both single hop and multihop EBGP sessions

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun033 |
| **Title / Test Name** | Add/delete non-default vrf with both single hop and multi hop EBGP sessions |
| **Description** | Add/delete non-default vrf with both single hop and multi hop EBGP sessions |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure single hop and multihop EBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. | |

#### Redistribute connected IPv4 routes into IBGP in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun034 |
| **Title / Test Name** | Redistribute connected IPv4 routes into IBGP in non-default vrf |
| **Description** | Redistribute connected IPv4 routes into IBGP in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Redistribute directly connected routes in IBGP and verify | |

#### Verify BFD singlehop over IBGP session for IPV4 address in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun035 |
| **Title / Test Name** | Redistribute connected IPv4 routes into IBGP in non-default vrf |
| **Description** | Redistribute connected IPv4 routes into IBGP in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure IBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Configure BFD with minimum timers and verify * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Modify the BFD timers and reverify the neighbors | |

#### Verify BFD single hop over EBGP session for IPV4 address in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun036 |
| **Title / Test Name** | Verify BFD single hop over EBGP session in non-default vrf |
| **Description** | Verify BFD single hop over EBGP session in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Configure BFD with minimum timers and verify * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Modify the BFD timers and reverify the neighbors | |

#### Verify BFD multihop hop over IBGP session for IPv4 address in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun037 |
| **Title / Test Name** | Verify BFD multi hop over EBGP session in non-default vrf |
| **Description** | Verify BFD multi hop over EBGP session in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Configure BFD with minimum timers and verify * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Modify the BFD timers and reverify the neighbors | |

#### Import IPv4 Static routes from one vrf to another and redistribute into IBGP for IPv4 address

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun038 |
| **Title / Test Name** | Import IPv4 Static routes from one vrf to another and redistribute into IBGP for IPv4 address |
| **Description** | Import IPv4 Static routes from one vrf to another and redistribute into IBGP for IPv4 address |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Inport static routes from another non-default vrf and redistribute them into BGP * Delete router BGP and readd | |

#### Import IPv4 Static routes from one vrf to another and redistribute into EBGP for IPv4 address

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun039 |
| **Title / Test Name** | Import IPv4 Static routes from one vrf to another and redistribute into EBGP for IPv4 address |
| **Description** | Import IPv4 Static routes from one vrf to another and redistribute into EBGP for IPv4 address |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Import static routes from another non-default vrf and redistribute them into BGP * Delete router BGP and readd | |

#### Clear IPv4 routes for IBGP on non-default vrf and verify relearning

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun040 |
| **Title / Test Name** | Clear IPv4 routes for IBGP on non-default vrf and verify relearning |
| **Description** | Clear IPv4 routes for IBGP on non-default vrf and verify relearning |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Clear ip routes/bgp routes and revierfy the steps | |

#### IPv4 ECMP in non-default vrf along with route leak into another vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun041 |
| **Title / Test Name** | IPv4 ECMP in non-default vrf along with route leak into another vrf |
| **Description** | IPv4 ECMP in non-default vrf along with route leak into another vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Configure ECMP within the VRF and verify the routes * Leak this route into another VRF and verify ECMP there as well | |

#### Verify BGP peer-group for IPv4 address family with EBGP neighbors non-default VRF

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun042 |
| **Title / Test Name** | Verify BGP peer-group with EBGP neighbors non-default VRF |
| **Description** | Verify BGP peer-group with EBGP neighbors non-default VRF |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * DUT1 is connected to DUT2 and DUT3 in the same AS (i.e. IBGP neighbors). Configure a peer-group on DUT1 and apply this peer group for both DUT2 and DUT3 neighbor statements. * Verify that all the IBGP neighbor relationships come up with peer group configuration. * Send traffic through the selected routes and verify that the traffic flows without any packet loss. | |

#### Verify BGPv4 route-map functionality in non-default VRF

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun043 |
| **Title / Test Name** | Verify BGPv4 route-map functionality in non-default VRF |
| **Description** | Verify BGPv4 route-map functionality in non-default VRF |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor relationship between the DUTs and inject IPv4 routes from the neighbors. * Match the IPv4 routes with the available matches * Set the matched routes with some actions * Routes not matching the match criteria should be dropped * Delete the import route map statement and verify * Run the traffic | |

#### Verify IBGP neighbor for BGPv6 in vrf for ipv6 address family

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun044 |
| **Title / Test Name** | Verify IBGP neighbor for BGPv6 in vrf for ipv6 address family |
| **Description** | Verify IBGP neighbor for BGPv6 in vrf for ipv6 address family |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv6 addresses under ipv6 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. | |

#### Verify EBGP neighbor for BGPv6 in vrf for ipv6 address family

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun045 |
| **Title / Test Name** | Verify EBGP neighbor for BGPv4 in vrf for ipv6 address family |
| **Description** | Verify EBGP neighbor for BGPv4 in vrf for ipv6 address family |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor commands using ipv6 addresses under ipv6 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. | |

#### Verify the EBGP peer connection and route advertisement under IPV6 address-family in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun046 |
| **Title / Test Name** | Verify EBGP peer connection and route advertisement under IPv6 address family in non-default vrf |
| **Description** | Verify EBGP peer connection and route advertisement under IPv6 address family in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor relationship between two DUTs and have multiple route advertisements. * Verify the IBGP neighbor relationships come up and the advertised routes are seen on the neighbor * Send bidirectional IPv6 traffic for these BGP learned routes and verify traffic * Clear off some of the neighbor relationship between the peers, verify that once they come back the traffic resumes properly. | |

#### Verify multihop EBGP under IPv6 address-family in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun047 |
| **Title / Test Name** | Verify multihop EBGP neighbor for ipv6 address family |
| **Description** | Verify multihop EBGP neighbor for ipv6 address family |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure multihop EBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. | |

#### Add/delete non default vrf with both single hop and multihop EBGP sessions for IPv6 address family

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun048 |
| **Title / Test Name** | Add/delete non-default vrf with both single hop and multi hop EBGP sessions for IPv6 address family |
| **Description** | Add/delete non-default vrf with both single hop and multi hop EBGP sessions for IPv6 address family |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure single hop and multihop EBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. | |

#### Redistribute connected IPv6 routes into IBGP in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun049 |
| **Title / Test Name** | Redistribute connected IPv6 routes into IBGP in non-default vrf |
| **Description** | Redistribute connected IPv6 routes into IBGP in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv6 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv6 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Redistribute directly connected routes in IBGP and verify | |

#### Verify BFD single hop over IBGP session for IPv6 address in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun050 |
| **Title / Test Name** | Verify BFD single hop over IBGP session for IPv6 address in non-default vrf |
| **Description** | Verify BFD single hop over IBGP session for IPv6 address in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure IBGP neighbor commands using ipv6 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv6 bgp vrf <vrfname>. * Configure BFD with minimum timers and verify * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Modify the BFD timers and reverify the neighbors | |

#### Verify BFD single hop over EBGP session for IPV6 address in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun051 |
| **Title / Test Name** | Verify BFD single hop over EBGP session in non-default vrf for IPv6 |
| **Description** | Verify BFD single hop over EBGP session in non-default vrf for IPv6 |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Configure BFD with minimum timers and verify * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Modify the BFD timers and reverify the neighbors | |

#### Verify BFD multihop hop over IBGP session for IPv6 address in non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun052 |
| **Title / Test Name** | Verify BFD multi hop over EBGP session in non-default vrf |
| **Description** | Verify BFD multi hop over EBGP session in non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor commands using ipv6 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv6 bgp vrf <vrfname>. * Configure BFD with minimum timers and verify * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Modify the BFD timers and reverify the neighbors | |

#### Import IPv6 Static routes from one vrf to another and redistribute into IBGP for IPv6 address

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun053 |
| **Title / Test Name** | Import IPv4 Static routes from one vrf to another and redistribute into IBGP for IPv6 address |
| **Description** | Import IPv4 Static routes from one vrf to another and redistribute into IBGP for IPv6 address |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv6 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Import static routes from another non-default vrf and redistribute them into BGP * Delete router BGP and readd | |

#### Import IPv6 Static routes from one vrf to another and redistribute into EBGP for IPv6 address

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun054 |
| **Title / Test Name** | Import IPv6 Static routes from one vrf to another and redistribute into EBGP for IPv4 address |
| **Description** | Import IPv6 Static routes from one vrf to another and redistribute into EBGP for IPv4 address |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Inport static routes from another non-default vrf and redistribute them into BGP * Delete router BGP and readd | |

#### Clear IPv6 routes for IBGP on non-default vrf and verify relearning

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun055 |
| **Title / Test Name** | Clear IPv6 routes for IBGP on non-default vrf and verify relearning |
| **Description** | Clear IPv6 routes for IBGP on non-default vrf and verify relearning |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv4 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Add and Remove some of the network statements under the address family on the peer BGP router and verify that those removed are not displayed any more. * Clear ip routes/bgp routes and revierfy the steps | |

#### Verify BGP peer-group for IPv6 address family with EBGP neighbors non-default VRF

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun056 |
| **Title / Test Name** | Verify BGP peer-group with EBGP neighbors non-default VRF |
| **Description** | Verify BGP peer-group with EBGP neighbors non-default VRF |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * DUT1 is connected to DUT2 and DUT3 in the same AS (i.e. IBGP neighbors). Configure a peer-group on DUT1 and apply this peer group for both DUT2 and DUT3 neighbor statements. * Verify that all the IBGP neighbor relationships come up with peer group configuration. * Send traffic through the selected routes and verify that the traffic flows without any packet loss. | |

#### Verify BGP4+ route-map functionality in non-default VRF

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun057 |
| **Title / Test Name** | Verify BGP4+ route-map functionality in non-default VRF |
| **Description** | Verify BGP4+ route-map functionality in non-default VRF |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure EBGP neighbor relationship between the DUTs and inject IPv6 routes from the neighbors. * Match the IPv6 routes with the available matches * Set the matched routes with some actions * Routes not matching the match criteria should be dropped * Delete the import route map statement and verify * Run the traffic | |

#### IPv6 ECMP in non-default vrf along with route leak into another vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfFun058 |
| **Title / Test Name** | IPv6 ECMP in non-default vrf along with route leak into another vrf |
| **Description** | IPv6 ECMP in non-default vrf along with route leak into another vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure BGP neighbor commands using ipv4 addresses under ipv6 address family unicast vrf. * Activate the neighbor through the neighbor x:x:x::x activate command. * After the BGP neighbor relationship comes up, verify through the show ipv4 bgp vrf <vrfname>. * Configure ECMP within the VRF and verify the routes * Leak this route into another VRF and verify ECMP there as well | |

### Reload cases

#### Verify non-default vrf after warm reboot

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfRel059 |
| **Title / Test Name** | Verify non-default vrf after a warm reboot |
| **Description** | Verify non-default vrf after a warm reboot |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure both IPv4 and IPv6 address families under non default vrf. Send traffic ans perform a warm reboot. | |

#### Verify non-default vrf after cold reboot

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfRel060 |
| **Title / Test Name** | Verify non-default vrf after a cold reboot |
| **Description** | Verify non-default vrf after a cold reboot |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Configure both IPv4 and IPv6 address families under non default vrf. Send traffic and perform a cold reboot. | |

### Scaling cases

#### Create maximum allowed VRFs

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfScl061 |
| **Title / Test Name** | Verify address family IPv4 and IPv6 in VRF instance |
| **Description** | Verify address family IPv4 and IPv6 in VRF instance |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Verify that address family IPv4 unicast vrf can be configured for BGP * Configure a non-default vrf with RD * Configure the address family IPv4 unicast under the vrf * Verify that the address family can be configured successfully. * Repeat the steps for IPv6 address family | |

#### Configure maximum VRFs, delete individual vrfs and in bulk

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfScl062 |
| **Title / Test Name** | Create maximum allowed VRFs, delete individual vrfs and in bulk |
| **Description** | Create maximum allowed VRFs, delete individual vrfs and in bulk |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Verify that address family IPv4 unicast vrf can be configured for BGP * Configure a non-default vrf with RD * Configure the address family IPv4 unicast under the vrf * Verify that the address family can be configured successfully. * Repeat the steps for IPv6 address family. * Delete a single vrf and verify the time taken. * Delete all the vrfs and verify for crash | |

#### Pump maximum IPv4 routes for IBGP session into non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfScl063 |
| **Title / Test Name** | Pump maximum IPv4 routes for IBGP session into non-default vrf |
| **Description** | Pump maximum IPv4 routes for IBGP session into non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Pump maximum allowed IPv4 routes in a single non-default vrf for IBGP session, send traffic and verify | |

#### Pump maximum IPv6 routes for EBGP session into non-default vrf

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfScl064 |
| **Title / Test Name** | Pump maximum IPv6 routes for EBGP session into non-default vrf |
| **Description** | Pump maximum IPv6 routes for EBGP session into non-default vrf |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Pump maximum allowed IPv6 routes in a single non-default vrf for EBGP session, send traffic and verify | |

#### Pump maximum IPv4 and IPv6 routes for IBGP session into scaled non-default vrfs

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfScl065 |
| **Title / Test Name** | Pump maximum IPv4 and IPv6 routes for IBGP session into scaled non-default vrfs |
| **Description** | Pump maximum IPv4 and IPv6 routes for IBGP session into scaled non-default vrfs |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Pump maximum allowed IPv4 and IPv6 routes in scaled non-default vrfs, send traffic and verify | |

#### Save and reload with maximum IPv4 and IPv6 routes for IBGP session into scaled non-default vrfs

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfScl066 |
| **Title / Test Name** | Save and reload with maximum IPv4 and IPv6 routes for IBGP session into scaled non-default vrfs |
| **Description** | Save and reload maximum IPv4 and IPv6 routes for IBGP session into scaled non-default vrfs |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Pump maximum allowed IPv4 and IPv6 routes in scaled non-default vrfs, send traffic and verify. * Save the config, reload the dut and reveirfy | |

### Negative cases

#### Same RD for different VRFs

|  |  |
| --- | --- |
| **Test Case Details** | |
| **Test case ID** | FtRtVrfNeg067 |
| **Title / Test Name** | Same RD for different VRFs |
| **Description** | Same RD for different VRFs |
| **Test Setup** | Topology 1 |
| **Automation Status** | Yes |
| **Pass/Fail Criteria** | Below verification should pass |
| **Test Procedure:**   * Same RD for different VRF interface, routes shouldn't be imported due to conflict | |

# Reference and Appendix

**To be updated with FS link**