# ROUTING CONSORTIUM

Border Gateway Protocol 4 (BGP)
Operations Test Suite

**Technical Document**

Revision 3.7


**University of New Hampshire**
**InterOperability Laboratory**
**Routing Consortium**
**http://www.iol.unh.edu**

**121 Technology Drive, Suite 2**
**Durham, NH 03824-3525**
**Phone: +1-603-862-3941**
**Fax: +1-603-862-4181**

# MODIFICATION RECORD

Version 1.0 Complete — April 20, 2000
Initial Release

Version 2.0 Complete — June 26, 2001

Version 2.1 Complete — January 23, 2002

Version 2.2 Complete — March 22, 2002
- Revised test procedure for test 1.18.d
- Modified Section 2

Version 2.3 Complete — April 1, 2002
- Added a section to test 1.6 (Address Prefix Limit)
- Revised observable results for test 1.15.c
- Added sections (f, g, h) to test 1.18
- Modified Section 2
- Added test 2.7 for IdleHold state.

Version 2.4 Complete — July 16, 2002
- Modified test 1.18 f
- Modified test 1.13 a
- Modified test 2.2 c
- Modified test 2.5 f
- Modified test 4.6 c
- Removed test 4.11 and 4.12
- Modified test 3.2 d, changed the BGP identifier to 0.0.0.0 instead of 255.0.0.5

Version 2.5 Complete — February 3, 2003
- Updated to draft-ietf-idr-bgp4-22
- Updated References and Discussions
- Added test 1.4 c
- Added tests 1.12 f and g
- Modified tests 1.15 b and c
- Updated results for test 1.16 a
- Added test 1.16 e, re-lettered other tests in 1.16
- Modified test 1.16 g
- Added test 1.16 h, changed the old 1.16.h to 1.16 i
- Added test 1.18 I
- Added test 2.1 b
- Added tests 2.2 a-c, e and i, re-lettered other tests in 2.2
- Added test 2.3 a, b and d, re-lettered other tests in 2.3
- Removed the old test 2.4 f and added a new one
- Updated results for tests in 2.4, 2.5 and 2.6 which involved state IdleHold
- Added test 2.5 c, re-lettered other tests in 2.5
- Removed section 2.7
- Removed test 3.2 f, g, and h

- Divided section 3.3 into 6 different test cases
- Added test 3.3 b, re-lettered other tests in 3.3
- Added test 3.7 b, 3.8 b, 3.9
- Renumbered Test 3.4 and 3.5 to 3.10 and 3.11 respectively
- Added test 4.4e

Version 2.6 Complete — October 30, 2003
- Revised test 2.2

Version 2.7 Complete — February 20, 2004
- Revised test 2.3

Version 2.8 Complete — June 7, 2004
- Removed test 3.9a
- Revised test 4.11

Version 3.0 Complete — September 21, 2004
- Test suite put in new format
- Updated to RFC 4271
- Updated Discussions to reflect new draft
- Made formatting/editorial changes
- Made editorial changes to tests 1.16, and 4.11
- Removed 1.16h and 3.3c
- Modified tests 1.15, and 3.1h
- Updated Test Results for 2.3d

Version 3.1 Complete — November 16, 2004
- Updated new test number scheme
- Updated Test Organization section

Version 3.2 Complete — December 9, 2004
- Added test 2.1c
- Added test 2.2 f, i, and j
- Split 2.3a into a and b
- Added 2.3 g-i
- Updated discussions in tests 2.2 and 2.3

Version 3.3 Complete — January 3, 2005
- Removed tests 2.2 i and 2.3 h, i
- Re-lettered test 2.2 j to 2.2 i and rolled back to the previous version of the test

Version 3.4 Complete — August 18, 2006
- Removed test 4.9

Version 3.5 Complete — June 4, 2007
- Renumbered Test 1.18g

| | |
|---|---|
| Version 3.6 Complete | July 30, 2007<br>- Fixed Typo on Test 1.5a.<br>- Reworded expected results on Test 3.8b.<br>- Reworded Test 3.10a.<br>- Reworded Tests 1.18f-g. |
| Version 3.7 Complete | February 27, 2009<br>- Clarified expected results on Test 1.5.C |

-

## ACKNOWLEDGMENTS

**The University of New Hampshire would like to acknowledge the efforts of the following individuals in the development of this test suite.**

| | |
|---|---|
| Eric Barrett | University of New Hampshire |
| David Bond | University of New Hampshire |
| Ethan Burns | University of New Hampshire |
| Michael Cleary | University of New Hampshire |
| Henry He | University of New Hampshire |
| Kimo Johnson | University of New Hampshire |
| Ray LaRocca | University of New Hampshire |
| Dr. William Lenharth | University of New Hampshire |
| Dan C. Maftei | University of New Hampshire |
| Catherine Rhoades | University of New Hampshire |
| Sagun Shakya | University of New Hampshire |
| Ying Shi | University of New Hampshire |
| Fanny Xu | University of New Hampshire |
| Mateusz Pusz | Intel Corporation |

# INTRODUCTION

**Overview**

The University of New Hampshire's InterOperability Laboratory (IOL) is an institution designed to improve the interoperability of standards based products by providing an environment where a product can be tested against other implementations of a standard. This suite of tests has been developed to help implementers evaluate the functioning of their Border Gateway Protocol 4 (from now on referred to as BGP) products. The tests do not determine if a product conforms to the BGP specification, nor are they purely interoperability tests. Rather, they provide one method to isolate problems within a device. Successful completion of all tests contained in this suite does not guarantee that the tested device will operate with other BGP devices. However, combined with satisfactory operation in the IOL's semi-production environment, these tests provide a reasonable level of confidence that the Router Under Test will function well in most multi-vendor BGP environments.

**Test Software and Descriptions**

The UNH IOL Testing Software is not a full BGP implementation; it is simply a packet generator that can transmit and receive packets. This allows the Testing Router to generate invalid packets. The Testing Software is not currently available to the public. The UNH IOL test descriptions outlined here are made available to members of the IPv4 Consortium.

# TEST ORGANIZATION

The tests contained in this document are organized to simplify the identification of information related to a test and to facilitate in the actual testing process. Each test contains an identification section that describes the test and provides cross-reference information. A detailed section discusses the background information and specifies how the test is to be performed. Each test contains the following information:

**Test Label:** The test label and title comprise the first line of the test block. The test label is composed by concatenating the short test suite name, the group number, and the test number within the group, separated by periods. The **Test Number** is the group and test number, also separated by a period. So, test label BGP_CONF.1.2 refers to the second test of the first test group in the BGP Conformance suite. The test number is 1.2.

**Purpose:** The Purpose is a short statement describing what the test attempts to achieve. It is usually phrased as a simple assertion of the feature or capability to be tested.

**References:** The References section lists cross-references to the specifications and documentation that might be helpful in understanding and evaluating the test and results.

**Last Modification:** The Last Modification is a record of the date when the test has been modified.

**Discussion:** The Discussion is a general discussion of the test and relevant section of the specification, including any assumptions made in the design or implementation of the test as well as known limitations.

**Test Setup:** The Test Setup section describes the configuration of all devices prior to the start of the test. Different parts of the procedure may involve configuration steps that deviate from what is given in the test setup. If a value is not provided for a protocol parameter, then the protocol's default is used for that parameter.

**Procedure:** This section of the test description contains the step-by-step instructions for carrying out the test. These steps include such things as enabling interfaces, unplugging devices from the network, or transmitting packet from a test station. The test procedure also cues the tester to make observations, which are interpreted in accordance with the observable results given for that test part.

**Observable Results:** This section lists observable results that can be examined by the tester to verify that the RUT is operating properly. When multiple observable results are possible, this section provides a short discussion on how to interpret them. The determination of a pass or fail for each test is usually based on how the RUT's behavior compares to the results described in this section.

**Possible Problems:** This section contains a description of known issues with the test procedure, which may affect test results in certain situations.

# DEFINITIONS

**Overview**
This section defines the conventions used in this test suite. In this document "BGP" refers to "BGP version 4".

**Acronyms**
Common acronyms are defined in this section.

**RUT**: **R**outer **U**nder **T**est
**TR**: **T**esting **R**outer
**N**: **N**etwork
**AS**: **A**utonomous **S**ystem

**Drawing conventions**

External BGP connection: 

Internal BGP connection: 

# REFERENCES

The following documents are referenced in this text:


| | |
|---|---|
| [CAPNEG] | "Capabilities Advertisement with BGP-4", Request for Comments 2842 |
| [RFC 4271] | "A Border Gateway Protocol 4 (BGP-4)", Request for Comments 4271 |
| [MD5] | "Protection of BGP Sessions via the TCP MD5 Signature Option", Request for Comments 2385 |
| [RFC 3107] | "Carrying Label Information in BGP-4", Request for Comments 3107 |
| [RFC 2858] | "Multiprotocol Extensions for BGP-4", Request for Comments 2858 |
| [RFC793] | "Transport Connection Protocol", Request for Comments 793 |
| [RFC904] | "Exterior Gateway Protocol", Request for Comments 904 |
| [RFC1997] | "BGP Communities Attribute", Request for Comments 1997 |
| [RFC2796] | "BGP Route Reflection An alternative to full mesh IBGP", Request for Comments 2796 |
| [RFC3065] | "Autonomous System Confederations for BGP", Request for Comments 3065 |
| [RFDAMPING] | "BGP Route Flap Damping", Request for Comments 2439 |

# TABLE OF CONTENTS

# 1. BASIC PROCESSING

**Overview**

The following tests are designed to verify the basic functionality of a BGP router.

**Discussion**

BGP employs four types of messages: OPEN, KEEPALIVE, UPDATE and NOTIFICATION.
OPEN messages are used to bring up a peering relationship between BGP routers. The Hold Time and the Optional Parameters are negotiated during the OPEN message exchange.
KEEPALIVE messages are used to maintain the BGP connection.
UPDATE messages are used to exchange routing information. A feasible route is described by a set of destinations (the NLRI) and a set of path attributes for the route. An unfeasible route is described only by the set of destinations. An UPDATE message may contain a feasible route OR many unfeasible routes OR a feasible route and many unfeasible routes.
NOTIFICATION messages are used when errors are detected. Upon receipt of a NOTIFICATION message, a BGP connection is closed. A NOTIFICATION message with error code Cease is used to gracefully close a BGP connection. Section 3 is dedicated to error handling.

Conceptually, a BGP router maintains three Routing Information Bases (RIBs): Adj-RIBs-In, Loc-RIB and Adj-RIBs-Out. Routes that are received from other BGP routers are present in the Adj-RIBs-In. There is one Adj-RIB-In per peer. The routes in the Adj-RIBs-In are kept for the duration of the BGP connection; therefore, periodic refreshment of routes is not necessary. Routes that will be used by the local BGP router must be present in the Loc-RIB, and the next hop for each of these routes must be present in the local BGP router's forwarding information base. Routes that will be advertised to other BGP routers are present in the Adj-RIB-Out. There is one Adj-RIB-Out per peer.

**Test BGP_CONF.1.1: Direct Connection Basic**

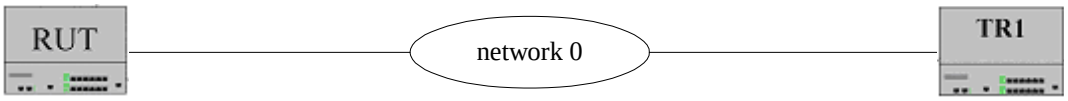
**Purpose:** To verify that a BGP router establishes a connection to a directly connected peer on TCP port 179.

**References:** [RFC 4271] – Sections 1 and 3

**Last Modification:** February 3, 2003

**Discussion:** BGP uses TCP [RFC793] as its transport protocol and listens on TCP port 179. Connections between BGP speakers of different ASs are referred to as "external" links. BGP connections between BGP speakers within the same AS are referred to as "internal" links. Similarly, a peer in a different AS is referred to as an external peer, while a peer in the same AS may be described as an internal peer. Internal BGP and external BGP are commonly abbreviated IBGP and EBGP.

**Test Setup:**




**Procedure:**

*Part A: Direct Connection of Internal Peers*

1. The RUT and TR1 are configured as internal peers, with a direct connection over network N0. Set the monitor to capture TCP packets on N0.
2. BGP is started on the routers.
3. Check the captured packets.

*Part B: Direct Connection of External Peers*

4. The RUT and TR1 are configured as external peers. Set the monitor to capture TCP packets on N0.
5. BGP is started on the routers.
6. Check the captured packets.

**Observable Results:**

- In Parts A and B, the TCP packets should show that the two BGP routers made a connection on TCP port 179.

**Possible Problems:** None.

## Test BGP_CONF.1.2: Indirect Connection

**Purpose:** To verify that a BGP router establishes a connection to an indirectly connected peer on TCP port 179.

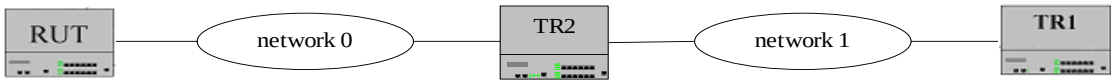
**References:** [RFC 4271] – Sections 1 and 3

**Last Modification:** February 3, 2003

**Discussion:** BGP uses TCP [RFC793] as its transport protocol and listens on TCP port 179. Connections between BGP speakers of different ASs are referred to as "external" links. BGP connections between BGP speakers within the same AS are referred to as "internal" links. Similarly, a peer in a different AS is referred to as an external peer, while a peer in the same AS may be described as an internal peer. Internal BGP and external BGP are commonly abbreviated IBGP and EBGP.

Two BGP routers can communicate remotely if they are configured to view each other as peers, and if they are in the same AS. A remote external connection can also be allowed through configuration.

**Test Setup:**




**Procedure:**

*Part A: Internal links across different networks*

1. The RUT and TR1 are configured as internal peers. TR2 is not running BGP. Set the monitor to capture TCP packets on networks N0 and N1.
2. Configure the RUT to have a route to N1 with TR2's IP Address on N0 as the next hop. TR1 has a route to N0 with TR2's IP Address on N1 as the next hop.
3. BGP is started on the routers.
4. Check the captured packets.

*Part B: External links across different networks*

5. The RUT and TR1 are configured as external peers. TR2 is not running BGP. Set the monitor to capture TCP packets on networks N0 and N1.
6. BGP is started on the routers.
7. Check the captured packets.

**Observable Results:**

- In Parts A and B, the TCP packets should show that the two BGP routers made a connection on TCP port 179.

**Possible Problems:** None.

## Test BGP_CONF.1.3: Routing Table Exchange

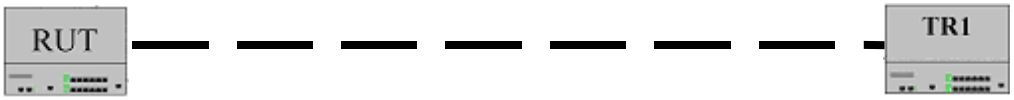
**Purpose:** To verify that a BGP router communicates its entire routing table to another BGP router after a BGP connection is established.

**References:** [RFC 4271] – Section 3

**Last Modification:** February 3, 2003

**Discussion:** Two systems form a TCP connection between one another. They exchange messages to open and confirm the connection parameters. The initial data flow is the portion of the BGP routing table that is allowed by the export policy, called the Adj-Ribs-Out. Incremental updates are sent as the routing tables change. BGP does not require periodic refresh of the entire BGP routing table.

**Test Setup:**




**Procedure:**

*Part A: Update Advertisement*

1. The RUT and TR1 are configured as external peers. Configure the RUT to advertise an address range A.
2. BGP is started on the routers.
3. Check the captured packets.

**Observable Results:**

- In Part A, once the connection is established, the RUT should send UPDATE messages to TR1 for all the routes to be advertised.

**Possible Problems:** None.

**Test BGP_CONF.1.4: Hold Time Negotiation**

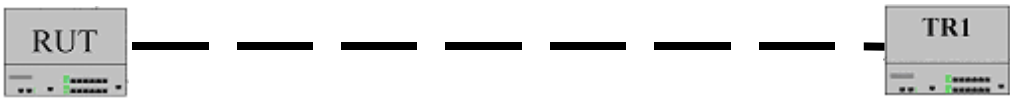
**Purpose:** To verify that a BGP router properly negotiates the Hold Time.

**References:** [RFC 4271] – Section 4.2

**Last Modification:** August 21, 2001

**Discussion:** This 2-octet unsigned integer indicates the number of seconds that the sender proposes for the value of the Hold Timer. Upon receipt of an OPEN message, a BGP speaker MUST calculate the value of the Hold Timer by using the smaller of its configured Hold Time and the Hold Time received in the OPEN message. The Hold Time MUST be either zero or at least three seconds. An implementation may reject connections on the basis of the Hold Time. The calculated value indicates the maximum number of seconds that may elapse between the receipt of successive KEEPALIVE, and/or UPDATE messages by the sender.

**Test Setup:**




**Procedure:**

*Part A: Router has Lower Hold Time*

1. Set the Hold Time of the RUT at 3 seconds. TR1's Hold Time is 7 seconds.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. Check the captured packets.

*Part B: Router has Higher Hold Time*

4. Set the Hold Time of the RUT at 7 seconds. TR1's Hold Time is 3 seconds.
5. BGP is started on the RUT and TR1. Wait until they establish the connection.
6. Check the captured packets.

*Part C: Router has an invalid Hold Time*

7. Set the Hold Time of the RUT at 2 seconds.

**Observable Results:**

- In Parts A and B, the RUT should have the actual Hold Time set at 3 seconds.
- In Part C, the RUT should not allow the Hold Timer to be set to the invalid value.

**Possible Problems:** None.

**Test BGP_CONF.1.5: Keepalive Timer**

**Purpose:** To verify that a BGP router sends KEEPALIVE messages every Keep Alive Timer interval.

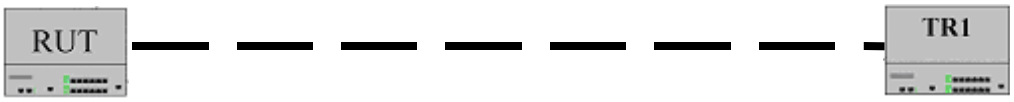
**References:** [RFC 4271] – Section 4.4

**Last Modification:** February 3, 2003

**Discussion:** BGP does not use any TCP-based keep-alive mechanism to determine if peers are reachable. Instead, KEEPALIVE messages are exchanged between peers often enough as not to cause the Hold Timer to expire. A reasonable maximum time between KEEPALIVE messages would be one third of the Hold Time interval. KEEPALIVE messages MUST NOT be sent more frequently than one per second. An implementation MAY adjust the rate at which it sends KEEPALIVE messages as a function of the Hold Time interval.

If the negotiated Hold Time interval is zero, then periodic KEEPALIVE messages MUST NOT be sent.

**Test Setup:**




**Procedure:**

*Part A: Keepalive Timer to be 1 second*

1. Set the Hold Time to 3 seconds and Keep Alive Timer to 1 second the RUT. TR1's Hold Time and Keep Alive Timer are also set to 3 seconds and 1 second respectively.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. Check the packets.

*Part B: Hold Time configured to be 0 seconds*

4. Set both Hold Time and Keep Alive Timer to 0 on the RUT.
5. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
6. Check the packets.

*Part C: Hold Time negotiated to be 0 seconds*

7. Set the Hold Time to 3 and the Keep Alive Timer to 1 on the RUT. TR1's Hold Time and Keep Alive Timer are set to 0
8. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
9. Check the packets.

**Observable Results:**

- In Part A, after the routers finish sending UPDATE messages to exchange routing table information, the RUT should send KEEPALIVE messages every second.
- In Part B, after the routers finish sending UPDATE messages to exchange routing table information, the RUT should NOT send KEEPALIVE messages. The connection should remain established.
- In Part C, after the routers finish sending UPDATE messages to exchange routing table information, the RUT should NOT send periodic KEEPALIVE messages but the RUT should still send the initial KEEPALIVE message acknowledging TR1s OPEN message. The connection should remain established.

**Possible Problems:** None.

## Test BGP_CONF.1.6: Cease NOTIFICATION Message

**Purpose:** To verify that a BGP router closes a BGP connection by sending a NOTIFICATION message with the special error code Cease.

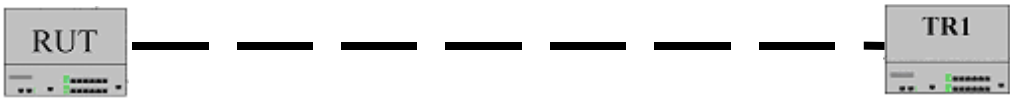
**References:** [RFC 4271] – Section 6.7

**Last Modification:** February 3, 2003

**Discussion:** In absence of any fatal errors (that are indicated in section 6 of RFC 4271), a BGP peer may choose at any given time to close its BGP connection by sending the NOTIFICATION message with Error Code Cease. However, the Cease NOTIFICATION message must not be used when a fatal error exists.

A BGP speaker MAY support the ability to impose an (locally configured) upper bound on the number of address prefixes the speaker is willing to accept from a neighbor. When the upper bound is reached, the speaker (under control of local configuration) may either (a) discard new address prefixes from the neighbor, or (b) terminate the BGP peering with the neighbor. If the BGP speaker decides to terminate its peering with a neighbor because the number of address prefixes received from the neighbor exceeds the locally configured upper bound, then the speaker must send to the neighbor a NOTIFICATION message with the Error Code Cease. The speaker MAY also log this locally.

**Test Setup:**




**Procedure:**

*Part A: BGP Connection Closing*

1. The RUT and TR1 are configured as external peers.
2. BGP is started on the routers. Wait until they establish the connection.
3. Close the BGP connection on the RUT. Check the packets.

*Part B: Address Prefix Limit*

4. If allowed via configuration, configure the RUT with a number of address prefixes it is willing to accept from TR1.
5. BGP is restarted on the routers. Wait until they establish the connection.
6. TR1 advertises arbitrary address prefixes until the number of address prefixes advertised surpasses the limit configured on the RUT. Check the packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code Cease.
- In Part B, when the number of address prefixes from TR1 surpasses the limit configured on the RUT, the RUT should either:
    1. Discard new address prefixes from TR1, OR
    2. Send a NOTIFICATION message with Error Code Cease and close the connection.

**Possible Problems:** None.

## Test BGP_CONF.1.7: Internal Update

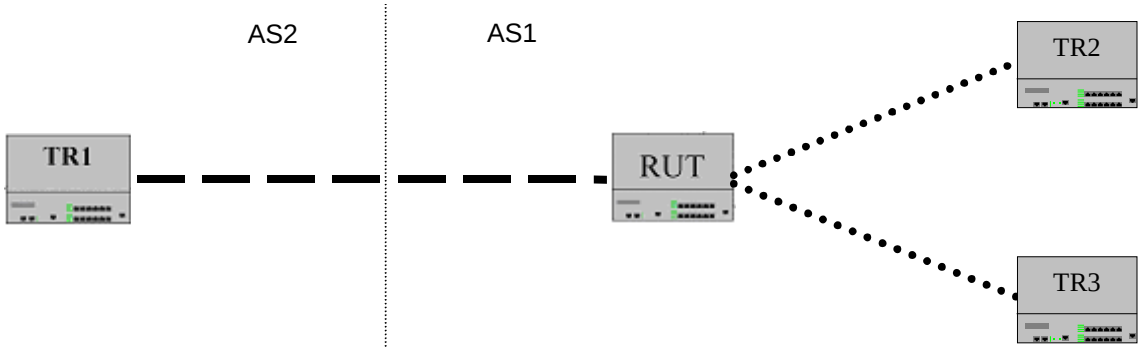
**Purpose:** To verify that a BGP router propagates internal updates only to external peers.

**References:** [RFC 4271] – Section 9.2

**Last Modification:** February 3, 2003

**Discussion:** When a BGP speaker receives an UPDATE message from an internal peer, the receiving BGP speaker shall not re-distribute the routing information contained in that UPDATE message to other internal peers, unless the speaker acts as a BGP Route Reflector [RFC2796].

**Test Setup:**




**Procedure:**

*Part A: Internal Update Advertisement*

1. The RUT and TR1 and the RUT and TR3 are configured as external peers. The RUT and TR2 are configured as internal peers.
2. BGP is started on the routers. Wait until they establish the connections.
3. TR3 sends an UPDATE message for some network N3 (the RUT, TR1 and TR2 do not have routes to N3). Check the packets.

**Observable Results:**

- In Part A, when the RUT receives the UPDATE message, it should modify its Loc-RIB, but should NOT send an UPDATE to TR2 since they are in the same AS. However, the RUT should send an UPDATE to TR1, which is in a different AS.

**Possible Problems:** None.

## Test BGP_CONF.1.8: External Update

**Purpose:** To verify that a BGP router propagates external updates to both internal and external peers.

**References:** [RFC 4271] – Sections 9.2

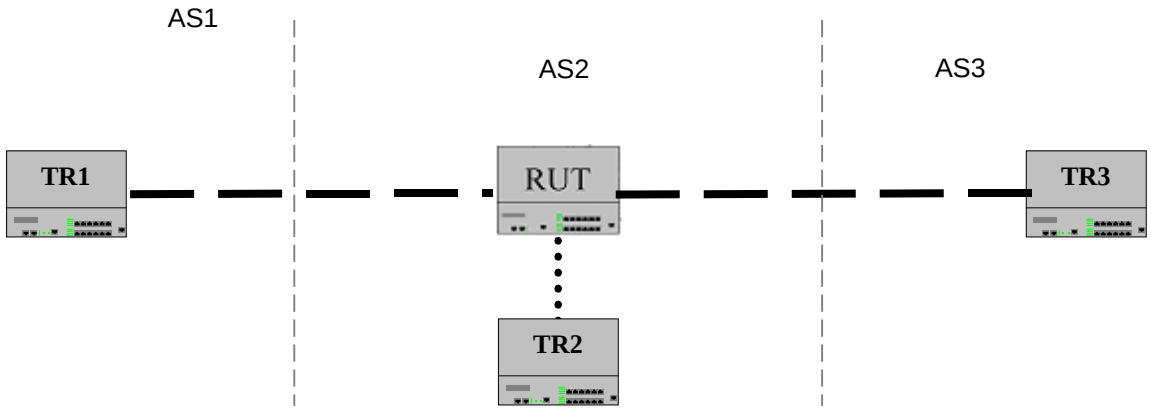
**Last Modification:** February 11, 2002

**Discussion:** The Update-Send process is responsible for advertising UPDATE messages to all peers. For example, it distributes the routes chosen by the Decision Process to other BGP speakers, which may be located in either the same autonomous system or a neighboring autonomous system.

As part of Phase 3 route selection process, the BGP speaker has updated its Adj-RIBs-Out and its Forwarding Table. All newly installed routes and all newly unfeasible routes for which there is no replacement route shall be advertised to external peers by means of UPDATE message.

Any routes in the Loc-RIB marked as unfeasible shall be removed. Changes to the reachable destinations within its own autonomous system shall also be advertised in an UPDATE message.

**Test Setup:**




**Procedure:**

*Part A: External Update Advertisement*

1. The RUT and TR1 and the RUT and TR3 are configured as external peer. The RUT and TR2 are configured as internal peers.
2. BGP is started on the routers. Wait until they establish the connections.
3. TR3 sends an UPDATE message for some network N4 (the RUT, TR1 and TR2 do not have routes to N4). Check the packets.

**Observable Results:**

- In Part A, when the RUT receives the UPDATE message, it should modify its Loc-RIB, and send UPDATE messages to TR1 and TR2, advertising the route to N3.

**Possible Problems:** None.

## Test BGP_CONF.1.9: Attribute Order

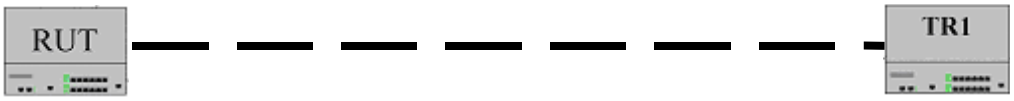
**Purpose:** To verify that a BGP router properly handles path attributes that are out of order.

**References:** [RFC 4271] – Section 5

**Last Modification:** August 21, 2001

**Discussion:** The sender of an UPDATE message should order path attributes within the UPDATE message in ascending order of attribute type. The receiver of an UPDATE message must be prepared to handle path attributes within the UPDATE message that are out of order.

**Test Setup:**




**Procedure:**

*Part A: Out of Order Attribute handling*

1. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
2. BGP is started on the routers. Wait until they synchronize.
3. TR1 sends an UPDATE for a new feasible route. The path attributes within the UPDATE are out of order (AS_PATH is first, ORIGIN is second).
4. Check the RUT's routing table.

**Observable Results:**

- In Part A, the RUT should have accepted the UPDATE, and should have the new route in its routing table.

**Possible Problems:** None.

**Test BGP_CONF.1.10: ORIGIN Attribute**

**Purpose:** To verify that a BGP router properly generates the ORIGIN attribute.

**References:** [RFC 4271] – Sections 4.3, 5.1.1

**Last Modification:** February 3, 2003

**Discussion:** ORIGIN is a well-known mandatory attribute. The ORIGIN attribute is generated by the speaker that originates the associated routing information. It shall be included in the UPDATE messages of all BGP speakers that choose to propagate this information to other BGP speakers.

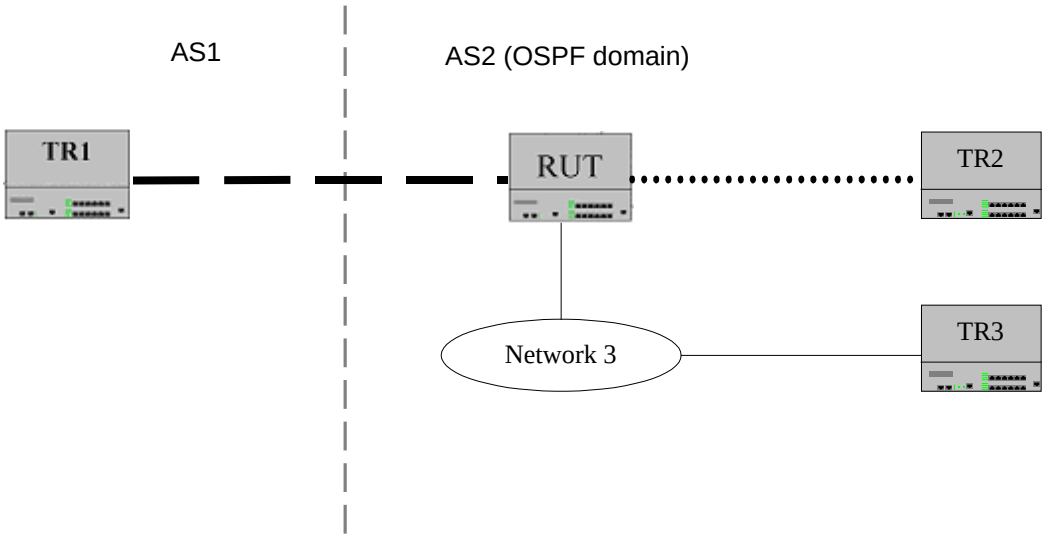
ORIGIN is a well-known mandatory attribute that defines the origin of the path information. The data octet can assume the following values:

0 IGP - Network Layer Reachability Information is interior to the originating AS

1 EGP - Network Layer Reachability Information learned via EGP protocol [RFC904]

2 INCOMPLETE - Network Layer Reachability Information learned by some other means

**Test Setup:**




**Procedure:**

*Part A: ORIGIN set to IGP*

1. The RUT and TR1 are configured as external peers. The RUT and TR2 are configured as BGP internal peers. The RUT and TR3 are running OSPF in AS2.
2. Configure the RUT to propagate OSPF learned routes to BGP.
3. BGP is started on the routers. Wait until they establish the connections.
4. TR3 advertises a route to some network N4 into OSPF (the RUT, TR1 and TR2 do not have routes to N3). Check the packets.

*Part B: ORIGIN set to EGP*

5. OSPF is disabled on the RUT and TR3 (the RUT should send an UPDATE with N4 listed as unfeasible). The RUT and TR3 are configured as EGP peers.
6. BGP is restarted on the routers. Wait until they establish the connections.
7. TR3 advertises a route to some network N4 into EGP.

*Part C: ORIGIN set to INCOMPLETE*

8. TR3's route to N4 is removed (the RUT should send an UPDATE with N4 listed as unfeasible).
9. Configure the RUT to advertise static routes.
10. BGP is restarted on the routers. Wait until they establish the connections.
11. Give the RUT a route to some network N4 (TR1 and TR2 do not have routes to N4). Check the packets.

**Observable Results:**

- In Part A, the RUT should send UPDATE messages to both TR1 and TR2. The UPDATE messages should contain the ORIGIN attribute with value 0 (IGP).
- In Part B, the RUT should send UPDATE messages to both TR1 and TR2. The UPDATE messages should contain the ORIGIN attribute with value 1 (EGP).
- In Part C, the RUT should send UPDATE messages to both TR1 and TR2. The UPDATE messages should contain the ORIGIN attribute with value 2 (INCOMPLETE).

**Possible Problems:** In Part B, the RUT may not support EGP.

## Test BGP_CONF.1.11: AS_PATH Attribute

**Purpose:** To verify that a BGP router properly handles the AS_PATH attribute.

**References:** [RFC 4271] – Sections 4.3, 5.1.2

**Last Modification:** August 21, 2001

**Discussion:** This attribute identifies the autonomous systems through which routing information carried in this UPDATE message has passed. The components of this list can be AS_SETs or AS_SEQUENCEs.

When a BGP speaker propagates a route, which it has learned from another BGP speaker's UPDATE message, it modifies the route's AS_PATH attribute based on the location of the BGP speaker to which the route will be sent:

a) When a given BGP speaker advertises the route to an internal peer, the advertising speaker shall not modify the AS_PATH attribute associated with the route.

b) When a given BGP speaker advertises the route to an external peer, then the advertising speaker update the AS_PATH attribute as follows:

1) if the first path segment of the AS_PATH is of type AS_SEQUENCE, the local system prepends its own AS number as the last element of the sequence (put it in the leftmost position with respect to the position of octets in the protocol message).

2) if the first path segment of the AS_PATH is of type AS_SET, the local system prepends a new path segment of type AS_SEQUENCE to the AS_PATH, including its own AS number in that segment.

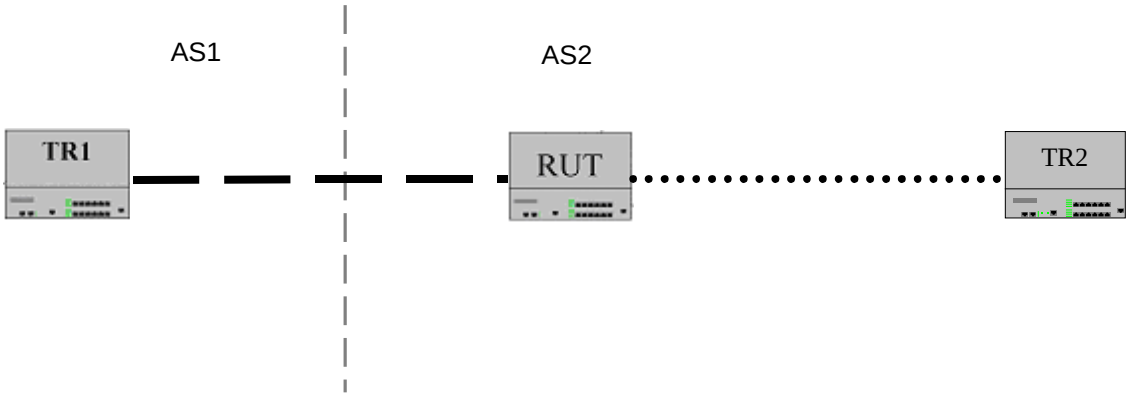
3) if the AS_PATH is empty, the local system creates a path segment of type AS_SEQUENCE, places its own AS into that segment, and places that segment into the AS_PATH.

When a BGP speaker originates a route then:

a) the originating speaker includes its own AS number in a path segment of type AS_SEQUENCE in the AS_PATH attribute of all UPDATE messages sent to an external peer. (In this case, the AS number of the originating speaker's autonomous system will be the only entry the path segment, and this path segment will be the only segment in the AS_PATH attribute).

b) the originating speaker includes an empty AS_PATH attribute in all UPDATE messages sent to internal peers. (An empty AS_PATH) attribute is one whose length field contains the value zero).

**Test Setup:**




**Procedure:**

*Part A: UPDATE to Internal Peer*

1. The RUT and TR2 are configured as internal peers.
2. BGP is started on the routers. Wait until they establish the connection.
3. Give the RUT a static route to some network N3 (TR2 does not have routes to N3). Check the packets.

*Part B: UPDATE to External Peer*

4. The RUT and TR1 are configured as external peers.
5. BGP is started on the routers. Wait until they establish the connection.
6. Give the RUT a static route to some network N3 (TR1 does not have routes to N3). Check the packets.

*Part C: UPDATE from External Peer*

7. Remove the static route on the RUT.
8. BGP is restarted on the routers. Wait until they establish the connections.
9. TR1 sends an UPDATE message to some network N3 with an AS_PATH set to (AS_SEQUENCE/AS1). Check the packets.

*Part D: UPDATE from Internal Peer*

10. BGP is restarted on the routers. Wait until they establish the connections.
11. TR2 sends an UPDATE message for N3 with an empty AS_PATH. Check the packets.

*Part E: Propagation from External Peer to External Peer*

12. TR2 is now in autonomous system AS3.
13. BGP is restarted on the routers. Wait until they establish the connections.
14. TR2 sends an UPDATE message for N3 with an AS_PATH set to (AS_SEQUENCE/AS3). Check the packets.

*Part F: UPDATE with AS_SET*

15. TR2 is replaced with a packet generator simulating a BGP router, configured in autonomous system AS3.
16. BGP is restarted on the routers. Wait until they establish the connections.
17. TR2 sends an UPDATE for a new route to N3, with its AS_PATH starting with (AS_SET/AS3, AS4). Check the packets.

**Observable Results:**

- In Part A, the RUT should send an UPDATE message for N3 to both TR1 and TR2. The UPDATE sent to TR2 should have an empty AS_PATH attribute.
- In Part B, the RUT should send an UPDATE to TR1 with an AS_PATH attribute set to (AS_SEQUENCE/AS2).

- In Part C, the RUT should send an UPDATE for N3 to TR2 with AS_PATH set to (AS_SEQUENCE/AS1).
- In Part D, the RUT should send an UPDATE for N3 to TR1 with the AS_PATH set to (AS_SEQUENCE/AS2).
- In Part E, the RUT should send an UPDATE for N3 to TR1 with AS_PATH set to (AS_SEQUENCE/AS2, AS3) (prepends its own AS number).
- In Part F, the RUT should send an UPDATE for N3 to TR1 with AS_PATH set to (AS_SEQUENCE/AS2)-(AS_SET/AS3-AS4) (prepends an AS_SEQUENCE segment containing its own AS number).

**Possible Problems:** None.

**Test BGP_CONF.1.12: NEXT_HOP Attribute**

**Purpose:** To verify that a BGP router properly handles the NEXT_HOP attribute.

**References:** [RFC 4271] – Sections 4.3, 5.1.3

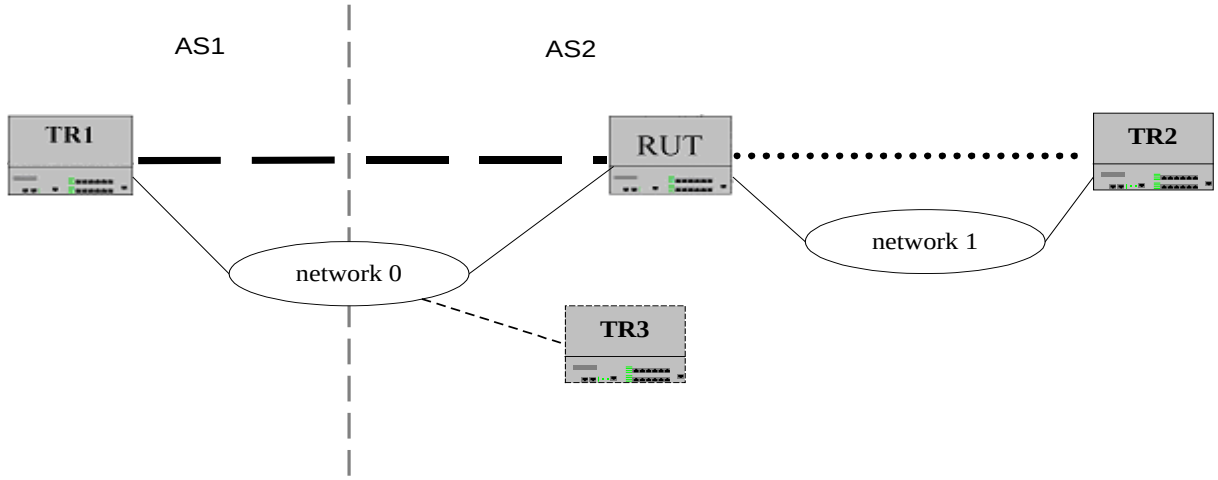
**Last Modification:** February 3, 2003

**Discussion:** This is a well-known mandatory attribute that defines the (unicast) IP address of the router that SHOULD be used as the next hop to the destinations listed in the Network Layer Reachability Information field of the UPDATE message.

> 1) When sending a message to an internal peer, if the route is not locally originated the BGP speaker SHOULD NOT modify the NEXT_HOP attribute, unless it has been explicitly configured to announce its own IP address as the NEXT_HOP. When announcing a locally originated route to an internal peer, the BGP speaker SHOULD use as the NEXT_HOP the interface address of the router through which the announced network is reachable for the speaker; if the route is directly connected to the speaker, or the interface address of the router through which the announced network is reachable for the speaker is the internal peer's address, then the BGP speaker SHOULD use for the NEXT_HOP attribute its own IP address (the address of the interface that is used to reach the peer).
>
> 2) When sending a message to an external peer X, and the peer is one IP hop away from the speaker:
>
> > - If the route being announced was learned from an internal peer or is locally originated, the BGP speaker can use for the NEXT_HOP attribute an interface address of the internal peer router (or the internal router) through which the announced network is reachable for the speaker, provided that peer X shares a common subnet with this address. This is a form of "third party" NEXT_HOP attribute.
> >
> > - Otherwise, if the route being announced was learned from an external peer, the speaker can use in the NEXT_HOP attribute an IP address of any adjacent router (known from the received NEXT_HOP attribute) that the speaker itself uses for local route calculation, provided that peer X shares a common subnet with this address. This is a second form of "third party" NEXT_HOP attribute.

A route originated by a BGP speaker SHALL NOT be advertised to a peer using an address of that peer as NEXT_HOP. A BGP speaker SHALL NOT install a route with itself as the next hop.

**Test Setup:**




**Procedure:**

*Part A: NEXT_HOP on Same Subnet*

1. The RUT and TR1 are configured as external peers. The RUT and TR2 are configured as internal peers. TR3 is not running BGP.
2. Configure the RUT to export static routes to BGP.
3. BGP is started on the routers. Wait until they establish the connections.
4. Give the RUT a static route to some network N2 (TR1 and TR2 do not have a route to N2), with the next hop set to TR3. Check the packets on N0.

*Part B: NEXT_HOP on Different Subnet*

5. Check the packets on N1.

*Part C: UPDATE Propagated with NEXT_HOP set to Self*

6. Remove the static route on the RUT.
7. BGP is restarted on the routers. Wait until they establish the connections.
8. TR2 sends an UPDATE message for some network N2, with the NEXT_HOP set to itself. Check the packets.

*Part D: UPDATE Propagated to Router on Different Subnet*

9. TR2 withdraws its path to N2. The RUT is NOT configured to set the NEXT_HOP to itself.
10. BGP is restarted on the routers. Wait until they establish the connections.
11. TR1 sends an UPDATE message for some network N2 with the NEXT_HOP set to TR3's IP Address on N0. Check the packets.

*Part E: NEXT_HOP Set to Self*

12. Configure the RUT to advertise itself as the NEXT_HOP.
13. BGP is restarted on the routers. Wait until they establish the connections.
14. TR1 sends an UPDATE message for some network N2. Check the packets.

*Part F: NEXT_HOP set to Peer*

15. TR1 withdraws its path to N2. Configure the RUT to advertise a static route with TR1's IP Address on N0 as the next hop.
16. BGP is restarted on the routers. Wait until they establish the connections.
17. Check the packets.

**Observable Results:**

- In Part A, the RUT should send an UPDATE message for N2 to TR1. The UPDATE for TR1 should have the NEXT_HOP attribute set to TR3's IP address on N0.

- In Part B, the RUT should send an UPDATE message for N2 to TR2. The NEXT_HOP attribute should be set to TR3's IP address on N0.
- In Part C, the RUT should propagate the UPDATE for N2 to TR1. The RUT should set the NEXT_HOP to an address on N0 (possibly its own address).
- In Part D, the RUT should propagate the UPDATE to TR2. The NEXT_HOP attribute in the RUT's UPDATE should be set to TR3's IP address on N0.
- In Part E, the RUT should propagate the UPDATE to TR2. The NEXT_HOP attribute in the RUT's UPDATE should be set to the RUT's IP address on N1.
- In Part F, the RUT should not send an UPDATE message to TR1 listing TR1 as the NEXT_HOP.

**Possible Problems:** None.

## Test BGP_CONF.1.13: MULTI_EXIT_DISC Attribute

**Purpose:** To verify that a BGP router properly handles the MULTI_EXIT_DISC attribute.

**References:** [RFC 4271] – Sections 4.3, 5.1.4

**Last Modification:** February 3, 2003

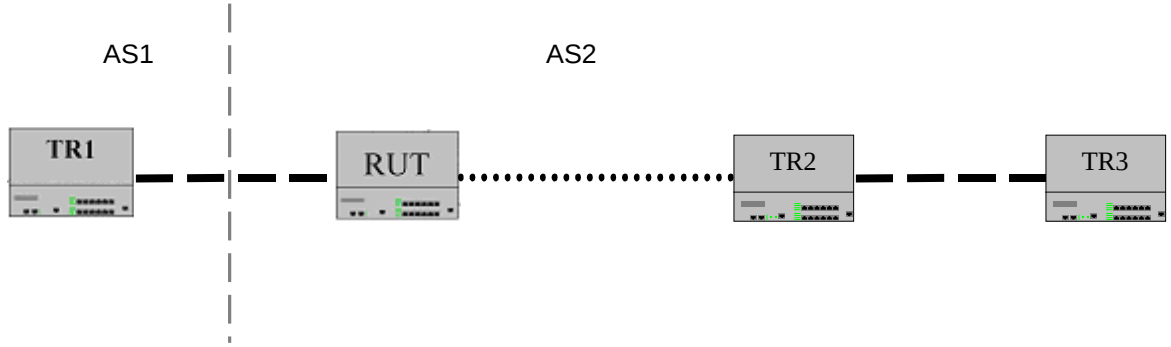
**Discussion:** This is an optional non-transitive attribute that is a four octet non-negative integer. The value of this attribute may be used by a BGP speaker's decision process to discriminate among multiple exit points to a neighboring autonomous system.

The MULTI_EXIT_DISC is an optional non-transitive attribute, which may be used on external (inter-AS) links to discriminate among multiple exit or entry points to the same neighboring AS. The value of the MULTI_EXIT_DISC attribute is a four octet unsigned number which is called a metric. All other factors being equal, the exit point with lower metric should be preferred. If received over EBGP, the MULTI_EXIT_DISC attribute MAY be propagated over IBGP to other BGP speakers within the same AS. The MULTI_EXIT_DISC attribute received from a neighboring AS MUST NOT be propagated to other neighboring ASs.

A BGP speaker MUST IMPLEMENT a mechanism based on local configuration, which allows the MULTI_EXIT_DISC attribute to be removed from a route. This MAY be done prior to determining the degree of preference of the route and performing route selection (decision process phases 1 and 2).

An implementation MAY also (based on local configuration) alter the value of the MULTI_EXIT_DISC attribute received over EBGP. If a BGP speaker is configured to alter the value of the MULTI_EXIT_DISC attribute received over EBGP, then altering the value MUST be done prior to determining the degree of preference of the route and performing route selection (Decision Process phases 1 and 2). See Section 9.1.2.2 for necessary restrictions on this.

**Test Setup:**




**Procedure:**

*Part A: UPDATE Originated with MED Attribute*

1. The RUT and TR1 are configured as external peers. The RUT and TR2 are configured as internal peers.
2. Configure the RUT to export static routes to BGP, and to use MED.
3. BGP is started on the routers. Wait until they establish the connections.
4. Give the RUT a static route to some network N4. Check the packets.

*Part B: UPDATE Received with MED*

5. Remove all static routes on the RUT.
6. BGP is restarted on the routers. Wait until they establish the connections.
7. TR1 sends an UPDATE message for some network N4 with the MED field. Check the packets.

*Part C: Internal Peer Propagates UPDATE from External Peer*

8. TR2 and TR3 are external peers.
9. BGP is restarted on the routers. Wait until they establish the connections.
10. TR3 sends an UPDATE message for some network N5 with the MED field. TR2 propagates it to the RUT, retaining the MED field. Check the packets.

**Observable Results:**

- In Part A, the RUT should send UPDATE messages for N4 to both TR1 and TR2. The UPDATE to TR1 should have the MULTI_EXIT_DISC attribute (with the configured value).
- In Part B, the RUT should propagate the UPDATE to TR2. The UPDATE to TR2 may include the MED attribute (if the RUT is configured to propagate MED in internal updates).
- In Part C, the RUT should propagate the UPDATE to TR1. The UPDATE to TR1 should not include the MED attribute received from TR2.

**Possible Problems:** None.

## Test BGP_CONF.1.14: LOCAL_PREF Attribute

**Purpose:** To verify that a BGP router properly handles the LOCAL_PREF attribute.

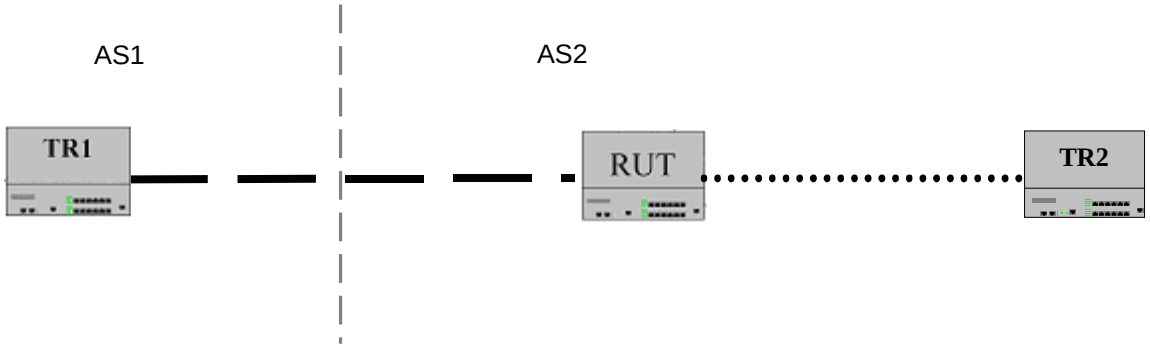
**References:** [RFC 4271] – Sections 4.3, 5.1.5

**Last Modification:** February 3, 2003

**Discussion:** LOCAL_PREF is a well-known mandatory attribute that SHALL be included in all UPDATE messages that a given BGP speaker sends to the other internal peers. A BGP speaker SHALL calculate the degree of preference for each external route and include the degree of preference when advertising a route to its internal peers. The higher degree of preference MUST be preferred. A BGP speaker uses the degree of preference learned via LOCAL_PREF in its decision process (see section 9.1.1).

A BGP speaker MUST NOT include this attribute in UPDATE messages that it sends to external peers, except for the case of BGP Confederations [RFC3065]. If it is contained in an UPDATE message that is received from an external peer, then this attribute MUST be ignored by the receiving speaker, except for the case of BGP Confederations [RF3065].

**Test Setup:**




**Procedure:**

*Part A: UPDATE Propagated from External Peer*

1. The RUT and TR1 are configured as external peers. The RUT and TR2 are set up as BGP internal peers.
2. BGP is started on the routers. Wait until they establish the connections.
3. TR1 sends an UPDATE message for some network N3 (the RUT and TR2 do not have routes to N3). Check the packets.

*Part B: UPDATE Propagated from Internal Peer*

4. BGP is restarted on the routers. Wait until they establish the connections.
5. TR2 sends an UPDATE message for some network N3 with the LOCAL_PREF attribute (the RUT and TR1 do not have routes to N3). Check the packets.

*Part C: LOCAL_PREF Attribute Generation*

6. BGP is restarted on the routers. Wait until they establish the connections.
7. Give the RUT a route to some network N3 (TR1 and TR2 do not have routes to N3). Check the packets.

*Part D: Erroneous LOCAL_PREF Attribute Handling*

8. TR1 is replaced with a packet generator that simulates a BGP router.

9. BGP is restarted on the routers. Wait until they establish the connections.
10. TR1 sends an UPDATE for some network N3 (the RUT and TR2 do not have routes to N3). The UPDATE has the LOCAL_PREF attribute included.
11. Check the packets.

**Observable Results:**

- In Part A, the RUT should propagate the UPDATE message for N3 to TR2. The UPDATE should contain the LOCAL_PREF attribute.
- In Part B, the RUT should propagate the UPDATE to TR1, but without the LOCAL_PREF attribute.
- In Part C, the RUT should send UPDATE messages for N3 to both TR1 and TR2. The UPDATE to TR1 should not have the LOCAL_PREF attribute. The UPDATE to TR2 should include the LOCAL_PREF attribute.
- In Part D, the RUT should ignore the LOCAL_PREF attribute received in the UPDATE from TR1. It should propagate the UPDATE to TR2, including a LOCAL_PREF attribute that is computed by itself, not the one received in the UPDATE.

**Possible Problems:** None.

**Test BGP_CONF.1.15: ATOMIC_AGGREGATE Attribute**

**Purpose:** To verify that a BGP router properly handles the ATOMIC_AGGREGATE attribute.

**References:** [RFC 4271] – Sections 4.3, 5.1.6 and 9.1.4

**Last Modification:** July 14, 2004

**Discussion:** ATOMIC_AGGREGATE is a well-known discretionary attribute.

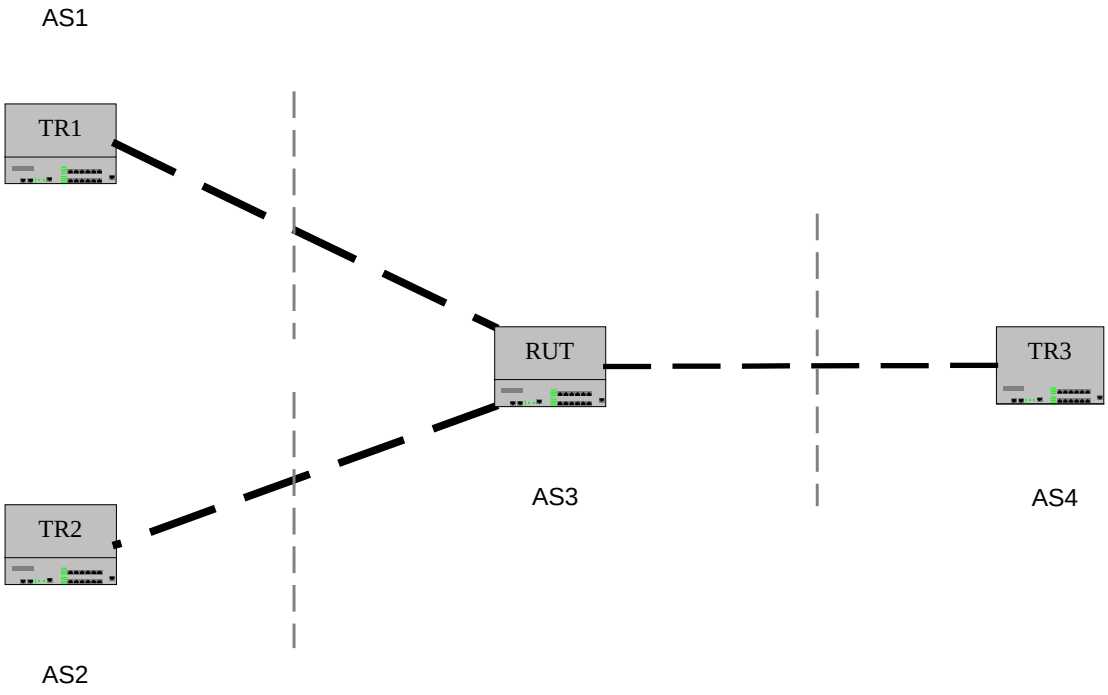
If a BGP speaker receives overlapping routes, the Decision Process MUST consider both routes based on the configured acceptance policy. If both a less and a more specific route are accepted, then the Decision Process MUST install in Loc-RIB either both the less and the more specific routes or aggregate the two routes and install in Loc-RIB the aggregated route, provided that both routes have the same value of the NEXT_HOP attribute.

When a BGP speaker aggregates several routes for the purpose of advertisement to a particular peer, the AS_PATH of the aggregated route normally includes an AS_SET formed from the set of ASs from which the aggregate was formed. In many cases the network administrator can determine that the aggregate can safely be advertised without the AS_SET and not form route loops.

If an aggregate excludes at least some of the AS numbers present in the AS_PATH of the routes that are aggregated as a result of dropping the AS_SET, the aggregated route, when advertised to the peer, SHOULD include the ATOMIC_AGGREGATE attribute.

A BGP speaker that receives a route with the ATOMIC_AGGREGATE attribute SHOULD NOT remove the attribute from the route when propagating it to other speakers.

**Test Setup:**



**Procedure:**

*Part A: No ATOMIC_AGGREGATE Attribute Generation*

1. The RUT and TR1, the RUT and TR2, and the RUT and TR3 are configured as external peers.
2. Configure the RUT to NOT aggregate overlapping routes.
3. BGP is started on the routers. Wait until they establish the connections.
4. TR1 sends an UPDATE message for 192.0.0.0/8. TR2 sends an UPDATE message for 192.1.0.0/16. Check the packets and the RUT's routing table.

*Part B: ATOMIC_AGGREGATE Attribute Generation*

5. Configure the RUT to aggregate overlapping routes under 192.0.0.0/8.
6. BGP is restarted on the routers. Wait until they establish the connections.
7. TR1 sends an UPDATE message for 192.0.0.0/8. TR2 sends an UPDATE message for 192.1.0.0/16. Check the packets and the RUT's routing table.

*Part C: ATOMIC_AGGREGATE Attribute Propagation*

8. TR1 is replaced with a packet generator that simulates a BGP router. The RUT and TR2 are no longer peers.
9. BGP is restarted on the routers. Wait until they establish the connections.
10. TR1 sends an UPDATE for 192.0.0.0/8, with the ATOMIC_AGGREGATE and the AGGREGATOR attributes included. Check the packets.

**Observable Results:**

- In Part A, the RUT should install both 192.0.0.0/8 and 192.1.0.0/16 in its routing table. The RUT should propagate the two UPDATE messages to TR2. The UPDATE messages should not include the ATOMIC_AGGREGATE or the AGGREGATOR attributes.
- In Part B, the RUT should install either an aggregate route 192.0.0.0/8 in its routing table or it should install both the component routes 192.0.0.0/8 and 192.1.0.0/16. The RUT should send an UPDATE message to TR2 for 192.0.0.0/8. It is recommended that the UPDATE include all AS used to form the aggregate in an AS_SET i.e. the AS_PATH should be set to (AS_SEQUENCE/AS2)-(AS_SET/AS1, AS3). If the RUT does not contain the AS_SET in its AS_PATH, it's recommended that it include the ATOMIC_AGGREGATE attribute in its UPDATE message.
- In Part C, the RUT should propagate the UPDATE message to TR2. It is recommended that the ATOMIC_AGGREGATE and AGGREGATOR attributes remain unchanged.

**Possible Problems:** None.

**Test BGP_CONF.1.16: Aggregation Path Attributes**

**Purpose:** To verify the correct handling of path attributes when a BGP router aggregates routes.

**References:** [RFC 4271] – Section 9.2.2.2

**Last Modification:** July 14, 2004

**Discussion:** Aggregation is the process of combining the characteristics of several different routes in such a way that a single route can be advertised. Routes can be aggregated by applying the following procedure separately to path attributes of the same type and to the Network Layer Reachability Information. Routes that have different MULTI_EXIT_DISC attribute SHALL NOT be aggregated.

Path attributes that have different type codes can not be aggregated together. Path attributes of the same type code may be aggregated, according to the following rules:

NEXT_HOP:

When aggregating routes that have different NEXT_HOP attribute, the NEXT_HOP attribute of the aggregated route SHALL identify an interface on the BGP speaker that performs the aggregation.

ORIGIN attribute:

If at least one route among routes that are aggregated has ORIGIN with the value INCOMPLETE, then the aggregated route MUST have the ORIGIN attribute with the value INCOMPLETE. Otherwise, if at least one route among routes that are aggregated has ORIGIN with the value EGP, then the aggregated route MUST have the ORIGIN attribute with the value EGP. In all other cases the value of the ORIGIN attribute of the aggregated route is IGP.

AS_PATH attribute:

If routes to be aggregated have identical AS_PATH attributes, then the aggregated route has the same AS_PATH attribute as each individual route.

When performing route aggregation, a conformant implementation SHALL be able to perform the following algorithm:

- determine the longest leading sequence of tuples (as defined above) common to all the AS_PATH attributes of the routes to be aggregated. Make this sequence the leading sequence of the aggregated AS_PATH attribute.

- set the type of the rest of the tuples from the AS_PATH attributes of the routes to be aggregated to AS_SET, and append them to the aggregated AS_PATH attribute.

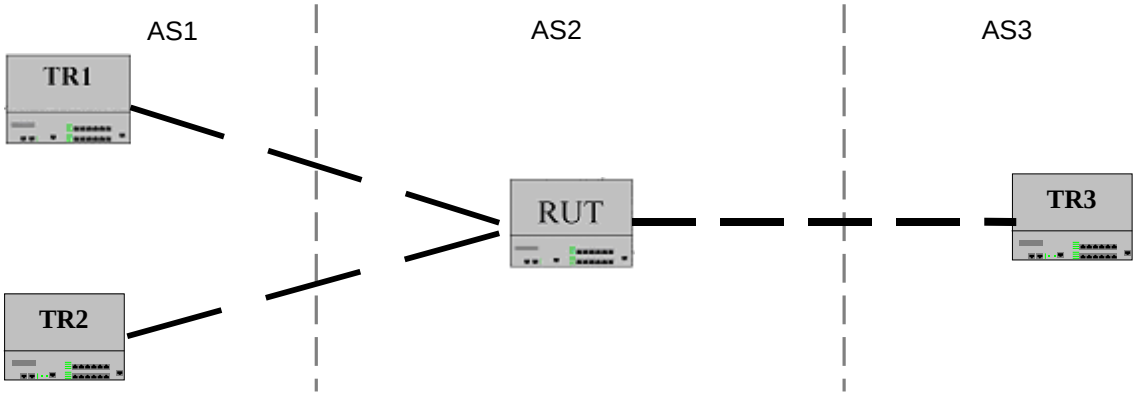
ATOMIC_AGGREGATE:

If at least one of the routes to be aggregated has ATOMIC_AGGREGATE path attribute, then the aggregated route SHALL have this attribute as well.

AGGREGATOR:

Any AGGREGATOR attributes from the routes to be aggregated MUST NOT be included in the aggregated route. The BGP speaker performing the route aggregation MAY attach a new AGGREGATOR attribute (see Section 5.1.7).

**Test Setup:** Configure the RUT and TR1 as external peers; TR1 is a packet generator simulating a BGP router. Configure the RUT and TR2 as external peers; the RUT and TR3 as external peers. Configure the RUT to aggregate routes below 192.1.0.0/16. Start the routers and wait until they synchronize.




**Procedure:**

*Part A: Aggregation with different NEXT_HOP Attribute*

1. The RUT and TR1, the RUT and TR2, and the RUT and TR3 are configured as external peers.
2. BGP is started on the routers. Wait until they establish the connections.
3. TR1 sends an UPDATE for 192.1.1.0/24 with NEXT_HOP set to itself. TR2 sends an UPDATE for 192.1.2.0/24 with NEXT_HOP set to itself.
4. Check the packets.

*Part B: No Aggregation with MED Attribute*

5. BGP is restarted on the routers. Wait until they establish the connections.
6. TR1 sends an UPDATE for 192.1.1.0/24 with MED set to 1. TR2 sends an UPDATE for 192.1.2.0/24 with MED set to 2 (from now on TR1 and TR2 will send UPDATE messages with the same NEXT_HOP).
7. Check the packets.

*Part C: Aggregation with different ORIGIN Attributes: INCOMPLETE and IGP*

8. BGP is restarted on the routers. Wait until they establish the connections.
9. TR1 sends an UPDATE for 192.1.1.0/24 with ORIGIN set to INCOMPLETE. TR2 sends an UPDATE for 192.1.2.0/24 with ORIGIN set to IGP.
10. Check the packets.

*Part D: Aggregation with different ORIGIN Attribute: EGP and IGP*

11. BGP is restarted on the routers. Wait until they establish the connections.
12. TR1 sends an UPDATE for 192.1.1.0/24 with ORIGIN set to EGP. TR2 sends an UPDATE for 192.1.2.0/24 with ORIGIN set to IGP.
13. Check the packets.

*Part E: Aggregation with identical AS_PATH Attribute*

14. BGP is restarted on the routers. Wait until they establish the connections.
15. TR1 sends an UPDATE for 192.1.1.0/24 with AS_PATH set to (AS_SEQUENCE/AS1-AS11). TR2 sends an UPDATE for 192.1.2.0/24 with AS_PATH set to (AS_SEQUENCE/AS1-AS11).
16. Check the packets.

*Part F: Aggregation of Routes with different AS_PATH Attributes from Routers in the Same AS*

17. BGP is restarted on the routers. Wait until they establish the connections.
18. TR1 sends an UPDATE for 192.1.1.0/24 with AS_PATH set to (AS_SEQUENCE/AS1-AS11).

19. TR2 is peered with another BGP router in AS12. That router sends an UPDATE for 192.1.2.0/24. TR2 propagates this UPDATE with the AS_PATH set to (AS_SEQUENCE/AS1-AS12).
20. Check the packets.

*Part G: Aggregation of Routes with different AS_PATH Attributes from Routers in Different AS's*

21. TR2 is no longer peered with the router in AS12. TR2 is moved to AS4.
22. BGP is restarted on the routers. Wait until they establish the connections.
23. TR1 sends an UPDATE for 192.1.1.0/24 with AS_PATH set to (AS_SEQUENCE/AS1). TR2 sends an UPDATE for 192.1.2.0/24; the AS_PATH will be set to (AS_SEQUENCE/AS4).
24. Check the packets.

*Part H: ATOMIC_AGGREGATE Attribute Handling*

25. TR2 is moved back to AS1. BGP is restarted on the routers. Wait until they establish the connections.
26. TR1 sends an UPDATE for 192.1.1.0/24 with the ATOMIC_AGGREGATE and the AGGREGATOR attributes. TR2 sends an UPDATE for 192.1.2.0/24.
27. Check the packets.

**Observable Results:**

- In Part A, the RUT should send an UPDATE to TR3, for 192.1.0.0/16, with the NEXT_HOP set to itself.
- In Part B, the RUT should not aggregate 192.1.1.0/24 and 192.1.2.0/24. It should propagate the two UPDATE messages to TR3.
- In Part C, the RUT should send an UPDATE to TR3, for 192.1.0.0/16, with ORIGIN set to INCOMPLETE.
- In Part D, the RUT should send an UPDATE to TR3, for 192.1.0.0/16, with ORIGIN set to EGP.
- In Part E, the RUT should send an UPDATE to TR3, for 192.1.0.0/16, with AS_PATH set to (AS_SEQUENCE/AS2-AS1-AS11).
- In Part F, the RUT should send an UPDATE to TR3, for 192.1.0.0/16. The AS_PATH should be set to (AS_SEQUENCE/AS2-AS1)-(AS_SET/AS11-AS12).
- In Part G, the RUT should send an UPDATE to TR3, for 192.1.0.0/16. The AS_PATH should be set to (AS_SEQUENCE/AS2)-(AS_SET/AS1-AS4).
- In Part H, the RUT should send an UPDATE to TR3, for 192.1.0.0/16. The UPDATE should include the ATOMIC_AGGREGATE attribute. The RUT should ignore the received AGGREGATOR attribute. The UPDATE should either have no AGGREGATOR attribute, or the AGGREGATOR attribute should be set to the RUT.

**Possible Problems:** None.

**Test BGP_CONF.1.17: Optional Attributes**

**Purpose:** To verify the correct handling of unrecognized optional attributes.

**References:** [RFC 4271] – Section 9

**Last Modification:** August 21, 2001

**Discussion:** If an optional non-transitive attribute is unrecognized, it is quietly ignored. If an optional transitive attribute is unrecognized, the Partial bit (the third high-order bit) in the attribute flags octet is set to 1, and the attribute is retained for propagation to other BGP speakers.

**Test Setup:**


**Procedure:**

*Part A: Non-transitive Attribute Handling*

1. The RUT and TR1 are configured as external peers; TR1 is a packet generator simulating a BGP router. The RUT and TR2 are configured as external peers.
2. BGP is started on the routers. Wait until they establish the connections.
3. TR1 send an UPDATE including an optional non-transitive attribute, Type Code 33. Check the packets.

*Part B: Partial Bit Clear Transitive Attribute Handling*

4. BGP is restarted on the routers. Wait until they establish the connections.
5. TR1 sends an UPDATE including an optional transitive attribute, Type Code 33. This attribute has the Partial bit clear. Check the packets.

*Part C: Partial Bit Set Transitive Attribute Handling*

6. BGP is restarted on the routers. Wait until they establish the connections.
7. TR1 sends an UPDATE including an optional transitive attribute, Type Code 33. This attribute has the Partial bit set. Check the packets.

**Observable Results:**

- In Part A, since the RUT does not recognize the attribute with Type Code 33, it should ignore it. Therefore, the RUT should propagate the UPDATE to TR2, without this attribute.
- In Part B, the RUT should set the Partial bit, and then propagate the UPDATE to TR2, including the modified attribute.
- In Part C, the RUT should propagate the UPDATE to TR2, without changing the attribute.

**Possible Problems:** None.

**Test BGP_CONF.1.18: Route Selection**

**Purpose:** To verify the route selection "tie breaking" algorithm implemented by a BGP router.

**References:** [RFC 4271] – Sections 9.1.2, 9.1.2.2

**Last Modification:** February 3, 2003

**Discussion:** If the NEXT_HOP attribute of a BGP route depicts an address that is not resolvable, or it would become unresolvable if the route was installed in the routing table the BGP route should be excluded from the Phase 2 decision function.

For each set of destinations for which a feasible route exists in the Adj-RIBs-In, the local BGP speaker identifies the route that has:

a) the highest degree of preference of any route to the same set of destinations, or
b) is the only route to that destination, or
c) is selected as a result of the Phase 2 tie breaking rules specified in 9.1.2.2.

The local speaker SHALL then install that route in the Loc-RIB, replacing any route to the same destination that is currently being held in the Loc-RIB. When the new BGP route is installed in the Routing Table, care must be taken to ensure that existing routes to the same destination that are now considered invalid are removed from the Routing Table. Whether or not the new BGP route replaces an existing non-BGP route in the Routing Table depends on the policy configured on the BGP speaker.

The local speaker MUST determine the immediate next-hop address from the NEXT_HOP attribute of the selected route (see section 5.1.3). If either the immediate next hop or the IGP cost to the NEXT_HOP (where the NEXT_HOP is resolved through an IGP route) changes, Phase 2: Route Selection must be performed again.

In its Adj-RIBs-In a BGP speaker may have several routes to the same destination that have the same degree of preference. The local speaker can select only one of these routes for inclusion in the associated Loc-RIB. The local speaker considers all routes with the same degrees of preference, both those received from internal peers, and those received from external peers.

The following tie-breaking procedure assumes that for each candidate route all the BGP speakers within an autonomous system can ascertain the cost of a path (interior distance) to the address depicted by the NEXT_HOP attribute of the route and follow the same route selection algorithm.

The tie-breaking algorithm begins by considering all equally preferable routes and then selects routes to be removed from consideration. The algorithm terminates as soon as only one route remains in consideration. The criteria must be applied in the order specified.

Several of the criteria are described using pseudo-code. Note that the pseudo-code shown was chosen for clarity, not efficiency. It is not intended to specify any particular implementation. BGP implementations MAY use any algorithm which produces the same results as those described here.

a) Remove from consideration all routes, which are not tied for having the smallest number of AS numbers present in their AS_PATH attributes. Note, that when counting this number, an AS_SET counts as 1, no matter how many ASs are in the set.

b) Remove from consideration all routes, which are not tied for having the lowest Origin number in their Origin attribute.

c) Remove from consideration routes with less-preferred MULTI_EXIT_DISC attributes. MULTI_EXIT_DISC is only comparable between routes learned from the same neighboring AS.

Routes that do not have the MULTI_EXIT_DISC attribute are considered to have the lowest possible MULTI_EXIT_DISC value and, therefore, are the most preferred. The lower the MULTI_EXIT_DISC value, the more preferred it is.

If a route is learned via IBGP, and the other IBGP speaker didn't originate the route, it is the neighbor AS from which the other IBGP speaker learned the route. If the route is learned via IBGP, and the other IBGP speaker originated the route, it is the local AS.

d) If at least one of the candidate routes was received from an external peer in a neighboring autonomous system, remove from consideration all routes which were received from internal peers.

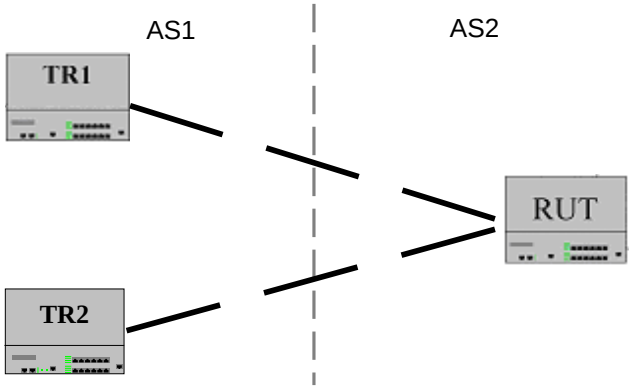
e) Remove from consideration any routes with less-preferred interior cost. The interior cost of a route is determined by calculating the metric to the NEXT_HOP for the route using the interior routing protocol(s). If the NEXT_HOP for a route is reachable, but no cost can be determined, then this step should be skipped (equivalently, consider all routes to have equal costs).

The lower the interior cost of a route, the more preferred the route is.

f) Remove from consideration all routes other than the route that was advertised by the BGP speaker whose BGP Identifier has the lowest value.

g) Prefer the route received from the lowest neighbor address.

**Test Setup:** Configure the RUT and TR1 as external peers; the RUT and TR2 as external peers. Configure the RUT to assign a higher degree of preference to routes received from TR1 than those received from TR2.



**Procedure:**

*Part A: Low Degree of Preference vs. High Degree of Preference*

1. The RUT and TR1 and the RUT and TR2 are configured as external peers. Configure the RUT to assign a higher degree of preference to routes received from TR1 than those received from TR2.
2. BGP is started on the routers. Wait until they establish the connections.
3. TR1 and TR2 send UPDATE messages for 192.1.1.0/24. The NEXT_HOP attributes are set to TR1 and TR2, respectively. The other path attributes are identical. Check the RUT's routing table.

*Part B: Short AS_PATH vs. Long AS_PATH*

4. From now on the RUT is configured to assign the same degree of preference to routes received from both TR1 and TR2.
5. BGP is restarted on the routers. Wait until they establish the connections.
6. TR1 sends an UPDATE for 192.1.1.0/24, with AS_PATH set to (AS_SEQUENCE/AS1, AS11), and NEXT_HOP set to itself. TR2 sends an UPDATE for 192.1.1.0/24, with AS_PATH set to (AS_SEQUENCE/AS1), and NEXT_HOP set to itself.
7. Check the RUT's routing table.

*Part C: Low ORIGIN vs. High ORIGIN*

8. From now on TR1 and TR2 advertise UPDATE messages with the NEXT_HOP field set to themselves.
9. BGP is restarted on the routers. Wait until they establish the connections.
10. TR1 sends an UPDATE for 192.1.1.0/24, with ORIGIN=0. TR2 sends an UPDATE for 192.1.1.0/24, with ORIGIN=2.
11. Check the RUT's routing table.

*Part D: Low MED vs. High MED*

12. From now on TR1 and TR2 advertise UPDATE messages with the same ORIGIN.
13. BGP is restarted on the routers. Wait until they establish the connections.
14. TR1 send an UPDATE for 192.1.1.0/24, with MED=20. TR2 send an UPDATE for 192.1.1.0/24, with MED=10.
15. Check the RUT's routing table.

*Part E: No MED vs. MED*

16. BGP is restarted on the routers. Wait until they establish the connections.
17. TR1 sends an UPDATE for 192.1.1.0/24, without MED. TR2 sends an UPDATE for 192.1.1.0/24, with MED=10.
18. Check the RUT's routing table.

*Part F: External Peer vs. Internal Peer*

19. From now on TR1 and TR2 send UPDATE messages without MED.
20. TR2 is moved to AS2. A new router TR3 is added to AS1. TR2 and TR3 are external peers
21. BGP is restarted on the routers. Wait until they establish the connections.
22. TR1 sends an UPDATE for 192.1.1.0/24. TR3 sends an UPDATE for 192.1.1.0/24 to TR2. TR2 propagates the message to the RUT.
23. Check the RUT's routing table.

*Part G: Low Interior Cost vs. High Interior Cost*

24. TR1 is moved to AS2. From now on TR1, TR2, and the RUT are IBGP peers.
25. The RUT is connected to N0, TR1 is connected to N0 and N1 and TR2 is connected to N0 and N2. BGP is enabled on N1 and N2 on TR1 and TR2 respectively.
26. Configure the RUT to have static routes to N1 and N2 with a cost of 20 to N1, and a cost of 10 to N2.
27. BGP is restarted on the routers. Wait until they establish the connections.
28. TR1 sends an UPDATE for 192.1.1.0/24. TR2 sends an UPDATE for 192.1.1.0/24.
29. Check the RUT's routing table.

*Part H: Low BGP Identifier vs. High BGP Identifier*

30. TR1's BGP Identifier is set to 1.1.1.1. TR2's BGP Identifier is set to 2.2.2.2.
31. BGP is restarted on the routers. Wait until they establish the connections.
32. TR1 sends an UPDATE for 192.1.1.0/24. TR2 sends an UPDATE for 192.1.1.0/24.
33. Check the RUT's routing table.

*Part I: Low IP Address vs. High IP Address*

34. BGP is stopped on TR2.
35. The RUT and TR1 are configured as IBGP peers on N0 and N1. TR1's IP Address on N0 is lower than its IP Address on N1.
36. BGP is restarted on the routers. Wait until they establish the connections.
37. TR1 sends an UPDATE for 192.1.1.0/24 on both N0 and N1.
38. Check the RUT's routing table.

**Observable Results:**

- In Part A, the RUT should have a route to 192.1.1.0/24, with TR1 as next hop.
- In Part B, the RUT should have a route to 192.1.1.0/24, with TR2 as next hop.
- In Part C, the RUT should have a route to 192.1.1.0/24, with TR1 as next hop.
- In Part D, the RUT should have a route to 192.1.1.0/24, with TR2 as next hop.
- In Part E, the RUT should have a route to 192.1.1.0/24, with TR1 as next hop.
- In Part F, the RUT should have a route to 192.1.1.0/24, with TR1 as next hop.
- In Part G, the RUT should have a route to 192.1.1.0/24, with TR2 as next hop.
- In Part H, the RUT should have a route to 192.1.1.0/24, with TR1 as next hop.
- In Part I, the RUT should have a route to 192.1.1.0/24, with TR1's IP Address on N0 as next hop.

**Possible Problems:** None.

# 2. BGP FINITE STATE MACHINE

**Overview:**

The following tests are designed to verify the correct functioning of the BGP finite state machine.

**Discussion:**

BGP operation can be specified in terms of a finite state machine (FSM).
The BGP finite state machine has the following states:
Idle, Connect, Active, OpenSent, OpenConfirm and Established

The events causing the BGP finite state machine to transition from one state to another are:

Administrative Events:
Manual Start [Event 1], Manual Stop [Event 2], Automatic Start [Event 3], Manual Start with passive TCP establishment [Event 4], Automatic Start with passive TCP establishment [Event 5], Automatic Start with bgp_stop_flap option set [Event 6] and Auto Stop [Event 7]

Timer Events:
Idle Hold timer expires [Event 8], Connect retry timer expires [Event 9], Hold time expires [Event 10], Keepalive timer expires [Event 11] and DelayBGP open timer expires [Event 12]

TCP Connection based Events:
TCP connection indication & valid remote peer [Event 13], RCV TCP connection indication with invalid source or destination [Event 14], TCP connection request sent received an ACK [Event 15], TCP connection confirmed [Event 16] and TCP connection fails [Event 17]

BGP Messages based Events:
BGP Open [Event 18], BGPOpen with BGP Delay Option Timer running [Event 19], BGPHeaderErr [Event 20], BGPOpenMsgErr [Event 21], Open collision dump [Event 22], NotifMsgVerErr [Event 23], NotifMsg [Event 24], KeepAliveMsg [Event 25], UpdateMsg [Event 26] and UpdateMsgErr [Event 27]

**Test BGP_CONF.2.1: Idle State**

**Purpose:** To verify the correct functionality of a BGP router in the Idle state.

**References:** [RFC 4271] – Section 8.2.2

**Last Modification:** December 9, 2004

**Discussion:** Initially the BGP peer FSM is in the Idle state. (Hereafter the BGP peer FSM will be shortened to BGP FSM.)

In this state BGP FSM refuses all incoming BGP connections for this peer. No resources are allocated to the peer. In response to a ManualStart event (Event 1) or an AutomaticStart event (Event 3), the local system:

- initializes all BGP resources for the peer connection,
- sets ConnectRetryCounter to zero,
- starts the ConnectRetryTimer with initial value,
- initiates a TCP connection to the other BGP peer,
- listens for a connection that may be initiated by the remote BGP peer, and
- changes its state to Connect.

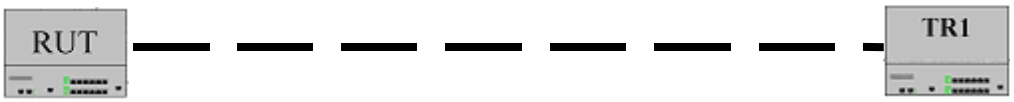
In response to a ManualStart_with_PassiveTcpEstablishment event (Event 4) or AutomaticStart_with_PassiveTcpEstablishment event (Event 5), the local system:

- initializes all BGP resources,
- sets the ConnectRetryCounter to zero,
- starts the ConnectRetryTimer with initial value,
- listens for a connection that may be initiated by the remote peer, and
- changes its state to Active.

The exact value of the ConnectRetryTimer is a local matter, but it SHOULD be sufficiently large to allow TCP initialization.

Any other event (Events 9-12, 15-28) received in the Idle state does not cause change in the state of the local system.

**Test Setup:**




**Procedure:**

*Part A: BGP with Active TCP Connection*

1. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
2. Start BGP on the RUT. Check the captured packets.

*Part B: BGP Router with Passive TCP Connection*

3. TR1 is unplugged.
4. Restart the RUT. TR1 is no longer in the RUT's ARP table.
5. Check the captured packets.

*Part C: Other Event on BGP Router*

6. TR1 is plugged in.
7. Restart BGP on the RUT.
8. Before the RUT sends BGP packets to initiate a connection, TR1 sends a NOTIFICATION message to the RUT.
9. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a TCP SYN segment and wait for ACK from TR1. After the RUT sends the TCP SYN segment, it transitions to state Connect.
- In Part B, the RUT should not send a TCP SYN segment and should transition to state Active.
- In Part C, the RUT should remain in state Idle.

**Possible Problems:** None.

**Test BGP_CONF.2.2: Connect State**

**Purpose:** To verify the correct functionality of a BGP router in state Connect.

**References:** [RFC 4271] – Section 8.2.2

**Last Modification:** January 3, 2005

**Discussion:** In this state, BGP FSM is waiting for the TCP connection to be completed.

In response to a ManualStop event (Event 2), the local system:
- drops the TCP connection,
- releases all BGP resources,
- sets ConnectRetryCounter to zero,
- stops the ConnectRetryTimer and sets ConnectRetryTimer to zero, and
- changes its state to Idle.

In response to the ConnectRetryTimer_Expires event (Event 9), the local system:
- drops the TCP connection,
- restarts the ConnectRetryTimer,
- stops the DelayOpenTimer and resets the timer to zero,
- initiates a TCP connection to the other BGP peer,
- continues to listen for a connection that may be initiated by the remote BGP peer, and
- stays in Connect state.

If the DelayOpenTimer_Expires event (Event 12) occurs in the Connect state, the local system:
- sends an OPEN message to its peer,
- sets the HoldTimer to a large value, and
- changes its state to OpenSent.

If the TCP connection succeeds (Event 16 or Event 17), the local system checks the DelayOpen attribute prior to processing. If the DelayOpen attribute is set to TRUE, the local system:
- stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,
- sets the DelayOpenTimer to the initial value, and
- stays in the Connect state.

If the DelayOpen attribute is set to FALSE, the local system:
- stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,
- completes BGP initialization
- sends an OPEN message to its peer,
- sets HoldTimer to a large value, and
- changes its state to OpenSent.

A HoldTimer value of 4 minutes is suggested.

If the DelayOpenTimer is not running, the local system:
- stops the ConnectRetryTimer to zero,
- drops the TCP connection,
- releases all BGP resources, and
- changes its state to Idle.

If an OPEN message is received while the DelayOpenTimer is running (Event 20), the local system:

- stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,
- completes the BGP initialization,
- stops and clears the DelayOpenTimer (sets the value to zero),
- sends an OPEN message,
- sends a KEEPALIVE message,
- and changes its state to OpenConfirm.

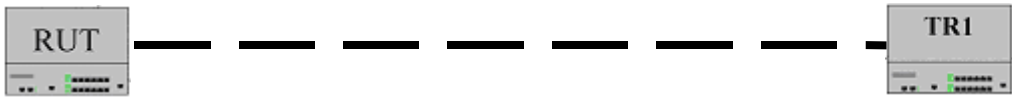
If BGP message header checking detects an error (Event 21) or OPEN message checking detects an error (Event 22) (see section 6.2), the local system:

- stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1, (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

In response to any other events (Events 8,10-11,13,19,23, 25-28) the local system:

- if the ConnectRetryTimer is running, stops and resets the ConnectRetryTimer (sets to zero),
- if the DelayOpenTimer is running, stops and resets the DelayOpenTimer (sets to zero),
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- performs peer oscillation damping if the DampPeerOscillations attribute is set to True, and
- changes its state to Idle.

**Test Setup:**




**Procedure:**

*Part A: BGP Router with Delay Open Flag Set*

1. Set the Delay Open Flag on the RUT. Set Open Delay Timer to n seconds.
2. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
3. BGP is started on the RUT and TR1.
4. TR1 sends a TCP SYN/ACK in response to the RUT's TCP SYN. TR1 does not send an OPEN message. Check the captured packets.

*Part B: BGP Router with Delay Open Timer Expires*

5. Wait for n seconds. Check the captured packets.

*Part C: OPEN Message Received*

6. BGP is restarted on the RUT and TR1. The Delay Open Flag is still set on the RUT.
7. TR1 sends an OPEN message before the Delay Open Timer expires.
8. Check the captured packets.

*Part D: BGP Router with Delay Open Flag not Set*

9. Configure the RUT so that the Delay Open Flag is not set.
10. BGP is restarted on the RUT and TR1.
11. TR1 sends a TCP SYN/ACK in response to the RUT's TCP SYN. Check the captured packets.

*Part E: Manual Stop Event on BGP Router*

12. BGP is restarted on the RUT and TR1. The Delay Open Flag is set on the RUT.
13. Before either the RUT or TR1 send an OPEN message, stop BGP on the RUT.
14. Check the captured packets.

*Part F: TCP Fails with Delay Open Flag Set*

15. Set the Delay Open Flag on the RUT with the Open Delay Timer set to n seconds.
16. BGP is restarted on the RUT and TR1.
17. TR1 sends a TCP SYN/ACK in response to the RUT's TCP SYN. Before the Open Delay Timer expires, TR1 sends a TCP FIN.
18. Check the captured packets.

*Part G: BGP Router's TCP Retransmission Timer Expires*

19. The Delay Open Flag is no longer set on the RUT. Statically configure the IP address of TR1 into the ARP table on the RUT.
20. TR1 is unplugged.
21. Start BGP on the RUT.
22. Check the captured packets.

*Part H: BGP Router's ConnectRetry Timer Expires*

23. Restart the RUT. Configure the RUT with a peer that is not on the network.
24. Statically configure the IP address of the new peer into the RUT's ARP table.
25. Check the captured packets.

*Part I: Other Event on BGP Router*

26. TR1 is plugged in.
27. Set the Delay Open Flag on the RUT. Set Open Delay Timer to n seconds.
28. BGP is restarted the RUT and TR1.
29. After TR1 sends a TCP SYN/ACK and before the Open Delay Timer Expires, TR1 sends a KeepAlive Message.
30. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should remain in state Connect.
- In Part B, the RUT should send an OPEN message to TR1 and transition to state OpenSent.
- In Part C, the RUT should send an OPEN message and transition to state OpenConfirm.
- In Part D, the RUT should acknowledge the TCP SYN/ACK segment from TR1, and then send an OPEN message (transitions to state OpenSent).
- In Part E, the RUT should transition to state Idle.
- In Part F, the RUT should transition to state Active.
- In Part G, the RUT should send a TCP SYN, which is not acknowledged. When the TCP retransmission timer expires, the RUT should transition to state Idle.
- In Part H, the RUT should send a TCP SYN. When the ConnectRetry timer expires, the RUT should resend a TCP SYN segment and remains in state Connect.
- In Part I, the RUT should transition to Idle state.

**Possible Problems:** None.

**Test BGP_CONF.2.3: Active State**

**Purpose:** To verify the correct functionality of a BGP router in state Active.

**References:** [RFC 4271] – Section 8.2.2

**Last Modification:** January 3, 2005

**Discussion:** In this state BGP FSM is trying to acquire a peer by listening for and accepting a TCP connection.

In response to a ManualStop event (Event 2), the local system:
- If the DelayOpenTimer is running and the SendNOTIFICATIONwithoutOPEN session attribute is set, the local system sends a NOTIFICATION with a Cease,
- releases all BGP resources including stopping the DelayOpenTimer
- drops the TCP connection,
- sets ConnectRetryCounter to zero,
- stops the ConnectRetryTimer and sets the ConnectRetryTimer to zero, and
- changes its state to Idle.

In response to a ConnectRetryTimer_Expires event (Event 9), the local system:
- restarts the ConnectRetryTimer (with initial value),
- initiates a TCP connection to the other BGP peer,
- continues to listen for TCP connection that may be initiated by remote BGP peer, and
- changes its state to Connect.

If the local system receives a DelayOpenTimer_Expires event (Event 12), the local system:
- sets the ConnectRetryTimer to zero,
- stops and clears the DelayOpenTimer (set to zero),
- completes the BGP initialization,
- sends the OPEN message to its remote peer,
- sets its hold timer to a large value, and
- changes its state to OpenSent.

A HoldTimer value of 4 minutes is also suggested for this state transition.

In response to a TCP connection succeeding (Event 16 or Event 17), the local system checks the DelayOpen optional attribute prior to processing.

If the DelayOpen attribute is set to TRUE, the local system:
- stops the ConnectRetryTimer and sets the ConnectRetryTimer to zero,
- sets the DelayOpenTimer to the initial value (DelayOpenTime), and
- stays in the Active state.

If the DelayOpen attribute is set to FALSE, the local system:
- sets the ConnectRetryTimer to zero,
- completes the BGP initialization,
- sends the OPEN message to its peer,
- sets its HoldTimer to a large value, and
- changes its state to OpenSent.

A HoldTimer value of 4 minutes is suggested as a "large value" for the HoldTimer.

If the local system receives a TcpConnectionFails event (Event 18), the local system:

- restarts ConnectRetryTimer (with initial value),
- stops and clears the DelayOpenTimer (sets the value to zero),
- releases all BGP resource,
- increments ConnectRetryCounter by 1,
- optionally performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

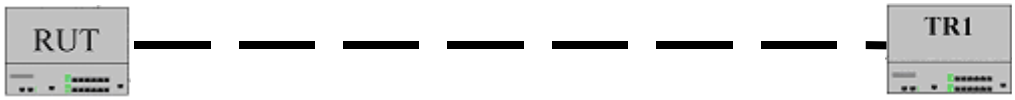
If an OPEN message is received and the DelayOpenTimer is running (Event 20), the local system:

- stops ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,
- stops and clears DelayOpenTimer (sets to zero),
- completes the BGP initialization,
- sends an OPEN message,
- sends a KEEPALIVE message,
- changes its state to OpenConfirm.

In response to any other event (Events 8,10-11,13,19,23,25-28), the local system:

- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by one,
- (optionally) performs peer oscillation damping if
- the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

**Test Setup:**




**Procedure:**

*Part A: BGP Router with Delay Open Flag Set*

1. Set the Delay Open Flag on the RUT. Set Open Delay Timer to n seconds.
2. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
3. Start the RUT and wait until it transitions to state Active.
4. BGP is started on TR1. TR1 sends a TCP SYN segment (initiates the connection).
5. Before the Delay Open Timer Expires, check the captured packets.

*Part B: Delay Open Timer Expires on BGP Router*

6. After the Delay Open Timer expires, check the captured packets.

*Part C: OPEN Message Received*

7. Restart BGP on the RUT. Set the Delay Open Flag on the RUT. Set Open Delay Timer to n seconds.
8. Start the RUT and wait until it transitions to state Active.
9. BGP is started on TR1. TR1 sends a TCP SYN segment (initiates the connection).
10. TR1 sends an OPEN message before the Open Delay Timer expires. Check the captured packets.

*Part D: BGP Router with Delay Open Flag not Set*

11. The Delay Open flag is no longer set on the RUT.
12. Start the RUT and wait until it transitions to state Active.

13. BGP is started on TR1. TR1 sends a TCP SYN segment (initiates the connection).
14. Check the captured packets.

*Part E: Manual Stop Event on BGP Router*

15. Restart BGP on the RUT and wait until it transitions to state Active.
16. Stop BGP on the RUT.
17. Check the captured packets.

*Part F: ConnectRetry Timer Expires on BGP Router*

18. TR1 is unplugged.
19. Restart the RUT and wait until it transitions to state Active.
20. Check the captured packets.

*Part G: TcpConnectionFails Event on BGP Router*

21. Restart BGP on the RUT and wait until it transitions to state Active.
22. TR1 sends a TCP SYN immediately followed by a TCP FIN segment.
23. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should remain in state Active.
- In Part B, the RUT should complete the connection initiated by TR1 by sending an OPEN message and transitioning to state OpenSent.
- In Part C, the RUT should send an OPEN message and transition to state OpenConfirm.
- In Part D, the RUT should complete the connection initiated by TR1. The RUT should send an OPEN message and transition to state OpenSent.
- In Part E, the RUT should transition to state Idle. The RUT should also delete all BGP routes from its routing table.
- In Part F, when the ConnectRetry timer fires the RUT should transition to state Connect.
- In Part G, the RUT should transition to state Idle.

**Possible Problems:** None.

**Test BGP_CONF.2.4: OpenSent State**

**Purpose:** To verify the correct functionality of a BGP router in state OpenSent.

**References:** [RFC 4271] – Section 8.2.2

**Last Modification:** February 3, 2003

**Discussion:** In this state BGP FSM waits for an OPEN message from its peer.

If a ManualStop event (Event 2) is issued in OpenSent state, the local system:
- sends the NOTIFICATION with a cease,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- sets the ConnectRetryCounter to zero, and
- changes its state to Idle.

If the HoldTimer_Expires (Event 10), the local system:
- sends a NOTIFICATION message with error code Hold Timer Expired,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

If a TcpConnectionFails event (Event 18) is received, the local system:
- closes the BGP connection,
- restarts the ConnectRetryTimer,
- continues to listen for a connection that may be initiated by the remote BGP peer, and
- changes its state to Active.

When an OPEN message is received, all fields are checked for correctness. If there are no errors in the OPEN message (Event 19), the local system:
- resets the DelayOpenTimer to zero,
- sets the BGP ConnectRetryTimer to zero,
- sends a KEEPALIVE message, and
- sets a KeepaliveTimer (via the text below)
- sets the HoldTimer according to the negotiated value (see Section 4.2),
- changes its state to OpenConfirm.

If the BGP message header checking (Event 21) or OPEN message check detects an error (Event 22)(see Section 6.2), the local system:
- sends a NOTIFICATION message with appropriate error code,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,

- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is TRUE, and
- changes its state to Idle.

If a connection in OpenSent state is determined to be the connection that must be closed, an OpenCollisionDump (Event 23) is signaled to the state machine. If such an event is received in OpenSent state, the local system:

- sends a NOTIFICATION with a Cease
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

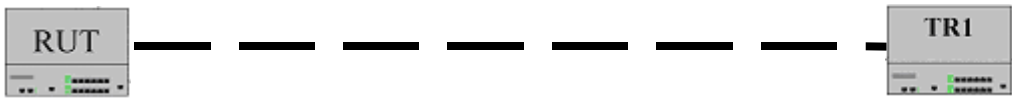
If a NOTIFICATION message is received with a version error (Event 24), the local system:

- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection, and
- changes its state to Idle.

In response to any other event (Events 9, 11-13,20,25-28), the local system:

- sends the NOTIFICATION with the Error Code Finite state machine error,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

**Test Setup:**




**Procedure:**

*Part A: BGP Router Receives Correct OPEN message*

1. RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1.
3. After the RUT sends an OPEN message, TR1 sends a correct OPEN message.
4. Check the captured packets.

*Part B: BGP Router Receives Incorrect OPEN message*

5. Restart the RUT and TR1.
6. After the RUT sends an OPEN message, TR1 sends an incorrect OPEN message.
7. Check the captured packets.

*Part C: BGP Router Receives TCP Connection Failure*

8. BGP is restarted on the RUT and TR1.

9. After the RUT sends an OPEN message, TR1 sends a TCP FIN segment.
10. Check the captured packets.

*Part D: Hold Timer Expires on BGP Router*

11. BGP is restarted on the RUT and TR1.
12. After the RUT sends an OPEN message, TR1 stops sending any messages.
13. Check the captured packets.

*Part E: BGP Router Receives NOTIFICATION with Version Error*

14. BGP is restarted on the RUT and TR1.
15. After the RUT sends an OPEN message, TR1 sends a NOTIFICATION message with a version error.
16. Check the captured packets.

*Part F: Manual Stop Event on BGP Router*

17. BGP is restarted on the RUT and TR1.
18. After the RUT sends an OPEN message, stop BGP on the RUT.
19. Check the captured packets.

*Part G: Other Event on BGP Router*

20. BGP is restarted on the RUT and TR1.
21. After the RUT sends an OPEN message, TR1 sends an UPDATE message.
22. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a KEEPALIVE message and transition to state OpenConfirm.
- In Part B, the RUT should send a NOTIFICATION message and transition to the state Idle.
- In Part C, the RUT should close the BGP connection, and transition to the state Active.
- In Part D, the RUT should set Hold Time to a large value, possibly 4 minutes or more, when it transitions from state Active to state OpenSent. After Hold Time, the RUT should send a NOTIFICATION with Error Code Hold Timer Expired, and transition to the state Idle.
- In Part E, the RUT should close connection with TRI, and transitionn to state Idle.
- In Part F, the RUT should send a NOTIFICATION message with Error Code Cease and transition to state Idle.
- In Part G, the RUT should send a NOTFICATION message with Error Code Finite State Machine Error, and transition to the state Idle.

**Possible Problems:** None.

**Test BGP_CONF.2.5: OpenConfirm State**

**Purpose:** To verify the correct functionality of a BGP router in state OpenConfirm.

**References:** [RFC 4271] – Section 8.2.2

**Last Modification:** February 3, 2003

**Discussion:** In this state BGP waits for a KEEPALIVE or NOTIFICATION message.

In response to a ManualStop event (Event 2) initiated by the operator, the local system:
- sends the NOTIFICATION message with Cease,
- releases all BGP resources,
- drops the TCP connection,
- sets the ConnectRetryCounter to zero,
- sets the ConnectRetryTimer to zero, and
- changes its state to Idle.

If the HoldTimer_Expires event (Event 10) occurs before a KEEPALIVE message is received, the local system:
- sends the NOTIFICATION message with the error code,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

If the local system receives a KeepaliveTimer_Expires event (Event 11), the system:
- sends a KEEPALIVE message,
- restarts the KeepaliveTimer, and
- remains in OpenConfirmed state.

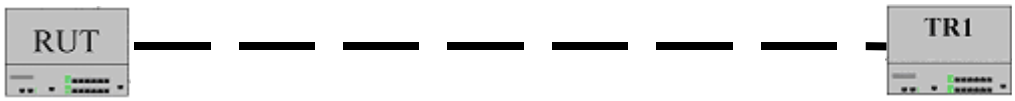
If an OPEN message is received, all fields are checked for correctness. If the BGP message header checking (BGPHeaderErr (Event 21)) or OPEN message check detects an error (see Section 6.2) (BGPOpenMsgErr (Event 22)), the local system:
- sends a NOTIFICATION message with appropriate error code,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

If the local system receives a KEEPALIVE message (KeepAliveMsg (Event 26)), the local system:
- restarts the HoldTimer and
- changes its state to Established.

In response to any other event (Events 9, 12-13, 20, 27-28), the local system:

- sends a NOTIFICATION with a code of Finite State Machine Error,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

**Test Setup:**




**Procedure:**

*Part A: BGP Router receives KEEPALIVE message*

1. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1.
3. After the RUT sends a KEEPALIVE message, TR1 sends a KEEPALIVE message.
4. Check the captured packets.

*Part B: Hold Timer Expires before KEEPALIVE message sent*

5. BGP is restarted on the RUT and TR1.
6. After the RUT sends a KEEPALIVE message, TR1 does not send a KEEPALIVE message for more than Hold Timer interval.
7. Check the captured packets.

*Part C: KEEPALIVE Timer Expires on a BGP Router*

8. Set the Hold Timer to a large value. BGP is restarted on the RUT and TR1.
9. Wait until the KEEPALIVE timer expires on the RUT and check the captured packets.

*Part D: BGP Router Receives NOTIFICATION message*

10. BGP is restarted on the RUT and TR1.
11. After the RUT sends a KEEPALIVE message, TR1 sends a NOTIFICATION message.
12. Check the captured packets.

*Part E: BGP Router Receives TCP Disconnect*

13. BGP is restarted on the RUT and TR1.
14. After the RUT sends a KEEPALIVE message, TR1 sends a TCP FIN segment.
15. Check the captured packets.

*Part F: Manual Stop Event on BGP Router*

16. BGP is restarted on the RUT and TR1.
17. After the RUT sends a KEEPALIVE message, stop BGP on the RUT.
18. Check the captured packets.

*Part G: Other Event on BGP Router*

19. BGP is restarted on the RUT and TR1.
20. After the RUT sends a KEEPALIVE message, TR1 sends an UPDATE message.
21. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should transition to state Established (possibly send UPDATE messages).
- In Part B, the RUT should send a NOTIFICATION message with Error Code Hold Timer Expired and transition to the state Idle.

- In Part C, the RUT should remain in state OpenConfirm.
- In Part D, the RUT should not send any message and should transition to the state Idle.
- In Part E, the RUT should transition to the state Idle.
- In Part F, the RUT should send a NOTIFICATION message with Cease and transition to the state Idle.
- In Part G, the RUT should send a NOTIFICATION message with a code of Finite State Machine Error and transition to state Idle.

**Possible Problems:** None.

**Test BGP_CONF.2.6: Established State**

**Purpose:** To verify the correct functionality of a BGP router in state Established.

**References:** [RFC 4271] – Section 8.2.2

**Last Modification:** February 3, 2003

**Discussion:** In the Established state, the BGP FSM can exchange UPDATE, NOTIFICATION, and KEEPALIVE messages with its peer.

In response to a ManualStop event (initiated by an operator) (Event 2), the local system:
- sends the NOTIFICATION message with Cease,
- sets the ConnectRetryTimer to zero,
- deletes all routes associated with this connection,
- releases BGP resources,
- drops the TCP connection,
- sets ConnectRetryCounter to zero, and
- changes its state to Idle.

If the HoldTimer_Expires event occurs (Event 10), the local system:
- sends a NOTIFICATION message with Error Code Hold Timer Expired,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

If the KeepaliveTimer_Expires event occurs (Event 11), the local system:
- sends a KEEPALIVE message, and
- restarts its KeepaliveTimer unless the negotiated HoldTime value is zero.

Each time the local system sends a KEEPALIVE or UPDATE message, it restarts its KeepaliveTimer, unless the negotiated HoldTime value is zero.

If the local system receives a NOTIFICATION message (Event 24 or Event 25) or a TcpConnectionFails (Event 18) from the underlying TCP, it:
- sets the ConnectRetryTimer to zero,
- deletes all routes associated with this connection,
- releases all the BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- changes its state to Idle.

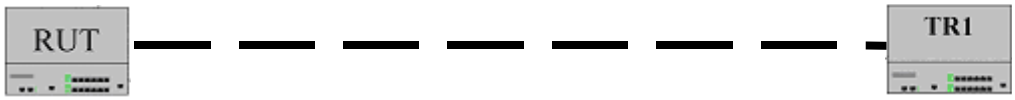
If the local system receives an UPDATE message, and the UPDATE message error handling procedure (see Section 6.3) detects an error (Event 28), the local system:
- sends a NOTIFICATION message with Update error,
- sets the ConnectRetryTimer to zero,
- deletes all routes associated with this connection,

- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

In response to any other event (Events 9, 12-13, 20-22) the local system:

- sends a NOTIFICATION message with Error Code Finite State Machine Error,
- deletes all routes associated with this connection,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the TCP connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

**Test Setup:**




**Procedure:**

*Part A: BGP Router Receives NOTIFICATION Message*

1. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. TR1 sends a NOTIFICATION message.
4. Check the captured packets.

*Part B: BGP Router Receives Incorrect UPDATE Message*

5. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
6. TR1 sends an incorrect UPDATE message.
7. Check the captured packets.

*Part C: BGP Router Receives TCP Disconnect*

8. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
9. TR1 sends a TCP FIN segment.
10. Check the captured packets.

*Part D: Hold Timer Expires before KEEPALIVE Message Sent*

11. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
12. TR1 does not send a KEEPALIVE message for more than Hold Timer interval.
13. Check the captured packets.

*Part E: KEEPALIVE Timer Expires on BGP Router*

14. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
15. TR1 does not send a KEEPALIVE message for more than Keep Alive Timer.
16. Check the captured packets.

*Part F: Manual Stop Event on BGP Router*

17. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
18. Stop BGP on the RUT.
19. Check the captured packets.

*Part G: Other Event on BGP Router*

20. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
21. TR1 sends an OPEN message.
22. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should transition to state Idle.
- In Part B, the RUT should send a NOTIFICATION message with Update error and transition to state Idle.
- In Part C, the RUT should transition to state Idle.
- In Part D, the RUT should send a NOTIFICATION message with Error Code Hold Timer Expired and transition to state Idle.
- In Part E, the RUT should send a KEEPALIVE message and remain in state Established.
- In Part F, the RUT should send a NOTIFICATION message with Error Code cease and transition to state Idle. The RUT should also delete all its routes.
- In Part G, the RUT should send a NOTIFICATION message with Error Code Finite State Machine Error and transition to state Idle.

**Possible Problems:** None.

# 3. ERROR HANDLING

**Overview:**

The following tests are designed to verify the correct handling of erroneous conditions.

**Discussion:**

When an error is detected, a BGP router is required to send a NOTIFICATION message with the appropriate Error Code, Error Subcode and Data fields, and to close the BGP connection. The "BGP connection is closed" means that the TCP connection has been closed and all resources for that BGP connection have been deallocated.

**Test BGP_CONF.3.1: Header Error**

**Purpose:** To verify the correct handling of errors in the BGP packet header.

**References:** [RFC 4271] – Section 6.1

**Last Modification:** July 14, 2004

**Discussion:** All errors detected while processing the Message Header are indicated by sending the NOTIFICATION message with Error Code Message Header Error. The Error Subcode elaborates on the specific nature of the error.

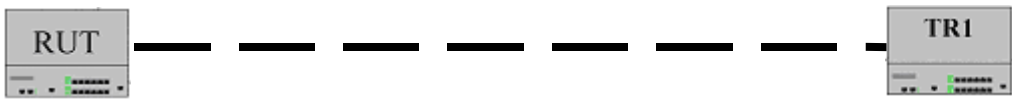
The expected value of the Marker field of the message header is all ones. If the Marker field of the message header is not as expected, then a synchronization error has occurred and the Error Subcode is set to Connection Not Synchronized.

If at least one of the following is true:
- if the Length field of the message header is less than 19 or greater than 4096, or
- if the Length field of an OPEN message is less than the minimum length of the OPEN message, or
- if the Length field of an UPDATE message is less than the minimum length of the UPDATE message, or
- if the Length field of a KEEPALIVE message is not equal to 19, or
- if the Length field of a NOTIFICATION message is less than the minimum length of the NOTIFICATION message,

then the Error Subcode MUST be set to Bad Message Length. The Data field MUST contain the erroneous Length field.

If the Type field of the message header is not recognized, then the Error Subcode MUST be set to Bad Message Type. The Data field MUST contain the erroneous Type field.

**Test Setup:**




**Procedure:**

*Part A: OPEN message with Invalid Marker Field*

1. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1. TR1 sends an OPEN message with the Marker field different than “all ones”.
3. Check the captured packets.

*Part B: KEEPALIVE message with Invalid Marker Field*

4. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
5. TR1 sends a KEEPALIVE message with the Marker field different than “all ones”.
6. Check the captured packets.

*Part C: OPEN Message Length Too Short*

7. BGP is restarted on the RUT and TR1. TR1 sends an OPEN message with the Length field in the message header set to 27 (the minimum length of an OPEN message is 29).

8. Check the captured packets.

*Part D: UPDATE Message Length Too Short*

9. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
10. TR1 sends an UPDATE message with the Length field in the message header set to 21 (the minimum length of an UPDATE message is 23).
11. Check the captured packets.

*Part E: KEEPALIVE Message Length Too Short*

12. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
13. TR1 sends a KEEPALIVE message with the Length field in the message header set to 18.
14. Check the captured packets.

*Part F: KEEPALIVE Message Length Too Long*

15. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
16. TR1 sends a KEEPALIVE message with the Length field in the message header set to 20.
17. Check the captured packets.

*Part G: NOTIFICATION Message Length Too Short*

18. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
19. TR1 sends a NOTIFICATION message with the Length field in the message header set to 20 (the minimum length of a NOTIFICATION message is 21).
20. Check the captured packets.

*Part H: Message Longer than Maximum Length*

21. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
22. TR1 sends an UPDATE message with the Length field in the message header set to 4098 (the maximum length of a BGP message is 4096).
23. Check the captured packets.

*Part I: Unknown Type Field*

24. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
25. TR1 sends a message with the Type field in the message header set to 7.
26. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Connection Not Synchronized.
- In Part B, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Connection Not Synchronized.
- In Part C, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Bad Message Length. The Data field should hold the value 27.
- In Part D, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Bad Message Length. The Data field should hold the value 21.
- In Part E, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Bad Message Length. The Data field should hold the value 18.
- In Part F, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Bad Message Length. The Data field should hold the value 20.
- In Part G, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Bad Message Length. The Data field should hold the value 20.
- In Part H, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Bad Message Length. The Data field should hold the value 4098.
- In Part I, the RUT should send a NOTIFICATION message with error code Message Header Error and error subcode Bad Message Type. The Data field should hold the value 7.

**Possible Problems:** None.

**Test BGP_CONF.3.2: OPEN Message Error**

**Purpose:** To verify the correct handling of errors in OPEN messages.

**References:** [RFC 4271] – Section 6.2

**Last Modification:** February 3, 2003

**Discussion:** All errors detected while processing the OPEN message are indicated by sending the NOTIFICATION message with Error Code OPEN Message Error. The Error Subcode elaborates on the specific nature of the error.

If the version number contained in the Version field of the received OPEN message is not supported, then the Error Subcode is set to Unsupported Version Number. The Data field is a 2-octets unsigned integer, which indicates the largest locally supported version number less than the version the remote BGP peer bid (as indicated in the received OPEN message).

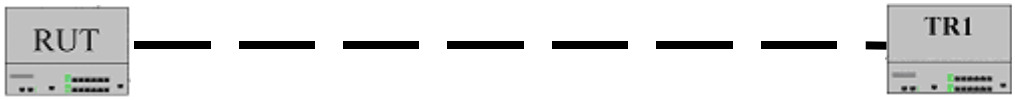
If the Autonomous System field of the OPEN message is unacceptable, then the Error Subcode is set to Bad Peer AS. The determination of acceptable Autonomous System numbers is outside the scope of this protocol.

If the Hold Time field of the OPEN message is unacceptable, then the Error Subcode MUST be set to Unacceptable Hold Time. An implementation MUST reject Hold Time values of one or two seconds. An implementation MAY reject any proposed Hold Time. An implementation which accepts a Hold Time MUST use the negotiated value for the Hold Time.

If the BGP Identifier field of the OPEN message is syntactically incorrect, then the Error Subcode is set to Bad BGP Identifier. Syntactic correctness means that the BGP Identifier field represents a valid IP host address.

If one of the Optional Parameters in the OPEN message is not recognized, then the Error Subcode is set to Unsupported Optional Parameters.

**Test Setup:**




**Procedure:**

*Part A: Unsupported Version Number in Version Field*

1. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1.
3. TR1 sends an OPEN message with the Version field set to 11.
4. Check the captured packets.

*Part B: Unacceptable Autonomous System Field*

5. BGP is restarted on the RUT and TR1.
6. TR1 sends an OPEN message with an unacceptable Autonomous System field.
7. Check the captured packets.

*Part C: Unacceptable Hold Time Field*

8. BGP is restarted on the RUT and TR1.
9. TR1 sends an OPEN message with the Hold time field set to 2 seconds.
10. Check the captured packets.
11. BGP is restarted on the RUT once again.
12. TR1 sends an OPEN message with the Hold time field set to 1 second.
13. Check the captured packets.

*Part D: Syntactically Incorrect BGP Identifier*

14. BGP is restarted on the RUT and TR1.
15. TR1 sends an OPEN message with the BGP Identifier field set to 0.0.0.0.
16. Check the captured packets.

*Part E: Error Handling of BGP OPEN message*

17. BGP is restarted on the RUT and TR1.
18. TR1 sends an OPEN message with optional parameters, with Parameter Type set to 11.
19. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code OPEN Message Error and error subcode Unsupported Version Number. The Data field should hold the largest version number supported by the RUT.
- In Part B, the RUT should send a NOTIFICATION message with error code OPEN Message Error and error subcode Bad Peer AS.
- In Part C, the RUT should send a NOTIFICATION message with error code OPEN Message Error and error subcode Unacceptable Hold Time after receiving each OPEN Message.
- In Part D, the RUT should send a NOTIFICATION message with error code OPEN Message Error and error subcode Bad BGP Identifier.
- In Part E, the RUT should send a NOTIFICATION message with error code OPEN Message Error and error subcode Unsupported Optional Parameters.

**Possible Problems:** None.

**Test BGP_CONF.3.3: UPDATE Message Length Error**

**Purpose:** To verify the correct handling of length errors in UPDATE messages.

**References:** [RFC 4271] – Section 6.3

**Last Modification:** July 14, 2004

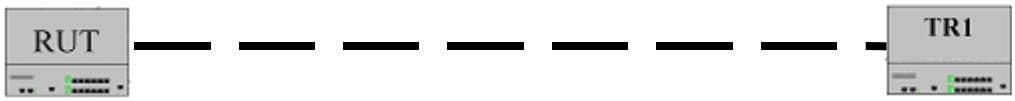
**Discussion:** All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

Error checking of an UPDATE message begins by examining the path attributes. If the Withdrawn Routes Length or Total Attribute Length is too large (i.e., if Withdrawn Routes Length + Total Attribute Length + 23 exceeds the message Length), then the Error Subcode is set to Malformed Attribute List.

If any recognized attribute has Attribute Flags that conflict with the Attribute Type Code, then the Error Subcode is set to Attribute Flags Error. The Data field contains the erroneous attribute (type, length and value).

If any recognized attribute has Attribute Length that conflicts with the expected length (based on the attribute type code), then the Error Subcode is set to Attribute Length Error. The Data field contains the erroneous attribute (type, length and value).

**Test Setup:**




**Procedure:**

*Part A: Unfeasible Route Length*

1. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. TR1 sends an UPDATE message with the Unfeasible Routes Length set to 4090.
4. Check the captured packets.

*Part B: Total Attribute Length Too Large*

5. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
6. TR1 sends an UPDATE message with the Total Attribute Length set to 4090.
7. Check the captured packets.

*Part C: Conflicting Attribute Flag and Type Code*

8. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
9. TR1 sends an UPDATE message with the high-order bit of the Attribute Flags for the ORIGIN attribute set to 1.
10. Check the captured packets.

*Part D: Conflicting Attribute Length and Type Code*

11. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
12. TR1 sends an UPDATE message with conflicting Attribute Length/Attribute Type.
13. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Malformed Attribute List.
- In Part B, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Malformed Attribute List.
- In Part C, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Attribute Flags Error. The Data field should contain the erroneous attribute (type, length and value).
- In Part D, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Attribute Length Error. The Data field should contain the erroneous attribute (type, length and value).

**Possible Problems:** None.

**Test BGP_CONF.3.4: Well-Known Attribute Error**

**Purpose:** To verify the correct handling of well-known attribute errors in UPDATE messages.

**References:** [RFC 4271] – Section 6.3

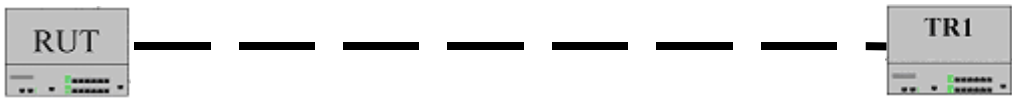
**Last Modification:** February 3, 2003

**Discussion:** All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

If any of the mandatory well-known attributes are not present, then the Error Subcode is set to Missing Well-known Attribute. The Data field contains the Attribute Type Code of the missing well-known attribute.

If any of the mandatory well-known attributes are not recognized, then the Error Subcode is set to Unrecognized Well-known Attribute. The Data field contains the unrecognized attribute (type, length and value).

**Test Setup:**




**Procedure:**

*Part A: Missing Well-Known Attribute*

1. The RUT and TR1 are configured as external peers, where the TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. TR1 sends an UPDATE message without the ORIGIN attribute.
4. Check the captured packets.

*Part B: Unrecognized Well-Known Attribute*

5. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
6. TR1 sends an UPDATE message with the high-order bit of the Attribute Flags field set to 0 (denoting a well-known attribute) and Attribute Type Code set to 11.
7. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Missing Well-known Attribute. The Data field should contain the value 1 (the Attribute Type Code for the ORIGIN attribute).
- In Part B, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Unrecognized Well-known Attribute. The Data field should contain the erroneous attribute (type, length and value).

**Possible Problems:** None.

**Test BGP_CONF.3.5: ORIGIN Attribute Error**

**Purpose:** To verify the correct handling of ORIGIN attribute errors in UPDATE messages.

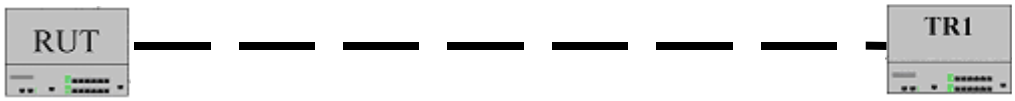
**References:** [RFC 4271] – Section 6.3

**Last Modification:** February 3, 2003

**Discussion:** All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

If the ORIGIN attribute has an undefined value, then the Error Subcode is set to Invalid Origin Attribute. The Data field contains the unrecognized attribute (type, length and value).

**Test Setup:**




**Procedure:**

*Part A: Undefined ORIGIN Attribute*

1. The RUT and TR1 are configured as external peers, where the TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. TR1 sends an UPDATE message with the ORIGIN attribute undefined (set to 5).
4. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Invalid Origin Attribute. The Data field should contain the erroneous attribute (type, length and value).

**Possible Problems:** None.

**Test BGP_CONF.3.6: NEXT_HOP Attribute Error**

**Purpose:** To verify the correct handling of NEXT_HOP attribute errors in UPDATE messages.

**References:** [RFC 4271] – Section 6.3

**Last Modification:** February 3, 2003

**Discussion:** All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

If the NEXT_HOP attribute field is syntactically incorrect, then the Error Subcode is set to Invalid NEXT_HOP Attribute. The Data field contains the incorrect attribute (type, length and value). Syntactic correctness means that the NEXT_HOP attribute represents a valid IP host address.

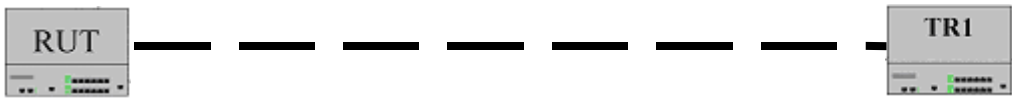
The IP address in the NEXT_HOP must meet the following criteria to be considered semantically correct:

a) It must not be the IP address of the receiving speaker

b) In the case of an EBGP where the sender and receiver are one IP hop away from each other, either the IP address in the NEXT_HOP must be the sender's IP address (that is used to establish the BGP connection), or the interface associated with the NEXT_HOP IP address must share a common subnet with the receiving BGP speaker.

If the NEXT_HOP attribute is semantically incorrect, the error should be logged, and the route should be ignored. In this case, no NOTIFICATION message should be sent, and connection should not be closed.

**Test Setup:**




**Procedure:**

*Part A: Syntactically Incorrect NEXT_HOP Attribute*

1. The RUT and TR1 are configured as external peers, where the TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. TR1 sends an UPDATE message with the NEXT_HOP attribute set to 224.0.0.5.
4. Check the captured packets.

*Part B: Semantically Incorrect NEXT_HOP Attribute*

5. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
6. TR1 sends an UPDATE message with the NEXT_HOP attribute set to an address that is NOT on the common subnet shared by the two routers.
7. Check the captured packets.

*Part C: NEXT_HOP Set to Receiving Speaker's IP Address*

8. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
9. TR1 sends an UPDATE message with the NEXT_HOP attribute set to the RUT's address on the common subnet.
10. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Invalid NEXT_HOP Attribute. The Data field should contain the erroneous attribute (type, length and value).
- In Part B, this is a semantically incorrect NEXT_HOP. The RUT should ignore the UPDATE, and should log the error. The connection should NOT be closed.
- In Part C, this is a semantically incorrect NEXT_HOP. The RUT should ignore the UPDATE, and should log the error. The connection should NOT be closed.

**Possible Problems:** None.

## Test BGP_CONF.3.7: AS_PATH Attribute Error

**Purpose:** To verify the correct handling of AS_PATH attribute errors in UPDATE messages.

**References:** [RFC 4271] – Section 6.3

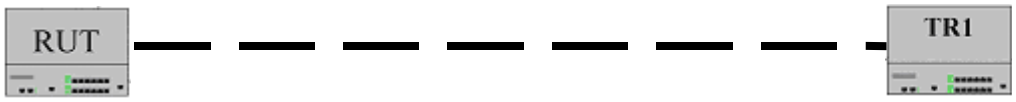
**Last Modification:** February 3, 2003

**Discussion:** All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

The AS_PATH attribute is checked for syntactic correctness. If the path is syntactically incorrect, then the Error Subcode is set to Malformed AS_PATH.

If the UPDATE message is received from an external peer, the local system MAY check whether the leftmost AS in the AS_PATH attribute is equal to the autonomous system number of the peer than sent the message. If the check determines that this is not the case, the Error Subcode is set to Malformed AS_PATH.

**Test Setup:**




**Procedure:**

*Part A: Syntactically Incorrect AS_PATH Attribute*

1. The RUT and TR1 are configured as external peers, where the TR1 is a packet generator simulating a BGP router.
2. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
3. TR1 sends an UPDATE message with the path segment type set to 11.
4. Check the captured packets.

*Part B: Malformed AS_PATH*

5. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
6. TR1 sends an UPDATE message. The leftmost AS in its AS_PATH Attribute is not set to its own Autonomous System.
7. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Malformed AS_PATH.
- In Part B, the RUT should either send a NOTIFICATION message with error code UPDATE Message Error and error subcode Malformed AS_PATH or install the UPDATE.

**Possible Problems:** None.

**Test BGP_CONF.3.8: NLRI Field Error**

**Purpose:** To verify the correct handling of NLRI field errors in UPDATE messages.

**References:** [RFC 4271] – Section 6.3

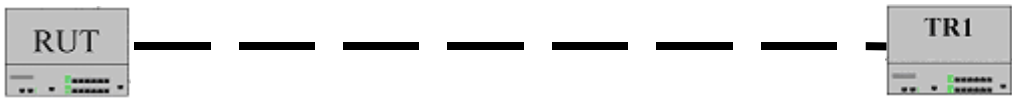
**Last Modification:** February 3, 2003

**Discussion:** All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

The NLRI field in the UPDATE message is checked for syntactic validity. If the field is syntactically incorrect, then the Error Subcode is set to Invalid Network Field. If a prefix in the NLRI field is semantically incorrect (e.g., an unexpected multicast IP address), an error should be logged locally, and the prefix should be ignored.

An UPDATE message that contains correct path attributes, but no NLRI, shall be treated as a valid UPDATE message.

**Test Setup:**




**Procedure:**

*Part A: Semantically Incorrect NLRI Field*

1. The RUT and TR1 as BGP peers, where the TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. TR1 sends an UPDATE message with the NLRI field containing the value 224.0.0.5.
4. Check the captured packets.

*Part B: NLRI Field Missing*

5. BGP is restarted on the RUT and TR1. Wait until they establish the connection.
6. TR1 sends an UPDATE message without the NLRI field.
7. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should not send a NOTIFICATION message, because a prefix in the NLRI field is semantically incorrect (e.g., an unexpected multicast IP address) and just ignore the prefix.
- In Part B, the RUT should accept the UPDATE message from TR1. Since there is no NLRI, no route will be seen on the RUT.

**Possible Problems:** None.

**Test BGP_CONF.3.9: Miscellaneous Attribute Errors**

**Purpose:** To verify the correct handling of miscellaneous errors in UPDATE messages.

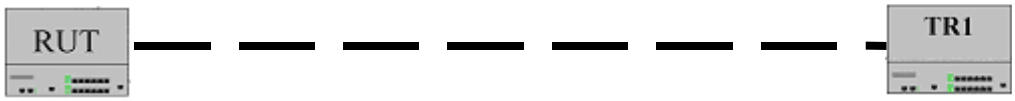
**References:** [RFC 4271] – Section 6.3

**Last Modification:** June 7, 2004

**Discussion:** All errors detected while processing the UPDATE message are indicated by sending the NOTIFICATION message with Error Code UPDATE Message Error. The error subcode elaborates on the specific nature of the error.

If any attribute appears more than once in the UPDATE message, then the Error Subcode is set to Malformed Attribute List.

**Test Setup:**




**Procedure:**

*Part A: Multiple Instances of Path Attribute*

1. The RUT and TR1 are configured as external peers, where the TR1 is a packet generator simulating a BGP router.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. TR1 sends an UPDATE message containing multiple instances of the ORIGIN Attribute.
4. Check the captured packets.

**Observable Results:**

- In Part A, the RUT should send a NOTIFICATION message with error code UPDATE Message Error and error subcode Malformed Attribute List.

**Possible Problems:** None.

**Test BGP_CONF.3.10: Hold Timer Expired**

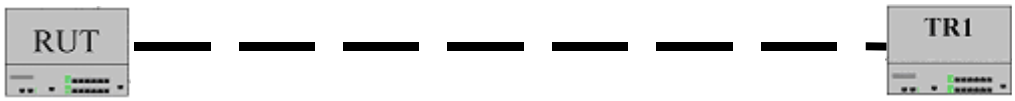
**Purpose:** To verify a BGP router's behavior when event Hold Timer Expired occurs.

**References:** [RFC 4271] – Section 6.5

**Last Modification:** August 21, 2001

**Discussion:** If a system does not receive successive KEEPALIVE and/or UPDATE and/or NOTIFICATION messages within the period specified in the Hold Time field of the OPEN message, then the NOTIFICATION message with Hold Timer Expired Error Code must be sent and the BGP connection closed.

**Test Setup:**




**Procedure:**

*Part A: Hold Timer Expired*

1. The RUT and TR1 are configured as external peers.
2. BGP is started on the RUT and TR1. Wait until they establish the connection.
3. TR1 ceases transmission of keep-alive messages. Check the captured packets.

**Observable Results:**

- In Part A, after the Hold Time period, the RUT should send a NOTIFICATION message with the error code Hold Timer Expired. The RUT should close the connection to TR1.

**Possible Problems:** None.

**Test BGP_CONF.3.11: Connection Collision Detection**

**Purpose:** To verify that, in the case of a connection collision, a BGP router properly closes one of the connections or accepts connection closure.

**References:** [RFC 4271] – Section 6.8

**Last Modification:** August 21, 2001

**Discussion:** If a pair of BGP speakers try simultaneously to establish a TCP connection to each other, then two parallel connections between this pair of speakers might well be formed. We refer to this situation as connection collision. Clearly, one of these connections must be closed.

Based on the value of the BGP Identifier a convention is established for detecting which BGP connection is to be preserved when a collision does occur. The convention is to compare the BGP Identifiers of the peers involved in the collision and to retain only the connection initiated by the BGP speaker with the higher-valued BGP Identifier.

Upon receipt of an OPEN message, the local system must examine all of its connections that are in the OpenConfirm state. A BGP speaker may also examine connections in an OpenSent state if it knows the BGP Identifier of the peer by means outside of the protocol. If among these connections there is a connection to a remote BGP speaker whose BGP Identifier equals the one in the OPEN message, then the local system performs the following collision resolution procedure:

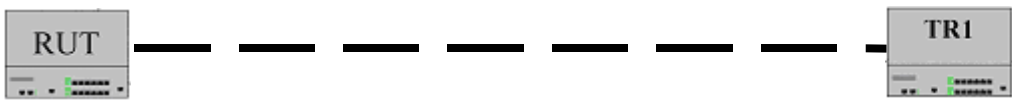
1. The BGP Identifier of the local system is compared to the BGP Identifier of the remote system (as specified in the OPEN message). Comparing BGP Identifiers is done by treating them as (4-octet long) unsigned integers.
2. If the value of the local BGP Identifier is less than the remote one, the local system closes BGP connection that already exists (the one that is already in the OpenConfirm state), and accepts BGP connection initiated by the remote system.
3. Otherwise, the local system closes newly created BGP connection (the one associated with the newly received OPEN message), and continues to use the existing one (the one that is already in the OpenConfirm state).

Unless allowed via configuration, a connection collision with an existing BGP connection that is in Established state causes closing of the newly created connection.

Note that a connection collision cannot be detected with connections that are in Idle, or Connect, or Active states.

Closing the BGP connection (that results from the collision resolution procedure) is accomplished by sending the NOTIFICATION message with the Error Code Cease.

**Test Setup:**



**Procedure:**

*Part A: BGP Router with Higher BGP Identifier in OpenConfirm state*

1. The RUT and TR1 are configured as external peers, where the TR1 is a packet generator simulating a BGP router.
2. Assign the RUT a BGP identifier higher than TR1.
3. Have the RUT initiate a connection to TR1, and keep the connection in state OpenConfirm (TR1 sends an OPEN message, but doesn't send a KEEPALIVE message).
4. TR1 initiates a connection with the RUT. Check the packets.

*Part B: BGP Router with Lower BGP Identifier in OpenConfirm state*

5. Assign the RUT a BGP identifier lower than TR1. BGP is stopped on the routers.
6. Have the RUT initiate a connection to TR1, and keep the connection in state OpenConfirm (TR1 sends an OPEN message, but doesn't send a KEEPALIVE message).
7. TR1 initiates a connection to the RUT. Check the packets.

*Part C: Connection Collision in Established state*

8. BGP is restarted on the RUT and TR1. A BGP connection is established.
9. TR1 initiates a connection to the RUT.
10. Check the packets.

**Observable Results:**

- In Part A, OPEN messages are first exchanged by the two BGP routers as they both open parallel connections with each other. After this is detected, a NOTIFICATION message should be sent with the Error Code Cease, closing the connection initiated by TR1.
- In Part B, OPEN messages are first exchanged by the two BGP routers as they both open parallel connections with each other. After this is detected, a NOTIFICATION message should be sent with the Error Code Cease, closing the connection initiated by the RUT.
- In Part C, the second connection should be closed, even if TR1's BGP Identifier is higher than the RUT's.

**Possible Problems:** None.

# 4. EXTENSIONS

**Overview:**

The following tests are designed to verify the behavior of BGP routers that implement the following extensions:

- Confederations;
- Route Reflection;
- Communities;
- Capabilities Negotiation;
- Multiprotocol Extensions;
- Carrying Label Information.

**Discussion:**

Confederations: This is an extension to BGP which may be used to create a confederation of autonomous systems which is represented as one single autonomous system to BGP peers external to the confederation. The intention of this extension is to aid in policy administration and reduce the management complexity of maintaining a large autonomous system.

Route Reflection: BGP deployments are configured such that that all BGP routers within a single AS must be fully meshed so that any external routing information must be re-distributed to all other routers within that AS. This represents a serious scaling problem that has been well documented with several alternatives proposed. Route Reflection is such a method, where one BGP router “reflects” routes to group of “clients”, which do not have to be fully meshed. The “route reflector” still has to be fully meshed with non-client BGP routers.

Communities: BGP supports transit policies via controlled distribution of routing information. Control over the distribution of routing information is presently based on either IP address prefixes or on the value of the AS_PATH attribute (or part of it). To facilitate and simplify the control of routing information the “communities” attribute is introduced to group destinations, so that the routing decision can also be based on the identity of a group. Such a scheme is expected to significantly simplify a BGP router's configuration that controls distribution of routing information.

Capabilities Negotiation: Currently BGP requires that when a BGP speaker receives an OPEN message with one or more unrecognized Optional Parameters, the speaker must terminate BGP peering. The Capabilities Negotiation extension is expected to facilitate introduction of new capabilities in BGP by providing graceful capability negotiation without requiring that BGP peering be terminated.

Multiprotocol Extensions: Currently BGP is capable of carrying routing information only for IPv4. Multiprotocol Extensions allow BGP to carry routing information for multiple Network Layer protocols (e.g., IPv6, IPX, etc...). The extensions are backward compatible - a router that supports the extensions can interoperate with a router that doesn't support the extensions.

Carrying Label Information: The Multiprotocol Label Switching (MPLS) architecture identifies situations in which the mapping between a label and a route must be distributed between BGP peers. The label mapping information is included in the BGP Update message that is used to distribute the route. This is done by utilizing BGP Multiprotocol Extensions attribute.

**Test BGP_CONF.4.1: Confederations (Propagating an UPDATE)**

**Purpose:** To verify the correct behavior of a BGP router participating in an AS confederation, when it propagates an UPDATE.

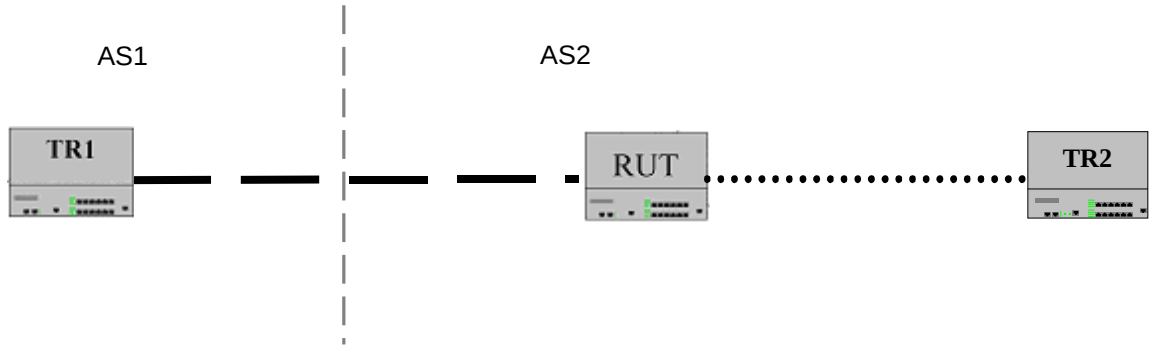
**References:** [RFC3065] – Section 6

**Last Modification:** February 11, 2002

**Discussion:** When a BGP router belongs to an AS which is part of a confederation, the rules to propagate UPDATE messages to a certain peer are:

- if a peer is in the same AS, the BGP router shall not modify the AS_PATH;
- if a peer is in a neighboring AS that is part of the confederation, then:
  - if the first path segment of AS_PATH is not AS_CONFED_SEQUENCE, the BGP router shall prepend a new path segment of type AS_CONFED_SEQUENCE, including its own confederation identifier in that segment;
  - if the first path segment of AS_PATH is AS_CONFED_SEQUENCE, the BGP router shall prepend its AS number to the AS_PATH;
- if a peer is in a neighboring AS that is not part of the confederation, first the AS_PATH must be "sanitized". This means that, if the AS_PATH starts with an AS_CONFED_SEQUENCE segment, that segment and any immediately following segments of type AS_CONFED_SET are removed. Then:
  - if the first path segment is of type AS_SEQUENCE, the BGP router shall prepend its own confederation identifier as the last element (the leftmost position);
  - if the first path segment is of type AS_SET or if the AS_PATH is empty, the BGP router shall prepend a new path segment of type AS_SEQUENCE, including its own confederation identifier in that segment.

**Test Setup:**




**Procedure:**

*Part A: UPDATE Propagation within AS*

1. The RUT and TR1 are configured as external peers. The RUT and TR2 are configured as internal peers.
2. BGP is started on the routers. Wait until they establish the connections.
3. TR1 sends an UPDATE for a new route to the RUT. Check the packets.

*Part B: UPDATE Propagation within Confederation to Router in Same AS*

4. AS1 and AS2 are part of a confederation, with confederation identifier 11.
5. BGP is restarted on the routers. Wait until they establish the connections.

6. TR2 sends an UPDATE for a new route to the RUT. Check the packets.

*Part C: UPDATE Propagation within Confederation to Router in Different AS*

7. TR2 is moved to AS3. AS3 is part of confederation 11.
8. BGP is restarted on the routers. Wait until they establish the connections.
9. TR2 sends an UPDATE for a new route to the RUT. The first AS_PATH segment in the UPDATE received from TR2 is (AS_CONFED_SEQUENCE/AS3). Check the packets.

*Part D: UPDATE from Router Outside Confederation*

10. AS1 is now outside the confederation.
11. BGP is restarted on the routers. Wait until they establish the connections.
12. TR2 sends an UPDATE for a new route to the RUT. The first AS_PATH segment in the UPDATE received from TR2 is (AS_CONFED_SEQUENCE/AS3). Check the packets.

*Part E: Single BGP Router in Confederation*

13. AS3 is now outside the confederation.
14. BGP is restarted on the routers. Wait until they establish the connections.
15. TR2 sends an UPDATE for a new route to the RUT. The first AS_PATH segment in the UPDATE received from TR2 is (AS_SEQUENCE/AS3). Check the packets.

*Part F: UPDATE message with only AS_CONFED_SEQUENCE in AS_PATH*

16. TR2 is replaced with a packet generator simulating a BGP router.
17. BGP is restarted on the routers. Wait until they establish the connections.
18. TR2 sends an UPDATE for a new route to the RUT. The AS_PATH attribute in the UPDATE message is set to (AS_CONFED_SEQUENCE/AS3). Check the packets.

*Part G: UPDATE message with AS_CONFED_SEQUENCE* and *AS_SET in the AS_PATH*

19. BGP is restarted on the routers. Wait until they establish the connections.
20. TR2 sends an UPDATE for a new route to the RUT. The AS_PATH attribute in the UPDATE message is set to (AS_CONFED_SEQUENCE/AS3), (AS_SET/AS4, AS5). Check the packets.

**Observable Results:**

- In Part A, the RUT should propagate the UPDATE to TR2. The AS_PATH attribute should be unchanged.
- In Part B, the RUT should propagate the UPDATE to TR1. The received AS_PATH should be empty. The RUT should prepend a new path segment of type AS_CONFED_SEQUENCE, with value 11. The updated AS_PATH shall be (AS_CONFED_SEQUENCE/11).
- In Part C, the received AS_PATH should be (AS_CONFED_SEQUENCE/AS3). The RUT should propagate the UPDATE to TR1. The updated AS_PATH should be (AS_CONFED_SEQUENCE/AS2, AS3).
- In Part D, the received AS_PATH is (AS_CONFED_SEQUENCE/AS3). The RUT should "sanitize" the AS_PATH in the received update, leaving an empty AS_PATH. Then it should propagate the UPDATE to TR1, with AS_PATH set to (AS_SEQUENCE/11).
- In Part E, the received AS_PATH should be (AS_SEQUENCE/AS3). The RUT should prepend its confederation identifier to AS_PATH, which should be (AS_SEQUENCE/11, AS3).
- In Part F, the RUT should propagate the UPDATE message to TR1 with AS_PATH set to (AS_SEQUENCE/11).
- In Part G, the RUT should propagate the UPDATE message to TR1 with AS_PATH set to (AS_SEQUENCE/11), (AS_SET/AS4, AS5).

**Possible Problems:** None.

**Test BGP_CONF.4.2: Confederations (Originating an UPDATE)**

**Purpose:** To verify the correct behavior of a BGP router participating in an AS confederation, when it originates an UPDATE.

**References:** [RFC3065] – Section 6

**Last Modification:** February 11, 2002

**Discussion:** When a BGP router belongs to an AS which is part of a confederation, the rules to originate an UPDATE messages to a certain peer are:

- if a peer is in the same AS, the BGP router shall include an empty AS_PATH;
- if a peer is in a neighboring AS that is part of the confederation , the BGP router shall include an AS_PATH containing an AS_CONFED_SEQUENCE segment with its own AS number;
- if a peer is in a neighboring AS that is not part of the confederation , the BGP router shall include an AS_PATH containing an AS_SEQUENCE segment with its own confederation identifier.

**Test Setup:**


**Procedure:**

*Part A: UPDATE Origination within Single-AS Confederation*

1. The RUT and TR1 are configured as internal peers in AS1, which is part of confederation 11.
2. BGP is started on the routers. Wait until they establish the connections.
3. Have the RUT originate an UPDATE and send it to TR1. Check the packets.

*Part B: UPDATE Origination within Multiple-AS Confederation*

4. TR1 is moved to AS2. AS2 is part of confederation 11.
5. BGP is restarted on the routers. Wait until they establish the connections.
6. Have the RUT originate an UPDATE and send it to TR1. Check the packets.

*Part C: UPDATE Origination outside Confederation*

7. TR1 is moved to AS3. AS3 is not part of confederation 11.
8. BGP is restarted on the routers. Wait until they establish the connections.
9. Have the RUT originate an UPDATE and send it to TR1. Check the packets.

**Observable Results:**

- In Part A, the UPDATE sent to TR1 should contain an empty AS_PATH.
- In Part B, the UPDATE sent to TR1 should have an AS_PATH set to (AS_CONFED_SEQUENCE/AS1).
- In Part C, the UPDATE sent to TR1 should have an AS_PATH set to (AS_SEQUENCE/11).

**Possible Problems:** None.

**Test BGP_CONF.4.3: Confederations (Attributes)**

**Purpose:** To verify the correct behavior of a BGP router participating in an AS confederation, regarding changes in the use of some fields in BGP messages.

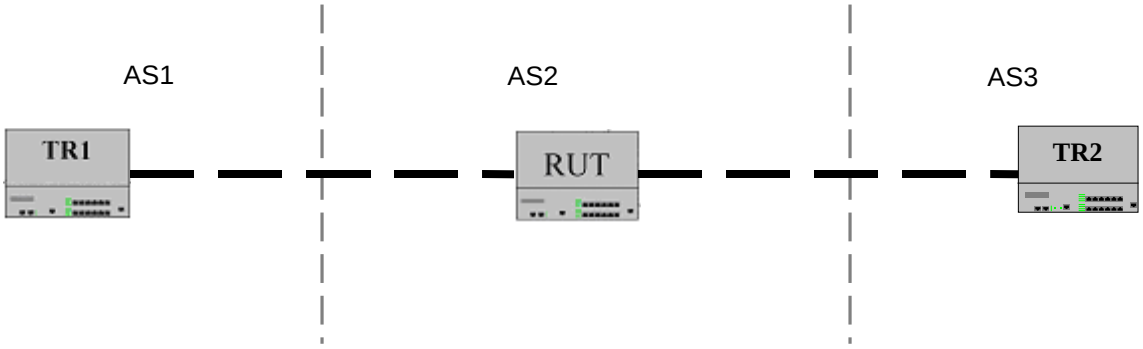
**References:** [RFC3065] – Section 7

**Last Modification:** February 3, 2003

**Discussion:** When a BGP router belongs to an AS which is part of a confederation, the rules to handle the NEXT_HOP, MED and LOCAL_PREF change as follows:

- the NEXT_HOP attribute may be propagated unchanged to peers in neighboring autonomous systems that are part of the same confederation;
- the MED attribute may be propagated unchanged to peers in neighboring autonomous systems that are part of the same confederation;
- the LOCAL_PREF attribute should be propagated to peers in neighboring autonomous systems that are part of the same confederation;

**Test Setup:**




**Procedure:**

*Part A: Handling of My Autonomous System field*

1. The RUT and TR1 are configured as external peers. The RUT and TR2 are configured as external peers. AS2 and AS3 are part of the same confederation (identifier 111).
2. BGP is started on the routers. Wait until they establish the connections.
3. Check the packets.

*Part B: Handling of NEXT_HOP, MED, and LOCAL_PREF fields*

4. BGP is restarted on the routers. Wait until they establish the connections.
5. TR1 sends an UPDATE for a new route to the RUT containing the MED field. Check the packets.

**Observable Results:**

- In Part A, the OPEN message from the RUT to TR2 should use the RUT's AS number (AS2) in the My Autonomous System field. The OPEN message from the RUT to TR1 should use the Confederation ID (111) in the My Autonomous System field.
- In Part B, the RUT should propagate the UPDATE to TR2. The NEXT_HOP attribute and the MED attribute may be unchanged. The RUT should compute a new LOCAL_PREF for the new route. The UPDATE to TR2 should contain the LOCAL_PREF attribute.

**Possible Problems:** None.

**Test BGP_CONF.4.4: Route Reflector**

**Purpose:** To verify the correct behavior of a BGP router when it is configured as a route reflector.

**References:** RFC2796

**Last Modification:** February 3, 2003

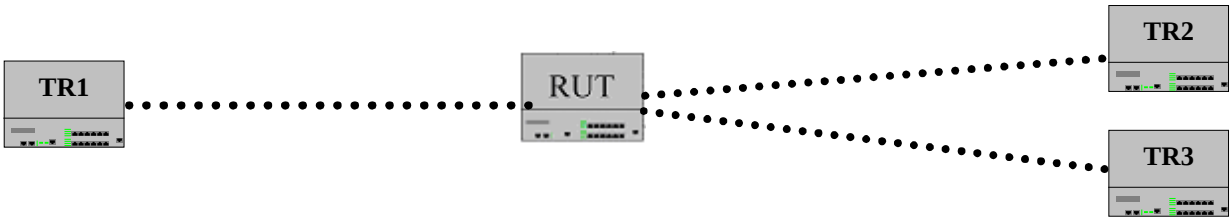
**Discussion:** A "route reflector" serves a cluster of clients, so that the clients do not have to be fully meshed. In order to ensure that all the BGP routers in the cluster of clients have consistent routing information, the route reflector "reflects" updates received from a client to the other clients. The propagation rules for peers outside the cluster are unchanged.

When a "route reflector" receives a route from an internal peer, it selects the best path based on its path selection rule. After the best path is selected, it must do the following depending on the type of the peer it is receiving the best path from:

- Reflect to all Clients any route received from a Non-Client internal peer.
- Reflect to all the Non-Client peers and also to the Client peers (hence the Client peers are not required to be fully meshed) any route received from a Client peer.

The ORIGINATOR_ID is a new optional, non-transitive BGP attribute of Type code 9. This attribute is 4 bytes long and it will be created by a "route reflector" in reflecting a route. This attribute will carry the ROUTER_ID of the originator of the route in the local AS. A BGP speaker should not create an ORIGINATOR_ID attribute if one already exists. A router which recognizes the ORIGINATOR_ID attribute should ignore a route received with its ROUTER_ID as the ORIGINATOR_ID.

**Test Setup:**




**Procedure:**

*Part A: UPDATE Propagation from Non-Client to Clients*

1. The RUT and TR1, the RUT and TR2 and the RUT and TR3 are configured as internal peers.
2. Configure the RUT as a route reflector, with a CLUSTER_ID = 11. The client cluster is made of TR2 and TR3.
3. BGP is started on the routers. Wait until they establish the connections.
4. TR1 originates an UPDATE for a new route. Check the packets.

*Part B: UPDATE Propagation from Client to Clients and Non-Clients*

5. TR2 originates an UPDATE for a new route.
6. Check the packets.

*Part C: UPDATE Propagation from Non-Client to Clients and Non-Clients*

7. TR3 is moved to another autonomous system.
8. BGP is restarted on the routers. Wait until they establish the connections.
9. TR3 sends an UPDATE for a new route containing the MED field. Check the packets.

*Part D: UPDATE Propagation to Clients, but not Non-Clients*

10. TR3 is moved back to the RUT's autonomous system. TR3 is not in the client cluster.
11. BGP is restarted on the routers. Wait until they establish the connections.
12. TR3 sends an UPDATE for a new route. Check the packets.

*Part E: Router receives UPDATE with its own ROUTER_ID as ORIGINATOR_ID*

13. TR1 is replaced with a packet generator simulating a BGP router.
14. BGP is restarted on the routers. Wait until they establish the connections.
15. TR1 sends an UPDATE for a new route with the ORIGINATOR_ID set to the ROUTER_ID of the RUT. Check the packets.

**Observable Results:**

- In Part A, the RUT should propagate the UPDATE to both TR2 and TR3. The NEXT_HOP, AS_PATH, and LOCAL_PREF attribute should be unchanged.
- In Part B, the RUT should propagate the UPDATE to TR1 and to TR3. The UPDATE to TR1 should include the CLUSTER_LIST attribute, which contains the CLUSTER_ID configured on the RUT (i.e., 11). Both UPDATE messages should include the ORIGINATOR_ID attribute, set to TR2's ROUTER_ID. The LOCAL_PREF attribute should be unchanged.
- In Part C, the RUT should propagate the UPDATE to both TR1 and TR2, following the normal rules. The MED attribute may be changed.
- In Part D, the RUT should propagate the UPDATE to TR2 only. The NEXT_HOP, AS_PATH, and LOCAL_PREF attributes should be unchanged. The UPDATE should include the ORIGINATOR_ID attribute set to TR3's ROUTER_ID.
- In Part E, the RUT should ignore the UPDATE received from TR1.

**Possible Problems:** None.

**Test BGP_CONF.4.5: Route Reflector to Non-Client**

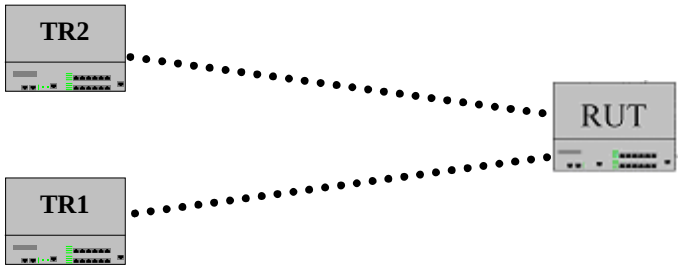
**Purpose:** To verify the correct behavior of a BGP route reflector when peered with a nonclient.

**References:** RFC2796

**Last Modification:** February 11, 2002

**Discussion:** The CLUSTER_LIST attribute, together with the CLUSTER_ID, is used to avoid routing loops and "black holes". Updates originated in a cluster should not be sent back into that cluster. A route reflector should ignore an UPDATE that includes a CLUSTER_LIST containing its own CLUSTER_ID.

**Test Setup:**




**Procedure:**

*Part A: Part A: Handling of CLUSTER_ID attribute*

1. The RUT and TR1 and the RUT and TR2 are configured as internal peers. Configure the RUT as a route reflector with CLUSTER_ID 5.0.0.0.
2. BGP is started on the routers. Wait until they establish the connections.
3. TR1 originates an UPDATE for a new route. The UPDATE includes the CLUSTER_LIST attribute, with CLUSTER_ID 5.0.0.0. Check the packets.

**Observable Results:**

- In Part A, the RUT should ignore the UPDATE. It should not install the new route, and should not propagate the UPDATE to TR2.

**Possible Problems:** None.

**Test BGP_CONF.4.6: Communities**

**Purpose:** To verify the correct behavior of a BGP router that implements the Communities Attribute extension.

**References:** [RFC1997]

**Last Modification:** February 3, 2003

**Discussion:** A community is a group of destinations that share some common property. The Communities Attribute is expected to significantly simplify a BGP router's configuration that controls distribution of routing information.

The Communities path attribute is an optional transitive attribute of variable length. The attribute consists of a set of four octet values, each of which specify a community. All routes with this attribute belong to the communities listed in the attribute.

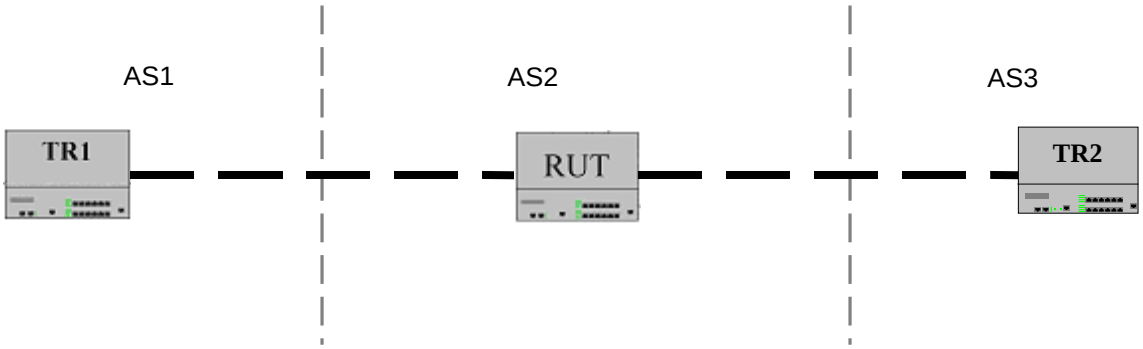
The following are well-known communities:

NO_EXPORT (0xFFFFFF01)
All routes received carrying a communities attribute containing this value MUST NOT be advertised outside a BGP confederation boundary (a stand-alone autonomous system that is not part of a confederation should be considered a confederation itself).

NO_ADVERTISE (0xFFFFFF02)
All routes received carrying a communities attribute containing this value MUST NOT be advertised to other BGP peers.

NO_EXPORT_SUBCONFED (0xFFFFFF03)
All routes received carrying a communities attribute containing this value MUST NOT be advertised to external BGP peers (this includes peers in other members autonomous systems inside a BGP confederation).

**Test Setup:**




**Procedure:**

*Part A: Originating NO_EXPORT Attribute*

1. The RUT and TR1 and the RUT and TR2 are configured as external peers.
2. Configure the RUT to consider some route part of the NO_EXPORT community.

3. BGP is started on the routers. Wait until they establish the connections.
4. Have the RUT originate an UPDATE for that route. Check the packets.

*Part B: Attaching Arbitrary COMMUNITIES Attribute*

5. BGP is restarted on the routers. Wait until they establish the connections.
6. Configure the RUT to attach a COMMUNITIES attribute to routes received from TR1.
7. TR1 sends an UPDATE for a new route, with no COMMUNITES attribute. Check the packets.

*Part C: Propagating NO_EXPORT Attribute*

8. BGP is restarted on the routers. Wait until they establish the connections.
9. TR1 sends an UPDATE for a new route, containing the COMMUNITIES attribute with value NO_EXPORT. Check the packets.

*Part D*: *Propagating NO_EXPORT_SUBCONFED Attribute*

10. AS2 and AS3 are now part of the same confederation.
11. BGP is restarted on the routers. Wait until they establish the connections.
12. TR1 sends an UPDATE for a new route, containing the COMMUNITIES attribute with value NO_EXPORT_SUBCONFED. Check the packets.

*Part E: Propagating NO_ADVERTISE Attribute*

13. TR2 is moved to AS2.
14. BGP is restarted on the routers. Wait until they establish the connections.
15. TR1 sends an UPDATE for a new route, containing the COMMUNITIES attribute with value NO_ADVERTISE. Check the packets.

**Observable Results:**

- In Part A, the RUT should send the UPDATE to both TR1 and TR2. The UPDATE should contain the COMMUNITIES attribute, with value NO_EXPORT.
- In Part B, the RUT should propagate the UPDATE to TR2. It should attach the COMMUNITIES attribute to the update.
- In Part C, the RUT should not propagate the UPDATE to TR2.
- In Part D, the RUT should not propagate the UPDATE to TR2.
- In Part E, the RUT should not propagate the UPDATE to TR2.

**Possible Problems:** None.

## Test BGP_CONF.4.7: Capabilities Advertisement

**Purpose:** To verify the correct behavior of a BGP router that implements the Capabilities Negotiation extension.

**References:** [draft-ietf-idr-bgp4-cap-neg-04]- Sections 3, 4

**Last Modification:** February 3, 2003

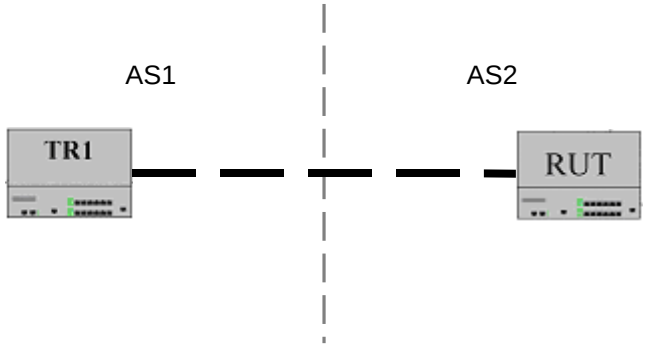
**Discussion:**
The purpose of Capabilities Negotiation is twofold:

- to communicate to a peer what capabilities a BGP router supports;
- to learn what capabilities the peer supports.

If a BGP router supports a certain capability, and it determines that its peer does not support that capability, the router may send a NOTIFICATION with Error Subcode Unsupported Capability. The Data field should contain the unsupported capability. In this case the speaker should attempt to re-establish a BGP connection with the peer without sending to the peer the Capabilities Optional Parameter.

A BGP router may transmit an OPEN message with a particular capability included more than once within the Optional Parameter.

**Test Setup:**




**Procedure:**
*Part A: Capabilities Not Negotiated*

1. The RUT and TR1 are configured as external peers.
2. TR1 does not support Capabilities Negotiation.
3. BGP is started on the routers.
4. Wait for the OPEN messages to be sent and check the packets.

*Part B: Same Capabilities*

5. BGP is restarted on the routers.
6. TR1 sends an OPEN message containing all Capabilities that the RUT does support. Check the packets.

*Part C: Capability included twice*

7. BGP is restarted on the routers.
8. TR1 sends an OPEN message containing all Capabilities that the RUT does support. One of these capabilities is included two times within the Optional Parameter.
9. Check the packets.

**Observable Results:**

- In Part A, the RUT should send an OPEN message containing the Capabilities Optional Parameter. TR1 should close the connection with Error Subcode Unsupported Optional Parameter. Upon receipt of the NOTIFICATION message, the RUT should attempt to reestablish the connection, without sending the Capabilities Optional Parameter.
- In Parts B and C, the BGP connection should be established normally.

**Possible Problems:** None.

**Test BGP_CONF.4.8: Multiprotocol**

**Purpose:** To verify the correct behavior of a BGP router that implements the Multiprotocol extension.

**References:** [RFC 2858] – Section 2

**Last Modification:** February 3, 2003

**Discussion:** The Multiprotocol extension allows BGP to carry routing information for multiple Network Layer protocols. This is achieved by the introduction of two new optional non-transitive attributes:

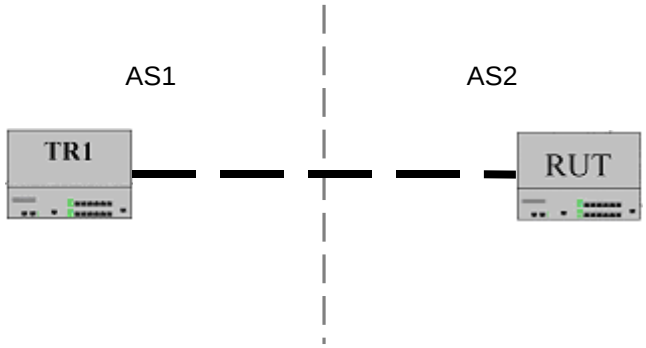
- MP_REACH_NLRI (Type Code 14) is used to:
  - advertise feasible routes;
  - advertise the Network Layer address of the router that should be used as the Next Hop to the destinations listed in the NLRI;
  - advertise some or all of the Subnetwork Points of Attachment (SNPAs) that exist within the local system.
- MP_UNREACH_NLRI (Type Code 15) is used to withdraw multiple unfeasible routes from service.

A BGP router that implements the Multiprotocol extension must use the Capability Negotiation extension. The Capability Code is 1. The Capability Length is set to 4, and the Capability Value lists the AFI/SAFI that the router supports.

An UPDATE message that carries the MP_REACH_NLRI must also carry the ORIGIN and the AS_PATH attributes (both in EBGP and in IBGP exchanges). Moreover, in IBGP exchanges such a message must also carry the LOCAL_PREF attribute. If the UPDATE message is received from an external peer, the local system shall check whether the leftmost AS in the AS_PATH attribute is equal to the autonomous system number of the peer than sent the message. If that is not the case, the local system shall send the NOTIFICATION message with Error Code UPDATE Message Error, and the Error Subcode set to Malformed AS_PATH.

An UPDATE message that carries no NLRI, other than the one encoded in the MP_REACH_NLRI attribute, should not carry the NEXT_HOP attribute. If such a message contains the NEXT_HOP attribute, the BGP speaker that receives the message should ignore this attribute.

**Test Setup:**




**Procedure:**

*Part A: Multiprotocol UPDATE Originated*

1. The RUT and TR1 are configured as external peers. that support the same protocols (AFI/SAFI). If a real router is not available, a packet generator is used for TR1.

2. BGP is started on the routers. Wait until they establish the connection.
3. Give the RUT a route belonging to one of the supported protocols.
4. Check the packets.

*Part B: Multiprotocol UPDATE Received*

5. Remove the route on the RUT.
6. Check the packets.

*Part C: UPDATE received with malformed AS_PATH*

7. BGP is started on the routers. Wait until they establish the connection.
8. TR1 sends an UPDATE message containing a MP_REACH_NLRI attribute for some route. The leftmost AS in the AS_PATH attribute is different than TR1's AS number. The NLRI field in the UPDATE is empty.
9. Check the packets.

*Part D: UPDATE Received with MP_REACH_NLRI*

10. BGP is started on the routers. Wait until they establish the connection.
11. TR1 sends an UPDATE message containing a MP_REACH_NLRI attribute for some route. The leftmost AS in the AS_PATH attribute is TR1's AS number. The NLRI field in the UPDATE is empty.
12. Check the packets.

*Part E: UPDATE Received with MP_UNREACH_NLRI*

13. BGP is started on the routers. Wait until they establish the connection.
14. TR1 sends an UPDATE message containing a MP_UNREACH_NLRI attribute for the previously advertised route.
15. Check the packets.

*Part F: UPDATE Sent to Internal Peer*

16. The RUT and TR1 are reconfigured as internal peers.
17. BGP is started on the routers. Wait until they establish the connection.
18. Have the RUT send an UPDATE message containing a MP_REACH_NLRI attribute for some route.
19. Check the packets.

**Observable Results:**

- In Part A, the RUT should send an UPDATE message containing the MP_REACH_NLRI attribute. The attribute should list the given destination in its NLRI, together with the Next Hop to that destination. Also, the UPDATE message should include the ORIGIN and AS_PATH attributes.
- In Part B, the RUT should send an UPDATE message containing the MP_UNREACH_NLRI attribute. The attribute should list the unfeasible route.
- In Part C, the RUT should send a NOTIFICATION message with Error Code UPDATE Message Error, and the Error Subcode set to Malformed AS_PATH.
- In Part D, the RUT should accept the UPDATE. The RUT should install the route in its Adj-RIB-In.
- In Part E, the RUT should accept the UPDATE. The RUT should remove the route from its Adj-RIB-In.
- In Part F, the RUT should send an UPDATE message containing the MP_REACH_NLRI attribute. The UPDATE should also include the LOCAL_PREF attribute.

**Possible Problems:** None.

**Test BGP_CONF.4.9: Basic MD5 Authentication**

**Purpose:** To verify that a router can perform basic MD5 authentication processing functionality.

**References:** [draft-przygienda-bgp-md5-00] – Section 3
RFC 2385] – Section 2

**Last Modification:** January 23, 2002

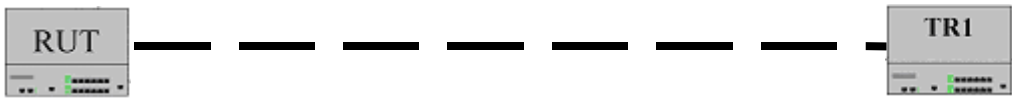
**Discussion:** The primary motivation for this option is to allow BGP to protect itself against the introduction of spoofed TCP segments into the connection stream. Of particular concern are TCP resets.

When a message with MD5 authentication is received,

- The digest is set aside,
- The appropriate algorithm and key are determined from the value of the Key Identifier field,
- The BGP-4 Authentication Key is written into the appropriate number (16 when keyed MD5 is used) of bytes starting at the offset indicated,
- Appropriate padding is added as needed, and
- A new digest calculated using the indicated algorithm.

If the calculated digest does not match the received digest, the message is discarded and appropriate Authentication failed NOTIFICATION sent. The connection is closed subsequently.

**Test Setup:** Connect the RUT to another Testing Router who is running BGP according to the figure.




**Procedure:**

*Part A: RUT Not Configured for Authentication*

1. Configure the RUT not to perform TCP MD5 authentication.
2. BGP is started on the routers. TR1 sends an OPEN message with a correct MD5 authentication header.
3. Check the packets.

*Part B: RUT Receives Authentication Header with Correct Digest*

4. Configure the RUT to perform TCP MD5 authentication, with a secret of ABCDEFGHIJKL.
5. BGP is restarted on the routers. TR1 sends an OPEN message with a MD5 authentication header (correct digest).
6. Check the packets.

*Part C: RUT Configured for Authentication, Receives Un-Authenticated BGP Message*

7. BGP is restarted on the routers. TR1 sends an OPEN message with no authentication header.
8. Check the packets.

**Observable Results:**

- In Part A, the RUT should discard the message, because the message contains an MD5 authentication header.

- In Part B, the RUT should accept the message and send an OPEN message to TR1.
- In Part C, the RUT should discard the message.

**Possible Problems:** None.

**Test BGP_CONF.4.10: Processing Route Advertisements**

**Purpose:** To verify the BGP router running Route Flap Damping properly processes route advertisements.

**References:** [2439] – Sections 4.8.1, 4.8.3

**Last Modification:** June 7, 2004

**Discussion:** If the new route is to be suppressed it is placed on a reuse list only if it would have been preferred to the current best route had the new route been accepted as stable. There is no reason to queue a route on a reuse list if after the route becomes usable it would not be used anyway due to the existence of a more preferred route. Such a route would not have to be reevaluated unless the preferred route became unreachable. As specified here, the less preferred route would be reevaluated and potentially used or potentially added to a reuse list when processing the withdrawal of a more preferred best route.

The following parameters are configurable in route flap damping:

cutoff threshold (cut)

- This is the value above which a route advertisement will be suppressed.

reuse threshold (reuse)

- This is the value below which a suppressed route will now be used again.

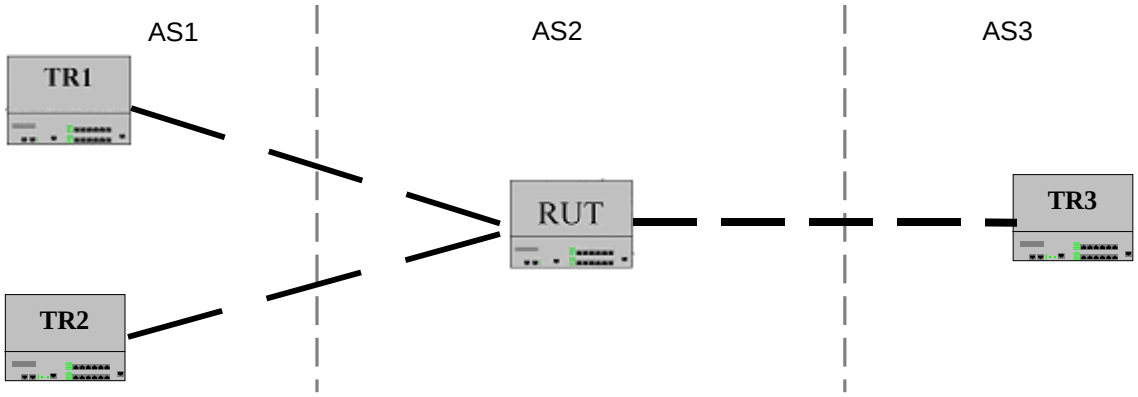
maximum hold down time (T-hold)

- This value is the maximum time a route can be suppressed no matter how unstable it has been prior to this period of stability.

decay half life while reachable (decay-ok)

- This value is the time duration in minutes or seconds during which the accumulated stability figure of merit will be reduced by half.

**Test Setup:** Set up the RUT according to the figure below and restart all routers.



**Procedure:**

*Part A: No damp history*

1. The RUT and TR1, the RUT and TR2, and the RUT and TR3 are configured as external peers.
2. Configure the RUT as follows:
   Penalty = 1000 (if configurable)
   Cutoff = 2000
   Reuse =800
   Half-life = 3 min
   Max suppress = 5 min
4. BGP is started on the routers. Wait until they establish the connections.
5. TR1 sends an UPDATE message with a route to 192.1.0.0/16 with NEXT_HOP set to itself.
6. Check the packets.

*Part B: !Suppressed && figure-of-merit < cut*

7. Configure the RUT to assign a higher degree of preference for TR1 than TR2.
8. BGP is started on the routers. Wait until they establish the connections.
9. TR2 a sends an UPDATE message with a route to 192.1.0.0/16 with NEXT_HOP set to itself.
10. TR1's route to 192.1.0.0/16 is flapped once. Check the packets.

*Part C: !Suppressed && figure-of-merit >= cut*

11. TR1's route to 192.1.0.0/16 is flapped two more times before the figure-of-merit decays to zero.
12. Check the packets.

*Part D: Suppressed && figure-of-merit < reuse*

13. Wait approximately 5 minutes.
14. Check the packets.

**Observable Results:**

- In Part A, the RUT should send an UPDATE to TR3 for 192.1.0.0/16 with the NEXT_HOP attribute set to TR1's IP address.
- In Part B, no changes in the RUT's routing table is made. The RUT should create a damp history for 192.1.0.0/16 that is associated with TR1.
- In Part C, the RUT should suppress 192.1.0.0/16 that is associated with TR1 and place it in the reuse list. The RUT should install 192.1.0.0/16 that is associated with TR2 in its routing table. The RUT should send an UPDATE message to TR3 for the changes.
- In Part D, the RUT should install 192.1.0.0/16 that is associated with TR1 in its routing table. The RUT should send an UPDATE to TR3 for the changes. The RUT should not insert 192.1.0.0/16 that is associated with TR2 in the reuse list.

**Possible Problems:** None.

**Test BGP_CONF.4.11: Processing Route Changes**

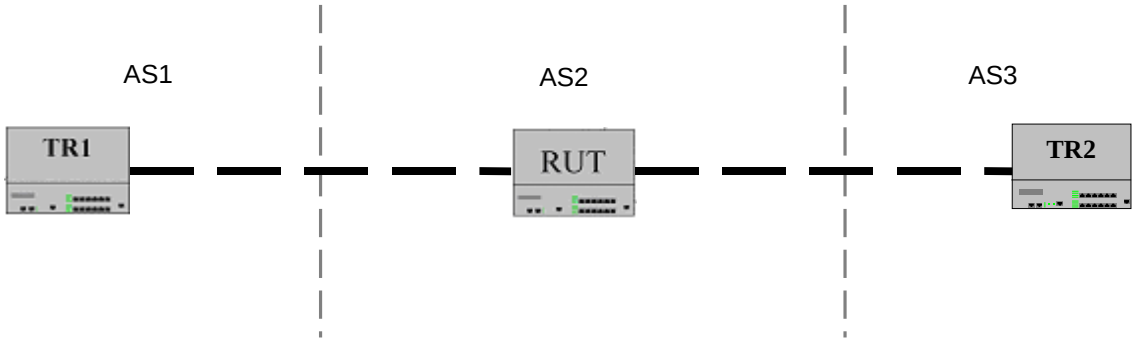
**Purpose:** To verify the BGP router running Route Flap Damping properly processes route changes.

**References:** [2439] – Section 4.8.4

**Last Modification:** January 23, 2002

**Discussion:** If a route is replaced by a peer router by supplying a new path, the route that is being replaced should be treated as if an unreachable were received (see Section 4.8.2). This will occur when a peer somewhere back in the AS path is continuously switching between two AS paths and that peer is not damping route flap (or applying less damping). There is no way to determine if one AS path is stable and the other is flapping, or if they are both flapping. If the cycle is sufficiently short compared to convergence times neither route through that peer will deliver packets very reliably. Since there is no way to affect the peer such that it chooses the stable of the two AS paths, the only viable option is to penalize both routes by considering each change as an unreachable followed by a route advertisement.

**Test Setup:** Set up the RUT according to the figure below and restart all routers.




**Procedure:**

*Part A: Route changes to the same destination*

1. The RUT and TR1 are configured as external peers, where TR1 is a packet generator simulating a BGP router. The RUT and TR2 are configured as external peers.
2. BGP is started on the routers. Wait until they establish the connection.
3. TR1 periodically switches between sending UPDATE messages with AS_PATH set to (AS_SEQUENCE/AS1, AS12) and (AS_SEQUENCE/AS1, AS14) to some network N3 until figure-of-merit >= cutoff threshold.
4. Check the packets.

**Observable Results:**

- In Part A, the RUT shall send an UPDATE to TR2 for the changes each time a switch takes place. When figure-of-merit for each route >= cutoff threshold, the RUT shall send an UPDATE to TR2 for the withdrawal of both routes.

**Possible Problems:** None.