Benchmarking Methodology
Internet-Draft
Intended status: Informational
Expires: 5 January 2026

Y. Cui
Y. Wei
X. Xie
Tsinghua University

4 July 2025

Specification of Test Case Structure for Protocol Testing Automation
draft-cui-bmwg-testcase-spec-00

Abstract

This document defines a standardized test case specification used in automated network protocol testing. The specification aims to provide a structured and interoperable format for representing test cases that evaluate the implementations of network protocols. This work facilitates the development of protocol-agnostic testing frameworks and improves the repeatability and automation of network testing.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-cui-bmwg-testcase-spec/.

Discussion of this document takes place on the bmwg Working Group mailing list (mailto:bmwg@ietf.org), which is archived at https://datatracker.ietf.org/wg/bmwg/. Subscribe at https://www.ietf.org/mailman/listinfo/bmwg/.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 January 2026.

Table of Contents

1. Introduction

Test cases have long served as a foundational element in evaluating network protocol implementations. Over the past several decades, informal conventions and community practices have emerged for describing and executing such tests. Despite this evolution, a formal, standardized specification for test case structure remains absent. As a result, test cases are often documented in ad hoc formats, leading to ambiguities in interpretation, inconsistent implementations, and challenges in sharing or reusing test artifacts across different environments.

The rise of automated network testing frameworks and protocol-agnostic validation systems has significantly raised the bar for test case clarity, precision, and interoperability. In these automated systems, even minor ambiguities in test definition can result in incorrect behavior, reduced repeatability, and difficulty in diagnosing failures. A standardized test case structure is therefore critical not only to eliminate misinterpretation during implementation but also to facilitate test intent preservation, enable reproducible testing, and support automated validation pipelines.

This document proposes a unified, machine-readable specification for network protocol test cases. The goal is to bridge the gap between legacy testing practices and modern automation needs by providing a clear, extensible, and interoperable schema for describing test logic, environment, parameters, and evaluation criteria. By aligning the test case definition with automation requirements, this specification lays the groundwork for improved consistency in protocol testing and advances in test automation research.

2. Definition and Acronyms

DUT: Device Under Test

CLI: Command Line Interface

Tester: A network device for protocol conformance and performance testing. It can generate specific network traffic or emulate particular network devices to facilitate the execution of test cases.

3. Test Case Classification

Protocol test cases can be broadly classified into the following categories. Each category serves a distinct validation goal and may require specialized tools or methodologies:

* Conformance Testing: Validates whether a protocol implementation adheres strictly to the normative requirements defined in the protocol specification. Conformance test suites are typically designed with exhaustive coverage and are essential for standard compliance.

* Functional Testing: Verifies that the implementation performs its intended functions correctly under both normal and edge-case conditions. Unlike conformance testing, which focuses on specification adherence, functional testing emphasizes operational behavior in representative scenarios, such as route convergence, state transitions, and control plane interactions.

* Performance and Scalability Testing: Evaluates key performance metrics such as throughput, latency, jitter, packet loss, and control plane convergence time. Tests in this category are often based on established benchmarking methodologies (e.g., [RFC2544], [RFC2889], [RFC5180]) and may also assess scalability under stress conditions, such as large-scale route distribution or high-volume traffic forwarding.

* Interoperability Testing: Ensures that multiple implementations of the same protocol (often from different vendors) can interoperate correctly. This type of testing focuses on real-world integration scenarios and is critical for multi-vendor environments. Interoperability testing is commonly performed in plugfests or consortium-organized test events.

* Robustness Testing: Assesses the implementation’s resilience against malformed, unexpected, or adversarial inputs. These tests may include protocol fuzzing, boundary condition violations, or replayed traffic anomalies. While related to security testing, robustness testing typically focuses on fault tolerance and graceful failure handling rather than exploit prevention.

4. Test Case Specification

Each test case MUST consist of the following components:

4.1. Metadata

* test-id: Unique identifier.

* title: A human-readable name of the test case.

* purpose: The summary of test objectives.

* category: One or more of the above classification types.

* protocol: Protocol(s) being tested (e.g., OSPFv2, BGP, TCP).

* references: Related RFCs or drafts.

4.2. Topology

Describes the setup of the test environment, including:

* Node roles (e.g., DUT, tester).

* Link characteristics (e.g., delay, bandwidth).

* Addressing scheme (IP, MAC, etc.).

In automated testing scenarios, it is common to construct a _minimal common topology_ for a given set of test cases. This refers to the simplest possible network setup that can support the execution of all selected test cases. The primary advantage of this approach is that the testbed needs to be instantiated only once prior to execution, after which a batch of automated tests can be run efficiently and repeatedly without further reconstruction.

4.3. Test Setup

The test setup specifies the initial configuration of all DUTs prior to test execution. Certain procedural steps may require temporary deviations from this baseline configuration. When protocol parameter values are not explicitly specified, implementations MUST use the protocol-defined default values.

4.4. Parameters

Parameters are a critical component of test cases, as any parameter left undefined may introduce ambiguity into the test case. When authoring a test case, it is essential to identify all relevant parameters required for its execution and provide explicit definitions for each.

* Protocol-specific knobs (e.g., HelloInterval for OSPF).

* Device configurations for interconnection.

* Traffic profiles (e.g., packet rate, pattern, size).

4.5. Test Procedure and Expected Behavior

To ensure consistency, each step in the test procedure MUST be defined unambiguously, such that different implementations or testing systems interpret and execute it in the same manner. Additionally, each expected behavior MUST be measurable, allowing for objective verification of test results.

* For each step:

  - Action, such as enabling interfaces, sending packets, or setting up new configurations.

  - Evaluation method, such as packet capture and API call.

  - Expected behavior.

4.6. Evaluation Criteria

* Functional: Compliance with expected state or output.
* Performance: Quantitative metrics (e.g., throughput >= 1Gbps).
* Pass/Fail thresholds.
* Result logging: Required telemetry or logs.

5. Application Scenarios

5.1. Use Case 1: Vendor Internal Testing

Network equipment manufacturers often validate their protocol implementation in controlled environments. Here, test cases focus on functional and performance validation of a single device (DUT), with testers simulating peers or traffic sources.

5.2. Use Case 2: Live Network Acceptance Testing

When new network equipment is introduced into a live production environment, operators often conduct acceptance testing to verify both backward compatibility and forward readiness. This includes ensuring interoperability and stability without disrupting existing services, as well as validating the device’s ability to meet the requirements of new service scenarios. The testing process may involve limited-scope deployment, traffic mirroring, or passive monitoring, helping ensure the DUT integrates smoothly while also supporting planned innovations or upgrades.

6. Usage Example

The following is a test case example for the OSPF HelloInterval mismatch scenario.

```
test-id: TC-OSPF-HI-001
title: OSPFv2 HelloInterval Mismatch Negative Test
purpose: >
  To verify that the DUT correctly rejects OSPF neighbor formation
  when receiving Hello packets with a mismatched HelloInterval value.
category:
  - Conformance Testing
protocol:
  - OSPFv2
references:
  - RFC2328 Section A.3.2
```

```
   topology:
     nodes:
       - name: TesterA
         role: tester
         interfaces:
           - name: PortTesterA_1
             ip: 192.0.2.100/24
             connected_to: DeviceA:PortDeviceA_1
       - name: DeviceA
         role: DUT
         interfaces:
           - name: PortDeviceA_1
             ip: 192.0.2.1/24
             connected_to: TesterA:PortTesterA_1
     links:
       - node1: TesterA:PortTesterA_1
         node2: DeviceA:PortDeviceA_1

   test-setup:
     DUT-initial-config:
       - Configure interface PortDeviceA_1 with IP 192.0.2.1/24
       - Enable OSPFv2 on PortDeviceA_1
       - Set HelloInterval = 10 seconds on PortDeviceA_1
     tester-initial-config:
       - Configure interface PortTesterA_1 with IP 192.0.2.100/24
       - Enable OSPFv2 on PortTesterA_1
       - Set HelloInterval = 5 seconds on PortTesterA_1

   parameters:
     OSPF.HelloInterval.DUT: 10s
     OSPF.HelloInterval.Tester: 5s
     Network.Mask: 255.255.255.0
     OSPF.Network.Type: Broadcast
     OSPF.Area: 0.0.0.0

   procedure:
     - step: 1
       action: >
         Initialize both DUT and Tester with the respective interface
         IPs and enable OSPF.
       evaluation: CLI log and interface status verification
       expected: >
         Interfaces are up and OSPF is enabled with configured
         HelloIntervals.
     - step: 2
       action: >
         Verify OSPF neighbor state between DUT and Tester.
       evaluation: >
```

```
        OSPF neighbor state via CLI and OSPF adjacency state table.
      expected: >
        No OSPF adjacency is formed. DUT interface remains in Down or
        Init state.

  evaluation-criteria:
    functional: >
      DUT must reject Hello packets with mismatched HelloInterval and
      not form adjacency.
    performance: null
    pass-fail: >
      PASS if no neighbor relationship is formed; FAIL if OSPF
      adjacency state reaches 2-Way or Full.
    logging:
      - OSPF neighbor state transitions on DUT
      - Packet capture if applicable
      - Interface and OSPF logs on DUT and Tester
```

7. Security Considerations

This document defines a test case specification format and does not introduce new protocols or alter protocol behaviors. However, the following considerations apply:

1. Test Artifacts Sensitivity: Test configurations and captured traffic may include sensitive information (e.g., IP addresses, authentication exchanges). Proper data sanitization and access control should be enforced during storage and sharing.

2. Fuzzing and Adversarial Inputs: Some robustness test cases may involve malformed or adversarial inputs. Care must be taken to ensure such inputs do not propagate beyond the test environment or affect uninvolved systems.

8. IANA Considerations

This document has no IANA actions.

9. Informative References

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <https://www.rfc-editor.org/rfc/rfc2544>.

[RFC2889] Mandeville, R. and J. Perser, "Benchmarking Methodology for LAN Switching Devices", RFC 2889, DOI 10.17487/RFC2889, August 2000, <https://www.rfc-editor.org/rfc/rfc2889>.

[RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <https://www.rfc-editor.org/rfc/rfc5180>.

Acknowledgments

This work is supported by the National Key R&D Program of China.

Contributors

Zhen Li
Beijing Xinertel Technology Co., Ltd.
Email: lizhen_fz@xinertel.com

Zhanyou Li
Beijing Xinertel Technology Co., Ltd.
Email: lizy@xinertel.com

Authors' Addresses

Yong Cui
Tsinghua University
Email: cuiyong@tsinghua.edu.cn

Yunze Wei
Tsinghua University
Email: wyz23@mails.tsinghua.edu.cn

Xiaohui Xie
Tsinghua University
Email: xiexiaohui@tsinghua.edu.cn