Windows Protocol Test Suites

Architecture, design, and usage for testing protocol implementations

Presented by
Obaro Ogbo
Senior Software Engineer
Microsoft
Agenda

- Background
- Test Suites Architecture
  - Application Design & Architecture
  - Test Case Design
  - SUT Control
- Usage
  - PTM Service
  - PTM Cli
  - Docker Image
Background

Windows Protocol Test Suites
Background – Interoperability

- Open connections to Microsoft products
  - Open protocols
  - Open APIs
  - Open Access
  - Open Source Compatibility
- Support for standards
- Data portability
- Open engagement
Background – Windows Protocols

Evaluates whether a protocol implementation meets certain interoperability requirements.

- Originally developed for in-house testing of Microsoft Open Specifications.
- Used to test/verify Windows behavior.
- Also used to test 3rd-party implementations.

Do not cover every protocol requirement, and do not certify an implementation, but can be a useful indication of interoperability.
Background – Test Suite Goals

- Cross platform
- Protocol features as tests
- Multiple protocols – RDP, SMB2, ADFamily, etc
Background – Protocol Test Methodology

1. Open Specification defines messages, sequences, behaviors
2. Develop protocol test suite (synthetic client) according to Open Specification
3. Run against Windows to verify Open Specification
4. Run against 3rd-party implementation to help identify & debug issues
5. Interoperability between partner’s server and Windows client
Architecture

Windows Protocol Test Suites
Architecture – Test Case Design

- **Design**
  - Test cases
  - Architecture
  - Interfaces
  - Consider undetermined partner implementation

- **Implementation**
  - SDK
  - Test cases

---

If xxx server sends message A to xxx client, the client MUST respond with message B.

If A is invalid, client MUST return error message C.

Test Case 1:
1. ...
2. Server.Send(msgA);
3. Expect client respond msgB, and
   • Assert (msgB.Field1 == valid1)
   • ...
   • Assert (msgB.FieldN == validN)

Test Case 2:
1. ...
2. Server.Send(Invalid msgA);
3. Expect client respond msgC, and
   • Assert (msgC.Status == ERROR)
Architecture – SUT Control

- SUT Remote Control Protocol
- Abstract control operations to SUT
  - PowerShell
  - Shell
  - Interactive
  - Managed
- SUT Control Agent (Managed) runs on SUT
  - C#
  - Java
  - C
  - Python
Usage

Windows Protocol Test Suites
Usage – Direct execution

- dotnet test
- Cross platform
- PowerShell & Shell batch scripts
  - RunAllTestCases
  - RunTestCasesByFilter
- Supports filtering
  - FullyQualifiedName
  - Name
  - TestCategory
Usage – PTM Service

https://github.com/microsoft/WindowsProtocolTestSuites/releases/tag/ptmservice@1.1.0

- User Friendly
  - Auto detection
  - GUI
- Web service
- Cross platform
- Supports multiple Test Suite versions
- Create custom filter rules
- Export configuration
Usage – PTM Cli

- Integrates with CI/CD
- Imports configuration
- Can be used with custom filter rules
- Automation friendly
- Cross platform
Usage – Docker Image

https://hub.docker.com/_/microsoft-windowsprotocoltestsuites

- Debian 11 base
- Supports PTM Service
- Supports Direct execution
- Supported Test Suites
  - File Server
  - RDP Server
  - RDP Client
Contact Us

testsuitehelp@microsoft.com
Please take a moment to rate this session.

Your feedback is important to us.