Wireshark Developer’s Guide

Version 3.5.0

Table of Contents

Preface

- 1. Foreword
- 2. Who should read this document?
- 3. Acknowledgements
- 4. About this document
- 5. Where to get the latest copy of this document?
- 6. Providing feedback about this document
- 7. Typographic Conventions
  - 7.1. Admonitions
  - 7.2. Shell Prompt and Source Code Examples

I. Wireshark Build Environment

- 1. Introduction
  - 1.1. Introduction
  - 1.2. What is Wireshark?
  - 1.3. Supported Platforms
    - 1.3.1. Unix and Unix-like platforms
    - 1.3.2. Microsoft Windows
  - 1.4. Development and maintenance of Wireshark
    - 1.4.1. Programming languages used
    - 1.4.2. Open Source Software
  - 1.5. Releases and distributions

- 1.5.1. Binary distributions
- 1.5.2. Source code distributions

1.6. Automated Builds (Buildbot)
- 1.6.1. Advantages
- 1.6.2. What does the Buildbot do?

1.7. Reporting problems and getting help
- 1.7.1. Website
- 1.7.2. Wiki
- 1.7.3. FAQ
- 1.7.4. Other sources
- 1.7.5. Q&A Site
- 1.7.6. Mailing Lists
- 1.7.7. Bug database (Bugzilla)
- 1.7.8. Reporting Problems
- 1.7.9. Reporting Crashes on UNIX-like platforms
- 1.7.10. Reporting Crashes on Windows platforms

2. Quick Setup

2.1. UNIX: Installation

2.2. Win32/64: Step-by-Step Guide
- 2.2.1. Recommended: Install Chocolatey
- 2.2.2. Install Microsoft Visual Studio
- 2.2.3. Install Qt
- 2.2.4. Install Python
- 2.2.5. Install Perl
- 2.2.6. Install Git
- 2.2.7. Install CMake
- 2.2.8. Install Asciidoctor, Xsltproc, And DocBook
- 2.2.9. Install winflexbison

2.2.10. Install and Prepare Sources

2.2.11. Open a Visual Studio Command Prompt

2.2.12. Generate the build files

2.2.13. Build Wireshark

2.2.14. Debug Environment Setup

2.2.15. Optional: Create User’s and Developer’s Guide

2.2.16. Optional: Create a Wireshark Installer

3. Work with the Wireshark sources

3.1. Introduction

3.2. The Wireshark Git repository

3.2.1. Git Naming Conventions

3.3. Browsing And Searching The Source Code

3.4. Obtaining The Wireshark Sources

3.4.1. Git Over SSH Or HTTPS

3.4.2. Development Snapshots

3.4.3. Official Source Releases

3.5. Update Your Wireshark Sources

3.5.1. Update Using Git

3.6. Build Wireshark

3.6.1. Building on Unix

3.6.2. Windows Native

3.7. Run Your Version Of Wireshark

3.7.1. Unix-Like Platforms

3.7.2. Windows Native

3.8. Debug Your Version Of Wireshark

3.8.1. Unix-Like Platforms

3.8.2. Windows Native

3.9. Make Changes To The Wireshark Sources
3.10. Contribute Your Changes
3.10.1. Creating Merge Requests
3.10.2. Updating Merge Requests
3.10.3. Some Tips For A Good Patch
3.10.4. Writing a Good Commit Message
3.10.5. Code Requirements
3.10.6. Backporting A Change
3.11. Binary Packaging
3.11.1. Packaging Guidelines
3.11.2. Debian: .deb Packages
3.11.3. Red Hat: .rpm Packages
3.11.4. macOS: .dmg Packages
3.11.5. Windows: NSIS .exe Installer
3.11.6. Windows: PortableApps .paf.exe Package
3.12. Mime Types
3.12.1. Display Filter
3.12.2. Coloring Rules
3.12.3. Filter List
3.12.4. Column List
4. Tool Reference
4.1. Introduction
4.2. Chocolatey
4.3. CMake
4.4. GNU Compiler Toolchain (UNIX And UNIX-like Platforms)
4.4.1. gcc (GNU Compiler Collection)
4.4.2. gdb (GNU Project Debugger)
4.4.3. make (GNU Make)

- 4.4.4. Ninja

4.5. Microsoft compiler toolchain (Windows native)
- 4.5.1. Official Toolchain Packages And Alternatives
- 4.5.2. cl.exe (C Compiler)
- 4.5.3. link.exe (Linker)
- 4.5.4. Visual C++ Runtime “Redistributable” Files
- 4.5.5. Windows Platform SDK

4.6. Documentation Toolchain
- 4.6.1. Asciidoctor
- 4.6.2. DocBook XML and XSL
- 4.6.3. xsltproc
- 4.6.4. HTML Help

4.7. Debugger
- 4.7.1. Visual Studio Integrated Debugger
- 4.7.2. Debugging Tools For Windows

4.8. bash
4.9. Python
4.10. Perl
4.11. Bison
4.12. Flex
4.13. Git client
4.14. Git Powershell Extensions (Optional)
4.15. Git GUI Client (Optional)
4.16. patch (Optional)
4.17. Windows: NSIS (Optional)
4.18. Windows: WiX Toolset (Optional)
4.19. Windows: PortableApps (Optional)

- 5. Library Reference
  - 5.1. Introduction
  - 5.2. Binary Library Formats
  - 5.3. Windows Automated Library Download
  - 5.4. Qt
  - 5.5. GLib And Supporting Libraries
  - 5.6. c-ares
  - 5.7. SMI (Optional)
  - 5.8. zlib (Optional)
  - 5.9. libpcap or Npcap (Optional, But Strongly Recommended)
  - 5.10. GnuTLS (Optional)
  - 5.11. Gcrypt
  - 5.12. Kerberos (Optional)
  - 5.13. LUA (Optional)
  - 5.14. MaxMindDB (Optional)
  - 5.15. WinSparkle (Optional)

II. Wireshark Development

- 6. How Wireshark Works
  - 6.1. Introduction
  - 6.2. Overview
  - 6.3. Capturing packets
  - 6.4. Capture Files
  - 6.5. Dissect packets
- 7. Introduction
  - 7.1. Source overview
  - 7.2. Coding Style
  - 7.3. The GLib library
- 8. Packet Capture

- 8.1. How To Add A New Capture Type To Libpcap
- 8.2. Extcap: Developer Guide
  - 8.2.1. Extcap command line interface
  - 8.2.2. Extcap Arguments
  - 8.2.3. Toolbar Controls

9. Packet Dissection

- 9.1. How packet dissection works
- 9.2. Adding a basic dissector
  - 9.2.1. Setting up the dissector
  - 9.2.2. Dissecting the protocol’s details
  - 9.2.3. Improving the dissection information
- 9.3. How to handle transformed data
- 9.4. How to reassemble split packets
  - 9.4.1. How to reassemble split UDP packets
  - 9.4.2. How to reassemble split TCP Packets
- 9.5. How to tap protocols
- 9.6. How to produce protocol stats
- 9.7. How to use conversations
- 9.8. *idl2wrs*: Creating dissectors from CORBA IDL files
  - 9.8.1. What is it?
  - 9.8.2. Why do this?
  - 9.8.3. How to use idl2wrs
  - 9.8.4. TODO
  - 9.8.5. Limitations
  - 9.8.6. Notes

10. Lua Support in Wireshark

- 10.1. Introduction
- 10.2. Example: Creating a Menu with Lua

10.3. Example: Dissector written in Lua
10.4. Example: Listener written in Lua

11. Wireshark’s Lua API Reference Manual
11.1. Saving Capture Files
11.1.1. Dumper
11.1.2. PseudoHeader

11.2. Obtaining Dissection Data
11.2.1. Field
11.2.2. FieldInfo
11.2.3. Global Functions

11.3. GUI Support
11.3.1. ProgDlg
11.3.2. TextWindow
11.3.3. Global Functions

11.4. Post-Dissection Packet Analysis
11.4.1. Listener

11.5. Obtaining Packet Information
11.5.1. Address
11.5.2. Column
11.5.3. Columns
11.5.4. NSTime
11.5.5. Pinfo
11.5.6. PrivateTable

11.6. Functions For New Protocols And Dissectors
11.6.1. Dissector
11.6.2. DissectorTable
11.6.3. Pref

- 11.6.4. Prefs
- 11.6.5. Proto
- 11.6.6. ProtoExpert
- 11.6.7. ProtoField
- 11.6.8. Global Functions

11.7. Adding Information To The Dissection Tree

- 11.7.1. TreeItem
- 11.7.2. Example

11.8. Functions For Handling Packet Data

- 11.8.1. ByteArray
- 11.8.2. Tvb
- 11.8.3. TvbRange

11.9. Custom File Format Reading And Writing

- 11.9.1. CaptureInfo
- 11.9.2. CaptureInfoConst
- 11.9.3. File
- 11.9.4. FileHandler
- 11.9.5. FrameInfo
- 11.9.6. FrameInfoConst
- 11.9.7. Global Functions

11.10. Directory Handling Functions

- 11.10.1. Dir
- 11.10.2. Example
- 11.10.3. Example

11.11. Utility Functions

- 11.11.1. Global Functions

11.12. Handling 64-bit Integers

11.12.1. Int64

11.12.2. UInt64

11.13. Binary encode/decode support

11.13.1. Struct

11.14. GLib Regular Expressions

11.14.1. GRegex

12. User Interface

12.1. Introduction

12.2. The Qt Application Framework

12.2.1. User Experience Considerations

12.2.2. Qt Creator

12.2.3. Source Code Overview

12.2.4. Coding Practices and Naming Conventions

12.2.5. Other Issues and Information

12.3. Human Interface Reference Documents

13. Wireshark Tests

13.1. Quick Start

13.2. Test suite structure

13.2.1. Test Coverage And Availability

13.2.2. Suites, Cases, and Tests

13.2.3. pytest fixtures

13.3. Listing And Running Tests

13.4. Listing And Running Tests (pytest)

13.5. Adding Or Modifying Tests

14. Creating ASN.1 Dissectors

14.1. About ASN.1

14.2. ASN.1 Dissector Requirements
  14.2.1. Building An ASN.1-Based Plugin

14.3. Understanding Error Messages
14.4. Hand-Massaging The ASN.1 File
14.5. Command Line Syntax
14.6. Generated Files
14.7. Step By Step Instructions
14.8. Hints For Using Asn2wrs
  14.8.1. ANY And Parameterized Types
  14.8.2. Tagged Assignments
  14.8.3. Untagged CHOICEs
  14.8.4. Imported Module Name Conflicts

14.9. Simple ASN.1-Based Dissector
14.10. Conformance (.cnf) Files
  14.10.1. Example .cnf File
  14.10.2. Example packet-protocol-template.h File
  14.10.3. Example packet-protocol-template.c File

14.11. Conformance File Directive Reference
  14.11.1. #.END
  14.11.2. #.EXPORTS
  14.11.3. #.FN_BODY
  14.11.4. #.MODULE_IMPORT And #.INCLUDE
  14.11.5. #.MODULE_IMPORT
  14.11.6. #.INCLUDE
  14.11.7. #.NO_EMIT And #.USER_DEFINED
  14.11.8. #.PDU and #.PDU_NEW
  14.11.9. #.REGISTER and #.REGISTER_NEW

15. This Document’s License (GPL)

List of Figures

3.1. GitLab Workflow

6.1. Wireshark function blocks

11.1. A progress bar in action

11.2. A text window in action

11.3. An input dialog in action

List of Tables

1. Typographic Conventions

8.1. Control packet:

8.2. Commands and application for controls

11.1. Default background colors

11.2. Default background colors