Requirements

This document contains requirements for an utility that allows Wireshark to interpret the binary representations of C-language structs. While C structs seldom are exchanged across networks, they are sometimes used in inter-process communication. The purpose of the utility described here is to provide Wireshark with the capability of automatically dissecting the binary representation of a C struct, as long as its definition is known.

The expected work flow for the utility is to read one or more C header files, which contain struct definitions, and output Wireshark dissectors, implemented in Lua scripts. A configuration file or source code annotations in the header files may be used when additional configuration is required.

Table 1 lists the functional requirements and their priority, while Table 2 lists the non-functional requirements.

Table 1: Functional Requirements

ID	Description	Priority
FR01	The utility shall be able to read basic C language struct definitions, and generate a Wireshark dissector for the binary representation of the structs.	High
FR02	The utility shall support structs with the following basic data types: int, float, char, boolean, structs, unions, array and enums.	High
FR03	The utility must support the following C preprocessor directives and macros: `#include`, `#define`, `#if`, `WIN32`, `_WIN32`, `_WIN64`, `__sparc__`, `__sparc` and `sun`	Medium
FR04	The dissector shall be able to recognize invalid values for a struct member. Allowed ranges should be specified by configuration.	Low
FR05	A struct may have a header and/or trailer (other registered protocol), which must be configurable.	Low
FR06	The dissector shall display each struct member, and support structs within structs.	Medium
FR07	Configuration must support custom handling of specific data types. E.g. a 'time_t' may be interpreted to contain a unixtime value, and be displayed as a date.	Low
FR08	Configuration must support integer members which indicate that a variable number of other structs (array of structs) are following the current struct.	High
FR09	Configuration must support integer members which represent enumerated named value or a bit string.	Medium
FR10	The dissectors must be able to handle binary input which size and endian depends on originating platform. Flags within message headers should signal the platform.	High

Table 2: Non-Functional Requirements

ID	Description	Priority
NR01	The utility shall be able to run on Windows and Solaris, 32bit and 64bit, Intel and Sparc platform.	High
NR02	The utility shall be able to run in batch mode.	High
NR03	The utility needs to have flexible configuration.	Medium
NR04	The configuration needs to be well documented.	Low