Chapter 1

Requirements

1.1 List of requirements

Table 1.1 lists the functional requirements, while Table 1.2 lists the non-functional requirements. Each requirement have a priority (Pri) and a complexity (Cmp): High (H), Medium (M) or Low (L).

Table 1.1: Functional Requirements

ID	Description	Pri.	Cmp.
FR1	The utility must be able to read basic C language struct definitions from C header files.	H	
FR1-A	The utility must support the following basic data types: int, float, char and boolean.	H	L
FR1-B	The utility must support members of type enums.	H	L
FR1-C	The utility must support members of type structs.	H	M
FR1-D	The utility must support members of type unions.	M	M
FR1-E	The utility must support member of type array.	H	M
FR2	The utility must be able to generate lua-script for Wireshark dissectors for the binary representation of C struct.	H	
FR2-A	The dissector shall be able to display simple structs.	H	L
FR2-B	The dissector shall be able to support structs within structs.	M	M
FR2-C	The dissector must support Wiresharks built-in filter and search on attributes.	H	L
FR3	The utility must support C preprocessor directives and macros.	H	
FR3-A	The utility shall support #include.	H	L
FR3-B	The utility shall support #define and #if.	H	L
FR3-C	The utility shall support `WIN32`, `_WIN32`, `_WIN64`, `__sparc__`, `__sparc` and `sun`.	M	H
FR4	The utility must support user configuration.	M	
FR4-A	The dissector shall be able to recognize invalid values for a struct member. Allowed ranges should be specified by configuration.	L	L
FR4-B	Configuration must support integer members which represent enumerated named value or a bit string.	M	L
FR4-C	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.	L	M
FR5	A struct may have a header and/or trailer (other registered protocol). The configuration must support the use of integer members to indicate the number of other structs that will follow in the trailer	L	H
FR6	The dissectors must be able to handle binary input which size and endian depends on originating platform.	M	
FR6-A	Flags must be specified for each platform.	M	M
FR6-B	Flags within message headers should signal the platform.	M	H
FR7	The utility shall support parameters from command line.	H	
FR7-A	Command line shall support parameters for c-header file.	H	L
FR7-B	Command line shall support for configuration file.	H	L
FR7-C	Command line shall support batch mode of c-header and configuration file.	L	M
FR7-D	When running batch mode, dissectors that already are generated, shall not be regenerated, if the source are not modified since last run.	L	M

Table 1.2: Non-Functional Requirements

ID	Description	Pri.	Cmp.
NR1	The utility shall be able to run on latest Windows and Solaris operating system.	M	L
NR2	The dissector shall be able to run on Windows x86, Windows x86-64, Solaris x86, Solaris x86-64 and Solaris SPARC.	M	M
NR3	The utilities user interface shall be command line. No clicking!.	H	L
NR4	The configuration shall have sufficient documentation to allow a person with no previous knowledge of the system to be able to use it to generate LUA-scripts after X hours of reading.	M	M
NR5	The configuration should have sufficient documentation to allow a person, already proficient with the system, to understand the code well enough to be able to extend it's functionality after Y hours of reading.	M	M
NR6	The utility code should follow standard python coding convention as specified by PEP8, and try to follow python style guidelines defined by PEP20.	H	L
NR7	The utilities code should be documented by python docstrings which should explain the use of the code. Python modules, classes, functions and methods should have docstrings.	M	L