|  |
| --- |
| [**TOC**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#toc) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Network Working Group | L. Degioanni | | Internet-Draft | CACE Technologies | | Expires: September 2, 2004 | F. Risso | |  | Politecnico di Torino | |  | G. Varenni | |  | CACE Technologies | |  | March 2004 | |

**PCAP下一代转储文件格式**

**PCAP-DumpFileFormat**

**Status of this Memo**

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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.”

The list of current Internet-Drafts can be accessed at [**http://www.ietf.org/ietf/1id-abstracts.txt**](http://www.ietf.org/ietf/1id-abstracts.txt).

The list of Internet-Draft Shadow Directories can be accessed at [**http://www.ietf.org/shadow.html**](http://www.ietf.org/shadow.html).

This Internet-Draft will expire on September 2, 2004.

**Copyright Notice**

Copyright © The Internet Society (2004). All Rights Reserved.

**Abstract**

This document describes a format to dump captured packets on a file. This format is extensible and it is currently proposed for implementation in the libpcap/WinPcap packet capture library.

**Updates**

* [27 Jul 2009] Guy Harris: added some missing reserved block types in Appendix B.
* [27 Jul 2009] Guy Harris: fixed a typo in Appendix B. The range of standardized blocks are in the range 0x00000000-0x7FFFFFFF.
* [ 8 Feb 2008] Gianluca Varenni: better documentation for the format of the timestamps. Renamed the if\_tsaccur option into if\_tsresol.
* [22 Oct 2007] Gianluca Varenni: added a note related to 64-bit alignment. Specified that the option length field is the length without padding. typos here and there. Added some option examples.
* [17 Oct 2007] Ulf Lamping: Major review: "Interface ID" in "ISB" now 32 bits. isb\_starttime/isb\_endtime depends on if\_tsaccur. Lot's of other editing ...
* [ 8 Oct 2007] Ulf Lamping: Fixed several typos. Grouped the block types into mandatory, optional, experimental, obsolete.
* [14 Sep 2006] Gianluca Varenni: Added the block type code for Arinc 429 in AFDX Encapsulation Information Block
* [23 May 2006] Gianluca Varenni: Added the block type code for IRIG Timestamp Block
* [23 Apr 2006] Gianluca Varenni: Cleaned up Appendix C a bit: we should use the LINKTYPE\_xxx values from libpcap, not the DLT\_xxx ones. Fixed the introduction to the appendix and added some comments.
* [21 Mar 2006] Gianluca Varenni: Added a preliminary version of Appendix C, detailing the Standardized Link Types.
* [21 Mar 2006] Gianluca Varenni: Added a preliminary version of Appendix B, detailing the Standardized Block Type codes.
* [21 Mar 2006] Gianluca Varenni: Added the Enhanced Packet Block in section 2.2. Fixed a typo in the list: it's Interface Statistics Block, and not Capture Statistics Block.
* [21 Mar 2006] Gianluca Varenni: Fixed some minor typos in the document.
* [21 Mar 2006] Gianluca Varenni: Fixed an error in Packet Block: option pack\_hash should have code 3.
* [21 Mar 2006] Gianluca Varenni: Added the definition of the Enhanced Packet Block.
* [12 Mar 2006] Gianluca Varenni: Added option if\_tsoffset in the Interface Description Block.

目录

[**PCAP下一代转储文件格式** 1](#_Toc459819571)

[**PCAP-DumpFileFormat** 1](#_Toc459819572)

[**Status of this Memo** 1](#_Toc459819573)

[**Copyright Notice** 1](#_Toc459819574)

[**Abstract** 2](#_Toc459819575)

[**Updates** 2](#_Toc459819576)

[**1.  目标** 5](#_Toc459819577)

[**2.  文件结构**（**General File Structure）** 5](#_Toc459819578)

[**2.1.  块结构（General Block Structure）** 5](#_Toc459819579)

[**2.2.  块类型（Block Types）** 6](#_Toc459819580)

[**2.3.  逻辑块层次结构（Logical Block Hierarchy）** 7](#_Toc459819581)

[**2.4.  物理文件的布局（Physical File Layout）** 7](#_Toc459819582)

[**2.5.  选项（Options）** 9](#_Toc459819583)

[**2.6.  数据格式（Data format）** 10](#_Toc459819584)

[**3.  块定义（Block Definition）** 11](#_Toc459819585)

[**3.1.  节头块（Section Header Block）** 11](#_Toc459819586)

[**3.2.  接口描述块（Interface Description Block）** 12](#_Toc459819587)

[**3.3. 增强分组块（Enhanced Packet Block）** 15](#_Toc459819588)

[**3.4.  简单分组块（Simple Packet Block）** 17](#_Toc459819589)

[**3.5.  分组块（Packet Block）** 18](#_Toc459819590)

[**3.6.  名称解析块（Name Resolution Block）** 20](#_Toc459819591)

[**3.7.  接口统计块（Interface Statistics Block）** 22](#_Toc459819592)

[**4. 实验块（Experimental Blocks）** 23](#_Toc459819593)

[**4.1. 替代性分组块（Alternative Packet Blocks）** 23](#_Toc459819594)

[**4.2. 压缩块（Compression Block）** 24](#_Toc459819595)

[**4.3. 加密块（Encryption Block）** 24](#_Toc459819596)

[**4.4. 固定长度块（Fixed Length Block）** 25](#_Toc459819597)

[**4.5. 目录块（Directory Block）** 26](#_Toc459819598)

[**4.6. 流量统计和监控块（Traffic Statistics and Monitoring Blocks）** 26](#_Toc459819599)

[**4.7.事件/安全块（Event/Security Block）** 26](#_Toc459819600)

[**5.  推荐的扩展名: .pcapng** 27](#_Toc459819601)

[**6. 怎样增加供应商/域特定扩展** 27](#_Toc459819602)

[**7.  结论** 27](#_Toc459819603)

[**Appendix A.  Packet Block Flags Word** 28](#_Toc459819604)

[**Appendix B.  Standardized Block Type Codes** 28](#_Toc459819605)

[**Appendix C.  Standardized Link Type Codes** 29](#_Toc459819606)

[**Appendix D.  Link Layer Headers** 33](#_Toc459819607)

[**Authors' Addresses** 34](#_Toc459819608)

[**Full Copyright Statement** 34](#_Toc459819609)

[**Intellectual Property** 35](#_Toc459819610)

[**Acknowledgment** 35](#_Toc459819611)

**1.  目标**

交换分组痕迹的问题变得越来越关键。不幸的是，现在这个问题仍然没有标准的解决方案。其中最能够被接受的分组交换格式之一是在libpcap中定义的，这是相当古老的，并且从可扩展性的角度来看不适合某些特定的应用程序。

该文件提出了转储数据包踪迹（packet traces）的新格式，它追求一下几个目标：

* Extensibility: aside of some common functionalities, third parties should be able to enrich the information embedded in the file with proprietary extensions, which will be ignored by tools that are not able to understand them.可扩展性：除了一些常见的功能，第三方应该能够在文件中嵌入丰富的信息，无法理解这些信息的工具可以之间忽略它。
* 可移植性：一个捕捉跟踪必须包含读取时所需要的所有信息，而不依赖于捕获数据时的网络、硬件和操作系统。
* 合并/附加数据：应该可以在给定文件的末尾添加数据，并将得到的文件仍然必须是可读的。

**2.  文件结构**（**General File Structure）**

**2.1.  块结构（General Block Structure）**

一个pcapng格式的文件有各种块（block）组成，块的结构描述如下：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Type（块类型） |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length（块长度） |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

/ Block Body （块内容） /

/ /\* variable length, aligned to 32 bits \*/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length（块长度） |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|  |
| --- |
| **Figure 1: 块结构描述** |

每个字段的含义如下:

* 块类型: 32 bits，该字段标识当前块的类型，不同类型的块有不同的类型值。
* 块长度：32 bits，当前块的总长度，单位为byte，不包含块体的块的总长度为12 byte。
* 块体: 块的内容，长度不固定，但需要4字节对齐。
* 块长度：32 bits，当前块的总长度，单位为byte，与块体前面的块长度重复。

**2.2.  块类型（Block Types）**

目前标准化的块类型代码在附录B中被指定，他们已被分为以下四类：

**强制性块（MANDATORY blocks）**必须在每个文件中出现至少一次：

* [Section Header Block ( Section Header Block (mandatory) )](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionshb) : it defines the most important characteristics of the capture file.[节头块([Section Header Block](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionshb))：](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionshb)它定义了捕获文件的最重要的特征。
* [接口描述块(Interface Description Block )](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionidb) ：它定义了用于捕获流量的接口的最重要的特征。

**OPTIONAL blocks can appear in a file:可选块（OPTIONAL blocks）**可以出现在文件中：

* [Enhanced Packet Block ( Enhanced Packet Block (optional) )](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionepb) : it contains a single captured packet, or a portion of it.[增强分组块（[Enhanced Packet Block](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionepb)）：](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionepb)它包含一个捕获数据包，或它的一部分，代表了原有的[分组块（Packet Block）的演进。](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionpb)
* [简单分组块 ( Simple Packet Block)](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionpbs) ：它包含一个捕获数据包，或它的一部分，以及很少的关于数据包描述信息。
* [名称解析块( Name Resolution Block )](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionnrb)：它定义由存在于数据包转储和规范名称的对应数字地址的映射。
* [接口统计块( Interface Statistics Block )](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionisb)：它定义了如何存储一些统计数据（例如分组丢弃，等等），它对于了解数据包捕获的条件是有用的。

**已过时块（OBSOLETE blocks）**不应该出现在新写入的文件（但在这里留下供参考）：

* [Packet Block ( Packet Block (obsolete!) )](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionpb) : it contains a single captured packet, or a portion of it.[分组块（Packet Block） ：](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionpb)它包含单个捕获分组，或它的一部分。它应该被视为过时的，有[增强分组块而取得。](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionepb)

**实验块（EXPERIMENTAL blocks）**被认为是有意义的，但作者认为他们应该被定义以前更深入的讨论：

* Alternative Packet BlocksAlternative Packet Blocks替代性包块
* Compression Block压缩块
* Encryption Block加密块
* Fixed Length Block固定长度块
* Directory Block目录块
* Traffic Statistics and Monitoring Blocks流量统计和监控模块
* Event/Security Blocks事件/安全模块

**2.3.  逻辑块层次结构（Logical Block Hierarchy）**

下图用树状结构的形式展示了当前所定义的块的层次结构：

Section Header

|

+- Interface Description

| +- Simple Packet

| +- Enhanced Packet

| +- Interface Statistics

|

+- Name Resolution

|  |
| --- |
| **Figure 2: 一个pcapng文件的逻辑块层次** |

**2.4.  物理文件的布局（Physical File Layout）**

一个pcapng格式的文件必须具有节头块（Section Header Block）开始。 However, more than one Section Header Block can be present on the dump, each one covering the data following it till the next one (or the end of file).在一个在在打发斯蒂芬在一个dump文件中可以存在多于一个节头块，每一个节头块的作用范围从紧随其后的数据开始，直至下一个节头块（或文件结束）。

In case an application cannot read a Section because of different version number, it must skip everything until the next Section Header Block.如果如果因为版本号的不同，应用程序可能无法读取一节（Section）的内容，必须跳过这一节，直到下一节头块。 Note that, in order to properly skip the blocks until the next section, all blocks must have the fields Type and Length at the beginning.需要注意的是，为了适当地跳过块，直到下一节，所有的块必须以类型和长度字段开始。This is a mandatory requirement that must be maintained in future versions of the block format.这是一个必须在块格式的未来版本进行维护的强制性要求。

[图3](mhtml:file://F:\pcap下一代转储文件格式.mht!https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#fssample-SHB)展示出了一个典型的文件结构中，整个文件只包含一个节头块。

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| SHB v1.0 | Data |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|  |
| --- |
| **Figure 3: 一个典型的文件结构** |

图4显示了包含三个节头块的文件，通常是文件拼接的结果。只能理解文件格式的1.0版本的应用程序可以跳过中间部分，并继续处理第三个节头块之后的数据包。

|-- 1st Section --|-- 2nd Section --|-- 3rd Section --|

| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| SHB v1.0 | Data | SHB V1.1 | Data | SHB V1.0 | Data |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|  |
| --- |
| **Figure 4: 包含3个节头块的文件.** |

[Figure 5](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#fssample-minimum)显示了一个堪比“经典libpcap格式的”pcapng文件结构，它包含一个节头块（SHB），一个单一的接口描述块（IDB）和几个增强分组块（EPB）。

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| SHB | IDB | EPB | EPB | ... | EPB |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|  |
| --- |
| **Figure 5: 一个经典的 pcapng文件结构** |

[**Figure 6**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#fssample-full) shows a complex example file. In addition to the minimum file above, it contains packets captured from three interfaces, and also includes some Name Resolution Blocks (NRB) and an Interface Statistics Block (ISB).

[Figure 6](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#fssample-full)是一个复杂的示例文件。除了上述的最小化的文件，它包含了来自三个接口捕获的数据包，并且还包括一些名称解析模块（NRB）和接口统计块（ISB）。

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| SHB | IDB | IDB | IDB | EPB | EPB | NRB | ... | EPB | ISB | NRB | EPB | EPB |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|  |
| --- |
| **Figure 6: 一个复杂的pcapng文件结构** |

最后一个例子应该很明显，该块结构使得相比于传统的libpcap格式的文件格式非常灵活。

**2.5.  选项（Options）**

所有的块体都有可能嵌入可选字段。Optional fields can be used to insert some information that may be useful when reading data, but that is not really needed for packet processing.可选字段可以用来插入一些信息，这些信息在读取数据时可能是有用的，而在数据包处理时并不真正需要。Therefore, each tool can either read the content of the optional fields (if any), or skip some of them or even all at once.因此，每个工具可以读取可选字段的内容（如果有的话），或跳过他们中的一些或甚至全部。

Skipping all the optional fields at once is straightforward because most of the blocks are made of a first part with fixed format, and a second optional part.大部分的块的第一部分具有固定的格式，第二部分是可选的内容，因此跳过所有的可选字段变得很简单。在通用块结构中中中中，块长度字段可以被用来跳过一切中间内容，直到下一个块。

Options are a list of Type - Length - Value fields, each one containing a single value:选项是一个TLV（类型-长度-值）记录的列表，每一个记录含有一个选项值，格式如下：

* Option Type (2 bytes): it contains the code that specifies the type of the current TLV record.选项类型（2字节）：包含指定当前TLV记录的类型的代码。Option types whose Most Significant Bit is equal to one are reserved for local use;最高位等于1的选项类型被保留给本地使用therefore, there is no guarantee that the code used is unique among all capture files (generated by other applications).。。因此，也不能保证所用的码是所有捕获文件中是唯一的（通过其他应用程序产生的）。In case of vendor-specific extensions that have to be identified uniquely, vendors must request an Option Code whose MSB is equal to zero.在具有被唯一标识特定供应商的扩展的情况下，供应商必须申请选项代码的MSB等于零。
* Option Length (2 bytes): it contains the actual length of the following 'Option Value' field without the padding bytes.选项长度（2字节）：它包含以下“选项值”字段的实际长度，不包含填充字节。
* Option Value (variable length): it contains the value of the given option, aligned to a 32-bit boundary.选项值（可变长度）：它包含给定选项的值，4字节对齐。The actual length of this field (ie without the padding bytes) is specified by the Option Length field.该字段（不包含填充字节）的实际长度由选项长度字段指定。

Options may be repeated several times (eg an interface that has several IP addresses associated to it) TODO: mention for each option, if it can/shouldn't appear more than one time.选项可以被重复几次（如：具有与其关联多个IP地址的接口）。要注意每个选项，看它是否能够/不应该出现多于一次。The option list is terminated by a Option which uses the special 'End of Option' code (opt\_endofopt).选项列表由一个特殊的选项（opt\_endofopt）做为结束标志。

The format of the optional fields is shown in[Figure 7展示了](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#formatopt).可选字段的格式。

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Option Code | Option Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

/ Option Value /

/ /\* variable length, aligned to 32 bits \*/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

/ /

/ . . . other options . . . /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Option Code == opt\_endofopt | Option Length == 0 |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|  |
| --- |
| **Figure 7: 选项的格式.** |

下面的代码可以永远存在于任何可选字段:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Code** | **Length** | **Description** | **Example(s)** |
| opt\_endofopt | 0 | 0 | 可选字段的结束选项，在一个块的This block cannot be repeated within a given list of options.选项列表中不能重复 |  |
| opt\_comment | 1 | 可变的 | 包含关联到当前块的注释的UTF-8字符串 | "This packet is the beginning of all of our problems" / "Packets 17-23 showing a bogus TCP retransmission, as reported in bugzilla entry 1486!" / "Captured at the southern plant" / "I've checked again, now it's working ok" / ... |

**2.6.  数据格式（Data format）**

**字节序（Endianess）**

Data contained in each section will always be saved according to the characteristics (little endian / big endian) of the dumping machine.包含在各节（section）中的数据将总是根据捕获数据包的主机的特性（小端/大端）被保存。This refers to all the fields that are saved as numbers and that span over two or more bytes.这里的数据包括所有的大小为2个字节或多个字节的所有数值类型的字段。

使用使用生成数据的主机的本机格式来保存每一节数据的方法是更加有效的，因为在这个主机上进行读/写操作时避免了数据的转换，这在产生/处理捕获数据文件时也是最常见的情况。

Please note: The endianess is indicated by the.请注意：字节顺序是由[节头块来标明，](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionshb)As this block can appear several times in a pcapng file, a single file can contain both endianess variants!由于此块还可以pcapng文件中出现多次，一个文件可以同时包含不同的字节序类型！

**Alignment对齐（**Alignment**）**

Most (all?) fields of this specification uses proper alignment for 16- and 32-bit values.本规范的大多数（所有？）字段使用2字节（16bits）对齐或16位和32位值。This makes it easier and faster to read/write file contents if using techniques like memory mapped files.

The alignment bytes (marked in this document eg with "aligned to 32 bits") should be filled with zero bytes (TODO: is this requirement a good idea for the sake of performance / do we want to allow bogus bytes here?).对齐字节时，填充的字节的值应被设置为0。

Please note: 64-bit values are not aligned to 64-bit boundaries.请注意：64位的数值不是64位边界对齐，This is because the file is naturally aligned to 32-bit boundaries only.这是因为该文件仅仅被自然对齐到32位边界。在读/写64位的数值时Special care should be taken when reading and writing such values.应特别注意，这点与TODO: the spec is not too consistent wrt how 64-bit values are saved.64位的数值的存储方式不太一致。in the Packet blocks we clearly specify where the low and high 32-bits of a 64-bit timestamp should be saved.在分组块中我们清楚地指定一个64位时间戳的低和高32位应该被怎样存储。In the SHB we do use the endianess of the machine when we save the section length.在SHB中，当我们保存块长度（section）时，我们使用的机器的字节顺序。

**3.  块定义（Block Definition）**

本节详细当前定义的块体的格式。

**3.1.  节头块（Section Header Block）**

节头块是强制性的，It identifies the beginning of a section of the capture dump file.它确定了捕获转储文件的一节的开头。The Section Header Block does not contain data but it rather identifies a list of blocks (interfaces, packets) that are logically correlated.节头块不包含数据，而是标识了逻辑上相关的块（接口块，数据块）列表。 Its format is shown in [Figure 8](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#formatSHB) 展示了 .其格式：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

0 | Block Type = 0x0A0D0D0A |

+---------------------------------------------------------------+

4 | Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

8 | Byte-Order Magic |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

12 | Major Version | Minor Version |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

16 | |

| Section Length |

| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

24 / /

/ Options (variable) /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 8:节头块格式** |

该字段的含义是：

* Block Type: The block type of the Section Header Block is the integer corresponding to the 4-char string "\r\n\n\r" (0x0A0D0D0A).块类型：节头块的块类型是对应于4字符的字符串为“\r\n\n\r”（0x0A0D0D0A）的整数。
* 块长度：在描述了这个块的总大小，参考2.1节（通用块结构）。
* 字节顺序幻数：它的值是十六进制数0x1A2B3C4D，这个数字可以用来区分当前节（section）采用的字节序是Little-Endian还是Big-Endian。
* 主版本号：number类型，当前值是1。
* 次版本号：number类型，当前值是0。
* 节长度：64位值，标明本节数据的长度，不包括节头块本身的长度。节长度值等于-1 (0xFFFFFFFFFFFFFFFF) 意味着当前节的大小没有定义，跳过这一节的唯一方法是解析它所包含的块。请注意，如果这个字段是有效的（即不-1），它的值始终是32bits对齐，因为所有的块被排列为32bits边界对齐。此外，在访问该字段时应特别注意：在该文件中所有块是32bits对齐的，但该字段是不能保证是64bits边界对齐的，这可能在64位工作站有问题。
* 选项：可选选项列表，参考[第2.5节](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionopt)。

Adding new block types or options would not necessarily require that either Major or Minor numbers be changed, as code that does not know about the block type or option could just skip it;添加新的块类型或选项并不一定要求主版本号或次版本号被改变，因为代码不知道该块类型或选项时可以只是跳过它; only if skipping a block or option does not work should the minor version number be changed.只有跳过块或选项时无法正常工作时次版本号需要进行更改。

除了在第2.5节Aside from the options defined in , the following options are valid within this block:定义的选项之外，以下选项是该块内是有效的：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Code** | **Length** | **Description** | **Example(s)** |
| shb\_hardware | 2 | 可变的 | An UTF-8 string containing the description of the hardware used to create this section. | "x86 Personal Computer" / "Sun Sparc Workstation" / ... |
| shb\_os | 3 | 可变的 | An UTF-8 string containing the name of the operating system used to create this section. | "Windows XP SP2" / "openSUSE 10.2" / ... |
| shb\_userappl | 4 | 可变的 | An UTF-8 string containing the name of the application used to create this section. | "dumpcap V0.99.7" / ... |

**3.2.  接口描述块（Interface Description Block）**

接口描述块是强制性的。This block is needed to specify the characteristics of the network interface on which the capture has been made.该块被用来指定进行捕获的网络接口的特性。 In order to properly associate the captured data to the corresponding interface, the Interface Description Block must be defined before any other block that uses it;为了把捕获的数据关联到相应的接口，该接口描述块必须在使用它的任何其它块之前定义; therefore, this block is usually placed immediately after the Section Header Block.因此，此块通常会紧跟着节头块放置。

An Interface Description Block is valid only inside the section which it belongs to.一个接口描述块只在其所在的一节内有效的。Figure 9给出了The structure of a Interface Description Block is shown in .一个接口说明块的结构描述：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

0 | Block Type = 0x00000001 |

+---------------------------------------------------------------+

4 | Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

8 | LinkType | Reserved |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

12 | SnapLen |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

16 / /

/ Options (variable) /

/ / +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length | +---------------------------------------------------------------+

|  |
| --- |
| **Figure 9: 接口描述块的格式.** |

该字段的含义是：

* Block Type: The block type of the Interface Description Block is 1.块类型：接口描述块的块类型为1。
* Block Total Length: total size of this block, as described in [Section 2.1 ( General Block Structure )](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionblock) .块长度：参考[2.1节（通用块结构）](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionblock)。
* LinkType: a value that defines the link layer type of this interface.链路类型：标识链路的类型The list of Standardized Link Layer Type codes is available in .，标准化链路类型代码列表参考附录C。
* SnapLen: maximum number of bytes dumped from each packet.快照长度（snaplen）：所保存的单个数据包的最大长度，一个The portion of each packet that exceeds this value will not be stored in the file.数据包数据数据包超过该值的部分将不被存储在文件中Options: optionally, a list of options (formatted according to the rules defined in ) can be present.。
* 选项：可选选项列表，参考[第2.5节（选项）](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionopt)。

Interface ID: Tools that write / read the capture file associate a progressive 16-bit number (starting from '0') to each Interface Definition Block.接口ID：在每个接口描述块中，每个写/读捕获文件的工具都关联一个递增的16位数字（从开始'0'）。This number is unique within each Section and uniquely identifies the interface (inside the current section);这个数字在每一节中是唯一的，唯一在当前节（Section）的标识接口; therefore, two Sections can have interfaces identified by the same identifiers.因此，两个节（Section）可以具有由相同的标识符标识的接口。This unique identifier is referenced by other blocks (eg Packet Block) to point out the interface the block refers to (eg the interface that was used to capture the packet).这种唯一的标识符被其它块（例如分组块）引用来指出所述块对应的接口（例如被用来捕获该包的接口）。(TODO - It would be nice, to have a "invalid Interface ID" defined, eg 0xFFFFFFFF)（TODO - 这将是很好，有一个“无效的接口ID”的定义，例如为0xFFFFFFFF）

In addition to the options defined in , the following options are valid within this block:除了在第2.5节​​定义的选项，以下选项在该块内是有效的：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Code** | **Length** | **Description** | **Example(s)** |
| if\_name | 2 | 可变的 | 包含用于捕获数据的设备名称的UTF-8字符串 | "eth0" / "\Device\NPF\_{AD1CE675-96D0-47C5-ADD0-2504B9126B68}" / ... |
| if\_description | 3 | 可变的 | 包含用于捕获数据的设备的描述的UTF-8字符串 | "Broadcom NetXtreme" / "First Ethernet Interface" / ... |
| if\_IPv4addr | 4 | 8 | 网络地址和子网掩码。This option can be repeated multiple times within the same Interface Description Block when multiple IPv4 addresses are assigned to the interface.当多个IPv4地址被分配给该接口，该选项可以重复多次，在同一个接口描述块内 | 192 168 1 1 255 255 255 0 |
| if\_IPv6addr | 5 | 17 | 网络地址和前缀长度（存储在最后一个字节）。 This option can be repeated multiple times within the same Interface Description Block when multiple IPv6 addresses are assigned to the interface.当多个IPv6地址被分配给该接口，该选项可以重复多次，在同一个接口描述块内。 | 2001:0db8:85a3:08d3:1319:8a2e:0370:7344/64 is written (in hex) as "20 01 0d b8 85 a3 08 d3 13 19 8a 2e 03 70 73 44 40" |
| if\_MACaddr | 6 | 6 | 接口硬件的MAC地址（48位） | 00 01 02 03 04 05 |
| if\_EUIaddr | 7 | 8 | 接口硬件EUI地址（64位），如果有的话 | TODO: give a good example |
| if\_speed | 8 | 8 | 接口速度（单位为bps） | 100000000 for 100Mbps |
| if\_tsresol | 9 | 1 | 时间戳分辨率。If the Most Significant Bit is equal to zero, the remaining bits indicates the resolution of the timestamp as as a negative power of 10 (eg 6 means microsecond resolution, timestamps are the number of microseconds since 1/1/1970).如果最高有效位等于零，其余的位表示时间戳的分辨率为10的负幂次方（例如，6表示10 ^ -6，也就是微秒的分辨率，时间戳是自从1/1/1970微秒的数量）。 If the Most Significant Bit is equal to one, the remaining bits indicates the resolution as as negative power of 2 (eg 10 means 1/1024 of second).如果最高有效位等于一，其余的位表示的分辨率为2的负幂次方（例如10表示1/1024秒）。 If this option is not present, a resolution of 10^-6 is assumed (ie timestamps have the same resolution of the standard 'libpcap' timestamps).如果此选项不存在，则假定为10 ^ -6分辨率（即时间戳有标准“的libpcap”时间戳相同的分辨率）。 | 6 |
| if\_tzone | 10 | 4 | 对于GMT支持时区 | TODO: give a good example |
| if\_filter | 11 | 可变的 | 流量捕获过滤器(e.g. "capture only TCP traffic"). 选项数据的第一个字节是所使用的过滤器的代码(e.g. if this is a libpcap string, or BPF bytecode, and more) | 00 "tcp port 23 and host 10.0.0.5" |
| if\_os | 12 | 可变的 | 安装该接口的计算机的操作系统名称的UTF-8字符串。This can be different from the same information that can be contained by the Section Header Block ( [Section 3.1 ( Section Header Block (mandatory) )](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionshb) ) because the capture can have been done on a remote machine.这可以与节头块包含相同信息的不同（因为捕获可以在远程机器上完成） | "Windows XP SP2" / "openSUSE 10.2" / ... |
| if\_fcslen | 13 | 1 | 一个整型值，指定了该接口对应的帧校验序列（位）的长度。对于链路层的FCS长度可以改变，分组块标志词可以使用（请参阅附录A） | 4 |
| if\_tsoffset | 14 | 8 | 一个64bits整型值，它定义了一个偏移量（以秒计）， 这个偏移量被添加到每个包的时间戳中以便得到一个数据包的绝对时间戳。 If the option is missing, the timestamps stored in the packet must be considered absolute timestamps.如果该选项被丢失，存储在分组中的时间戳，将被认为是绝对时间戳。The time zone of the offset can be specified with the option if\_tzone.偏移的时区可以用选项if\_tzone指定 | 1234 |

**3.3. 增强分组块（Enhanced Packet Block）**

一个增强分组块是用来存储来自网络的数据包的标准容器。The Enhanced Packet Block is optional because packets can be stored either by means of this block or the Simple Packet Block, which can be used to speed up dump generation.增强型分组块是可选的，因为数据包可以通过该块或简单分组块进行存储。Figure 10展示了The format of an Enhanced Packet Block is shown in .增强分组块的格式。

增强分组块是原来分组块（Packet Block）之上进行的改进：

* it stores the Interface Identifier as a 32bit integer value.它存储接口标识符为一个32bits整数，这是当一个捕获存储来自大量接口的数据包时的需求。
* 不同于[分组块，](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionpb)当前数据包与前一个数据包之间被丢弃的数据包的个数没有保存在块头中，而是当作一个选项保存在块体中。

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

0 | Block Type = 0x00000006 | +---------------------------------------------------------------+

4 | Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

8 | Interface ID |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

12 | Timestamp (High) |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

16 | Timestamp (Low) |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

20 | Captured Len |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

24 | Packet Len |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

28 / /

/ Packet Data /

/ /\* variable length, aligned to 32 bits \*/ /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

/ /

/ Options (variable) /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 10: 增强分组块格式.** |

增强型分组块具有以下字段：

* Block Type: The block type of the Enhanced Packet Block is 6.块类型：增强型分组块的块类型是6。
* Block Total Length: total size of this block, as described in 块 .长度：块长度：描述了这个块的总大小，具体参考2.1节（通用块结构）。
* Interface ID: it specifies the interface this packet comes from;接口ID：它指定该数据包来自于的接口， ) of this field.这里的值应该与其接口描述块（参考3.2节）中的值一致。
* Timestamp (High) and Timestamp (Low): high and low 32-bits of a 64-bit quantity representing the timestamp.时间戳（高）和时间戳（低）：表示一个64bits时间戳的高、低32位。The timestamp is a single 64-bit unsigned integer representing the number of units since 1/1/1970.时间戳是一个64bits无符号整数，表示自1/1/1970以来所经过的时间单位数，The way to interpret this field is specified by the 'if\_tsresol' option (see [具体具体的时间单位有](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#formatidb) ) of the Interface Description block referenced by this packet.由接口描述块的“if\_tsresol”选项的值确定。Please note that differently from the libpcap file format, timestamps are not saved as two 32-bit values accounting for the seconds and microseconds since 1/1/1970.请注意，不同于libpcap的文件格式，这里的时间戳保存的不是1/1/1970以来经过的秒和微秒数，只是把一个64bits整数当作两个32bits整数来保存。
* Captured Len: number of bytes captured from the packet (ie the length of the Packet Data field).捕获长度：当前数据包被捕获的字节数（即分组数据字段的长度）).，这将是实际分组长度和快照长度（在Figure 9中定义）中的最小值。The value of this field does not include the padding bytes added at the end of the Packet Data field to align the Packet Data Field to a 32-bit boundary这个字段的值不包括为了32bits边界对齐而填充的字节长度。
* Packet Len: actual length of the packet when it was transmitted on the network.分组长度：表示在网络上传输的数据包的实际长度。It can be different from Captured Len if the user wants only a snapshot of the packet.如果用户仅仅希望保留当前数据包的快照时，这个值可以与捕获长度（Captured Len）不相同。
* Packet Data: the data coming from the network, including link-layer headers.分组数据：从网络中捕获的数据，包括链路层报头。The actual length of this field is Captured Len.这个字段的实际长度是捕获长度（Captured Len）。 The format of the link-layer headers depends on the LinkType field specified in the Interface Description Block (see .链路层报头的格式取决于接口描述块中链路类型（LinkType）字段的值，参考3.2节和[附录D（链路层头）](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#appendixLinkHeaders) 。
* Options: optionally, a list of options (formatted according to the rules defined in ) can be present.选项：可选选项列表，选项格式参见第2.5节（选项）

除了在第2.5节​​定义的选项，以下选项在该块内是有效的：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Code** | **Length** | **Description** | **Example(s)** |
| epb\_flags | 2 | 4 | 包含链路层信息的标志词。一个完整的规格定义参见附录A | 0 |
| epb\_hash | 3 | 可变的 | 此选项包含该分组的哈希值。 The first byte specifies the hashing algorithm, while the following bytes contain the actual hash, whose size depends on the hashing algorithm, and hence from the value in the first bit.第一字节指定的哈希算法，而接下来的字节包含实际的散列，其大小取决于所述散列算法。 The hashing algorithm can be: 2s complement (algorithm byte = 0, size=XXX), XOR (algorithm byte = 1, size=XXX), CRC32 (algorithm byte = 2, size = 4), MD-5 (algorithm byte = 3, size=XXX), SHA-1 (algorithm byte = 4, size=XXX).散列算法可以是：二进制补（算法字节= 0，大小= XXX），异或（算法字节= 1，大小= XXX）的CRC32（算法字节= 2，大小= 4），MD-5（算法字节= 3，大小= XXX），SHA-1（算法字节= 4，大小= XXX）。 The hash covers only the packet, not the header added by the capture driver: this gives the possibility to calculate it inside the network card.散列仅覆盖该分组，而不是由捕获驱动添加的报头。The hash allows easier comparison/merging of different capture files, and reliable data transfer between the data acquisition system and the capture library.散列使得不同的捕获文件的比较/合并，以及在数据采集系统和捕获库之间可靠的传输数据更加容易 (TODO: the text above uses "first bit", but shouldn't this be "first byte"?!?) | TODO: give a good example |
| epb\_dropcount | 4 | 8 | 一个64位整数值，表示在当前包和前一个包之间被接口和操作系统丢弃的数据包数量 | 0 |

**3.4.  简单分组块（Simple Packet Block）**

简单分组块是可选的，用于存储来自网络的分组的轻量级容器。

简单分组块类似于分组块（见第3.5节），但它更小、处理起来更简单，只包含了一组最小信息集合。当性能或占用空间是关键因素时，这个块将优先于标准分组块被使用，如在持续流量转储应用程序。一个捕获文件可以同时包含分组块和简单的数据包块：例如硬件资源成为瓶颈时，捕捉工具可以从分组块切换到简单分组块。

在简单分组块中不包含接口ID字段。因此，必须假定所有的简单分组块都是在第一个接口描述块中指定的接口上捕获的。

Figure 11展示了简单的分组块的格式：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

0 | Block Type = 0x00000003 |

+---------------------------------------------------------------+

4 | Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

8 | Packet Len |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

12 / /

/ Packet Data /

/ /\* variable length, aligned to 32 bits \*/ /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 11:简单分组块格式** |

在简单分组块具有以下字段：

* Block Type: The block type of the Simple Packet Block is 3.块类型：简单的分组块的块类型为3。
* Block Total Length: total size of this block, as described in 块 .块长度：描述了这个块的总大小。
* Packet Len: actual length of the packet when it was transmitted on the network.分组长度：在网络上传输的数据包的实际长度。Can be different from captured len if the packet has been truncated by the capture process.Packet Data: the data coming from the network, including link-layers headers.
* 分组数据：从网络中捕获的数据，包括链路层报头。The actual length of this field is Captured Len.The length of this field can be derived from the field Block Total Length, present in the Block Header, and it is the minimum value among the SnapLen (present in the Interface Description Block) and the Packet Len (present in this header).此字段的长度可以根据块头中的块长度字段计算得到，而且它是Snaplen（在接口描述块中）和Packet Len（在当前块中）中的最小值。

The Simple Packet Block does not contain the timestamp because this is often one of the most costly operations on PCs.简单分组块不包含时间戳，因为这通常是在PC上的最耗时的操作之一。 Additionally, there are applications that do not require it;此外，有些应用也不需要它; eg an Intrusion Detection System is interested in packets, not in their timestamp.例如入侵检测系统只对数据包感兴趣，而不是在它们的时间戳。

A Simple Packet Block cannot be present in a Section that has more than one interface because of the impossibility to refer to the correct one (it does not contain any Interface ID field).简单分组块不能存在于具有两个以上接口的节中，因为无法指向正确的接口。

The Simple Packet Block is very efficient in term of disk space: a snapshot whose length is 100 bytes requires only 16 bytes of overhead, which corresponds to an efficiency of more than 86%.简单分组块是在磁盘空间利用方面非常有效：一个长度为100字节的快照，只需要16个字节的开销，其对应于超过86％的有效性。

**3.5.  分组块（Packet Block）**

分组块被标记为已过时，更好地使用增强分组块来代替它。

分组块是用于存储来自网络的信息包的标准容器。分组块是可选的，因为数据包可以通过该块或简单分组块进行存储。Figure 12展示了The format of an Enhanced Packet Block is shown in .分组块的格式。

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

0 | Block Type = 0x00000002 |

+---------------------------------------------------------------+

4 | Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

8 | Interface ID | Drops Count |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

12 | Timestamp (High) |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

16 | Timestamp (Low) |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

20 | Captured Len |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

24 | Packet Len |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

28 / /

/ Packet Data /

/ /\* variable length, aligned to 32 bits \*/ /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

/ /

/ Options (variable) /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 12:分组块的格式** |

分组块具有以下字段：

* Block Type: The block type of the Packet Block is 2.块类型：增强型分组块的块类型是2。
* Block Total Length: total size of this block, as described in 块 .长度：块长度：描述了这个块的总大小，具体参考2.1节（通用块结构）。
* Interface ID: it specifies the interface this packet comes from;接口ID：它指定该数据包来自于的接口， ) of this field.这里的值应该与其接口描述块（参考3.2节）中的值一致。
* Drops Count: a local drop counter.丢包数：局部计数器。It specifies the number of packets lost (by the interface and the operating system) between this packet and the preceding one.它记录了在该分组和前一个分组之间由接口和操作系统丢弃数据包的数目。The value xFFFF (in hexadecimal) is reserved for those systems in which this information is not available.值xFFFF（十六进制）被保留用于在此信息不可用的系统。
* Timestamp (High) and Timestamp (Low): timestamp of the packet.时间戳（高）和时间戳（低）：该数据包的时间戳， ).时间戳的格式与在增强分组块中的定义是相同的。
* 捕获长度：当前数据包被捕获的字节数（即分组数据字段的长度）).，这将是实际分组长度和快照长度（在Figure 9中定义）中的最小值。The value of this field does not include the padding bytes added at the end of the Packet Data field to align the Packet Data Field to a 32-bit boundary这个字段的值不包括为了32bits边界对齐而填充的字节长度。
* Packet Len: actual length of the packet when it was transmitted on the network.分组长度：表示在网络上传输的数据包的实际长度。It can be different from Captured Len if the user wants only a snapshot of the packet.如果用户仅仅希望保留当前数据包的快照时，这个值可以与捕获长度（Captured Len）不相同。
* Packet Data: the data coming from the network, including link-layer headers.分组数据：从网络中捕获的数据，包括链路层报头。The actual length of this field is Captured Len.这个字段的实际长度是捕获长度（Captured Len）。 The format of the link-layer headers depends on the LinkType field specified in the Interface Description Block (see .链路层报头的格式取决于接口描述块中链路类型（LinkType）字段的值，参考3.2节和[附录D（链路层头）](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#appendixLinkHeaders) 。
* Options: optionally, a list of options (formatted according to the rules defined in ) can be present.选项：可选选项列表，选项格式参见第2.5节（选项）

除了在第2.5节​​定义的选项，以下选项在该块内是有效的：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Code** | **Length** | **Description** | **Example(s)** |
| pack\_flags | 2 | 4 | 见增强分组块的epb\_flags | 0 |
| pack\_hash | 3 | variable | 见增强分组块的epb\_hash | TODO: give a good example |

**3.6.  名称解析块（Name Resolution Block）**

名称解析块用来把数字地址（存在于捕获的数据包）及其相应的规范名称关联起来，这是可选的。通过在文件中保存名称，这可以防止名称解析中的延迟时间。名称解析块避免了在跟踪捕获文件时发出大量的DNS的请，并且允许在未联网的计算机上读取捕获数据时进行名称解析。

名称解析块通常被放置在文件的开头，名称解析块可以通过处理捕获文件的工具进行二次添加，如网络分析仪的工具来添加。

名称解析块的格式如图Figure 13所示：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

0 | Block Type = 0x00000004 |

+---------------------------------------------------------------+

4 | Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

8 | Record Type | Record Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

12 / Record Value /

/ /\* variable length, aligned to 32 bits \*/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

. .

. . . . other records . . . .

. .

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Record Type == end\_of\_recs | Record Length == 00 |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

/ /

/ Options (variable) /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 13:名称解析块的格式** |

名称解析模块具有以下字段：

* Block Type: The block type of the Name Resolution Block is 4.块类型：名称解析块的块类型为4。
* Block Total Length: total size of this block, as described in块长度：描述了这个块的总大小，具体参考2.1节（通用块结构）。

This is followed by a zero-terminated list of records (in the TLV format), each of which contains an association between a network address and a name.这之后紧跟着一组以零终止符结束的TLV格式的记录列表，其每一条记录包含一个网络地址和名字之间的关联。There are three possible types of records:有三种类型的记录：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Code** | **Length** | **Description** | **Example(s)** |
| nres\_endofrecord | 0 | 0 | 它标识了名称解析记录的结束，也可能是其它选项（如果有的话）的开始 |  |
| nres\_ip4record | 1 | 可变的 | 指定IPv4地址（包含在第一4字节），接着包含该地址的DNS条目的一个或多个零结尾字符串 | 127 0 0 1 "localhost" |
| nres\_ip6record | 2 | 可变的 | 指定IPv6地址（包含在第一16个字节），然后是包含该地址的DNS条目的一个或多个零结尾字符串 | TODO: give a good example |

每个记录值都是32bits边界对齐的，相应的记录长度反映了记录值的实际长度。

在名称解析记录列表后，是可选的选项列表。

除了在第2.5节定义的选项，以下选项在该块内有效：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Code** | **Length** | **Description** | **Example(s)** |
| ns\_dnsname | 2 | 可变的 | 包含主机名称（DNS服务器）的UTF-8字符串，用来进行名称解析 | "our\_nameserver" |
| ns\_dnsIP4addr | 3 | 4 | DNS服务器的IPv4地址 | 192 168 0 1 |
| ns\_dnsIP6addr | 4 | 16 | DNS服务器的IPv6地址 | TODO: give a good example |

**3.7.  接口统计块（Interface Statistics Block）**

接口统计块是可选的，它包含给定接口的捕获统计信息。在当前节（Section）中，接口相关的The statistics are referred to the interface defined in the current Section identified by the Interface ID field.统计信息被接口ID字段中标识的接口引用。An Interface Statistics Block is normally placed at the end of the file, but no assumptions can be taken about its position - it can even appear multiple times for the same interface.接口统计块通常被放置在该文件的末尾，但是没有明确要求关于其放置位置 - 它甚至可以出现多次对相同的接口。

The format of the Interface Statistics Block is shown in Figure 14展示了 .接口统计块的格式

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

0 | Block Type = 0x00000005 |

+---------------------------------------------------------------+

4 | Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

8 | Interface ID |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

12 | Timestamp (High) |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

16 | Timestamp (Low) |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

20 / /

/ Options (variable) /

/ /

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 14: 接口统计块的格式.** |

这些字段的含义如下：

* 块类型：接口统计块的块类型为5。
* 块长度：描述了这个块的总大小。
* 接口ID：它指定了那些统计信息相关的接口， ) of this field.这里的值应该与当前节（Section）的接口描述块（参考3.2节）中的值一致。
* 时间戳：本次统计引用的时间，时间戳的格式与增强分组块的定义是相同的。
* 选项：可选选项列表。

所有的统计信息字段都被定义成了选项，以便应对那些没有完整的统计信息集合的系统。除了在第2.5节定义的选项，以下选项是该块内有效：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Code** | **Length** | **Description** | **TODO: give a good example** |
| isb\_starttime | 2 | 8 | 捕获开始时间，The format of the timestamp is the same already defined in the Enhanced Packet Block ( [Section 3.3 ( Enhanced Packet Block (optional) )](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionepb) ).时间戳的格式与增强分组块中的定义是相同的（3.3节） | TODO: give a good example |
| isb\_endtime | 3 | 8 | 捕获结束时间，The format of the timestamp is the same already defined in the Enhanced Packet Block ( [Section 3.3 ( Enhanced Packet Block (optional) )](https://translate.googleusercontent.com/translate_c?depth=1&hl=zh-CN&prev=search&rurl=translate.google.com&sl=en&u=https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html&usg=ALkJrhjD3sDDJLuD7c2N7Qrp3wZQBY7yoA#sectionepb) ).时间戳的格式与增强分组块中的定义是相同的（3.3节）。 | TODO: give a good example |
| isb\_ifrecv | 4 | 8 | 捕获开始以来，从物理接口接收到的包数 | 100 |
| isb\_ifdrop | 5 | 8 | 捕获开始以来，因资源缺乏由接口丢弃的包数 | 0 |
| isb\_filteraccept | 6 | 8 | 捕获开始以来，由过滤器接受的数据包数量 | 100 |
| isb\_osdrop | 7 | 8 | 捕获开始以来，由操作系统丢弃的数据包数量 | 0 |
| isb\_usrdeliv | 8 | 8 | 捕获开始以来，提交给用户的数据包数量，这个值可能与'isb\_filteraccept - isb\_osdrop'不相等，因为捕获结束时可能还有一些数据包躺在操作系统的缓冲区中 | 0 |

这里所有的统计字段都是64位值，以当前节采用的字节序表示。 在访问这些字段是要特别小心，因为所有的块被排列为32位边界对齐，这些字段都不能保证在64位边界上也是对齐的。

**4. 实验块（Experimental Blocks）**

**4.1. 替代性分组块（Alternative Packet Blocks）**

Can some other packet blocks (besides the ones described in the previous paragraphs) be useful?

**4.2. 压缩块（Compression Block）**

压缩块是可选的，一个文件可以包含任意数量的压缩块。压缩块，正如它的名字一样，被用来保存压缩的数据。它的格式如Figure 15所示：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

| Block Type = ? |

+---------------------------------------------------------------+

| Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Compr. Type | |

+-+-+-+-+-+-+-+-+ |

| |

| Compressed Data |

| |

| /\* variable length, byte-aligned \*/ |

| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 15: 压缩块格式** |

这些字段的含义如下：

* 块类型：压缩块的块类型尚未分配。
* 块长度：描述了这个块的总大小。
* 压缩类型：指定压缩算法。 Possible values for this field are 0 (uncompressed), 1 (Lempel Ziv), 2 (Gzip), other??该字段可能的值是0（未压缩），1 (Lempel Ziv), 2 (Gzip), other??
* 压缩数据：压缩后的数据

**4.3. 加密块（Encryption Block）**

加密块是可选的，被用来保存加密的数据，一个文件可以包含任意数量的加密块。它的格式如Figure 16所示：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

| Block Type = ? |

+---------------------------------------------------------------+

| Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Encr. Type | |

+-+-+-+-+-+-+-+-+ |

| |

| Encrypted Data |

| |

| /\* variable length, byte-aligned \*/ |

| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 16: 加密块格式** |

这些字段的含义如下：

* 块类型：加密块的块类型尚未分配。
* 块长度：描述了这个块的总大小。
* 加密类型：指定加密算法。
* 加密数据：加密后的数据。一旦解密，生成其它块。

**4.4. 固定长度块（Fixed Length Block）**

固定长度块是可选的，A file can contain an arbitrary number of these blocks.一个文件可以包含这些块的任意数量。A Fixed Length Block can be used to optimize the access to the file.固定长度的块，可以用来优化对文件的访问，Its format is shown in .其格式描述如图Figure 17所示。A Fixed Length Block stores records with constant size.固定长度块存储固定大小的记录。It contains a set of Blocks (normally Packet Blocks or Simple Packet Blocks), of wihich it specifies the size.

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+---------------------------------------------------------------+

| Block Type = ? |

+---------------------------------------------------------------+

| Block Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Cell Size | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |

| |

| Fixed Size Data |

| |

| /\* variable length, byte-aligned \*/ |

| |

| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Block Total Length |

+---------------------------------------------------------------+

|  |
| --- |
| **Figure 17: Fixed Length Block format.** |

这些字段的含义如下：

* Block Type: The block type of the Fixed Length Block is not yet assigned.块类型：固定长度块的块类型尚未分配。
* Block Total Length: total size of this block, as described in块长度：描述了这个块的总大小。
* Cell size: the size of the blocks contained in the data field.单元大小：包含在数据字段中的块的大小。
* Fixed Size Data: data of this block.固定长度数据：此块的数据。

**4.5. 目录块（Directory Block）**

如果存在，此块包含以下信息：

* number of indexed packets (N)索引的数据包数量（N）
* 包含已经建立索引的任何数据包长度和位置信息的表（N entries）

目录块后面必须至少有N个数据包，否则它必须被视为无效。它可用于高效地装载文件的一部分到内存中，并支持内存映射文件的操作。此块可以作为一个文件处理的结果被网络分析工具添加到文件中。

**4.6. 流量统计和监控块（Traffic Statistics and Monitoring Blocks）**

一个或多个块可以被定义为包含网络统计或流量监控信息。它们被用来存储从RMON或Netflow探针，或从其他网络监控工具收集的数据。

**4.7.事件/安全块（Event/Security Block）**

该块可以被用来存储事件。事件可能包含了通用信息（例如网络负载超过50％，服务器关闭...）或安全警报。一个事件可以是：

* skipped, if the application doesn't know how to do with it跳过，如果应用程序不知道怎么用它做
* 独立于包的处理。换言之，应用程序跳过数据包，只处理警报
* 处理相关的数据包。例如，一个安全工具，可以仅仅加载与安全告警相关的数据包；一个监视工具可以跳过服务器当机时捕获的数据包。

**5.  推荐的扩展名: .pcapng**

在本文档中定义的“PCAP下一代转储文件格式”文件的扩展名推荐为“.pcapng”。

至少在“Windows世界中”，文件是由扩展其文件名区分。这种扩展名在技术上不是必需的，因为应用程序应能够通过文件开头的“魔法字节”自动检测pcapng文件格式。但是，使用扩展名可以更轻松地处理文件（如视觉上分辨文件格式），所以建议 - 虽然不是必需的 - 使用.pcapng作为扩展名按照此规范文件。

请注意：为了避免混乱（如.CAP就对应了不同的捕获文件格式），应该避免使用”.pcapng”之外的其它扩展名。

**6. 怎样增加供应商/域特定扩展**

首选方法是增加新的块类型或者在已存在的块中增加新的选项。

**7.  结论**

本文件中所提出的文件格式应该是非常通用的，并满足了广泛的应用。在最简单的情况下，它可以通过使用简单分组块来进行原始数据包的转储。在最复杂的情​​况下，它可以被用作不同类型的信息的储存库。在任何情况下，该文件仍然易于解析和应用始终可以跳过不感兴趣的数据；与此同时，不同的应用程序可以共享该文件，他们每一个可以受益的由他人制造的信息。两个或更多的文件可以合并获得另一个有效的文件。

|  |
| --- |
| [**TOC**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#toc) |

**Appendix A.  Packet Block Flags Word**

The Packet Block Flags Word is a 32-bit value that contains link-layer information about the packet.

The meaning of the bits is the following:

|  |  |
| --- | --- |
| **Bit Number** | **Description** |
| 0-1 | Inbound / Outbound packet (00 = information not available, 01 = inbound, 10 = outbound) |
| 2-4 | Reception type (000 = not specified, 001 = unicast, 010 = multicast, 011 = broadcast, 100 = promiscuous). |
| 5-8 | FCS length, in bytes (0000 if this information is not available). This value overrides the if\_fcslen option of the Interface Description Block, and is used with those link layers (e.g. PPP) where the length of the FCS can change during time. |
| 9-15 | Reserved (must be set to zero). |
| 16-31 | link-layer-dependent errors (Bit 31 = symbol error, Bit 30 = preamble error, Bit 29 = Start Frame Delimiter error, Bit 28 = unaligned frame error, Bit 27 = wrong Inter Frame Gap error, Bit 26 = packet too short error, Bit 25 = packet too long error, Bit 24 = CRC error, other?? are 16 bit enough?). |

|  |
| --- |
| [**TOC**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#toc) |

**Appendix B.  Standardized Block Type Codes**

Every Block is uniquely identified by a 32 bit integer value, stored in the Block Header.

As pointed out in [**Section 2.1**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionblock), Block Types codes whose Most Significant Bit (bit 31) is set to 1 are reserved for local use by the application.

All the remaining Block Type codes (0x00000000 to 0x7FFFFFFF) are standardized by this document. A request should be sent to the authors of this document to add a new Standard Block Type code to the specification.

Here is a list of the Standardized Block Type Codes.

|  |  |
| --- | --- |
| **Block Type Code** | **Description** |
| 0x00000000 | Reserved ??? |
| 0x00000001 | [**Interface Description Block**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionidb) |
| 0x00000002 | [**Packet Block**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionpb) |
| 0x00000003 | [**Simple Packet Block**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionpbs) |
| 0x00000004 | [**Name Resolution Block**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionnrb) |
| 0x00000005 | [**Interface Statistics Block**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionisb) |
| 0x00000006 | [**Enhanced Packet Block**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionepb) |
| 0x00000007 | IRIG Timestamp Block (requested by Gianluca Varenni <gianluca.varenni@cacetech.com>, CACE Technologies LLC) |
| 0x00000008 | Arinc 429 in AFDX Encapsulation Information Block (requested by Gianluca Varenni <gianluca.varenni@cacetech.com>, CACE Technologies LLC) |
| 0x0A0D0D0A | [**Section Header Block**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#sectionshb) |
| 0x0A0D0A00-0x0A0D0AFF | Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode. |
| 0x000A0D0A-0xFF0A0D0A | Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode. |
| 0x000A0D0D-0xFF0A0D0D | Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode. |
| 0x0D0D0A00-0x0D0D0AFF | Reserved. Used to detect trace files corrupted because of file transfers using the FTP protocol in text mode. |

|  |
| --- |
| [**TOC**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#toc) |

**Appendix C.  Standardized Link Type Codes**

NOTE: we should decide if we want to have this list here or in a separate document. It may make sense to have this list in a separate document and describe the format of a frame for each different linktype, or specify that the frame format is proprietary of a company and not public. There are a large number of encapsulations that are really vague and unspecified at all (even the name does not make any sense). Moreover, we should decide if we want to have \*all\* the linktypes (LINKTYPE\_XXX) defined by libpcap, or just a subset of them, thus trying to remove the big mess and confusion of similar headers.

Here is a list of the Standardized Link Type Codes.

|  |  |  |
| --- | --- | --- |
| **Link Type Name** | **Link Type Code** | **Description** |
| LINKTYPE\_NULL | 0 | No link layer information. A packet saved with this link layer contains a raw L3 packet preceded by a 32-bit host-byte-order AF\_ value indicating the specific L3 type. |
| LINKTYPE\_ETHERNET | 1 | D/I/X and 802.3 Ethernet |
| LINKTYPE\_EXP\_ETHERNET | 2 | Experimental Ethernet (3Mb) |
| LINKTYPE\_AX25 | 3 | Amateur Radio AX.25 |
| LINKTYPE\_PRONET | 4 | Proteon ProNET Token Ring |
| LINKTYPE\_CHAOS | 5 | Chaos |
| LINKTYPE\_TOKEN\_RING | 6 | IEEE 802 Networks |
| LINKTYPE\_ARCNET | 7 | ARCNET, with BSD-style header |
| LINKTYPE\_SLIP | 8 | Serial Line IP |
| LINKTYPE\_PPP | 9 | Point-to-point Protocol |
| LINKTYPE\_FDDI | 10 | FDDI |
| LINKTYPE\_PPP\_HDLC | 50 | PPP in HDLC-like framing |
| LINKTYPE\_PPP\_ETHER | 51 | NetBSD PPP-over-Ethernet |
| LINKTYPE\_SYMANTEC\_FIREWALL | 99 | Symantec Enterprise Firewall |
| LINKTYPE\_ATM\_RFC1483 | 100 | LLC/SNAP-encapsulated ATM |
| LINKTYPE\_RAW | 101 | Raw IP |
| LINKTYPE\_SLIP\_BSDOS | 102 | BSD/OS SLIP BPF header |
| LINKTYPE\_PPP\_BSDOS | 103 | BSD/OS PPP BPF header |
| LINKTYPE\_C\_HDLC | 104 | Cisco HDLC |
| LINKTYPE\_IEEE802\_11 | 105 | IEEE 802.11 (wireless) |
| LINKTYPE\_ATM\_CLIP | 106 | Linux Classical IP over ATM |
| LINKTYPE\_FRELAY | 107 | Frame Relay |
| LINKTYPE\_LOOP | 108 | OpenBSD loopback |
| LINKTYPE\_ENC | 109 | OpenBSD IPSEC enc |
| LINKTYPE\_LANE8023 | 110 | ATM LANE + 802.3 (Reserved for future use) |
| LINKTYPE\_HIPPI | 111 | NetBSD HIPPI (Reserved for future use) |
| LINKTYPE\_HDLC | 112 | NetBSD HDLC framing (Reserved for future use) |
| LINKTYPE\_LINUX\_SLL | 113 | Linux cooked socket capture |
| LINKTYPE\_LTALK | 114 | Apple LocalTalk hardware |
| LINKTYPE\_ECONET | 115 | Acorn Econet |
| LINKTYPE\_IPFILTER | 116 | Reserved for use with OpenBSD ipfilter |
| LINKTYPE\_PFLOG | 117 | OpenBSD DLT\_PFLOG |
| LINKTYPE\_CISCO\_IOS | 118 | For Cisco-internal use |
| LINKTYPE\_PRISM\_HEADER | 119 | 802.11+Prism II monitor mode |
| LINKTYPE\_AIRONET\_HEADER | 120 | FreeBSD Aironet driver stuff |
| LINKTYPE\_HHDLC | 121 | Reserved for Siemens HiPath HDLC |
| LINKTYPE\_IP\_OVER\_FC | 122 | RFC 2625 IP-over-Fibre Channel |
| LINKTYPE\_SUNATM | 123 | Solaris+SunATM |
| LINKTYPE\_RIO | 124 | RapidIO - Reserved as per request from Kent Dahlgren <kent@praesum.com> for private use. |
| LINKTYPE\_PCI\_EXP | 125 | PCI Express - Reserved as per request from Kent Dahlgren <kent@praesum.com> for private use. |
| LINKTYPE\_AURORA | 126 | Xilinx Aurora link layer - Reserved as per request from Kent Dahlgren <kent@praesum.com> for private use. |
| LINKTYPE\_IEEE802\_11\_RADIO | 127 | 802.11 plus BSD radio header |
| LINKTYPE\_TZSP | 128 | Tazmen Sniffer Protocol - Reserved for the TZSP encapsulation, as per request from Chris Waters <chris.waters@networkchemistry.com> TZSP is a generic encapsulation for any other link type, which includes a means to include meta-information with the packet, e.g. signal strength and channel for 802.11 packets. |
| LINKTYPE\_ARCNET\_LINUX | 129 | Linux-style headers |
| LINKTYPE\_JUNIPER\_MLPPP | 130 | Juniper-private data link type, as per request from Hannes Gredler <hannes@juniper.net>. The corresponding DLT\_s are used for passing on chassis-internal metainformation such as QOS profiles, etc.. |
| LINKTYPE\_JUNIPER\_MLFR | 131 | Juniper-private data link type, as per request from Hannes Gredler <hannes@juniper.net>. The corresponding DLT\_s are used for passing on chassis-internal metainformation such as QOS profiles, etc.. |
| LINKTYPE\_JUNIPER\_ES | 132 | Juniper-private data link type, as per request from Hannes Gredler <hannes@juniper.net>. The corresponding DLT\_s are used for passing on chassis-internal metainformation such as QOS profiles, etc.. |
| LINKTYPE\_JUNIPER\_GGSN | 133 | Juniper-private data link type, as per request from Hannes Gredler <hannes@juniper.net>. The corresponding DLT\_s are used for passing on chassis-internal metainformation such as QOS profiles, etc.. |
| LINKTYPE\_JUNIPER\_MFR | 134 | Juniper-private data link type, as per request from Hannes Gredler <hannes@juniper.net>. The corresponding DLT\_s are used for passing on chassis-internal metainformation such as QOS profiles, etc.. |
| LINKTYPE\_JUNIPER\_ATM2 | 135 | Juniper-private data link type, as per request from Hannes Gredler <hannes@juniper.net>. The corresponding DLT\_s are used for passing on chassis-internal metainformation such as QOS profiles, etc.. |
| LINKTYPE\_JUNIPER\_SERVICES | 136 | Juniper-private data link type, as per request from Hannes Gredler <hannes@juniper.net>. The corresponding DLT\_s are used for passing on chassis-internal metainformation such as QOS profiles, etc.. |
| LINKTYPE\_JUNIPER\_ATM1 | 137 | Juniper-private data link type, as per request from Hannes Gredler <hannes@juniper.net>. The corresponding DLT\_s are used for passing on chassis-internal metainformation such as QOS profiles, etc.. |
| LINKTYPE\_APPLE\_IP\_OVER\_IEEE1394 | 138 | Apple IP-over-IEEE 1394 cooked header |
| LINKTYPE\_MTP2\_WITH\_PHDR | 139 | ??? |
| LINKTYPE\_MTP2 | 140 | ??? |
| LINKTYPE\_MTP3 | 141 | ??? |
| LINKTYPE\_SCCP | 142 | ??? |
| LINKTYPE\_DOCSIS | 143 | DOCSIS MAC frames |
| LINKTYPE\_LINUX\_IRDA | 144 | Linux-IrDA |
| LINKTYPE\_IBM\_SP | 145 | Reserved for IBM SP switch and IBM Next Federation switch. |
| LINKTYPE\_IBM\_SN | 146 | Reserved for IBM SP switch and IBM Next Federation switch. |

|  |
| --- |
| [**TOC**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#toc) |

**Appendix D.  Link Layer Headers**

The Packet Data field of the Packet Blocks won't start with the actual network data captured, but with some link type specific "meta data". The format of this meta data depends on the link type used. TODO: mention example code in libpcap that lists these link headers.

|  |
| --- |
| [**TOC**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#toc) |

**Authors' Addresses**

|  |  |
| --- | --- |
|  | Loris Degioanni |
|  | CACE Technologies |
|  | 1949 Fifth Street #103 |
|  | Davis, CA 95616 |
|  | United States |
| **Phone:** | +1 530 758 2790 |
| **Email:** | [**loris.degioanni@cacetech.com**](mailto:loris.degioanni@cacetech.com) |
| **URI:** | [**http://www.winpcap.org/loris/**](http://www.winpcap.org/loris/) |
|  |  |
|  | Fulvio Risso |
|  | Politecnico di Torino |
|  | Corso Duca degli Abruzzi, 24 |
|  | Torino 10129 |
|  | Italy |
| **Phone:** | +39 011 564 7008 |
| **Email:** | [**fulvio.risso@polito.it**](mailto:fulvio.risso@polito.it) |
| **URI:** | [**http://staff.polito.it/fulvio.risso/**](http://staff.polito.it/fulvio.risso/) |
|  |  |
|  | Gianluca Varenni |
|  | CACE Technologies |
|  | 1949 Fifth Street #103 |
|  | Davis, CA 95616 |
|  | United States |
| **Phone:** | +1 530 758 2790 |
| **Email:** | [**gianluca.varenni@cacetech.com**](mailto:gianluca.varenni@cacetech.com) |
| **URI:** | [**http://www.winpcap.org/gianluca/**](http://www.winpcap.org/gianluca/) |

|  |
| --- |
| [**TOC**](http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#toc) |

**Full Copyright Statement**

Copyright © The Internet Society (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an “AS IS” basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

**Intellectual Property**

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP 11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

**Acknowledgment**

Funding for the RFC Editor function is currently provided by the Internet Society.

附录：pcap文件格式说明

# 文件格式

* PCAP文件格式也有过一些变化（其中一次官方的改动就是开始支持纳秒级别的时间戳），下面只介绍2.4版中的这种应用最广的格式。
* 从1998年的libpcap0.4就这个格式没变过，估计也不会变了（除了PCAPng格式）。
* 总体来说，文件结构是file header后面跟着0或若干条捕获数据包的记录，如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| File Header | Packet Header | Packet Data | Packet Header | Packet Data | … |

## ile Header

typedef struct pcap\_hdr\_s {

guint32 magic\_number; /\* magic number \*/

guint16 version\_major; /\* major version number \*/

guint16 version\_minor; /\* minor version number \*/

gint32 thiszone; /\* GMT to local correction \*/

guint32 sigfigs; /\* accuracy of timestamps \*/

guint32 snaplen; /\* max length of captured packets, in octets \*/

guint32 network; /\* data link type \*/

} pcap\_hdr\_t;

**magic\_number :**

用来识别文件格式本身和字节顺序。一般，写入的程序用自己的字节序在文件中写入0xa1b2c3d4。读取的程序如果读出0xd4c3b2a1，就表示后面所有的字段都要将字节序反转。

对于纳秒精度的文件，写入程序应在文件中写入0xa1b23c4d，读取程序再进行相应的判别。

**version\_major, version\_minor :**

文件格式的版本号，最新的版本是2.4

**thiszone :**

GMT (UTC)和后面packet header的时间戳所用的时区的时间差（单位：秒）。 实际上，时间戳用的一般都是GMT时间, 所以thiszone一般都设为0。

**sigfigs :**

时间戳的精度，实际上，一般设置为0

**snaplen :**

每个数据包的最大存储长度（该值设置所抓获的数据包的最大长度，如果所有数据包都要抓获，将该值设置为65535； 例如：想获取数据包的前64字节，可将该值设置为64）

**network :**

链路类型, specifying the type of headers at the beginning of the packet (例，以太网是1, 详见[tcpdump.org's link-layer header types page](http://www.tcpdump.org/linktypes.html),<--这里还有些关于DLT-Value的解释)

常用类型：

0 BSD loopback devices, except for later OpenBSD

1 Ethernet, and Linux loopback devices

6 802.5 Token Ring

7 ARCnet

8 SLIP

9 PPP

10 FDDI

100 LLC/SNAP-encapsulated ATM

101 “raw IP”, with no link

102 BSD/OS SLIP

103 BSD/OS PPP

104 Cisco HDLC

105 802.11

108 later OpenBSD loopback devices (with the AF\_value in network byte order)

113 special Linux “cooked” capture

114 LocalTalk

## Record (Packet) Header

每个捕获的包都以Packet Header开始(any byte alignment possible):

typedef struct pcaprec\_hdr\_s {

guint32 ts\_sec; /\* timestamp seconds \*/

guint32 ts\_usec; /\* timestamp microseconds \*/

guint32 incl\_len; /\* number of octets of packet saved in file \*/

guint32 orig\_len; /\* actual length of packet \*/

} pcaprec\_hdr\_t;

**ts\_sec :**

时间戳高位，精确到seconds（值是自从January 1, 1970 00:00:00 GMT以来的秒数来记），其实就是UN\*X的time\_t，可以用ANSI C time.h中的time()函数来获取这个值。如果时间戳不是基于GMT的，需调整global header中的thiszone

**ts\_usec :**

时间戳低位。一般的pcap文件中，精确到microseconds （数据包被捕获时候的微秒数，是自ts-sec的偏移量）

在纳秒精度的文件中，这个字段是数据包被捕获时候的纳秒数

注意：这个字段的值不能大于1秒

**incl\_len :**

捕获且存储在PCAP文件中的数据包长度，其实就是min[orig\_len,snaplen]

**orig\_len :**

这个数据包在网络中的原始长度。

## Packet Data

紧跟着包头的数据包内容，长度为incl\_len。