Binary Node List Specification

Version 1.0.0.0

H. Utku Maden

April 5, 2020

Chapter 1

Foreword

1.1 Changelog

1.0.0.0 This is the first revision of this document. No changes can exist.

1.2 Purpose

1.2.1 Purpose of "Binary Node List"s

The purpose of a "Binary Node List" file is to give programmers an easy to implement file format that can contain generic arrays of elements or raw binary data in an easy to access manner.

1.2.2 Purpose of this document

The purpose of this document is to define a specification all "Binary Node List" files must adhere to. Any file violating this document shall not be considered a "Binary Node List" except those adhering to later versions of the specification.

1.3 Conventions

This section of this document describes some conventions the document will adhere to from now on.

Constant Value Convention Constant values in this document are referred to via names for easy of readabilty. They are represented as teal colored capital words with underscores (_) as word separators. For example `BNL_MAGIC_WORD` is a well defined constant in this document. All refered constants are documented in this document at the section "Constants". Please refer to this section for their descriptions and their values. If a constant reference has an asterisk (*), this means any constant that starts with the expression to the left of the asterisk, or ends with the expression to the right of the asterisk. For example `BNL_TYPE_*` refers to all constants of the "node type" constant group.

Primitive Size Convention The primitive sizes this document follow the convention of the C# language conventions. Please refer to the section "Definitions" for their exact definition. Some examples following other languages' conventions are also given. They are represented as lowercase blue words. For example `byte` is an unsigned 1-byte wide integer type.

Number Base Representation Convention This document follows number base representation conventions of the C family of languages. Expressions starting with `0x` are numbers in base-16, otherwise they are in base-10. Octal bases are never used in this document.

Requisite Convention Throughout this document some words like *must*, *should*, and *may*have reserved meanings when formatted as given. Please refer to the section "Definitions" for their reserved meanings.

1.4 Definitions

Must Any file that is "Binary Node List" compliant adheres to the statement including this word.

Should The authors of this file format recommend statements with this word are followed.

May Statements with this word imply that this statement can be followed if the developer chooses to do so, or are generally implementation dependent.

Endianness Endianness is the order of bytes in a multiple byte wide integer numbers.

Little Endian These integers are ordered from the least significant byte to the most significant byte.

Big Endian These integers are ordered from the most significant byte to the least significant byte.

1.4.1 Data Types

This sub section describes the naming convention of data types and their equivelants in a couple other languages.

Type (C#)	C/C++[1] (Win32[2])	Java	English
byte	**uint8_t** (or **UINT8**)	**byte**[3]	An unsigned 1-byte wide integer type.
short	**int16_t** (or **INT16**)	**short**	A signed 2-bytes wide integer type.
int	**int32_t** or (or **INT32**)	**int**	A signed 4-bytes wide integer type.
float	**float**	**float**	An IEEE 754 compliant single precision floating point number (4-byte wide)
long	**int64_t** (or **INT64**)	**long**	A signed 8-bytes wide integer type.
double	**double**	**double**	An IEEE 754 compliant double precision floating point number (8-bytes wide)

Table 1.1: Data types as used by this document (first column), in other programming languages and in English.

[1]Some of these keywords are defined in the header `stdint.h`.

[2]These types are defined in the header `BaseTsd.h`. It may lead to unportable code.

[3]Java does not implement an unsigned 1-byte wide integer type. Implementors should be aware of this fact.

Chapter 2

File Specification

2.1 File Header

Every valid Binary Node List *must* have a valid header. The header is 16 **byte**s long. Is is layed out as described in the table below.

Offset	Length	Type	Description
0x00	0x04	**int**	*Must* be equal to BNL_MAGIC_WORD.
0x04	0x04	**int**	Version of Binary Node List this file was rendered with. *Should* be in the range of BNL_MIN_VERSION and BNL_MAX_VERSION, both inclusive.
0x08	0x04	**int**	Number of nodes this Binary Node List has. *Must* be greater than zero.
0x0C	0x04	**int**	Reserved for later use. *Should* be zero.

2.1.1 Implementation Suggestions

The programmer *should* read 16 bytes from the input stream to verify the file. The magic word *must* be validated, and the file version *should* be within the expected range. If the file version is out of range, the file *may* be read if it is not critical for proper operation of the target program.

2.2 Node Format

The file head is preceded by the main node list. The number of elements the list contains is know at this point in time.

Each node consists of a 16 **byte** node head, followed by a node name string and the node content (if any).Every array field in a node *must* be 16 **byte** aligned. This is reminded where needed. The node head format is described below.

Offset	Length	Type	Description
0x00	0x04	**int**	Node content type. Must be a value in the constant group Type Value. (e.g BNL_TYPE_ANY)
0x04	0x04	**int**	Node array index.
0x08	0x04	**int**	Node key length.
0x0C	0x04	**int**	Node content length (in bytes)

The node head is followed by the node key. The node key is a string that *must* be aligned to 16 **byte**s. If the node key length is zero, this field *must* be zero bytes long. It *must* not contain a terminating null character, however strings not aligned to 16 **byte**s must be padded with null characters.

The node key is followed by the node content. The node content is a generic buffer aligned to 16 **byte**s. The buffer *must* be padded with zero if the length is not 16 **byte**s aligned. If a string is contained in the buffer, it *should* not contain a terminating null character.

Chapter 3

Appendix

3.1 Constants

Name	Value (32-bit hex)	Description
BNL_MAGIC_WORD	0x6C6E6225	File format magic word equivalent to the string "%bnl".
BNL_VERSION_MIN	0x01000000	Minimum acceptable file version by this document's specification.
BNL_VERSION_MAX	0x01FFFFFF	Maximum acceptable file version by this document's specification.

Table 3.1: Table of generic constants.

Type Name	Value (hex)	Type Description
BNL_TYPE_ANY	0x00	Type for generic streams.
BNL_TYPE_STRING	0x01	Type for a ASCII or UTF-8 encoded **string**.
BNL_TYPE_VEC1B	0x11	Type for a one dimentional array of **byte**s.
BNL_TYPE_VEC2B	0x12	Type for a two dimentional array of **byte**s.
BNL_TYPE_VEC3B	0x13	Type for a three dimentional array of **byte**s.
BNL_TYPE_VEC4B	0x14	Type for a four dimentional array of **byte**s.
BNL_TYPE_VEC1S	0x21	Type for a one dimentional array of **short**s.
BNL_TYPE_VEC2S	0x22	Type for a two dimentional array of **short**s.
BNL_TYPE_VEC3S	0x23	Type for a three dimentional array of **short**s.
BNL_TYPE_VEC4S	0x24	Type for a four dimentional array of **short**s.
BNL_TYPE_VEC1I	0x41	Type for a one dimentional array of **int**s.
BNL_TYPE_VEC2I	0x42	Type for a two dimentional array of **int**s.
BNL_TYPE_VEC3I	0x43	Type for a three dimentional array of **int**s.
BNL_TYPE_VEC4I	0x44	Type for a four dimentional array of **int**s.
BNL_TYPE_VEC1F	0x4A	Type for a one dimentional array of **float**s.
BNL_TYPE_VEC2F	0x4B	Type for a two dimentional array of **float**s.
BNL_TYPE_VEC3F	0x4C	Type for a three dimentional array of **float**s.
BNL_TYPE_VEC4F	0x4D	Type for a four dimentional array of **float**s.
BNL_TYPE_VEC1L	0x81	Type for a one dimentional array of **long**s.
BNL_TYPE_VEC2L	0x82	Type for a two dimentional array of **long**s.
BNL_TYPE_VEC3L	0x83	Type for a three dimentional array of **long**s.
BNL_TYPE_VEC4L	0x84	Type for a four dimentional array of **long**s.
BNL_TYPE_VEC1D	0x8A	Type for a one dimentional array of **double**s.
BNL_TYPE_VEC2D	0x8B	Type for a two dimentional array of **double**s.
BNL_TYPE_VEC3D	0x8C	Type for a three dimentional array of **double**s.
BNL_TYPE_VEC4D	0x8D	Type for a four dimentional array of **double**s.

Table 3.2: Table of Type Constants