

# データファイル書式規約

## データファイルの書式についての説明

2008-04-08

任天堂株式会社発行

このドキュメントの内容は、機密情報であるため、厳重な取り扱い、管理を行ってください。

## 目次

1	はじめに .....	4
2	XMLファイル規約 .....	4
2.1	XMLファイルの構造 .....	4
2.1.1	XML宣言 .....	4
2.1.2	ルートエレメント .....	4
2.1.3	headエレメント .....	4
2.1.4	create エレメント .....	5
2.1.5	title エレメント .....	5
2.1.6	comment エレメント .....	5
2.1.7	generator エレメント .....	5
2.1.8	body エレメント .....	5
2.2	XMLファイルの例 .....	6
3	バイナリファイル規約 .....	6
3.1	バイナリファイルの構造 .....	6
3.1.1	ヘッダ部 .....	6
3.1.2	データ部 .....	7
3.2	バイナリファイルヘッダ .....	7
3.2.1	signature .....	7
3.2.2	byteOrder .....	7
3.2.3	version .....	8
3.2.4	fileSize .....	8
3.2.5	headerSize .....	8
3.2.6	dataBlocks .....	8
3.3	データブロックヘッダ .....	8
3.3.1	kind .....	8
3.3.2	size .....	8

## コード

コード 2-1	XMLファイル例 .....	6
---------	----------------	---

## 表

表 2-1	headエレメント .....	5
表 2-2	createエレメント .....	5
表 2-3	generatorエレメント .....	5
表 3-1	バイナリファイルヘッダ .....	7
表 3-2	データブロックヘッダ .....	8

## 図

図 3-1	バイナリファイルの構造図 .....	6
-------	--------------------	---

**改訂履歴**

改訂日	改訂内容
2008-04-08	改訂履歴の書式変更と文章の修正。
2004-05-11	初版。

# 1 はじめに

NITRO-System では、ツール等から出力されるデータファイルの書式に最低限の規約を設けています。この文書では、NITRO-System のデータファイル書式規約について説明しています。

## 2 XMLファイル規約

近年では、テキスト形式のデータを扱い易くする為に、テキスト形式のデータファイルにXMLを採用するケースが増えています。NITRO-System では、テキストファイルに XML 形式を用いる場合の規約が定められています。NITRO-System で提供される XML 形式のテキストファイルは、この規約に従っています。

### 2.1 XMLファイルの構造

---

NITRO-System の XML ドキュメントでは、ルートエレメント内に **head** エレメントと **body** エレメントが1つずつ格納されているという基本構造と、**head** エレメント内に格納すべき情報のみが規定されています。ドキュメントの本体となる情報は、各種ファイルフォーマット毎に規定され、**body** エレメントに格納されます。

NITRO-System で規定される XML ドキュメント内には、以下に示すようなエレメントが存在します。これらのエレメントやエレメント内の属性などの順序は、特に明示されている場合を除き、規定されていません。XML ドキュメントを解釈するときは、エレメントタグを確認して処理する必要があります。

#### 2.1.1 XML宣言

---

XML ファイルは、XML 宣言から始まります。XML 宣言は、XMLプロセッサがXMLドキュメントとして書かれている内容を正しく処理するために書かれています。

#### 2.1.2 ルートエレメント

---

ルートエレメントには、ドキュメント名が記述されています。ルートエレメントは **version** 属性を1つ持ちます。**version** 属性には、ドキュメントのバージョンが記述されます。ルートエレメントには、**head** エレメントと **body** エレメントが1つずつ、この順序で格納されます。

#### 2.1.3 headエレメント

---

**head** エレメントは、作成者や作成時刻、コメントなどのドキュメント全体に関する情報を格納するエレメントです。**head** エレメントには、以下の 4 つのエレメントが格納されます。

表 2-1 head エlement

create	ドキュメントの作成に関する情報。
title	ドキュメントのタイトル。
comment	ドキュメントに対するコメント。
generator	ジェネレータに関する情報。

### 2.1.4 create エlement

create エlementには、ドキュメント作成に関する情報が格納されます。このElementは、以下に示す4つの属性のみを持ち、Elementの内容はありません。

表 2-2 create エlement

user	作成者。
host	作成マシン名。
date	作成日時。書式は、YYYY-MM-DDThh:mm:ss とします。これは、ISO8601 の一形式で、日付と地方時を拡張形式で表した形となっています。（例:2002-01-17T17:03:00）
source	データの元になった、ソースデータが存在する場合、そのデータを示す文字列です。新規に作成された場合などには、この属性は存在しません。

### 2.1.5 title エlement

title エlementには、ドキュメントのタイトル(文字データ)が格納されます。

### 2.1.6 comment エlement

comment エlementには、ドキュメントに対するコメント(文字データ)が格納されます。このElementの存在は、任意となっています。

### 2.1.7 generator エlement

generator エlementには、ドキュメントのジェネレータに関する情報が格納されます。このElementは、以下に示す2つの属性のみを持ち、内容はありません。ドキュメントがアプリケーションにより機械的に生成された場合には、必ずこのElementが存在します。

表 2-3 generator エlement

name	ジェネレータ名
version	ジェネレータのバージョン

### 2.1.8 body エlement

body エlementには、ドキュメントの本体となる情報が格納されます。この中身は、各種データフォーマット毎に規定されます。

## 2.2 XMLファイルの例

以下に、NITRO-System で規定されるXMLドキュメントの例を示します。

### コード 2-1 XMLファイル例

```
<?xml version="1.0" encoding="Shift-JIS"?>

<sample version="1.0">

  <head>
    <create user="UserName" host="HostName" date="2002-01-18T10:15:00" source="s.xml" />
    <title>NITRO-System XML</title>
    <comment>XMLドキュメントの例</comment>
    <generator name="ToolName" version="1.0" />
  </head>

  <body>
    <!-- この中は、各種データフォーマットごとに規定されます。-->
  </body>

</sample>
```

## 3 バイナリファイル規約

NITRO-System では、比較的規模の大きいバイナリファイル用にバイナリファイル規約を規定しています。ここで規定される項目は、バイナリファイルの構造に関する基本的な部分のみであり、データの内容は各種バイナリファイルフォーマット毎に規定されます。

### 3.1 バイナリファイルの構造

バイナリファイルは、下の図のようにヘッダ部とデータ部の2つに分けられており、ヘッダ部が必ず先に格納される構造となっています。ヘッダ部の後に続くデータ部には、各種バイナリデータを含んでいるデータブロックが複数個格納されます。

バイナリファイルヘッダ	ヘッダ部
データブロック1	データ部
データブロック2	
データブロック3	
⋮	

図 3-1 バイナリファイルの構造図

#### 3.1.1 ヘッダ部

ヘッダ部には、最低16バイトの大きさを持つバイナリファイルヘッダが1つだけ格納されます。このバイナリファイルヘッ

ダには、バイナリファイルの種類を識別する為の情報が格納されます。バイナリファイルヘッダには、各種バイナリファイルフォーマット毎に固有な情報を付加することが可能となっています。なお、バイナリファイルヘッダの大きさは、必ず4バイトの整数倍(4バイトアライメント)となります。

### 3.1.2 データ部

データ部には、複数個のデータブロックが格納されます。各データブロックの先頭には、8バイトの大きさを持つデータブロックヘッダが置かれます。データブロックは、その先頭にデータブロックヘッダを置くことだけが規定されており、データブロックの内容は、各種バイナリデータフォーマット毎に規定されます。

データブロックの大きさは、最低でも4バイトの整数倍(4バイトアライメント)となっています。なお、各種バイナリファイルフォーマット毎の規定により、データブロックのサイズを、さらに大きな単位にアライメントされている場合もあります。また、データブロックが格納される順番に関しても、各種バイナリファイルフォーマット毎に規定されます。(データブロックが順不同であるか、決まった順番に並んでいるかは、各種バイナリファイルフォーマット毎に異なります。)

## 3.2 バイナリファイルヘッダ

バイナリファイルヘッダは、以下のような構造となっています。バイナリファイルヘッダは、最低16バイトの大きさを持ち、バイナリファイルの種類を示すためのシグネチャと、そのバージョン番号、及び、エンディアン判定用のコード、バイナリファイルの大きさ、そしてバイナリファイルに含まれるデータブロック数が格納されます。

表 3-1 バイナリファイルヘッダ

タイプ	パラメータ名	意味	サイズ
char[4]	signature	ファイル・シグネチャ。	4 バイト
u16	byteOrder	バイトオーダーマーク(エンディアン判定用)。	2 バイト
u16	version	バイナリファイルフォーマットのバージョン番号。	2 バイト
u32	fileSize	バイナリファイルの大きさ。	4 バイト
u16	headerSize	バイナリファイルヘッダの大きさ。	2 バイト
u16	dataBlocks	データブロックの数。	2 バイト

### 3.2.1 signature

signature には、バイナリファイルの種類を判定する為のファイル・シグネチャが格納されます。ファイル・シグネチャは4バイトのコードであり、エンディアンに関係なく上位バイトから順番にメモリに格納されます。逆に、他のデータは全てエンディアンに依存します。

ファイル・シグネチャは4文字の ASCII 文字で構成され、各バイナリファイルフォーマット間で重ならないように管理されています。

### 3.2.2 byteOrder

byteOrder には、エンディアン判定用のバイトオーダーマーク(Zero Width No Break Space)0xfeff が格納されます。バイトデータ列は、データのエンディアンが ビッグエンディアンのときは 0xfe,0xff の順で、リトルエンディアンのときは 0xff,0xfe の順で格納されます。

なお、NITRO 用のバイナリファイルでは、リトルエンディアンでデータを作成しています。この情報は、同じファイルフォーマットをバイトオーダーが異なるゲーム機と共用する場合を想定して設定されています。

### 3.2.3 version

version には、バイナリファイルフォーマットのバージョン番号が格納されます。上位バイトと下位バイトをそれぞれ、メジャーバージョン(整数値)と、マイナーバージョン(小数値)を示します。バージョン番号が 1.2 の場合、0x0102 という値となります。

### 3.2.4 fileSize

fileSize には、バイナリファイルの全体の大きさが格納されます。この大きさには、バイナリファイルヘッダの大きさも含まれます。

### 3.2.5 headerSize

headerSize には、バイナリファイルヘッダの大きさが格納されます。規定されているバイナリファイルヘッダの大きさは 16 バイトですので、通常は 16 が格納されます。バイナリファイルヘッダに各種バイナリファイルフォーマット毎に固有な情報が付加されている場合、headerSize には固有情報が付加された全体の大きさが格納されます。なお、バイナリファイルヘッダの大きさは、必ず 4 バイトの整数倍 (4 バイトアライメント) となっています。

バイナリファイルに含まれる最初のデータブロックは、バイナリファイル (バイナリヘッダ) の先頭から、headerSize 分だけ先に進めた位置に存在します。

### 3.2.6 dataBlocks

dataBlocks には、バイナリファイルのデータ部に含まれるデータブロックの数が格納されます。dataBlocks は、負号無し 16 ビットですので、1 つのバイナリファイルには、最大で 65535 個のデータブロックを格納する事が可能となっています。

## 3.3 データブロックヘッダ

バイナリファイルのデータ部に含まれるデータブロックは、以下のような構造を持つデータブロックヘッダから始まります。このデータブロックヘッダは 8 バイトの大きさを持ち、データブロックの種類と大きさが格納されます。

表 3-2 データブロックヘッダ

タイプ	パラメータ名	意味	サイズ
u32	kind	データブロックの種類。	4 バイト
u32	size	データブロックの大きさ。	4 バイト

### 3.3.1 kind

kind には、データブロックの種類を表す 4 バイトのコードが格納されます。このコードは、各種バイナリファイルフォーマット毎に規定されます。即ち、バイナリファイルのシグネチャが異なれば、データブロックの kind が同じであっても、データブロックの内容は異なります。通常データブロックの種類には、4 文字の ASCII 文字で構成されます。

### 3.3.2 size

size には、データブロックの大きさが格納されます。データブロックの大きさは、このデータブロックヘッダの大きさも含まれます。データブロックの大きさは、必ず 4 の倍数 (4 バイトアライメント) となっています。データブロックが複数個ある場合には、今のデータブロックヘッダの先頭から、この size 分だけ先に進めた位置に次のデータブロックが存在します。



© 2004-2008 Nintendo

任天堂株式会社の許諾を得ることなく、本書に記載されている内容の一部あるいは全部を無断で複製・複写・転写・頒布・貸与することを禁じます。