AMF3の真実

The knowledge of AMF implementation.

Presented by wacky

FxUG in Toyama #3　2011.4.16

はじめに

- 最近AMF3のEncode/Decodeを実装してみました。
- そこで得た知識を共有したいと思います！

- ３０分後には...
  - AMFの基本構造が分かっている
  - AMFの得手不得手が分かっている
  - BlazeDSの弱点も分かっている

What is AMF3?

- Action Message Formatの略、データ形式の一種。
- 仕様はAdobeから公開されている。
- ActionScript3(Flex2以降)の型に対応している。
- Flash Player 9から使用可能。

AMF3仕様:

http://opensource.adobe.com/wiki/display/blazeds/Developer+Documentation

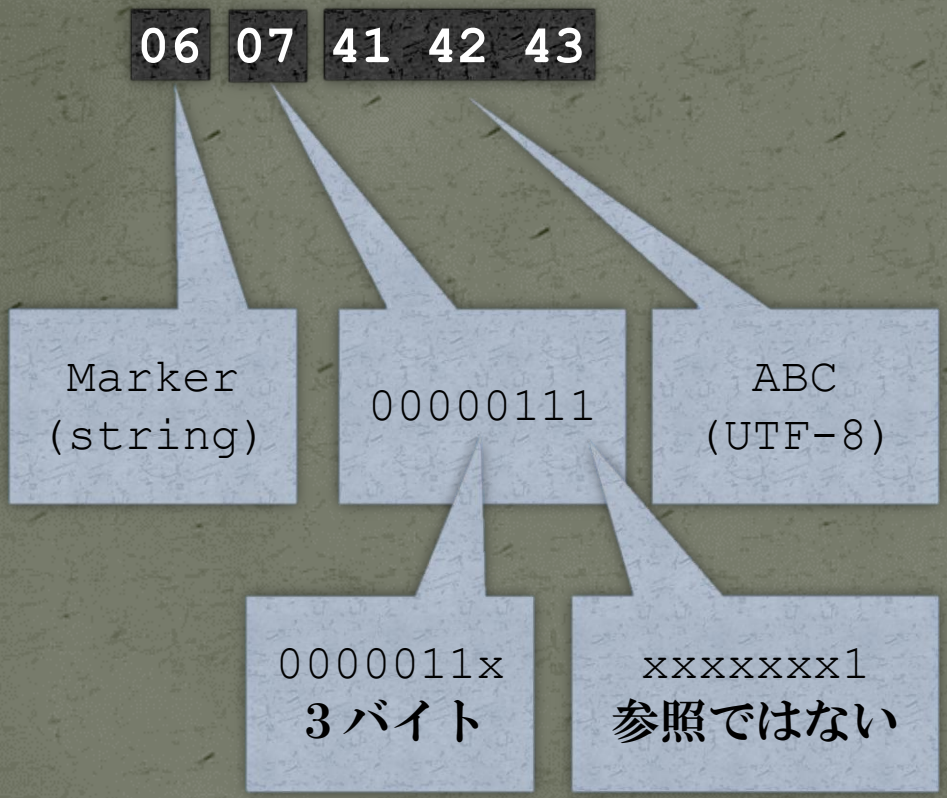
http://opensource.adobe.com/wiki/download/attachments/1114283/JP_amf3_spec_121207.pdf

http://opensource.adobe.com/wiki/download/attachments/1114283/amf3_spec_05_05_08.pdf

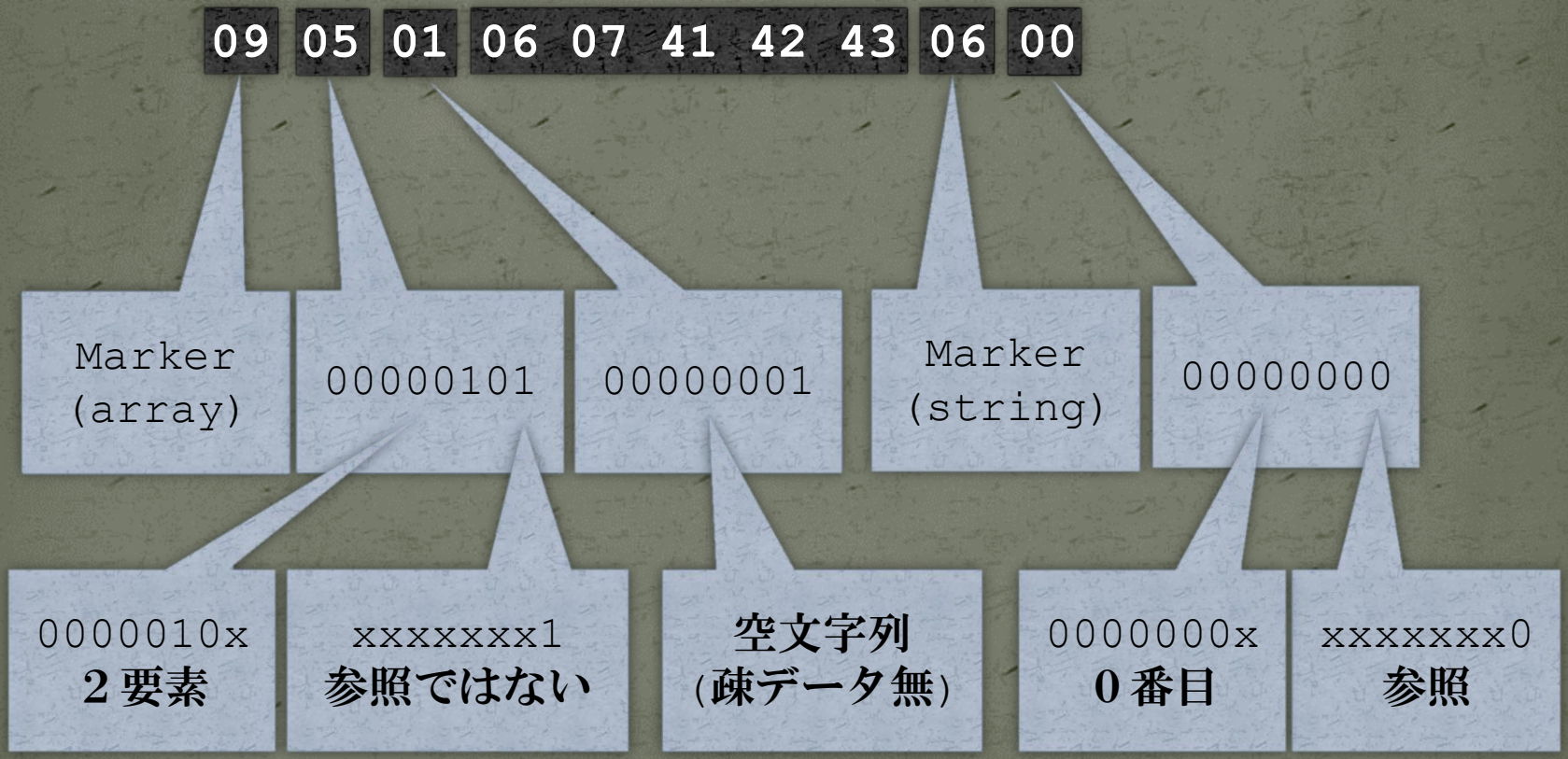
AMF3の立ち位置

	AMF	JSON	XML
仕様	Adobe	ECMA Script RFC4627	W3C
利用方法(Flex)	言語仕様 - RemoteObject - ByteArray	外部ライブラリ - as3corelib	言語仕様 - XML(E4X) - XMLDocument
利用方法(Java)	外部ライブラリ - BlazeDS	外部ライブラリ - JSONIC - JSON-lib	言語仕様 - JAXP - DOM/SAX
可読性	×	○	◎
データ量	○	△	×
型情報	○	×	△
オブジェクト参照	○	×	△
まとめ	良くも悪くも バイナリ形式	XMLより軽量… (いろんな意味で)	◎標準的 ×メモリ消費大

AMF3バイト配列を読む！(文字列)



AMF3バイト配列を読む！(配列)



AMF3の内部データ形式

Marker	Data Type	Size	Ref	Remarks
0x00	undefined	1 byte	-	Markerのみ
0x01	null	1 byte	-	Markerのみ
0x02	false	1 byte	-	Markerのみ
0x03	true	1 byte	-	Markerのみ
0x04	integer	2〜5 byte	-	可変長（データ精度は29bit）
0x05	double	9 byte	-	IEEE754
0x06	string	2 ~ 2^{28} byte	S	UTF-8形式、最大約256MB
0x07	XML-doc	2 ~ 2^{28} byte	O	XMLDocument ※下位互換用
0x08	date	9 byte	O	1970/1/1からの経過ミリ秒
0x09	array		O	疎と密な内部構造を持つ
0x0A	object		OT	ArrayCollection, ユーザ定義クラス
0x0B	XML	2 ~ 2^{28} byte	O	XML(E4X)
0x0C	ByteArray	2 ~ 2^{28} byte	O	

参照テーブル

- AMF3では、３種類の参照情報を管理している。
  - 文字列
  - オブジェクト(Object, Array, Date, XML, ByteArray)
  - Traits情報（オブジェクトの型情報）
    - プロパティー名リストのようなもの

- 既出の文字列は、2~5バイトで送信可能。
- 既出のオブジェクトも、2~5バイトで送信可能。
  - ただし、同一インスタンスの場合のみ
- 既出のTraits情報も、2~5バイトで送信可能。

AMF3データ構成編
～終了～

整数 (int)のEncodeサイズ

- 整数(int)は可変バイトでEncodeされる。

16進表記	10進表記	Type	Size
0x80000000 ～ 0xEFFFFFFF	-2147483648 ～ -268435755	double	9 byte
0xF0000000 ～ 0xFFFFFFFF	-268435456 ～ -1	integer	5 byte
0x00000000 ～ 0x0000007F	0 ～ 127	integer	2 byte
0x00000080 ～ 0x00003FFF	128 ～ 16383	integer	3 byte
0x00004000 ～ 0x001FFFFF	16384 ～ 2097151	integer	4 byte
0x00200000 ～ 0x0FFFFFFF	2097152 ～ 268435455	integer	5 byte
0x10000000 ～ 0x7FFFFFFF	268435456 ～ 2147483647	double	9 byte

※SizeはMarkerに必要な1 byteを含む。
※int領域全体での平均サイズは、実は約8.5 byte...
※Vector内のintは固定バイト(4 byte)となる。

整数 (long)のEncodeサイズ

- 整数(long)は固定でMarker + 8バイトにEncode
- ただし、形式はdouble形式。
  - つまり、桁落ちします…
- longの範囲(64bit)
  - -9223372036854775808～9223372036854775807
- 正しくEncode→Decodeできる範囲(53bit)
  - -9007199254740992～9007199254740992

※元々、 Flex上では表現できない数字だけど…

オブジェクトのEncodeサイズ(１)

- サンプルデータ（異なる値）

```
// Sample: Dynamic値
var target:Array = [];
for (var k:int = 0; k < 1000; k++) {
    target.push({ index: k, message: "Message" + k });
}

// Sample: Sealed値
var target:Array = [];
for (var k:int = 0; k < 1000; k++) {
    target.push(new Bean(k, "Message" + k));
}
```

オブジェクトのEncodeサイズ(２)

- サンプルデータ（同じ値、別インスタンス）

```
// Sample: Dynamic値
var target:Array = [];
for (var k:int = 0; k < 1000; k++) {
    target.push({ index: 999, message: "Message999" });
}

// Sample: Sealed値
var target:Array = [];
for (var k:int = 0; k < 1000; k++) {
    target.push(new Bean(999, "Message999"));
}
```

オブジェクトのEncodeサイズ(３)

- サンプルデータ（同一インスタンス）

```
// Sample: Dynamic値
var target:Array = [];
var item:Object = { index: 999, message: "Message999" };
for (var k:int = 0; k < 1000; k++) {
    target.push(item);
}

// Sample: Sealed値
var target:Array = [];
var item:Bean = new Bean(999, "Message999");
for (var k:int = 0; k < 1000; k++) {
    target.push(item);
}
```

オブジェクトのEncodeサイズ(４)

データ	参照未使用	Flash Player (比率)	BlazeDS (比率)
異なる値(Dynamic)	32,766 byte	19,779 byte (60.3%)	16,781 byte (51.2%)
異なる値(Sealed)	31,766 byte	16,781 byte (52.8%)	16,781 byte (52.8%)
同じ値(Dynamic)	33,004 byte	10,027 byte (30.3%)	7,029 byte (21.2%)
同じ値(Sealed)	32,004 byte	7,029 byte (21.9%)	7,029 byte (21.9%)
同一インスタンス(Dynamic)	33,004 byte	2,035 byte (6.1%)	2,034 byte (6.1%)
同一インスタンス(Sealed)	32,004 byte	2,034 byte (6.3%)	2,034 byte (6.3%)

AMF3は圧縮形式なの？

- 同一データを重複送信しないだけ。圧縮すると減る。

	参照未使用	参照未使用 (Compress)	Flash Player	Flash Player (Compress)
異なる値(Dynamic)	32,766 byte	5,029 byte (15.3%)	19,779 byte (60.3%)	3,726 byte (11.3%)
異なる値(Sealed)	31,766 byte	3,947 byte (12.4%)	16,781 byte (52.8%)	3,695 byte (11.6%)
同じ値(Dynamic)	33,004 byte	161 byte (0.4%)	10,027 byte (30.3%)	91 byte (0.2%)
同じ値(Sealed)	32,004 byte	144 byte (0.4%)	7,029 byte (21.9%)	81 byte (0.2%)
同一インスタンス (Dynamic)	33,004 byte	161 byte (0.4%)	2,035 byte (6.1%)	56 byte (0.1%)
同一インスタンス (Sealed)	32,004 byte	144 byte (0.4%)	2,034 byte (6.3%)	55 byte (0.1%)

新データタイプ

- 実は、AMF3の仕様書に載っていない
  新しいデータタイプが存在...
- Flash Player 10を使用すると発生…

Marker	Data Type	Ref	Remarks
0x0D	Vector.<int>	O	要素は固定４バイト
0x0E	Vector.<uint>	O	要素は固定４バイト
0x0F	Vector.<Number>	O	要素は固定８バイト(double)
0x10	Vector.<Object>	O	
0x11	Dictionary	O	Flash Player 9と非互換？

BlazeDSが微妙な件...

- Java5対応されていない
  - Enum: 文字列にエンコード。デコードは未対応。
  - Generics: 無視される。※List<Integer>とか厳しい…
- Unicodeのサロゲートペア未対応
  - Encode: サロゲートペア１文字が、6バイト(UTF-8)にEncodeされてしまう。※正しくは４バイト
  - Decode: Flash Player10でEncodeしたサロゲートペアをDecodeすると例外が発生。
- Flash Player10未対応
  - Vectorを扱えない。
  - Dictionaryを扱えない。

※BlazeDS-4.0.0.14931で確認

お疲れさまぁ～
m(__)m