

Adobe Systems Inc.**AMF 0 規格仕様**カテゴリ: **ActionScript** シリアライゼーション

Action Message Format -- AMF 0

著作権表記

Copyright (c) Adobe Systems Inc. (2002-2006). All Rights Reserved.

概要

Action Message Format (AMF) は、ActionScript オブジェクトグラフのシリアライズに用いられるコンパクトなバイナリ形式です。シリアライズされた AMF エンコードドオブジェクトグラフは、セッション間でアプリケーションのパブリックステートを永続・抽出するために使用したり、厳格に型付けされたデータ交換を介した、2 つのエンドポイント間での通信を可能にします。

AMF は 2001 年に Flash Player 6 とともに登場し、Flash Player 7 および ActionScript 2.0 のリリース時には変更が施されませんでした。AMF 書式のバージョンヘッダは当初「0」に設定されたため、この書式は AMF 0 と呼ばれています。

メモ: Flash Player 9 のリリース時には、ActionScript 3.0 および最新の ActionScript バーチャルマシン (AVM+) の投入にあわせて、新しいバージョンの AMF である AMF 3 が登場しています。なお、AMF 0 は引き続き Flash Player 6 以降の全バージョンの Flash Player にてサポートされています。

目次

- 1 はじめに
 - 1.1 目的
 - 1.2 記法基準
 - 1.2.1 ABNF (Augmented BNF)
 - 1.3 基本規則
 - 1.3.1 スtringと UTF-8 について
- 2 AMF 0 のデータ型
 - 2.1 データ型の概要
 - 2.2 Number データ型
 - 2.3 Boolean データ型
 - 2.4 String データ型
 - 2.5 Object データ型
 - 2.6 Movieclip データ型
 - 2.7 null データ型
 - 2.8 undefined データ型
 - 2.9 Reference データ型
 - 2.10 ECMA Array データ型
 - 2.11 Object End データ型
 - 2.12 Strict Array データ型
 - 2.13 Date データ型
 - 2.14 Long String データ型
 - 2.15 Unsupported データ型

2.16	RecordSet データ型
2.17	XML Document データ型
2.18	Typed Object データ型
3	AMF 0 エクステンション
3.1	AVM+ タイプマーカ
4	AMF 0 の使用
4.1	AMF パケットと NetConnection
4.1.1	AMF パケットのバージョン
4.1.2	AMF コンテキストヘッダ
4.1.3	AMF メッセージ
5	引用規格

1 はじめに

1.1 目的

Action Message Format (AMF) は、ActionScript オブジェクトグラフのシリアライズに用いられるコンパクトなバイナリ形式です。シリアライズされた AMF エンコードドオブジェクトグラフは、セッション間でアプリケーションステートを永続・抽出するために使用したり、厳格に型付けされたデータの交換を介した、2 つのエンドポイント間での通信を可能にします。AMF 0 と呼ばれる AMF の初期バージョンは、ActionScript オブジェクトのシリアライズ、厳格な型指定情報の維持、およびアプリケーションデータのパブリックステートの取得を行うことができます。また、AMF 0 は参照を用いた複雑なオブジェクトの送信をサポートしており、オブジェクトグラフ内の冗長的なインスタンスが送信されることを防止できるとともに、エンドポイントにおける各種関係の復元および循環参照の回避を可能にします。

1.2 記法基準

1.2.1 ABNF (Augmented BNF)

本スペックの型定義には ABNF (Augmented Backus-Naur Form) 記法 [RFC2234] が用いられています。本文書を読み進むには、この記法に精通している必要があります。

1.3 基本規則

U8	=	符号なしのバイト、8 ビットのデータ、オクテット
U16	=	ビッグエンディアン (network) バイトオーダーで記された符号なしの 16 ビット整数
S16	=	ビッグエンディアン (network) バイトオーダーで記された符号付きの 16 ビット整数
U32	=	ビッグエンディアン (network) バイトオーダーで記された符号なしの 32 ビット整数
DOUBLE	=	ネットワークバイトオーダーで記された 8 バイト IEEE-754 倍精度浮動小数点値 (符号ビットをローメモリで表現)
KB	=	キロバイト (1024 バイト)
GB	=	ギガバイト (1,073,741,824 バイト)

1.3.1 スtringと UTF-8 について

AMF 0 は、Stringのエンコードに(非修正の)UTF-8 を使用します。UTF-8 は、8 ビット Unicode Transformation Format の略にあたります。通常、UTF-8 Stringはバイト長ヘッダで始まり、その後、一連の可変長(1~4 オクテット)のエンコード済み Unicode コードポイントが続きます。形式およびデータ型によっては、AMF が、幾分改変されたバイト長ヘッダを使用することがあります。これらの亜種については以下に詳細を定義するとともに、本文書において明確に表示しています。

ABNF 記法 [RFC3629] では、UTF-8 を以下の体裁で表します。

```

UTF8-char          =   UTF8-1 | UTF8-2 | UTF8-3 | UTF8-4
UTF8-1             =   %x00-7F
UTF8-2             =   %xC2-DF UTF8-tail
UTF8-3             =   %xE0 %xA0-BF UTF8-tail | %xE1-EC 2(UTF8-tail)
                    | %xED %x80-9F UTF8-tail | %xEE-EF 2(UTF8-tail)
UTF8-4             =   %xF0 %x90-BF 2(UTF8-tail) | %xF1-F3 3(UTF8-tail)
                    | %xF4 %x80-8F 2(UTF8-tail)
UTF8-tail          =   %x80-BF

```

シリアライズされた UTF-8 Stringには、通常、文字コンテンツの前に配置され、直後のコンテンツのバイト長を示すヘッダが含まれます。標準のヘッダは 16ビット整数です。本文書ではこのタイプを以下の体裁で参照しています。

```

UTF-8              =   U16 *(UTF8-char)

```

16 ビットのバイト長ヘッダの場合、UTF-8 でエンコードされるStringの論理最大値は 65,535 バイト(つまり 64KB)になります。

さらに長いStringを扱うには、32 ビットのバイト長が必要になることがあります。本文書ではこのタイプを以下の体裁で参照しています。

```

UTF-8-long         =   U32 *(UTF8-char)

```

32 ビットのバイト長ヘッダの場合、UTF-8 でエンコードされるStringの論理最大値は 4,294,967,295 バイト(つまり 4GB)になります。

一部の場面では、それ以降にダイナミックプロパティが存在しないことを示すために、特別なケースとして空のStringが用いられることがあります。

```

UTF-8-empty        =   U16          ; UTF8-char コンテンツが存在しないことを示す
                        ; バイト長ゼロ表記。表記例: 0x0000

```

2 AMF 0 のデータ型

2.1 データ型の概要

AMF 0 には 16 種類の主要タイプマーカが用意されています。タイプマーカの長さは 1 バイトであり、それ以降に続くエンコード済みデータの種類を示します。

marker = U8

使用可能なタイプマーカの一覧を以下に示します(値の形式は 16 進法)。

number-marker	=	0x00	
boolean-marker	=	0x01	
string-marker	=	0x02	
object-marker	=	0x03	
movieclip-marker	=	0x04	; リザーブ、非サポート
null-marker	=	0x05	
undefined-marker	=	0x06	
reference-marker	=	0x07	
ecma-array-marker	=	0x08	
object-end-marker	=	0x09	
strict-array-marker	=	0x0A	
date-marker	=	0x0B	
long-string-marker	=	0x0C	
unsupported-marker	=	0x0D	
recordset-marker	=	0x0E	; リザーブ、非サポート
xml-document-marker	=	0x0F	
typed-object-marker	=	0x10	

メモ: AMF 0 には、Flash Player 9 および AMF 3 の登場にあわせて、AMF 3 でのシリアライゼーションへの切り替えを示すための特別なタイプマーカが追加されています。これにより、NetConnection リクエストを AMF 0 で開始し、AMF 3 の効率的なエンコーディングを活用する最初の複雑なデータ型に遭遇するとともに、AMF 3 に切り替えられるようになります。

avmplus-object-marker = 0x11

タイプマーカの後は、エンコードされた実際の型データを続けることができます。ただし、マーカが 1 つの値のみしか示し得ないような場合 (null など) は、他の情報をエンコードする必要はありません。

value-type = number-type | boolean-type | string-type | object-type | null-marker | undefined-marker | reference-type | ecma-array-type | strict-array-type | date-type | long-string-type | xml-document-type | typed-object-type

object-end-type は、object-type または typed-object-type の一連のプロパティの終了、もしくは ECMA 配列の連想セクションの終了を示す目的でのみ使用します。

Movieclip および Recordset のデータ型は、シリアライゼーション時にはサポートされていません。これらのマーカには、将来の使用に備えてリザーブステータスが与えられています。

2.2 Number データ型

AMF 0 の Number データ型は ActionScript の Number のエンコードに用いられます。Number タイプマーカに続くデータには、ネットワークバイトオーダーで記された 8 バイト IEEE-754 倍精度浮動小数点値 (符号ビットをローメモリで表現) を用います。

```
number-type          = number-marker DOUBLE
```

2.3 Boolean データ型

AMF 0 の Boolean データ型は、ActionScript 1.0 または 2.0 のプリミティブな Boolean、および ActionScript 3.0 Boolean のエンコードに用いられます。なお、Object (非プリミティブ) バージョンの ActionScript 1.0 または 2.0 ブール値はシリアライズできません。Boolean タイプマーカの後は、符号なしのバイトを続けます。バイト値ゼロが false、ゼロ以外の値 (通常は 1 を使用) が true をそれぞれ表します。

```
boolean-type         = boolean-marker U8          ; 0 は false、0 以外なら true
```

2.4 String データ型

AMF のすべてのストリングは UTF-8 でエンコードされます。ただし、バイト長ヘッダの形式は場面によって異なります。AMF 0 の String データ型は標準のバイト長ヘッダ (例: U16) を使用します。UTF-8 でエンコードに 65535 バイト以上が必要とされるような長いストリングの場合は、AMF 0 Long String データ型を使用するようにします。

```
string-type          = string-marker UTF-8
```

2.5 Object データ型

AMF 0 の Object データ型は、匿名の ActionScript オブジェクトのエンコードに用いられます。登録済みクラスを持たない、型指定のなされたオブジェクトはすべて、匿名の ActionScript オブジェクトとして扱われるべきです。オブジェクトグラフ内に同一のオブジェクトインスタンスがある場合は、AMF 0 を利用して参照で送信するようにします。

reference データ型を利用して、情報が必要以上に繰り返しシリアライズされることや、循環参照による無限ループが発生することを防止するようにしてください。

```
object-property      = (UTF-8 value-type) |  
                      (UTF-8-empty object-end-marker)  
anonymous-object-type = object-marker *(object-property)
```

2.6 Movieclip データ型

このデータ型はサポート対象外です。将来の使用に備えてリザーブされています。

2.7 null データ型

null データ型は null タイプマーカによって表されます。値に他の情報をエンコードする必要はありません。

```
null-type = null-marker
```

2.8 undefined データ型

undefined データ型は undefined タイプマーカによって表されます。値に他の情報をエンコードする必要はありません。

```
undefined-type = undefined-marker
```

2.9 Reference データ型

AMF 0 は複雑なオブジェクトを匿名オブジェクト、型指定のなされたオブジェクト、配列、または ecma-array のいずれかで定義します。複雑なオブジェクトに完全一致するインスタンスがオブジェクトグラフ内に複数回現れる場合、当該オブジェクトは必ず参照で送られる必要があります。reference データ型では、符号なしの 16 ビット整数を利用して、シリアライズ済みオブジェクトのテーブルのインデックスにポイントします。なお、インデックスは 0 から始まります。

```
reference-type = reference-marker U16 ; 他の複雑なオブジェクトに  
; ポイントするインデックス
```

16 ビットの符号なし整数が用いられるため、参照で送信できる一意的な複雑オブジェクトの論理最大数は 65,535 個になります。

2.10 ECMA Array データ型

ECMA Array (連想配列) は、ActionScript の Array に数値以外のインデックスが含まれる場合に使用されます。このデータ型は複雑なデータ型として扱われるため、繰り返し発生するインスタンスは参照で送ることができます。数値およびそれ以外の違いを問わず、インデックスはすべて整数ではなく、ストリング「キー」として扱われます。シリアライゼーションの観点から考えた場合、このデータ型は匿名オブジェクトに似ています。

```
associative-count = U32  
ecma-array-type = associative-count *(object-property)
```

連想配列カウンタは 32 ビットであるため、連想配列に含まれるエン트리数の論理最大値は 4,294,967,295 になります。

2.11 Object End データ型

object-end-marker は、匿名オブジェクト、型指定のなされたオブジェクトまたは連想配列における一連のオブジェクト属性の終了を合図する特殊タイプとして用いられます。これらのデータ型以外での使用は想定されていません。このマーカには必ず前に空の UTF-8 ストリングが付き、この組み合わせで object end タイプが形成されます。

```
object-end-type = UTF-8-empty object-end-marker ; 0x00 0x00  
; 0x09
```

2.12 Strict Array データ型

Strict Array には数値のインデックスのみを含むことができます。この際、AMF 0 は緻密、希薄のどちらのインデックスにも対応します。インデックス間の希薄な領域にある未定義エントリは、`undefined` としてシリアライズされます。

```
array-count          =    U32
strict-array-type    =    array-count *(value-type)
```

配列カウンタが 32 ビットであるため、配列に含まれるエントリ数の論理最大値は 4,294,967,295 になります。

2.13 Date データ型

ActionScript の Date は、UTC (協定世界時) タイムゾーンの 1970 年 1 月 1 日 0 時を原点とした経過時間を、ミリ秒で換算してシリアライズされます。このデータ型の仕様では、タイムゾーン差分情報を用いた操作に対応することができます。しかし、ネットワーク上で日付情報をシリアライズする際には、タイムゾーンを変更することが慣習的ではないため、タイムゾーンの差分を記入したり、使用することは推奨されません。タイムゾーンに関しては、必要に応じて個別にクエリをかけることが推奨されます。

```
time-zone            =    S16          ; リザーブ
                                ; 非サポート
                                ; 0x0000 に設定
                                ; するようにします
date-type            =    date-marker DOUBLE time-zone
```

2.14 Long String データ型

AMF 0 では、UTF-8 に変換すると 65535 バイト以上となるストリングをエンコードする際に、Long String を使用します。UTF-8 にエンコードされたストリングのバイト長ヘッダは、標準の 16 ビット整数ではなく 32 ビット整数です。

```
long-string-type     =    long-string-marker UTF-8-long
```

2.15 Unsupported データ型

特定のデータ型がシリアライズできない場合は、当該データ型の代わりとして特殊マーカの `unsupported` が使用できます。一部のエンドポイントでは、このタイプマーカの遭遇時にエラーが発生することがあります。このデータ型には、他の情報をエンコードする必要はありません。

2.16 RecordSet データ型

このデータ型はサポート対象外であり、将来の使用に備えてリザーブされています。

2.17 XML Document データ型

ActionScript 1.0/2.0 の XMLDocument、および ActionScript 3.0 の `flash.xml.XMLDocument` は、XML 文書の DOM 表現を提供するものです。ただし、シリアライズ後は当該文書をストリング表現化したものが用いられます。XML document データ型は常に UTF-8 Long String としてエンコードされます。

```
xml-document-type    =    xml-document-marker UTF-8-long
```

2.18 Typed Object データ型

厳格な型指定のなされたオブジェクトが当該クラスへのエイリアス登録を有する場合は、型の名前もシリアライズされます。Typed Object は複雑なデータ型として扱われ、繰り返し発生するインスタンスは参照で送信できます。

```
class-name          = UTF-8
object-type         = object-marker class-name *(object-property)
```

3 AMF 0 エクステンション

3.1 AVM+タイプマーカ

Flash Player 9 の発表にあわせて、ActionScript 3.0 と最新の AMF+をサポートするための AMF 3 が登場しました。これに伴い、AMF 0 エンコーディング環境を AMF 3 に切り替えられるよう、AMF 0 形式が拡張されており、AMF 0 には新しいタイプマーカの `avmplus-object-marker` が追加されています。このマーカの存在は、直後のオブジェクトが AMF 3 形式であることを示します(詳しくは [AMF3] を参照)。

なお、AMF 3 をサポートためのアップデートが行われていないレガシーAMF 0 システムでは、未知のタイプエラーが発生するはずです。

4 AMF 0 の使用

4.1 AMF パケットと NetConnection

ActionScript データ型のシリアライズ処理に加えて、NetConnection は、サーバにメッセージを送信し、非同期的にリモートサービス呼び出すために AMF を使用します。この際、複数のメッセージが単一の AMF パケットにまとめられます。

```
amf-packet          = version header-count *(header-type) message-count
                    *(message-type)
```

パケットに含まれるヘッダ数およびメッセージ数は、符号なしの 16 ビット整数で表されます。

```
header-count        = U16
message-count       = U16
```

NetConnection リクエストごとに含むことができるヘッダ数およびメッセージ数の論理最大値は、それぞれ 65,535 になります。

amf-packet の主要構成要素の定義を以下に示します。

4.1.1 AMF パケットのバージョン

AMF パケットの最初の 2 バイトは、値がどのバージョンの AMF でエンコードされているかの指定です。AMF パケットの一般構造は、常に AMF 0 でフォーマットされます。ただし、ヘッダ値およびメッセージ本文の値は、他の AMF バージョン (AMF 3 など) でエンコードされることがあります。

```
version          =      U16                ; AMF 0 の場合は必ず値を 0 に設定
```

4.1.2 AMF コンテキストヘッダ

ヘッダは、AMF パケットの残りの部分とそれに続くすべてのメッセージの処理コンテキストを提供します。このコンストラクトの主な用途としては、残り部分のパケットの暗号化、および(または)サーバへのユーザ認証(ユーザ名/パスワード)が挙げられます。パケットには、複数のコンテキストヘッダを含めることができます。

通常、ヘッダ名には、当該コンテキストヘッダによって呼び出されるリモート処理またはメソッドを示す名前を使用します。メソッドを指定する場合は、URI フォーマット様式を遵守し、オブジェクトとディレクトリパス間の区切り文字として前方スラッシュ (/) を使用します。ヘッダが Flash Player 向けであるような場合は、NetConnection インスタンスのクライアント上で識別可能なメソッド名をターゲットにするようにします。

```
header-name      =      UTF-8
```

ヘッダには、ブール値の「must understand」フラグも含まれます。ブール値が true である場合、このフラグは、エンドポイントでヘッダが理解できなかった際に処理を中止し、エラーを発することを指示します。

must understand が true に設定されたヘッダが Flash Player に送信され、当該 NetConnection インスタンスのクライアントオブジェクトがこのヘッダを処理するためのメソッドを有していない場合、Flash Player は NetConnection オブジェクトの onStatus ハンドラを呼び出します。

```
must-understand  =      U8                ; value == 0 の場合は false
                                     ; value != 0 の場合は true
```

ヘッダのバイト長が明らかな場合は、これを指定することで、リモートエンドポイントのメモリ割り当てを最適化できます。それ以外の場合は、このフィールドに (U32)-1 と記し、長さが不明であることを指定するようにします。

```
header-length    =      U32                ; ヘッダのバイト長
                                     ; わからない場合は (U32)-1
header-type      =      header-name must-understand header-length
                                     value-type
```

オブジェクト参照のインデックスは各コンテキストヘッダ固有のものとして扱われることに注意してください。シリアライズおよびデシリアライズは、新たなヘッダを処理するたびに参照インデックスを 0 にリセットする必要があります。

4.1.3 AMF メッセージ

AMF メッセージには、処理が求められる個別のトランザクションに関する情報が含まれており、リモートでの処理内容、成功時および失敗時に呼び出されるローカルクライアントでの処理内容、および、その処理時に用いられるデータが示されています。レスポンスメッセージの構造はリクエストメッセージのそれと同じです。

AMF メッセージ最初のフィールドは、リモートで呼び出す処理、関数またはメソッドを示すターゲット URI です。ターゲット URI の正確な書式は本仕様では定義されていません。クライアントでは、特定のサーバ実装にて確立された命名慣習を使用する必要があります。命名慣習の一例としては、サービス階層 (クラスパッケージなど) を前方スラッシュ、サービス名と処理名 (パブリッククラスメソッドなど) をピリオドでそれぞれ区切り、完全に有効なサービス名 (クラス名など) を指定するような表記法があります。

例: "com/mycompany/Services.getQuote"

AMF メッセージ 2 つ目のフィールドは、レスポンス URI です。レスポンス URI では、クライアントからの呼び出しにレスポンスを照合させるために使用する一意的な処理名を指定します。リモートエンドポイントは、エラー発生時にこのフィールドを使用します。したがって、このフィールドは、たとえリクエストの成功時に値がクライアントに返されないような場合でも必要になります。

例: "/1"

メッセージにリモートエンドポイントからのレスポンスが含まれる場合は、ターゲット URI が、レスポンスを処理するためにローカルクライアント上のどのメソッド (AMF リクエストオリジネータなど) が呼び出されるかを指定します。

例: "/1/onResult" または "/1/onStatus"

レスポンスのターゲット URI は、リクエストのレスポンス URI に成功時の「/onResult」接尾辞、または失敗時の「/onStatus」接尾辞を付けて設定します。

AMF のすべてのストリング同様に、ターゲット URI およびレスポンス URI のストリングは UTF-8 でエンコードされます。

```
target-uri          = UTF-8
response-uri       = UTF-8
```

AMF メッセージ 3 つ目のフィールドは、メッセージ本文のバイト長です。このフィールドは、あらかじめ各メッセージをデコードすることなく、AMF パケットを分割しなければならないような場面で有用です。長さを的確に判断できないような場合は、値として (U32)-1 を指定できます。

```
message-length     = U32                ; メッセージのバイト長
                  ; わからない場合は (U32)-1
```

4 つ目 (最後) のフィールドはメッセージ本文です。本文には、処理に関連付けられた実際のデータが含まれます。メッセージがクライアントリクエストである場合は、本文に、リモート処理・メソッドに渡されるクライアントのパラメータデータが含まれます。一連の引数は **Strict Array** データ型で表現するようにします。

メッセージがリモートエンドポイントのレスポンスである場合は、メッセージ本文に結果が含まれます。

また、リモートエンドポイントが受信したクライアントデータおよび(または)リクエストされた処理上でエラーが検出された場合、リモートエンドポイントはレスポンスメッセージの本文でエラー情報を提供しません。

メッセージタイプの一般的なフォーマットは以下の通りです。

```
message-type          =   target-uri response-uri message-length value-type
```

オブジェクト参照インデックスは、各メッセージ本文固有のものとして扱われることに注意してください。シリアライズおよびデシリアライズは、新たなメッセージを処理するごとに参照インデックスを 0 にリセットする必要があります。

5 引用規格

- [AMF3] Adobe Systems Inc. "Action Message Format -- AMF 3", June 2006.
- [RFC2234] D. Crocker., et. al. "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.