翻译:许江华
Adobe Systems Inc.
AMF 3 Specification
Category: ActionScript Serialization 类别：AS序列化

Action Message Format -- AMF 3

Abstract 概览

Action Message Format (AMF) is a compact binary format that is used to serialize ActionScript object graphs. Once serialized an AMF encoded object graph may be used to persist and retrieve the public state of an application across sessions or allow two endpoints to communicate through the exchange of strongly typed data.
AMF(Action Message Format 动作信息格式)是用来序列化AS(ActionScript 动作脚本)实例对象(object graphs)的压缩的二进制格式。序列化的AMF编码的实例对象可用来持久化，并且在不同的会话中获得应用的公共状态，或者允许在两个端点(比如客户户端和服务器端 --译者注)通过强类型数据交换进行通信。

AMF was introduced in Flash Player 6 in 2001 and remained unchanged with the introduction of ActionScript 2.0 in Flash Player 7 and with the release of Flash Player 8. This version of AMF is referred to as AMF 0 (See [AMF0]). In Flash Player 9, Action Script 3.0 was introduced along with a new ActionScript Virtual Machine (AVM+) - the new data types and language features made possible by these improvements prompted AMF to be updated. Given the opportunity to release a new version of AMF, several optimizations were also made to the encoding format to remove redundant information from serialized data. This specification defines this updated version of AMF, namely AMF 3.
AMF引进于2001年的FlashPlayer6，并且在引入AS2.0的FlashPlayer7和FlashPlayer8中没有改变的保留了。这个版本的AMF参考于AMF0(查阅[AMF0])。在FlashPlayer9中，AS3.0同新的AS虚拟机(AVM+)一起被引进—新的数据类型和语言特性的改进致使AMF升级成为可能，给了一个发布新的AMF版本的机会，新版本的AMF在序列化数据的时候做了一些优化，使得编码格式去除了一些冗余信息。升级后的AMF版本便是AMF3。

Table of Contents 目录(略)

1 Introduction 介绍

1.1 Purpose 目的

Action Message Format (AMF) is a compact binary format that is used to serialize ActionScript object graphs. Once serialized an AMF encoded object graph may be used to persist and retrieve the public state of an application across sessions or allow two endpoints to communicate through the exchange of strongly typed data. (译者注：之前有翻译)
The first version of AMF, referred to as AMF 0, supports sending complex objects by reference which helps to avoid sending redundant instances in an object graph.
第一个版本的AMF，即AMF0，支持在避免了在对象图中发送冗余的实例的通过引用发送复杂的对象。
It also allows endpoints to restore object relationships and support circular references while avoiding problems such as infinite recursion during serialization.
他也允许端点存储对象关系，并且支持在避免问题--如在序列化时无穷的递归--的情况下的循环

引用。

A new version of AMF, referred to as AMF 3 to coincide with the release of ActionScript 3.0, improves on AMF 0 by sending object traits and strings by reference in addition to object instances.
新版本的AMF，即AMF3，与AS3.0版本保持一致，在通过引用发送除对象实例外的对象特性和字符串做了改进。
AMF 3 also supports some new data types introduced in ActionScript 3.0.
AMF3也支持在AS3.0中的一些新的数据类型。

1.2 Notational Conventions 标记转换

1.2.1 Augmented BNF 参数化的BNF

Type definitions in this specification use Augmented Backus-Naur Form (ABNF) syntax [RFC2234].
在这个规格中类型定义使用参数化的巴克斯范式(ABNF)语法[RFC2234]。
(译者注：
BNF is a formal meta-syntax used to express context-free Grammars. BNF is one of the most commonly used meta-syntactic notation s for specifying the syntax of programming languages, command sets, PDUs, and similar things. However, pure BNF is rather limited, so the two variations EBNF and ABNF have become more popular.)

The reader should be familiar with this notation before reading this document.
读者在阅读这个文档之前应该熟悉这个注解。

1.3 Basic Rules 基本规则

Throughout this document bytes are assumed to be octets, or 8-bits.
这个文档的字节是8位的。

U8	An unsigned byte (8-bits, an octet) 无符号字节（8位，一个8位字节）
U16	An unsigned 16-bit integer in big endian (network) byte order 在网络中的字节顺序中的无符号的占用16个二进制位的整数
U32	An unsigned 32-bit integer in big endian (network) byte order 在网络中的字节顺序中的无符号的占用32个二进制位的整数
DOUBLE	8 byte IEEE-754 double precision floating point value in network byte order (sign bit in low memory). 在网络中的字节循序的（符号位在低存储）8字节的IEEE-754双精度浮点数
MB	A megabyte or 1048576 bytes. 兆字节

More complicated data type rules require special treatment which is outlined below.
更多的复杂的数据类型规则需要特殊的处理，概括如下：

1.3.1 Variable Length Unsigned 29-bit Integer Encoding

可变长度的无符号29位整数的编码

AMF 3 makes use of a special compact format for writing integers to reduce the number of bytes required for encoding.
AMF3使用一种特别的压缩格式来写整数，用于压缩编码的字节数量。
As with a normal 32-bit integer, up to 4 bytes are required to hold the value however the high bit of the first 3 bytes are used as flags to determine whether the next byte is part of the integer.
对于一个正常的32-bit的整数，需要4个字节来存储，然而前3个字节的最高位是用于标识下一个字节是不是整数的一部分
With up to 3 bits of the 32 bits being used as flags, only 29 significant bits remain for encoding an integer. This means the largest unsigned integer value that can be represented is $2^{29} – 1$
在32-bit中多达3个bit是用来标志的，所以对编码的一个整数仅仅有29个bit有意义。这意味着最大的无符号的整数值是2^{29}-1。

```
        (hex)               :              (binary)
0x00000000 - 0x0000007F     :  0xxxxxxx
0x00000080 - 0x00003FFF     :  1xxxxxxx 0xxxxxxx
0x00004000 - 0x001FFFFF     :  1xxxxxxx 1xxxxxxx 0xxxxxxx
0x00200000 - 0x3FFFFFFF     :  1xxxxxxx 1xxxxxxx 1xxxxxxx xxxxxxxx
0x40000000 - 0xFFFFFFFF     :  throw range exception
```

In ABNF syntax, the variable length unsigned 29-bit integer type is described as follows:
在ABNF语法中，可变长度的无符号的29位的整型数据类型描述如下：

```
U29         =  U29-1 | U29-2 | U29-3 | U29-4
U29-1       =  %x00-7F
U29-2       =  %x80-FF %x00-7F
U29-3       =  %x80-FF %x80-FF %x00-7F
U29-4       =  %x80-FF %x80-FF %x80-FF %x00-FF
```

1.3.2 Strings and UTF-8

AMF 0 and AMF 3 use (non-modified) UTF-8 to encode strings. UTF-8 is the abbreviation for 8-bit Unicode Transformation Format. UTF-8 strings are typically preceded with a byte-length header followed by a sequence of variable length (1 to 4 octets) encoded Unicode code-points.
AMF0和AMF3使用UTF-8来编码字符串（没有变化的）。UTF-8是8-bit的Unicode Transformation Format（统一的传输格式）的缩写。UTF-8字符串是典型的字节长度为头，紧跟着编码Unicode编码点的可变长度（1到4个字节）的字节序列。
AMF 3 uses a slightly modified byte-length header; a detailed description is provided below and referred to throughout the document.
AMF3使用一个轻微改变的字节长度头；适用于整个文档的具体描述如下。

```
        (hex)              :              (binary)
0x00000000 - 0x0000007F    :  0xxxxxxx
0x00000080 - 0x000007FF    :  110xxxxx 10xxxxxx
0x00000800 - 0x0000FFFF    :  1110xxxx 10xxxxxx 10xxxxxx
0x00010000 - 0x0010FFFF    :  11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

In ABNF syntax, [RFC3629] describes UTF-8 as follows:
在ABNF句法，[RFC3629]描述UTF-8如下：

```
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
```

For AMF 3 a string can be encoded as a string literal or a string reference.
对于AMF3，一个字符串可以作为字面值或引用来编码。
A variable length unsigned 29-bit integer is used for the header and the first bit is flag that specifies which type of string is encoded.
可变长度的无符号的29-bit的整数用于头，并且第一个bit标识字符串的编码。
If the flag is 0, a string literal is encoded and the remaining bits are used to encode the byte-length of the UTF-8 encoded String.
标识为0时，字符串是字面上编码的，余下的位数是用来编码UTF-8的编码的字节的长度。
If the flag is 1, then a string reference is encoded and the remaining bits are used to encode an index to the implicit string reference table.
标识为1时，是字符串引用的编码，余下的位数是用来编码直指隐含的字符串引用表的一个索引。

```
U29S-ref      =  U29    ; The first (low) bit is a flag with
                        ; value 0. The remaining 1 to 28
                        ; significant bits are used to encode a
                        ; string reference table index (an
                        ; integer).

U29S-value    =  U29    ; The first (low) bit is a flag with
                        ; value 1. The remaining 1 to 28
                        ; significant bits are used to encode the
                        ; byte-length of the UTF-8 encoded
                        ; representation of the string

UTF-8-empty   =  0x01   ; The UTF-8-vr empty string which is
                        ; never sent by reference.

UTF-8-vr      =  U29S-ref | (U29S-value *(UTF8-char))
```
（译者注：标识位的意义前后定义的不一样）

Note that this encoding imposes some theoretical limits on the use of Strings.
注意：这个编码增加了字符串使用上的理论上的限制。
The number of unique Strings that can be sent by reference is limited to $2^{28} - 1$, and the byte-length of each UTF-8 encoded String is limited to $2^{28} - 1$ bytes (approx 256 MB).
一个被引用发送的字符串的最大长度是：2^{28}-1，每一个UTF-8编码的字符串长度是2^{28}-1字节（大概256MB）。

2. Technical Summary 技术摘要

2.1 Summary of improvements 改进的摘要

The following is a table of the improvements and changes in AMF 3:
以下是AMF3的改进和改变的目录：

Object traits can now be sent by reference
对象的特性现在可以被引用发送
Strings can now be sent by reference
字符串可以被引用发送
int/uint type support
对int/uint类型的支持
flash.utils.ByteArray type support, can also be sent by reference
对flash.utils.ByteArray 类型的支持，也可以通过引用发送
flash.utils.IExternalizable support
对flash.utils.IExternalizable的支持。
Variable length encoding scheme for integers to reduce data size
旨在压缩数据大小的可变长度的整数的编码策略
References are sent using variable length integer
可变长度的整数的引用发送
String UTF-8 length uses variable length integer
UTF-8的字符串长度使用可变长度的整数
Array count uses variable length integer
数组的长度使用可变长度的整数
A single Array type marker covers both strict and ECMA Arrays
一个单独的Array数组数据类型覆盖了严格的和 ECMA两个数组
Dates no longer send timezone information
日期不再发送时区信息
Dates can now be sent by reference
日期可以通过引用发送
XMLDocument UTF-8 length uses variable length integer
XMLDocument的UTF-8的长度使用可变长度整数
XMLDocument can now be sent by reference
XMLDocument可以通过引用发送
XML type support, can also be sent by reference
XML也可以通过应用发送
XML UTF-8 length uses variable length integer
ByteArray type length uses variable length integer
Boolean true and false are now sent as one byte type markers
Boolean类型使用一个字节标识。
Unsupported type marker has been removed
不支持类型标志被删除了
Reserved RecordSet and Movieclip type markers have been removed
保留了RecordSet和Movieclip类型标识被删除了。

2.2 Reference Tables

In AMF 3, Strings, Complex Objects (which in AMF 3 are defined as anonymous Objects, typed Objects, Arrays, Dates, XMLDocument, XML, and ByteArrays) and an Object Type's Traits can now be sent by reference.
在AMF3中，字符串，复杂对象（他是AMF3中定义的匿名对象，类型化的对象，数组，日期，xml文档，xml, 字节数组）

This means that instead of sending redundant information, these components of AMF can simply refer to an earlier occurrence of a component.
这意味着不发送冗余信息，这些AMF组件可以简单的参考先前的版本。
This reference is an integer forming a zero-based index that is encoded in the component information, typically in the first number that appears after the relevant type marker (see the type definitions for Object, Array, Date, XMLDocument, XML and ByteArray below for exact details).
这个引用从0开始索引的整数，被编码在组件信息中，典型地出现在相关的类型标识（详细情况查阅下面Object, Array, Date, XMLDocument, XML and ByteArray类型定义）之后第一个数字。
These indexes form a virtual "table" of references that a deserializer and serializer must maintain when reading and writing AMF 3 formatted data.
这些索引形式是一个引用的虚拟表，在读写AMF3格式的数据的时候必须有一个反序列化器和序列化其来维护这个表。

Note that 3 separate reference tables are used for Strings, Complex Objects and Object Traits respectively.
注意：不同的引用表被用于对应的字符串，复杂对象和对象特性。

3 AMF3 Data Types

3.1 Overview

There are 13 types in AMF 3. A type marker is one byte in length and describes the type of encoded data that follows.
有13个AMF3的类型。一个类型标识占用一个字节，描述跟着他的编码的数据类型。

```
marker                   =  U8
```

The set of possible type markers are listed below (values are represented in hexadecimal format):
所有可能的类型标识罗列如下（值是用16进制表示的）：

```
undefined-marker     =  0x00
null-marker          =  0x01
false-marker         =  0x02
true-marker          =  0x03
integer-marker       =  0x04
double-marker        =  0x05
string-marker        =  0x06
xml-doc-marker       =  0x07
date-marker          =  0x08
array-marker         =  0x09
object-marker        =  0x0A
xml-marker           =  0x0B
byte-array-marker    =  0x0C
```

Type markers may be followed by the actual encoded type data, or if the marker represents a single possible value (such as null) then no further information needs to be encoded.
典型的标识后有真正的比啊类型数据，或者如果标识标识一个单独可能的值（如null）,就没有编码的信息了。

```
value-type    =  undefined-marker | null-marker | false-marker |
                 true-marker | integer-type | double-type |
                 string-type | xml-doc-type | date-type |
                 array-type | object-type | xml-type |
                 byte-array-type
```

AMF 3 makes use of three reference tables for strings, objects and traits (the characteristics of objects that define a strong type such as the class name and public member names).
AMF3使用三个引用表，分别用于字符串，对象和特性（一个强类型的对象特征有如类的名字和公共成员名字）
These tables are considered implicit as they are not encoded as a unique entity in the format.
这些表是隐含着的，因为他们不是作为一个在格式上唯一编码的实体。
Each type that can be sent by reference may instead be encoded using an index to the appropriate reference table.
每一个可以被引用发送的实体，会使用合适的引用表中的一个索引来编码。
Strings can be sent by reference using an index to the string table.
字符串能使用字符串引用表中的索引被发送。
Object, Array, XML, XMLDocument, ByteArray, Date and instances of user defined Classes can be sent by reference using an index to the object table.
对象，数组，XML，XMLDocument，ByteArray，Date 和 用法定义的类实例可以使用对象表的索引来进行引用发送。
Objects and instances of user defined Classes have trait information which can also be sent by reference using an index to the traits table.
对象和自定义实例的特性信息也使用特性表中的索引来进行引用发送。

3.2 undefined Type 未定义类型

The undefined type is represented by the undefined type marker. No further information is encoded for this value.
未定义类型由未定义类型标识来表示。对于这个值没有其他的信息。

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

Note that endpoints other than the AVM may not have the concept of undefined and may choose to represent undefined as null.
注意：不是AVM的其他终端可能没有未定义的概念，可能会选择null来表示未定义的类型。

3.3 null Type

The null type is represented by the null type marker. No further information is encoded for this value.
Null类型由null类型标识来表示。对于这个值没有其他的信息。

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

3.4 false Type

The false type is represented by the false type marker and is used to encode a Boolean value of false.
False类型用false类型标识来表示，并且用false的Boolean值来编码。
Note that in ActionScript 3.0 the concept of a primitive and Object form of Boolean does not exist. No further information is encoded for this value.
注意：在AS3.0中，基本的Object形式的Boolean概念不复存在。对于这个值没有其他的信息。

```
false-type            =  false-marker
```

3.5 true type

The true type is represented by the true type marker and is used to encode a Boolean value of true. Note that in ActionScript 3.0 the concept of a primitive and Object form of Boolean does not exist. No further information is encoded for this value.
（译者：和false类似）

```
true-type          =  true-marker
```

3.6 integer type

In AMF 3 integers are serialized using a variable length unsigned 29-bit integer.
在AMF3中，整数是使用可变长度的无符号的29-bit整数来序列化。
The ActionScript 3.0 integer types - a signed 'int' type and an unsigned 'uint' type - are also represented using 29-bits in AVM+.
AS3.0的整数类型，int表示有符号的，uint标识无符号的，在AVM+中也使用29-bit来表示。
If the value of an unsigned integer (uint) is greater or equal to 2^{29} or if the value of a signed integer (int) is greater than or equal to 2^{28} then it will be represented by AVM+ as a double and thus serialized in using the AMF 3 double type.
如果uint整数大于或等于2^{29},int整数大于等于2^{28}，在AVM+中会用双精度浮点数来表示，并用AMF3的双精度浮点数来序列化他。

```
integer-type         =  integer-marker U29
```

3.7 double type

The AMF 3 double type is encoded in the same manner as the AMF 0 Number type.
AMF3的双精度浮点型和AMF0的的双精度浮点型的编码是一样的。
This type is used to encode an ActionScript Number or an ActionScript int of value greater than or equal to 228 or an ActionScript uint of value greater than or equal to 229.
The encoded value is always an 8 byte IEEE-754 double precision floating point value in network byte order (sign bit in low memory).
编码的值在网络序列中总是一个8个字节的IEEE-754的双精度浮点值（符号位在低位内存）

```
double-type          =  double-marker DOUBLE
```

3.8 String type

ActionScript String values are represented using a single string type in AMF 3 - the concept of string and long string types from AMF 0 is not used.
AMF3中，AS的字符串的值用一个字符串类型来表示，AMF0中的字符串和长字符串的概念不使用了。
Strings can be sent as a reference to a previously occurring String by using an index to the implicit string reference table.
Strings are encoding using UTF-8 - however the header may either describe a string literal or a string reference.
（译者：以上都是之前翻译了的）
The empty String is never sent by reference.
空的字符串不会被应用发送

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

3.9 XMLDocument type

ActionScript 3.0 introduced a new XML type (see 3.13) however the legacy XMLDocument type is

retained in the language as flash.xml.XMLDocument.
虽然遗产类XMLDocument被保留了下了， AS3.0还是引进了XML类型。
Similar to AMF 0, the structure of an XMLDocument needs to be flattened into a string representation for serialization.
和AMF0类似，XMLDocument的结构需要转换为字符串表示来序列化。
As with other strings in AMF, the content is encoded in UTF-8.
在AMF中，因为需要字符串，其内容用UTF-8来编码。

XMLDocuments can be sent as a reference to a previously occurring XMLDocument instance by using an index to the implicit object reference table.
对于当前产生的XMLDocument实例，可以使用隐含的对象引用表的一个索引来进行引用发送。

```
U29X-value     =  U29   ; The first (low) bit is a flag with
                        ; value 1. The remaining 1 to 28
                        ; significant bits are used to encode the
                        ; byte-length of the UTF-8 encoded
                        ; representation of the XML or
                        ; XMLDocument.

xml-doc-type   =  xml-doc-marker (U29O-ref | (U29X-value
                  *(UTF8-char)))
```

Note that this encoding imposes some theoretical limits on the use of XMLDocument. The byte-length of each UTF-8 encoded XMLDocument instance is limited to 2^{28} - 1 bytes (approx 256 MB).
（译者：以上都是之前翻译了的）

3.10 Date type

In AMF 3 an ActionScript Date is serialized simply as the number of milliseconds elapsed since the epoch of midnight, 1st Jan 1970 in the UTC time zone.
在AMF3中， AS日期简单地序列化为一个从公元1970年1月1日零点，使用UTC时区的到现在的毫秒数。
Local time zone information is not sent.
本地时区信息不被发送。

Dates can be sent as a reference to a previously occurring Date instance by using an index to the implicit object reference table.

```
U29D-value   =  U29        ; The first (low) bit is a flag with
                           ; value 1. The remaining bits are not
                           ; used.

date-time    =  DOUBLE     ; A 64-bit integer value transported
                           ; as a double.

date-type    =  date-marker (U29O-ref | (U29D-value date-time))
```

3.11 Array type

ActionScript Arrays are described based on the nature of their indices, i.e. their type and how they are positioned in the Array.
AS的数组是基于他的自然索引来描述的，例如：他们的类型和他们在数组中的位置
The following table outlines the terms and their meaning:

术语和他的意义列出如下：

strict	contains only ordinal (numeric) indices 仅仅包含索引
dense	ordinal indices start at 0 and do not contain gaps between successive indices (that is, every index is defined from 0 for the length of the array) 索引从0开始，不包含连续索引中的间隙（也就是说：每一个索引对于数组的长度来说都是从0开始定义的）
sparse	contains at least one gap between two indices 在两个索引中至少包含一个间隙
associative	contains at least one non-ordinal (string) index (sometimes referred to as an ECMA Array) 至少包含一个非顺序（字符串）索引（一些时候是ECMA数组）

AMF considers Arrays in two parts, the dense portion and the associative portion.
AMF把数组分为两个部分，密集部分和关联部分。
The binary representation of the associative portion consists of name/value pairs (potentially none) terminated by an empty string.
对于关联部分的二进制表示由以空字符串结束的名/值对（可能没有）组成。
The binary representation of the dense portion is the size of the dense portion (potentially zero) followed by an ordered list of values (potentially none).
密集部分的二进制表示是跟着顺序的列表值（可能没有）的密集部分的大小（可能是零）
The order these are written in AMF is first the size of the dense portion, an empty string terminated list of name/value pairs, followed by size values.
在AMF中写入的顺序是第一个密集部分的大小，空的字符串。

```
U29A-value    =  U29    ; The first (low) bit is a flag with
                        ; value 1. The remaining 1 to 28
                        ; significant bits are used to encode the
                        ; count of the dense portion of the
                        ; Array.

assoc-value   =  UTF-8-vr value-type

array-type    =  array-marker (U29O-ref | (U29A-value
                 (UTF-8-empty | *(assoc-value) UTF-8-empty)
                 *(value-type)))
```

3.12 Object type

A single AMF 3 type handles ActionScript Objects and custom user classes. The term 'traits' is used to describe the defining characteristics of a class. In addition to 'anonymous' objects and 'typed' objects, ActionScript 3.0 introduces two further traits to describe how objects are serialized, namely 'dynamic' and 'externalizable'. The following table outlines the terms and their meanings:

Anonymous	an instance of the actual ActionScript Object type or an instance of a Class without a registered alias (that will be treated like an Object on deserialization) 真实的AS对象实例或没有别名注册的类对象实例（反序列化时，他会像对象一样处理）

Typed	an instance of a Class with a registered alias 有别名注册的类对象实例
Dynamic	an instance of a Class definition with the dynamic trait declared; 使用动态特性声明的类定义的实例 public variable members can be added and removed from instances dynamically at runtime 在运行时，可以动态的给实例添加和删除公共成员变量。
Externalizable	an instance of a Class that implements flash.utils.IExternalizable and completely controls the serialization of its members (no property names are included in the trait information). 实现了flash.utils.IExternalizable接口的类实例并且完全控制他的成员的序列化（在他们的特性信息中没有属性名）

In addition to these characteristics, an object's traits information may also include a set of public variable and public read-writeable property names defined on a Class (i.e. public members that are not Functions).
除了这些特性，对象特性信息也可能包括一组在类中定义公共变量和公共可读写的属性名（例如：公共成员但不是函数）。
The order of the member names is important as the member values that follow the traits information will be in the exact same order.
成员名字的顺序是重要的，因为在特性信息中的成员值也是同样的顺序。
These members are considered sealed members as they are explicitly defined by the type.
这些成员是封装的成员，因为他们是被类型显示定义的。

If the type is dynamic, a further section may be included after the sealed members that lists dynamic members as name / value pairs.
如果类型是动态的，在封装好的成员之后的部分会列出动态成员的名/值对
One continues to read in dynamic members until a name that is the empty string is encountered.
读取动态成员指导名称是空的字符串。

Objects can be sent as a reference to a previously occurring Object by using an index to the implicit object reference table. Further more, trait information can also be sent as a reference to a previously occurring set of traits by using an index to the implicit traits reference table.

U29O-ref	U29	; The first (low) bit is a flag 低位第一位（bit）是一个标志。 ; (representing whether an instance ; follows 表示是否跟着个实例对象) with value 0 to imply that ; this is not an instance but a ; reference. 为0时表示不是一个实例而是一个引用。 The remaining 1 to 28 ; significant bits are used to encode an ; object reference index (an integer). 余下的1到28个有效的位是用来编码一个对象引用的索引（一个整数）

U29O-traits-ref	U29	; The first (low) bit is a flag with ; value 1. 第一个低位标志位值为1 The second bit is a flag ; (representing whether a trait ; reference follows) with value 0 to ; imply that this objects traits are ; being sent by reference. 第二个位是个标志位，0表示引用发送对象特性 The remaining ; 1 to 27 significant bits are used to ; encode a trait reference index (an ; integer). 余下的27位表示引用值（一个整数）
U29O-traits-ext	U29	; The first (low) bit is a flag with ; value 1. The second bit is a flag with ; value 1. The third bit is a flag with ; value 1. 前三位标志位都是1 The remaining 1 to 26 ; significant bits are not significant ; (the traits member count would always ; be 0). 余下的26位无意义（特性成员的数量必须是0）
U29O-traits	U29	; The first (low) bit is a flag with ; value 1. The second bit is a flag with ; value 1. The third bit is a flag with ; value 0. 第一个标志位为1，第二个标志位为1，第三个标志位为0。 The fourth bit is a flag ; specifying whether the type is ; dynamic. 第四个标志位表示类型是不是动态的。 A value of 0 implies not ; dynamic, a value of 1 implies dynamic. 0表示不是动态的，1表示是动态的。 ; Dynamic types may have a set of name ; value pairs for dynamic members after ; the sealed member section. 在封装好的成员之后，动态类型可能拥有一组动态成员的名/值对。 The remaining 1 to 25 significant bits are ; used to encode the number of sealed ; traits member names that follow after ; the class name (an integer). 余下的25位用来编码封装好的特性成员的名成的数量（一个整数），在他之后是类名。

class-name	UTF-8-vr	; Note: use the empty string for ; anonymous classes. 注意：对于匿名类，使用空的字符串。
dynamic-member	UTF-8-vr value-type	; Another dynamic member follows ; until the string-type is the ; empty string.

		跟着的另一个动态成员，直到是个空的字符串类型。

object-type	object-marker (U29O-ref | (U29O-traits-ext class-name *(U8)) | U29O-traits-ref | (U29O-traits class-name *(UTF-8-vr))) *(value-type) *(dynamic-member)))

Note that for `U29O-traits-ext`, after the class-name follows an indeterminable number of bytes as `*(U8)`.
注意对于`U29O-traits-ext`，类名之后跟着的`*(U8)`字节的不可决定的成员。
This represents the completely custom serialization of "externalizable" types.
这是表示的完全自定义序列化的外部化类型。
The client and server have an agreement as to how to read in this information.
客户和服务端有一个协议读取这部分信息。

3.13 XML type

ActionScript 3.0 introduces a new XML type that supports E4X syntax.
AS3.0引进了一个新的XML类型，支持E4X句法。
For serialization purposes the XML type needs to be flattened into a string representation.
XML类型转换为字符串表示进行序列化。
As with other strings in AMF, the content is encoded using UTF-8.
像AMF的其他字符串一样，使用UTF-8进行编码。
XML instances can be sent as a reference to a previously occurring XML instance by using an index to the implicit object reference table.

```
xml-type       =  xml-marker (U29O-ref |
                  (U29X-value *(UTF8-char)))
```

Note that this encoding imposes some theoretical limits on the use of XML. The byte-length of each UTF-8 encoded XML instance is limited to 2^{28} - 1 bytes (approx 256 MB).

3.14 ByteArray type

ActionScript 3.0 introduces a new type to hold an Array of bytes, namely ByteArray.
AS3.0引进了新的类型表示字节数组，即ByteArray。
AMF 3 serializes this type using a variable length encoding 29-bit integer for the byte-length prefix followed by the raw bytes of the ByteArray.
AMF3用可变长度的29-bit的整数编码字节的长度作为前缀，其后跟着ByteArray的原始字节来序列化。
ByteArray instances can be sent as a reference to a previously occurring ByteArray instance by using an index to the implicit object reference table.

```
U29B-value     =  U29    ; The first (low) bit is a flag with
                         ; value 1. The remaining 1 to 28
                         ; significant bits are used to encode the
                         ; byte-length of the ByteArray.

bytearray-type  =  bytearray-marker (U29O-ref | U29B-value *(U8))
```

Note that this encoding imposes some theoretical limits on the use of ByteArray. The maximum byte-length of each ByteArray instance is limited to 2^{28} - 1 bytes (approx 256 MB).

4. Usages of AMF 3

4.1 NetConnection and AMF 3

In addition to serializing ActionScript types, AMF can be used in the asynchronous invocations of remote services.
因为序列化的AS类型，AMF可用于异步的远程调用服务。

A simple messaging structure is used to send a batch of requests to a remote endpoint.
一个简单的消息结构是使用对远程端点发送一批请求。
The format of this messaging structure is AMF 0 (See [AMF0].
这个消息结构的格式AMF0(参考[AMF0])
A context header value or message body can switch to AMF 3 encoding using the special avmplus-object-marker type.
使用一个特殊的avmplus-object-marker的类型可使一个上下文头值或消息体切换到AMF3编码。
Similar to AMF 0, AMF 3 object reference tables, object trait reference tables and string reference tables must be reset each time a new context header or message is processed.
和AMF0类似，AMF3对象引用表，对象特性引用表和字符串引用表必须在每次新的上下文头或消息处理完时复位一次。

4.1.1 NetConnection in ActionScript 3.0

The qualified class name for NetConnection in ActionScript 3.0 is flash.net.NetConnection.
NetConnection在AS3.0中的合适的类名是flash.net.NetConnection。
This class continues to use a responder to handle result and status responses from a remote endpoint, however, a strongly typed Responder class is now required. The fully qualified class name is flash.net.Responder.
这个类继续使用一个响应器处理从远程端点来的结果和状态，然而，现在需要强类型的响应器类，类的全名是flash.net.Responder
For events other than normal result and status responses NetConnection dispatches events for which the developer can add listeners.
除了结果和状态响应的事件，也可以响应开发真可能添加的监听器分派的事件。
These events are outlined below:
这些事件有：

asyncError	Dispatched when an exception is thrown asynchronously - i.e. from native asynchronous code. 当异步的抛出异常时分派，如从本地异步代码
ioError	Dispatched when an input or output error occurs that causes a network operation to fail. 引起网络运行失败的输入/输出错误发生时分派
netStatus	Dispatched when a NetConnection object is reporting its status or error condition. NetConnection对象报告他的状态或错误的条件时分派
securityError	Dispatched if a call to NetConnection.call() attempts to connect to a server outside the caller's security sandbox. 当NetConnection试图连接调用者的安全沙箱之外的服务器时发布

To handle an AMF context header a suitable method needs to be available, matching the header name.
为了处理AMF上下文头，需要一个可用的合适的方法，并且和头的名称相匹配。

NetConnection is now a sealed type so either it must be subclassed or an object with a suitable implementation needs to be set for the NetConnection client property.
NetConnection 现在是一个封装好的类型，因为他必须是某个子类或者实现合适的接口可以对 他的客户属性做需要的设定。

4.2 ByteArray, IDataInput and IDataOutput

ActionScript 3.0 introduced a new type to support the manipulation of raw data in the form of an Array of bytes, namely flash.utils.ByteArray.
为了支持字节数组形式的原始数据的操作， AS3.0引入了一个新的数据类型，即 flash.utils.ByteArray。
To assist with ActionScript Object serialization and copying, ByteArray implements flash.utils.IDataInput and flash.utils.IDataOutput.
为了辅助AS对象的序列化和复制， ByteArray实现了flash.utils.IDataInput和flash.utils.IDataOutput 两个接口。
These interfaces specify utility methods that help write common types to byte streams.
这两个接口定义了使得常用类型转换为字节流的工具方法。
Two methods of interest are IDataOutput.writeObject and IDataInput.readObject.
常用的两个方法是DataOutput.writeObject和IDataInput.readObject。
These methods encode objects using AMF.
这两个方法使用AMF格式编码实例对象。
The version of AMF used to encode object data is controlled by the ByteArray.
编码实例对象的AMF版本是由ByteArray控制的。
objectEncoding method, which can be set to either AMF 3 or AMF 0.
objectEncoding方法，可以适当是AMF3或者AMF0。
An enumeration type, flash.net.ObjectEncoding, holds the constants for the versions of AMF - ObjectEncoding.AMF0 and ObjectEncoding.AMF3 respectively.
一个枚举的类型flash.net.ObjectEncoding，拥有AMF版本的常量，分别为ObjectEncoding.AMF0 和ObjectEncoding.AMF3 。
Note that ByteArray.writeObject uses one version of AMF to encode the entire object. Unlike NetConnection, ByteArray does not start out in AMF 0 and switch to AMF 3 (with the objectEncoding property set to AMF 3).
注意：ByteArray使用AMF的一个版本编码整个对象。不像NetConnection，ByteArray不在AMF0 中开始输出并且切换到AMF3(通过设定objectEncoding属性到AMF3)
Also note that ByteArray uses a new set of implicit reference tables for objects, object traits and strings for each readObject and writeObject call.
还有注意：ByteArray对每一个对象的读写调用都使用一组新的对象，对象特性和字符串引用表。

5. Normative References 标准的参考文档

[AMF0]	Adobe Systems Inc. "Action Message Format - AMF 0", 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.