# Protocol序列化协议说明

## 概念与格式

一下名称代表在序列化协议中的一些专有概念：

### Protocol序列化协议

定义：

组成：协议描述 和 数据包格式

详细解析

协议描述：包含对象定义的描述信息，传输值对象和枚举通讯前需要从服务端获取并解析

数据包格式：

### 协议描述

定义：包含序列化对象定义的描述内容，为byte数组，并先使用zlib后QuickLZ进行两次压缩，使用自定义对象和枚举通讯前需要从服务端获取并解析<协议描述>

格式：[[类型描述]](#_类型描述), [类型描述] …

详细解析

### 类型描述

用于序列化传输的类型定义，目前包含[[值对象定义]](#_值对象定义)和[[枚举定义]](#_包头)，详见以下内容

### 枚举定义

定义：定义枚举常量，对应JAVA 的enum，即枚举名称和枚举数值的对应映射关系

格式：定义标识(0x00) + 类标识 + 类名 + [枚举数值 + 枚举名称] …

详细解析

定义标识：枚举定义标识固定为 0x00, 共1 bytes

类标识：为short整数，序列化时使用该标识映射对应JAVA类

类名：枚举类型的JAVA类名，UTF8字节数组

枚举数值：枚举名称对应的数值，为short 类型整数

枚举名称：枚举名称，UTF8字节数组

范例：

0, 0, 1, 0, 11, 116, 101, 115, 116, 46, 83, 116, 97, 116, 117, 115, 0, 2, 0, 3, 78, 69, 87, 0, 3, 79, 76, 68

枚举标识: 1

枚举名: test.Status

枚举值: [NEW => 0 , OLD => 1]

### 值对象定义

定义：定义值对象类型，对应JAVA 的class

格式：定义标识(0x01) + 类标识 + 类名 + 字段数量 + [字段名称] …

详细解析

定义标识：值对象定义标识固定为 0x01, 共1 bytes

类标识：为short整数，序列化时使用该标识映射对应JAVA类

类名：枚举类型的JAVA类名，UTF8字节数组

字段数量：值对象的字段数量，为short 类型整数

字段名称：字段名称对应的UTF8字节数组

范例：

1, 0, 1, 0, 11, 116, 101, 115, 116, 46, 80, 101, 114, 115, 111, 110, 0, 2, 0, 2, 105, 100, 0, 4, 110, 97, 109, 101

值对象标识: 1

类名：test.Person

字段: [id = > 0, name = > 1]

# 数据包格式

定义：

组成：数据标识 + 数据内容

数据标识：byte数值，其中前 4 bit为数据类型，后4 bit根据数据类型不同而有不同含义

数据类型：用4个bit表示，为类型标识的高4位( 和0xF0进行按位与运算的结果)

|  |  |  |
| --- | --- | --- |
| 名称 | 值 | 说明 |
| *OBJECT* | 0xF0 | 值对象类型 |
| *STRING* | 0xE0 | 字符串类型 |
| *ARRAY* | 0xD0 | 数组类型 |
| *MAP* | 0xC0 | 字典类型 |
| *BYTE\_ARRAY* | 0xB0 | byte数组类型 |
| *DATE\_TIME* | 0xA0 | 日期类型 |
| *COLLECTION* | 0x90 | 集合类型 |
| *ENUM* | 0x50 | 枚举类型 |
| *BOOLEAN* | 0x20 | 布尔类型 |
| *NUMBER* | 0x10 | 数字类型 |
| *INTERNAL* | 0x00 | 内部类型 |

### OBJECT

值对象类型

对象数组类型：

数据标识： 0xF0 代表*值对象实例*

0xF1 代表*值对象引用*

数据内容：

1. 对象实例：

数据内容：对象标识 + 字段数量 + 字段内容

对象标识：[[对象定义]](#_值对象定义)唯一标识，用变长表示的32位整数，参见VERINT32

字段数量：用变长表示的32位整数，参见VERINT32

字段内容：数量为字段数量的[[数据包]](#_数据包格式)重复

1. 集合引用：

对象内容：参见[[对象引用]](#_对象引用)

范例：

F0 01 02 12 01 E0 04 4E 41 4D 45

代表标识为 1 的值对象,

有两个属性,

1. INT64值 1
2. 字符串值 “NAME”

### STRING

字符串类型：

数据标识： 0xE0 代表*字符串实例*

0xE1 代表*字符串引用*

0xE2 代表*压缩的字符串对象*

数据内容：

1. 字符串实例：参见[[字符串对象]](#_字符串格式：字节长度_+_UTF8字节数组)
2. 字符串引用：参见[[字符串引用]](#_字符串引用)
3. 压缩的字符串对象：同[[字符串对象]](#_字符串格式：字节长度_+_UTF8字节数组)相似，

格式：字节长度 + QuickLZ压缩的UTF8字节数组

范例：

E0 03 41 42 43 代表字符串 “ABC”

### ARRAY

对象数组类型：

数据标识： 0xD0 代表*数组实例*

0xD1 代表*数组引用*

数据内容：

1. 数组实例：

数据内容：数组长度 + 数组内容

数组长度：用变长表示的32位整数，参见VERINT32

数组内容：数量为数组长度的 [[数据包]](#_数据包格式) 重复

1. 数组引用：

数据内容：参见[[对象引用]](#_对象引用)

范例：

D0 01 11 01 ：代表数组 [1]

D0 05 11 01 11 02 11 03 11 04 11 05 ：代表数组 [1, 2, 3, 4]

### MAP

Key-Value集合类型

对象数组类型：

数据标识： 0xC0 代表*集合实例*

0xC1 代表*集合引用*

数据内容：

1. 集合实例：

数据内容：集合长度 + 数组内容

集合长度：用变长表示的32位整数，参见VERINT32

集合内容：数量为数组长度的 [KEY[[数据包]](#_数据包格式) + VALUE[[数据包]](#_数据包格式)] 重复

1. 集合引用：

数据内容：参见[[对象引用]](#_对象引用)

范例：

C0 01 E0 03 6B 65 79 11 64 代表MAP{“key” : 100}

### BYTE\_ARRAY

BYTE[]类型：

数据标识： 0xB0 代表*字节数组*

数据内容：格式: 长度 + 内容

1. 长度：用变长表示的32位整数，参见VERINT32
2. 内容：字节数组[…]

范例：

B0 03 00 01 00 代表字节数组[00, 01, 00]

### DATE\_TIME

布尔类型：

数据标识： 0xA0 代表*用UNIX时间戳表示的时间*

数据内容： 距离 1970-01-01 00:00:00 的秒数, 用变长表示的64位整数，参见VERINT64

### COLLECTION

对象列表类型：

数据标识： 0x90 代表*列表实例*

0x91 代表*列表引用*

数据内容：

1. 列表实例：

数据内容：列表长度 + 列表内容

列表长度：用变长表示的32位整数，参见VERINT32

列表内容：数量为列表长度的 [[数据包]](#_数据包格式) 重复

1. 列表引用：

数据内容：参见[[对象引用]](#_对象引用)

范例：

90 01 11 01 ：代表列表[1]

90 05 11 01 11 02 11 03 11 04 11 05 ：代表列表[1, 2, 3, 4]

### ENUM

枚举类型：

数据标识： 0x50 代表*枚举实例*

数据内容：枚举标识 + 枚举索引

枚举标识：[[枚举定义]](#_包头)的唯一标识, 用变长表示的32位整数，参见VERINT32

枚举索引：枚举的值下标, 用变长表示的32位整数，参见VERINT32

范例：

50 01 00 ： 代表标识为1，值为0的枚举值

### BOOLEAN

布尔类型：

数据标识： 0x20 代表*false, 假*

0x21 代表*true, 真*

数据内容：无

范例：

false编码后表示为 0x20

true编码后表示为 0x21

### NUMBER

数值类型

数据标识： 0x11 代表*INT32+, 32位正整数*

0x12 代表*INT64+, 64位正整数*

0x13 代表*FLOAT, 32位浮点数*

0x14代表*DOUBLE, 64位浮点数*

0x19代表*INT32-, 32位负整数*

0x1A 代表*INT64-, 64位负整数*

数据内容：

1. 32位正整数:

变长表示的32位整数，参见VERINT32

1. 64位正整数:

变长表示的64位整数，参见VERINT64

1. 32位正整数:

变长表示的32位整数，参见VERINT32

1. 64位正整数:

变长表示的64位整数，参见VERINT64

1. 32位浮点数:

4个字节, 浮点数的内存表示(BIG-ENDIAN)

1. 64位负整数:

8个字节, 浮点数的内存表示(BIG-ENDIAN)

范例：

INT32的 0 编码后表示为 0x11 00

INT32的100编码后表示为 0x11 64

INT32的1024编码后表示为 0x11 82 04 00

INT64的 -100000000编码后表示为0x1A 84 05 F5 E1 00

Float的0.1编码后表示为0x13 3D CC CC CD

Double的0.1编码后表示为0x14 3F B9 99 99 99 99 99 9A

### INTERNAL

内部类型

数据标识： 0x00 代表*UNKOWN, 未知类型, 无法处理*

0x01 代表*NULL, 代表空值*

数据内容：无

范例：

NULL值编码后表示为 0x01

协议描述数据内容所涉及到的定义如下：

### 字符串对象

格式：字节长度 + UTF8字节数组

如：字符串 ”ABC”的表示为 03 41 42 43

### 字符串引用

在一个消息包中出现的字符串，在首次出现时会加入[字符串引用池]，并给予自增的唯一标识，以后则通过引用标识出现。

引用标识格式：变长表示的32位整数，参见VERINT32

### 对象引用

在一个消息包中出现的对象，在首次出现时会加入[对象引用池]，并给予自增的唯一标识，以后则通过引用标识出现。

引用标识格式：变长表示的32位整数，参见VERINT32

\*\*\*注意\*\*\*：字符串引用和对象引用使用两个不同的引用池

### VERINT32 ：

变长的正整数, 值范围0 - 0x7fffffff

数据体长度为1-5字节，具体根据首字节解释结果区分

首字节：

1. 0x00 – 0x7f : 数据体长度 1 字节, 值范围 0 -0x7F
2. 0x81: 数据体长度 2 字节, 值范围 0x80 – 0xFF
3. 0x82: 数据体长度 3 字节, 值范围 0x0100 – 0xFFFF
4. 0x83: 数据体长度 4 字节, 值范围 0x010000 – 0xFFFFFF
5. 0x84: 数据体长度 5 字节, 值范围 0x01000000 – 0x7FFFFFF

### VERINT64

变长的正整数, 值范围0 - 0x00ffffffffffffffL

数据体长度为1-8字节，具体根据首字节解释结果区分

首字节：

1. 0x00 – 0x7f : 数据体长度 1 字节, 值范围 0 -0x7F
2. 0x81: 数据体长度 2 字节, 值范围 0x80 – 0xFF
3. 0x82: 数据体长度 3 字节, 值范围 0x0100 – 0xFFFF
4. 0x83: 数据体长度 4 字节, 值范围 0x010000 – 0xFFFFFF
5. 0x84: 数据体长度 3 字节, 值范围 0x01000000 – 0xFFFFFFF
6. 0x85: 数据体长度 5 字节, 值范围 0x0100000000 – 0xFFFFFFFFF
7. 0x86: 数据体长度 6 字节, 值范围 0x010000000000 – 0xFFFFFFFFFFF
8. 0x87: 数据体长度 7 字节, 值范围 0x01000000000000 – 0xFFFFFFFFFFFFF
9. 0x88: 数据体长度 8 字节, 值范围 0x0100000000000000 – 0xFFFFFFFFFFFFFFF