TIBIA PROTOCOL DOCUMENTATION PROJECT

Written by Khaos
since 11.1.2006.

THIS IS A WORK IN PROGRESS!

If some of your friends cannot reach this document directly from us,
it's for a good reason probably!

Always wanted to make your own packet parser, a packet hack or even a ... khm ... client? Perhaps modify your OTserv with more power or construct your own server from scratch? Well seek out no more. This document is all that an experienced programmer that knows how to deal with sockets needs to know to make a program that (mis)uses Tibia protocol.

I am entirely not responsible for the data stated in this document.

Note: I do not endorse use of any hack. Hell, I don't use any myself. I simply feel that it's not fair to use them and that it ruins game not only for others, but also for me.

These icons are used:
 - supported/required in protocol version 7.1
 - supported/required in protocol version 7.4 ← this protocol is not verified
 - supported/required in protocol version 7.5
 - supported/required in protocol version 7.55
 - supported/required in protocol version 7.6
 - supported/required in all 7 protocol versions
 - correct usage & function is unknown; this is a wild guess
 - correct usage & function is unknown by the author; this is a wild guess
 - per-protocol changes are undocumented (perhaps there aren't any, this is untested)
 - this thing repeats several times (guess how many)
 - packet has no parameters, byte comes alone

Table of Contents

1. Principles

Tibia protocol is organized in messages and packets.

1.1. Messages

Generally messages are in this format:

2 bytes	Message length
<message specific>	Packets, one after another

You should wait until the entire message has been received and then start processing it.

1.2. Packets

Each packet begins with one byte that identifies it. Then you must at least receive it according to the packet specific syntax to be able to go on.

1 byte	Packet id
<packet specific>	Packet data, one after another

2. Gameworld Connection Packet Specifications & Listings

There is a lot of packets. We'll discuss the known ones here – they're main reason for this documentation. I've mostly discovered them by analyzing OTserv – otherwise I wouldn't know it! Some are still being discovered (as in new branch revmovesys; there appear some so-far OTserv-unused and unknown new bytes)

Another thing you need to know is that Tibia's got two connections in the logon process – the so called charlist connection and the so called gameworld connection. In this version of document, I've decided not to describe charlist connection; I'll make a new chapter later.

2.1. Simple data formats

Byte = 1 byte
Short = 2 bytes
Int = 4 bytes
Long = 8 bytes

It is important that low-endian is used, e.g. number 258 is written and transmitted like this (in hex):
0x02 0x01
Make sure you transmit it correctly if you want to do it byte per byte.

2.2. Complex data formats

These are formats that are made up of some other formats.

2.2.1. String

Short	Length
<string dependant>	characters; it's not null terminated

2.2.2. Position

Short	X coordinate
Short	Y coordinate
Byte	Z coordinate

2.2.3. ThingDescription

Short	ItemId

How we go on depends on ItemId and specific item's properties.
Some even depend on some properties of thing sent before some others. Like, LookType is one such that decides what will go on next.

0x61 = creature appearing for first time

Long	Remove
Long	CreatureID
String	Name
Byte	Health percent
Byte	Direction
Byte	Looktype
LookType=0: Integer	ItemID
LookType!=0: Byte	HeadType
LookType!=0: Byte	BodyType
LookType!=0: Byte	LegsType
LookType!=0: Byte	FeetType
Byte	unknown
Byte	unknown
Integer	speedIndex
Byte	skull

Byte	? ? ? ? shield

0x62 = creature appearing for second etc. time

Long	CreatureID
Byte	Health percent
Byte	Direction
Byte	Looktype
LookType=0: Integer	ItemID
LookType!=0: Byte	HeadType
LookType!=0: Byte	BodyType
LookType!=0: Byte	LegsType
LookType!=0: Byte	FeetType
Byte	? unknown
Byte	? unknown
Integer	speedIndex
Byte	? ? ? ? skull
Byte	? ? ? ? shield

0x63 = creature with turning around update

Long	CreatureID
Byte	Direction

Others, only if stackable, splash or fluid container

Integer	countOrSubType

2.2.4. MapDescription

Server transmits a map in some pretty variable-size messages. They depend on numerous states of the client; the current location, current byte being processed and perhaps more. It is quite difficult to write a proper parser of the map, and requires all possible details of a certain item to work properly. Saving the map in memory and updating it is even trickier, because it depends on even more properties of the item. For example, grounds must always be added to the bottom; there are several layers of topitems and so on.

This dataformat has also changed between the versions. It is certain that CipSoft is handling this in a pretty different way, but this is what has so far been conceived.

First of all, depending on the location, we must prepare to receive different floors, and even in different order. The Z coordinate of the player's position can be a number of 0-14. In case it's 7 or less, it's in ground level or above. In case it's 8 or more, it's underground.

For ground levels, we receive all 8 ground floor levels – marked 0 through 7, inclusive. We receive them from 7 to 0 – backwards (from ground upwards). For underground, we receive 2 floors above us, our level and 1 level under us; those are levels z-2, z-1, z and z+1. They're received in this direction: from ground downward. If z is 14, we do not receive floor 15.

2.3. Server-transmitted packets

Here's a general list of gameworld packets (names made up, and listed as in The Outcast's source).

Note: some data formats might differ between protocols, don't forget to refer to correct protocol for data format!

These packets are transmitted by the server.

Packet ID (hex)	Name	Syntax
0x0A	P_MAPINIT	Long playerCreatureId Int unknownU16 (Byte reportBugs?)
0x0B	P_GMACTIONS	Byte unknown (32x)
0x14	P_LOGONERROR	String errorMessage
0x15	P_POPUP	String errorMessage
0x16	P_TOOMANYPLAYERS	String errorMessage
0x1e	P_PING	
0x64	P_PLAYERLOC	Position pos
0x65	P_MOVENORTH	MapDescription (18,1)
0x66	P_MOVEEAST	MapDescription (1,14)
0x67	P_MOVESOUTH	MapDescription (18,1)
0x68	P_MOVEWEST	MapDescription (1,14)
0x69	P_TILEUPDATE	Position pos TileDescription td
0x6a	P_ADDITEM	Position pos ThingDescription thing
0x6b	P_REPLACEITEM	Position pos Byte stackpos ThingDescription thing
0x6c	P_REMOVEITEM	Position pos Byte stackpos
0x6d	P_MOVETHING	Position pos1 Byte stackpos Position pos2
0x6e	P_CONTAINER	Byte index Short containerIcon Byte slotCount ThingDescription item
0x6f	P_CONTAINERCLOSE	Byte index
0x70	P_ADDITEMCONTAINER	Byte index ThingDescription itm
0x71	P_TRANSFORMITEMCONTAINER	Byte index Byte slot

		* ThingDescription itm
0x72	P_REMOVEITEMCONTAINER	* Byte index * Byte slot
0x78	P_INVENTORYEMPTY	* Byte invSlot
0x79	P_INVENTORYITEM	* Byte invSlot * ThingDescription itm
0x7d	P_TRADEREQ	* String otherperson * Byte slotCount * n*... ThingDescription itm
0x7e	P_TRADEACK	* String otherperson * Byte slotCount * n*... ThingDescription itm
0x7f	P_TRADECLOSE	*
0x82	P_LIGHTLEVEL	* Byte lightlevel * Byte lightcolor
0x83	P_MAGICEFFECT	* Position pos * Byte magiceffect
0x84	P_ANIMATEDTEXT	* Position pos * Byte color * String message
0x85	P_DISTANCESHOT	* Position pos1 * ? Byte stackposition * Position pos2
0x86	P_CREATURESQUARE	?. * Long creatureid * Byte squarecolor
0x8c	P_CREATUREHEALTH	* Long creatureid * Byte healthPercent
0x8d	P_CREATURELIGHT	? * Long creatureid * Byte ? * Byte ?
0x8e	P_SETOUTFIT	* Long creatureid * Byte lookType * Byte headType * Byte bodyType * Byte legsType * Byte feetType // can extended look go here too?
0x8f	P_CREATURESPEED	YIKES! I didnt handle this!
0x96	P_TEXTWINDOW	* Long windowId * ? Byte icon * ? Byte maxlength * String message
0xA0	P_STATUSMSG	* Status status

0xA1	P_SKILLS	* Skills skills
0xA2	P_ICONS	* Byte icons
0xA3	P_CANCELATTACK	*
0xAA	P_CREATURESPEAK	It's advanced, special section
0xAB	P_CHANNELSDIALOG	* Byte channelCount * n*... (Int channelId String channelName)
0xAC	P_CHANNEL	? * Int channelId * String channelName
0xAD	P_OPENPRIV	* String playerName
0xB4	P_TEXTMESSAGE	* Byte msgClass * String string
0xB5	P_CANCELWALK	* Byte direction
0xBE	P_FLOORUP	Advanced topic; read separate text
0xBF	P_FLOORDOWN	Advanced topic; read separate text
0xC8	P_OUTFITLIST	* Byte lookType * Byte headType * Byte bodyType * Byte legsType * Byte feetType * Byte firstModel * Byte lastModel
0xD2	P_VIPADD	* Long guid * String name * Byte isOnline
0xD3	P_VIPLOGIN	* Long guid
0xD4	P_VIPLOGOUT	* Long guid