Hotline Protocol

v1.1.1

By Virtual1, PDF'd by Wade Tregaskis (aka Sys)
virtualftp.neotek.net
http://2100sw.8m.com/

18/2/2000

Notes

You can download the latest version of this document from the 2100 Software web site (http://2100sw.8m.com/), as well as the Pacific Media & Design Hotline server, hotline://pmd.dhs.org/

This guide was generated during several days of intense use of OTSessionWatcher, to get the protocol figured out in preparation for the development of HotSocket, a RealBasic socket-based class for use as a Hotline Client/Server interface.

Thanks goes out to XAW and his development of the BHC, (Basic Hotline Client) whose RealBasic source code gave me (Virtual1) the insight necessary to begin to understand what I was seeing in the Session Watcher. Thanks also to the creators of mBot, without whose greed and lack of interest in releasing source code led me to make HotSocket, and thus, the need to make this guide. ;)

Chapter 1: Numbers & Strings

Anywhere there is a number that is preceeded by a length, (such as if the number is the only part of an object, like Socket or Icon) then the number can be a short OR a long. Hotline client and server software will always pick the smaller of the two when sending, though it does not hurt them to receive a long that is zero.

Numbers appear to be stored as "two's complement".

0 = 00 00 00 00 (you can send as a short 00 00)
1 = 00 00 00 01 (you can send as a short 00 01)
65535 = 00 00 FF FF (you can send as a short FF FF)
65536 = 00 01 00 00
2 = 7F FF FF FF
-2147483648 = 80 00 00 00 (now counting backwards toward zero)
-1 = FF FF FF FF

This is how 2's complement works. The only place you'd have to worry about this is if you ran into a file > 2.1gb that was returning a negative filesize or something. Remember that icons can be negative numbers.

It might be simpler to just send everything you can as a long. Some items must be sent as shorts if they don't have a length indicated in the protocol, such as all length indicators and some items in filelist/userlist entries. Anywhere you see short() or long(), it means that you MUST send it that way, because there is no length indicator. Anywhere you see number(), you need to send the length as a short, followed by the number, in your chosen format.

There are a few oddball exceptions. Icon numbers are numbers, and normally the server will send them in Number format. (length followed by the number) Userlists however, send the Socket, Icon, and Status objects without length bytes, (all as shorts) Filelists have the same limitation. In the event of a negative icon number, (it can happen, and does work) the icon will be sent as a SHORT two's complement number. They are very easy to convert fortunately... just lop off the the first two characters of the number. -3 changes from FF FF FF FD to FF FD. This limits your numeric range to -32768 <-> +32767.

Strings are always sent as a length (always a short) followed by the string's characters. Strings marked as "encoded" have each character of the string EOR'd with $FF. i.e. y=chr(255-asc(x)) It's not meant to be hard to crack, just hard to READ and easy to DO. Normal string format is commonly referred to as a "pascal string", if you happen to know it by that name.

Chapter 2: Objects

Objects are sent under the following format:

- object header
  - short (object ID number)
  - short (object length)
    Note: This does not count these four header bytes
- object data

if it's a number >=0 and <65536:
- short (number)

if it's a number >65535:
- long (number)

if it's a number <0:
- long (2^32+number)

if it is a string:
- string encoded strings have all chars EOF $FF)

if it is a filelistentry
- file type four characters, or "fldr" if folder, or
  "alis" if unresolved alias
- file creator four characters, or long(0) if folder
- long (contained files if folder, file size in bytes if doc/app)
- long (0) unused, always zero
- long (filename length)
- string (filename)

if it is a Path
- short (directory levels)
  - one or more directory levels
    - 00 just one chr(0), not sure what it's for
    - short (length of dir name)
    - string (dir name)

if it is a userlistentry
- short (socket)
- short (icon)
- short (status)
- short (length of nick)
- string (nick)

if it is a datetime
- eight bytes of date/time code

Integer objects are preceeded by a length for a reason. Do not assume that just because the object you are expecting can only be a number 0-50 ,that it will have to be sent as a short. It could be sent as a long, and we don't want to break the socket for such a simple misunderstanding. The reverse is true for longs, they may be sending an icon number that is 5, and decide to save a few bytes and send it as a short. Be careful.

Client objects and their ID numbers:

ID#	Name	Object Type
100	errormsg	string
101	message	string
102	nick	string
103	socket	number
104	icon	number
105	login	encoded string (NOT encoded in transaction #352)
106	password	encoded string
107	xferID	number - the ID number of the file transfer
108	xfersize	number - size of file xfer, in bytes
109	parameter	number - specifies icon for broadcast
110	privs	eight bytes - can make 64 flags, only use 27
111	???	
112	status	number (0 = black & active)
113	ban	short (0 = kick, 1 = ban)
114	chatwindow	Four random bytes?
115	subject	string - the new subject of a chat window
200	fileentry	file list entry
201	filename	string
202	path	path
203	"flt"	???
204	???	
205	infolongtype	string
206	infocreator	string
207	infosize	number
208	infocreated	datetime
209	infomodified	datetime
210	comment	string
211	newfilename	string
212	targetpath	path
213	infotype	string - the 4-character MacOS file type
300	userlistentry	user list entry

Chapter 3: Transactions

Transaction are sent under the following format:

- header
  - short (transaction class) 0 = info/request, 1 = reply
  - short (transaction ID number) server replies are always zero
  - long (task number)
  - long (error code) valid if this is a reply, 0 = ok, 1 = err
  - long (length of data block)
  - long (length of data block, again)
- data
  - short (number of objects in transaction)
  - objects can be one, many, or none

It would be wise to assume that objects can be passed in any and possibly random order. Ensure that your code allows for this, otherwise your client or server may not handle new or alien clients/servers.

Transaction IDs, classes, types, names, and objects:

ID#	Cls	Init	Type	Name	Object(s)
101	0	Client	request	GetNews	None
0	1	Server	reply	GetNews	message
102	0	Server	info	NewPost	message
103	0	Client	request	PostNews	message
104	0	Server	info	Broadcast	message
104	0	Server	info	Error	parameter, message
104	0	Server	info	PrivateMessage	socket, nick, message
105	0	Client	info	SendChat	message, chatwindow, parameter
106	0	Server	info	RelayChat	message, chatwindow
107	0	Client	request	Login	login, password, nick, icon
108	0	Client	request	SendPM	socket, message
109	0	Server	info	Agreement	message
110	0	Client	request	Kick	socket, ban
111	0	Server	info	Disconnected	message
112	0	Client	request	CreatePchatWith	socket
0	1	Server	reply	CreatePchatWith	chatwindow, socket, icon, status, nick

ID#	Cls	Init	Type	Name	Object(s)
113	0	Server	info	InvitedToPchat	chatwindow, socket, nick
113	0	Client	info	AddToPchat	socket,chatwindow
114	0	Client	Info	RejectPchat	chatwindow
115	0	Client	request ¬ RequestJoinPchat ¬		chatwindow
0	1	Server	reply	JoiningPchat	userlistentry, ¬ userlistentry, subject
116	0	Client	Info	LeavingPchat	chatwindow
117	0	Server	Info	JoinedPchat	chatwindow, socket, icon, status, nick
118	0	Server	Info	LeftPchat	chatwindow, socket
119	0	Server	Info	ChangeSubject	chatwindow, subject
120	0	Client	Info	ChangeSubject	chatwindow, subject
200	0	Client	request ¬ FolderList		(path)
0	1	Server	reply	FolderList	file entry
202	0	Client	request	Download	file name, path
0	1	Server	reply	Download	xfersize, xferID
203	0	Client	request	Upload	filename, path, xfersize
0	1	Server	reply	Upload	XferID
204	0	Client	request	MoveToTrash ¬	filename, path
205	0	Client	request	CreateFolder ¬	filename, path
206	0	Client	request	GetFileInfo ¬	filename, path
0	1	Server	reply	GetFileInfo	info type, info long type, info creator, file name, info created, info modified, info size, comment
207	0	Client	request	SetFileInfo	filename, path, new file name / comment
208	0	Client	request	MoveFile	file name, path, target path
209	0	Client	request	MakeAlias	file name, path, target path
300	0	Client	request	GetUserList	(none)

ID#	Cls	Init	Type	Name	Object(s)
0	0	Server	reply	GetUserList	user list entry
301	0	Server	info	UserChange	socket, icon, nick, status
302	0	Server	info	UserLeave	socket
303	0	Client	request	GetUserInfo	socket
0	1	Server	reply	GetUserInfo	message, nick
304	0	Client	info	ChangeNickIconStatus	icon, nick, status
350	0	Client	request	CreateUser	login, password, nick, privs
351	0	Client	request	DeleteUser	login
352	0	Client	request	OpenUser	login (NOT ENCODED)
0	1	Server	reply	OpenUser	login, password, privs, nick
353	0	Client	request	ModifyUser	nick, login, password, privs

Transactions dealing with files always include the filename. If the path is not included, root folder can be assumed. If the file is being moved or aliased, targetpath may also be included. If not, root is assumed as the target.

Notes:

• Transaction #105 (SendChat) is chat. When sent with a parameter of 1, it becomes an emote.

• Server reply to #352 always returns string(ctrl-G) as password.

• #353 must send a password string (chr(0)) if password was not changed. Returning string (ctrl-G) will result in that being the user's new password!

• Unless otherwise specified, a successful task reply will have an error code of 0 and no objects. Unsuccessful tasks will reply with an error code of 1 and the error message object.

• Server transaction #104 "Error" is used for when client sends a non-request that fails, such as trying to send public chat when they don't have chat privs.

• Reply to #303 (get info) will be missing the Nick object if you're getting info on a "ghost". (the HL client returns "Unnamed User")

• Task is a reply to a request. The object(s) included in the Task are dependent on what the request was. The Task can be matched back

to its request by using the task number portion of the header. It's probably possible to reuse task numbers, but don't re-issue a task number in a request until the current instance of that task number has been replied to! I have noticed that while the client can create tasks, the server cannot. This makes sense, because a server would eventually crash or eat up all available memory if it had to remember tasks until complete, assuming it was up a week or so and had clients leaving tasks hanging.

Chapter 4: Logging In

Before sending a login, you must establish a "pipe". Do this by connecting over TCP to the server (the default port is 5500), and then exchanging this "handshake" with the server:

CLIENT HELLO

- "TRTPHOTL" identifies this is a hotline client
- short (1) Minimum server version this client is compatible with?
- short (2) Client version?

TRTPHOTL (0) (1) (0) (2)

SERVER HELLO

- "TRTP"
- long (errorcode) - 0 = OK you are connected, 1 = rejected

TRTP (0) (0) (0) (0)

Once these have been exchanged, you can assume you are connected to a HL server and can proceed to login. Until you have received a success reply to your login transaction, the only other transaction you can submit is a request for disconnect.

Once logged in, you are by no means required to request a userlist, request news, or do anything else for that matter. The normal client will send the login and then immediately fire off a request for the agreement, userlist and news, before even receiving confirmation of a successful login.

Chapter 5: Further Notes

I have seen many admins and co-admins running around kicking idle users, saying they are "taking up bandwidth". I was wondering if this was true, and did some pondering. A user that is completely idle (no file xfers) by themselves will take zero bandwidth. There will be some bandwidth needed though for each time a user in the user list goes idle, goes active, changes nick or icon, leaves, arrives, or someone posts public chat. Each of these events requires a task to be sent to every user online, though the amount of data sent is quite small. News posts also go to all users, and those can be relatively large in comparison to the other transactions. It sounds kind of silly, but it is in everyone's best interest that on busy file serving server, you should be quiet and use chat only sparingly.