

Specifications for Building an *Application-Level* Firewall Proxy to Support RealAudio

Introduction

This document provides information to allow firewall developers to support the RealAudio Player-Server communications. The information is provided for the sole purpose of designing firewall software which supports RealAudio.

There are two types of firewall proxies that can be built to support RealAudio, Transparent and Application-Level.

Transparent Proxy Firewalls

A Transparent Proxy Firewall operates by monitoring network traffic and only letting through connections matching certain protocols. The Transparent Firewall relies on knowing details of all protocols it will support. Transparent proxies can perform their function without the client or server applications being modified or configured. For information on how to build a Transparent Proxy, please refer to the document entitled, *Specifications for Building a Transparent Firewall Proxy to Support RealAudio*.

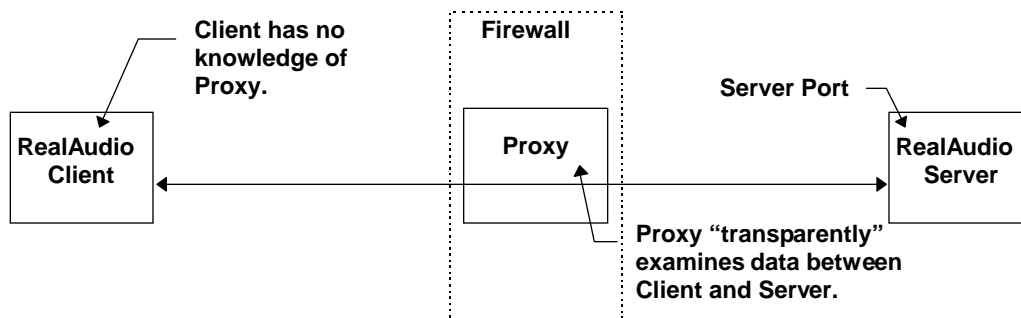


Figure 1 - Transparent Firewall

Application-Level Proxy Firewalls

Unlike a Transparent Proxy Firewall, an Application-Level Proxy firewall relies on the application inside the firewall having knowledge of the firewall proxy. All connections are made to the Proxy with the proxy then connecting to the desired external host. This means that an Application-Level Proxy needs to know about the client application connecting and what its Proxy protocol is, but does not need to know about the low level protocol details.

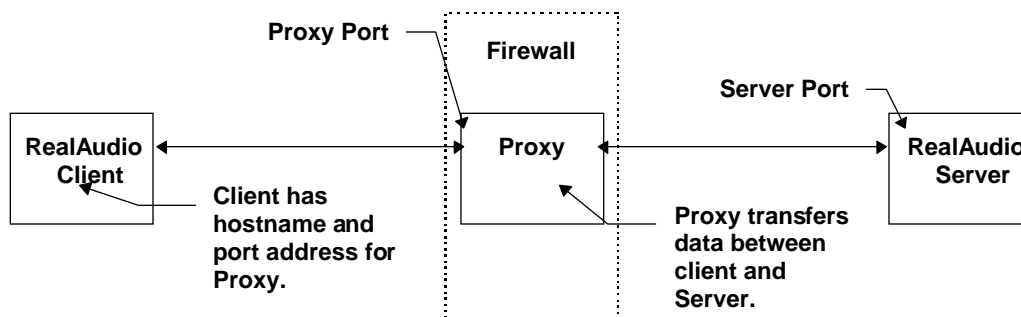


Figure 2 - Application-Level Proxy

RealAudio Communications

RealAudio Client / Server Communications

A RealAudio Client (Player) sets up either one or two network connections with the RealAudio Server. In the standard configuration, utilizing UDP, two connections are used: a full-duplex TCP connection, used for control and negotiation, and a simplex UDP connection from the RealAudio Server to the Player for audio data delivery. The UDP connection is used for the audio to allow the RealAudio Server and Player to handle error correction rather than the protocol. This allows a RealAudio Server to provide a better Audio stream when packet loss occurs. In TCP-Only mode, a single full-duplex TCP connection is used for both control and for audio data delivery from the RealAudio Server to the Player. The standard TCP connection port on a RealAudio Server is 7070. The following diagrams show both these configurations.

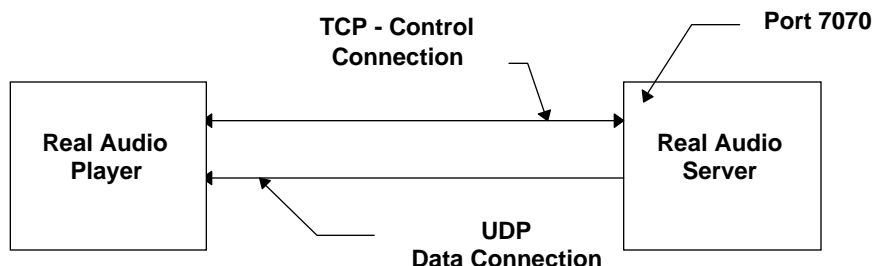


Figure 3 - RealAudio Client / Server Communications : Normal Mode (UDP)

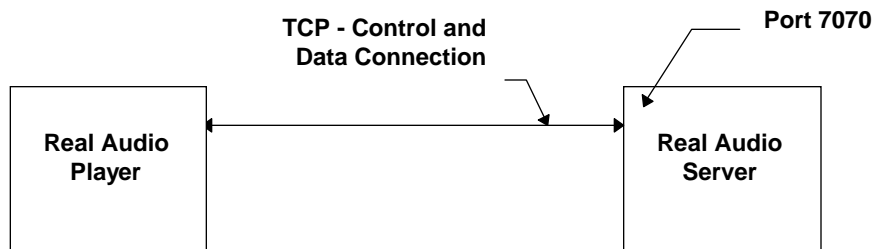


Figure 4 -RealAudio Client / Server Communications : TCP-Only Mode

Real Audio and Firewalls

A Transparent Proxy requires no explicit support in either the Player or RealAudio Server. It must know about the protocol being used between the Player and RealAudio Server but there is no actual connection to the proxy by either the Player or RealAudio Server. The details of producing a Transparent Proxy are covered in separate document.

The Application-Level Proxy is only supported by RealAudio 2.0 Players and requires the Player to have been configured with the hostname and port number used by the Application-Level Proxy. In this case the Player always connects to the Proxy and passes information to it which can then be used by the Proxy to find the RealAudio Server, and connect to it. Apart from an initial setup protocol between the Proxy and the Player, the dialogue is identical to that used between a Player and a RealAudio Server but all data flows via the Proxy, rather than directly to the RealAudio Server.

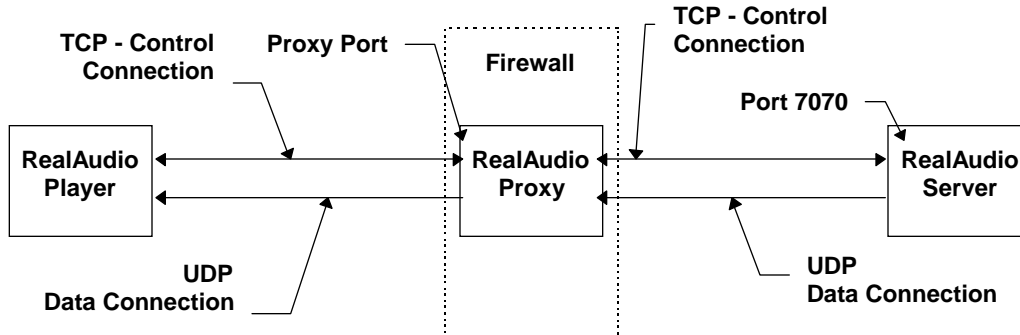


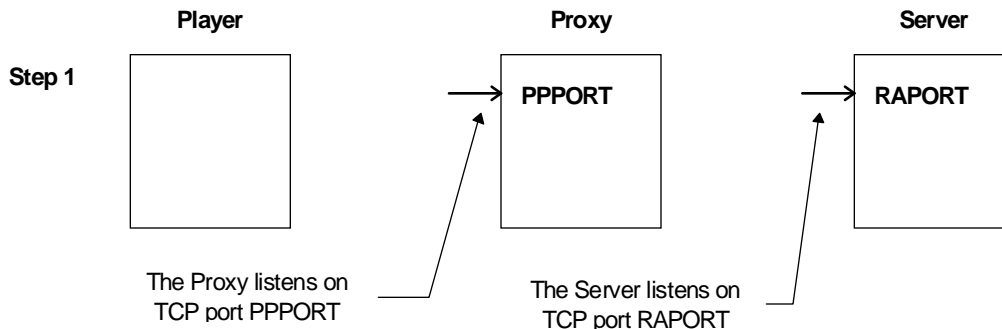
Figure 5 - RealAudio Application-Level Proxy Communications

Application-Level Proxy Handshaking

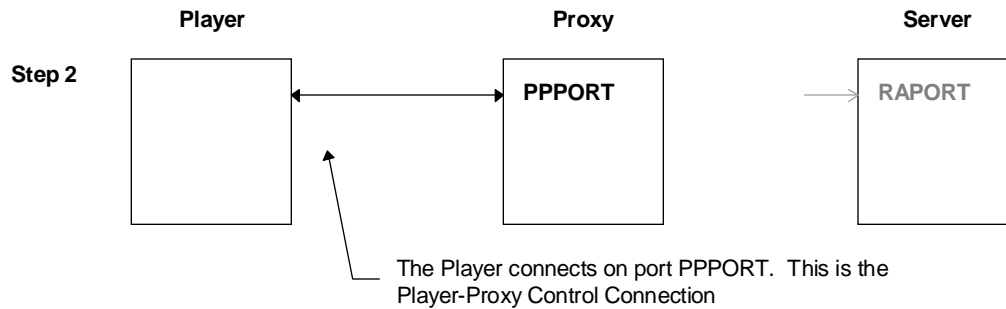
Application Level Proxies are only supported by RealAudio 2.0 Players and later. The interactions between the Player and the Proxy is identical to that between the Player and a Real Server, with an initial exchange to establish the Proxy connection. This is described in general terms below including schematic Diagrams that show the progression of messages and actions in the interaction. The connections from prior steps in the diagrams are grayed out in subsequent steps where they are not an active part of that step. All descriptions of messages refer to structured RealAudio Proxy Protocol messages. These messages are defined in Table 2 - RealAudio Proxy Protocol, Version 1, that follow this sample description.

Handshake and Communications Description

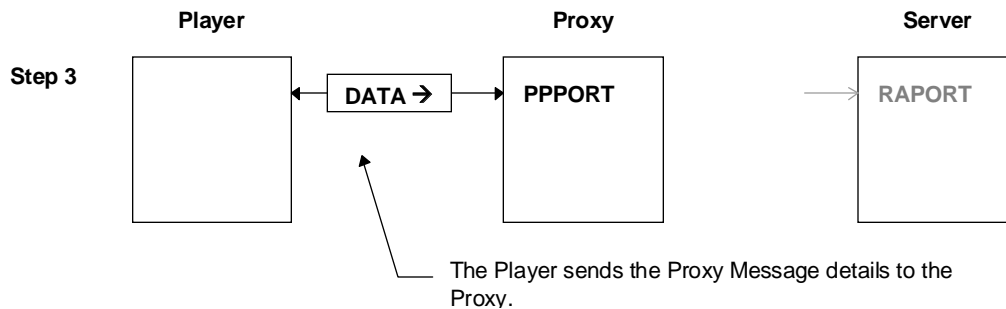
1. The Proxy listens on its defined TCP port, PPPORT. All Players within the firewall must be setup with the Proxy hostname and TCP port. The RealAudio Server is outside the firewall and is passively listening on TCP port RAPORT, usually port 7070, for any incoming connections.



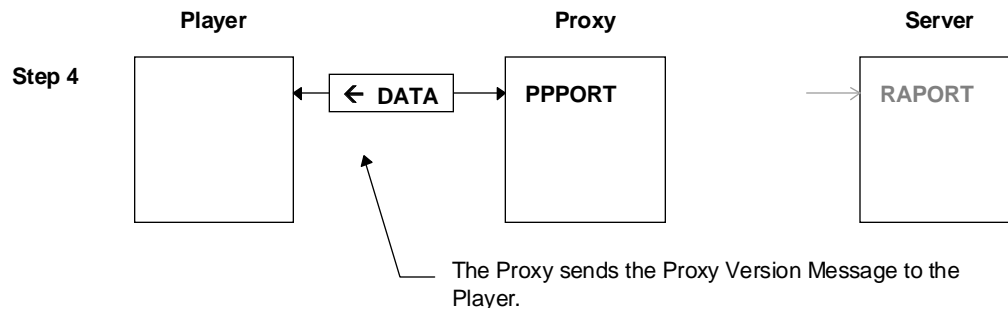
2. The Player connects (via TCP) to the Proxy on PPPORT port. This is a control connection between the Proxy and the Player.



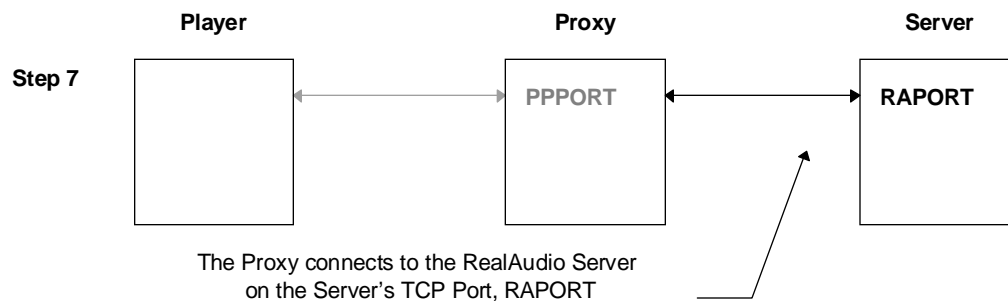
3. The Player sends a set of simple messages to the Proxy. The Proxy Protocol Version message must be the first message sent and the End Proxy message must be the last. There is no specified order for any other messages.
 - a) Proxy Protocol Version
 - b) Server hostname
 - c) TCP port number for the Server, call this RAPORT
 - d) Type of data connection, UDP or TCP
 - e) Offset to the location of the UDP data Port number in the subsequent communication between the Player and the Server. This offset is relative to the start of the normal Player / Server communication. This offset is not sent in TCP Only Mode.
 - f) End Proxy



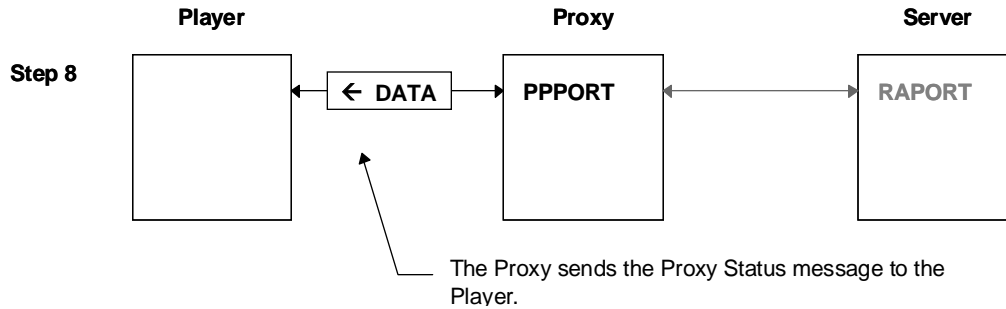
4. The Proxy responds to the Player with the Proxy Protocol Version message.



5. If the Proxy Protocol Version sent by the Proxy is not the same as that sent (i.e. supported) by the Player. The Player and the Proxy terminate the communication and the Player displays an error.
6. The Proxy reads in all incoming messages from the Player and collects and saves the following information
- a) Proxy Protocol Version
 - b) Server hostname
 - c) TCP port number for the Server, call this RAPORT
 - d) Type of data connection, UDP or TCP
 - e) Offset to UDP Port number
 - f) Proxy End
7. Once all of the required information in step 6 has been received by the Proxy from the Player. The Proxy opens a TCP connection to the RealAudio Server on TCP port RAPORT. The hostname and port for the RealAudio Server are those received in step 6b & c. This completes the TCP Control connection between the Player and the RealAudio Server, via the Proxy.



8. The Proxy sends a status message with a success code to the Player. If all required information is not received or if a error occurs in completing the connection to the server then a status message with an error code and message is sent to the Player.



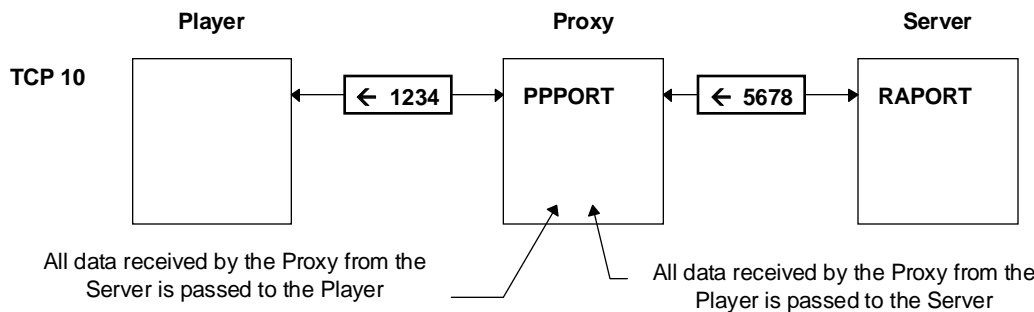
9. The Player examines the Status message code received from the Proxy and if it contains a Status of Success, it then begins its normal Player / Server interaction. The existing TCP control connection is kept open and is used for the ongoing Player / Server communications.

If the Status Message code received from the Proxy by the Player is an Error, the Player displays the corresponding error message and any error string supplied with the Status Message. The Player will then close the TCP control connection to the Proxy ending all interactions.

Depending on whether the data connection is **UDP** or **TCP**, this is determined by the data collected in step 6e, follow one of the set of steps below

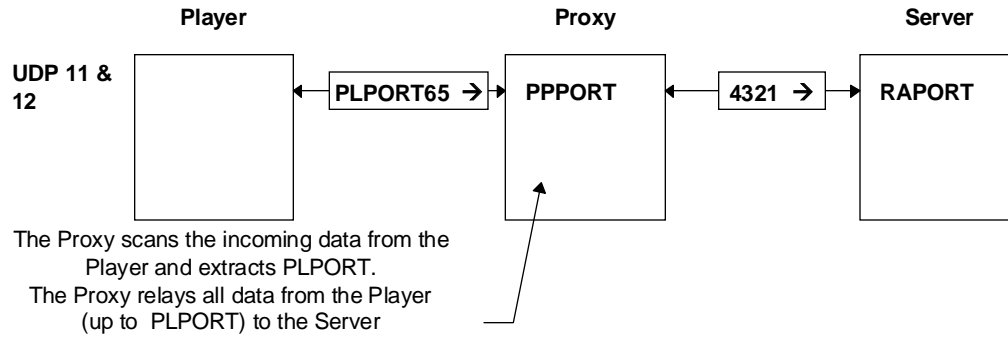
TCP-Only Connection

10. The Proxy transfers every byte read from the Player on the TCP Control connection to the Server on its TCP Control connection. The reverse is also true, all data received from the Server by the Proxy is also transferred to the Player.

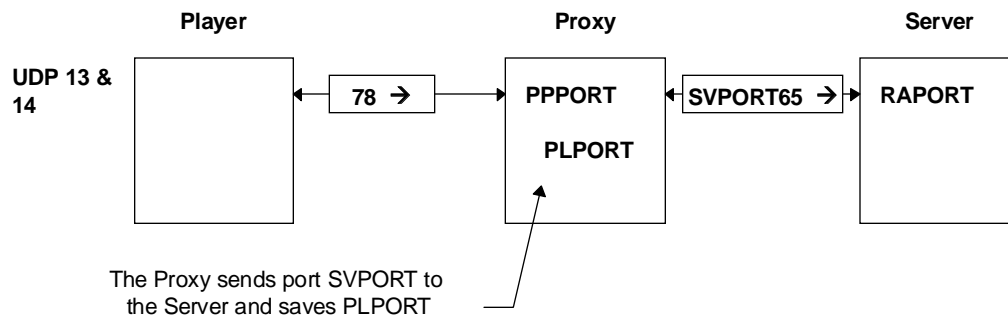


UDP Data Connection

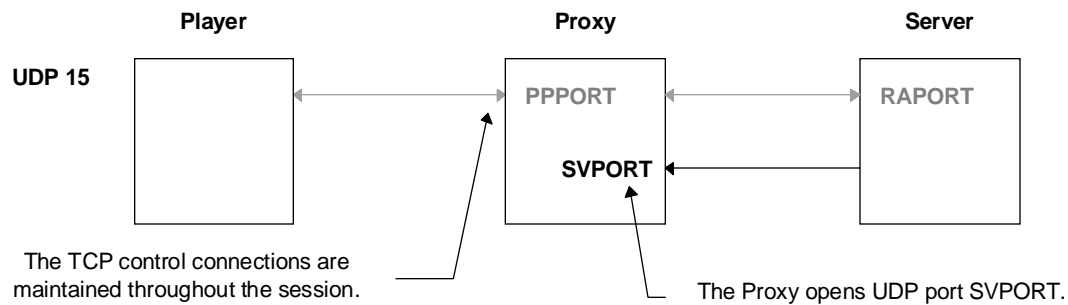
10. The Proxy transfers every byte read from the Player on the TCP control connection to the RealAudio Server on its TCP control connection. The reverse is also true, all data received from the RealAudio Server by the Proxy is also transferred to the Player.
11. After receiving the End Proxy message from the Player, the Proxy begins counting the number of bytes of incoming data from the Player. This will allow the Proxy to locate the requested UDP Port number in the data stream sent between the Player and the RealAudio Server. The port number is located after the number of bytes, indicated in the UDP Offset Message. (i.e. if the offset is 4 bytes then the Port number is located in the 5th and 6th byte.). The Port number is always two bytes long.
12. When the port number offset value collected in step 6d is reached in the Proxy. The Proxy extracts the Player Port number from the data stream. This Port is used by the Player to receive the incoming Audio Data from the Server. This port is PLPORT



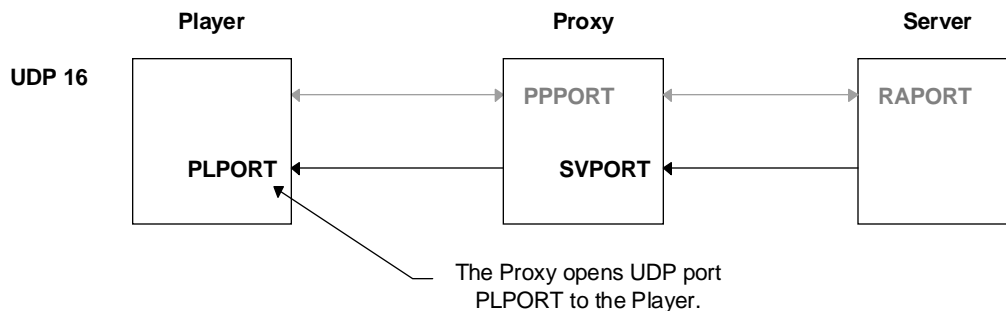
13. The Proxy then allocates a fresh port number to be used in making a connection between the Proxy and the RealAudio Server. This is SVPORT.
14. The Port number (PLPORT) located in step 12 is substituted in the data stream by the new port number (SVPORT) and the data stream is then passed onto the RealAudio Server.



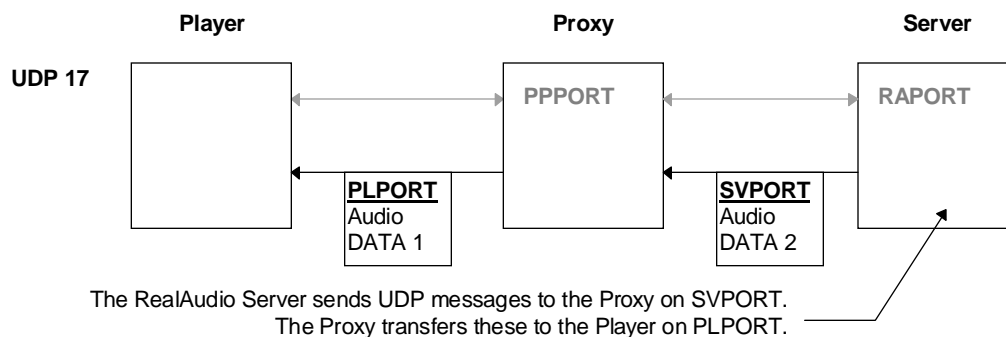
15. The Proxy opens UDP port SVPORT to the RealAudio Server.



16. The Proxy opens a UDP port (PLPORT) to the Player. This completes the data connection between the Player and the RealAudio Server via the Proxy.



17. All data received by the Proxy on UDP port SVPORT from the Server is forwarded to the Player on UDP port PLPORT.



18. When any of the connections are closed all other open connections in this Server / Proxy / Player interaction should also be closed. This will terminate the Player / Proxy interaction and also the Server / Proxy interaction.

Message Definitions for Proxy Server Interaction

Messages are encoded in a standard format. This allows unknown messages to be skipped over and ignored. The standard format is as follows

Table 1 - RealAudio Proxy Protocol Message Format

← Data Flow	2 Bytes	1 Byte	n Bytes
	Length of opcode & data portion of message	Opcode, defines message type	Optional Data

All numeric fields are in Network byte order

Table 2 - RealAudio Proxy Protocol, Version 1

Opcode	Op Name	Data Length Bytes	Data Contents	Purpose
0	end	0 (implied)		Signals end of Player / Proxy communication.
1	version	1	1	Specifies the version of the Proxy protocol
2	hostname	string length	host name as ascii string	Specifies the RealAudio server hostname. This can be an ascii IP address
3	port	2	port number	Specifies the port number for the RealAudio Server.
4	port_off	2	port offset	Specifies the number of bytes to the location of the Port number in the incoming data stream once Player / Proxy communications have completed.
5	use_tcp	1	0 or 1	Specifies whether the data connection will be TCP or UDP.
6	status	1+string length	error code + ascii string	Provides a status on operations. This includes a 1 byte code plus a optional ascii string.

The following Table lists the current set of status values defined in the Protocol. If a player receives a status that is not included in this table it will display the supplied string and terminate the Proxy interaction

Table 3 - RealAudio Proxy Protocol Status codes

Status Number	Status Description
0	Success
1	No Connect, could not connect to RealAudio Server
2	Bad Info, Bad information was passed in the Proxy dialog
3	Bad Version, the RealAudio Proxy Protocol version number does not match
4	Bad Opcode, a message with an invalid opcode was received

The following table details an example of each message. All numeric values are encoded in network byte order as integers All strings are in ascii and are not nul ('\0') terminated. The middle three columns of the

table can be taken as an example message layout Where the data is listed as None this signifies that no data is sent with this message.

Table 4 - Sample RealAudio Proxy Protocol Messages

Op Name	Length 2 Bytes	Opcode 1 Bytes	Data	Notes
end	1	0	<i>None</i>	
version	2	1	1	network byte order
hostname	18	2	www.realaudio.com	ascii
port	3	3	7070	network byte order
port_off	3	4	15	network byte order
use_tcp	2	5	0	network byte order
status	36	6	2Permission Denied, Invalid Username	1 byte number followed by ascii string

Pseudo Code of Player Proxy Interactions

The following section uses a C Style pseudo code to describe the operations in both the Player and the Proxy during a Firewall interaction. All descriptions of messages refer to structured RealAudio Proxy Protocol messages. These messages are defined in Table 2 - RealAudio Proxy Protocol, Version 1

Player Pseudo Code	Proxy Pseudo Code
<pre> Player-side: { player_side_dialog; if success normal server_player stuff else error processing } player_side_dialog: { send version message send hostname message send port message send port_off message send use_tcp message send end message await message if (message != version or an unknown version) return bad_proxy; await message switch (message) { </pre>	<pre> Proxy-side: { proxy_side_dialog; if success { if (use_tcp == 0 { open a udp port while (off < port_off) { copy every byte from the player to the server } read two bytes. send the port number of the socket we just opened. } do { read from player => send to the server read byte server => send to the player if (use_tcp == 0) { read audio packet from server => send to player } } while (all tcp connections are open); close all tcp connections; } else close the connection; } proxy-side-dialog: { send version message; await message; if (message != version or unknown version) { send status = bad first message or unknown version return bad_player; } done = 0; do { await message; switch (message) { </pre>

<pre> case status: if (status == error) return status; done = 1; default: return invalid message } while (!done); send end return success; } </pre>	<pre> case hostname: stash hostname; case port: stash port; case port_off: stash port_off; case use_tcp: stash use_tcp; case end: done = 1; } if (all required information received) done = 1; } while (!done); connect to the server's host:port if (connection OK) send status = success; else send status = failure; return success; } </pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------