

# **Novell BorderManager: *A Beginner's Guide to Configuring Filter Exceptions***

**Third Edition, Revision 2**  
November 27, 2002

**Craig Johnson**  
Novell Support Connection SysOp  
Craig Johnson Consulting  
<http://nscsysop.hypermart.net/>

# Table of Contents

---

<b>Table of Contents .....</b>	<b>2</b>
<b>Table of Figures.....</b>	<b>8</b>
<b>Acknowledgements .....</b>	<b>12</b>
<b>About the Author .....</b>	<b>13</b>
<b>Licensing .....</b>	<b>14</b>
<b>Official Disclaimer.....</b>	<b>15</b>
<b>What This Book is About.....</b>	<b>16</b>
<b>What's New .....</b>	<b>18</b>
Please Read This! .....	18
What's Different About BorderManager 3.7? .....	19
What's New in the Third Edition, Revision 2 .....	20
What's New in the Third Edition .....	20
What's New in the Second Edition .....	21
<b>Printing This Book .....</b>	<b>22</b>
<b>Chapter 1 - The Network Configuration .....</b>	<b>23</b>
<b>Chapter 2 - The Basics.....</b>	<b>25</b>
How Packet Filtering Works .....	25
Stateful Filter Exceptions.....	25
ACK Bit Filters.....	26
Filters and the Relationship to NAT and Routing .....	27
What Are Port Numbers? .....	28
How Routing Works .....	30
Setting up the Default Route .....	32
Public and Private IP Address Networks.....	36
Secondary IP Addresses .....	38
NAT (Routing) versus Proxy .....	40
Dynamic NAT - for Outbound Traffic .....	42
NAT Implicit Filtering .....	43
Disabling NAT Implicit Filtering in INETCFG.....	43
Disabling NAT Implicit Filtering at the Server Console Prompt.....	44
Security Implications for Disabling NAT Implicit Filtering .....	44
Static NAT - For Inbound Traffic.....	45
Static NAT and Filtering .....	46
Setting up Static NAT .....	47
Static NAT versus Reverse Proxy Acceleration .....	50
Viewing & Capturing TCP/IP Traffic .....	51
Static NAT Example Debug Trace.....	52
Default BorderManager Filters .....	53
What are Default Filters? .....	53
The BorderManager 3.x Default Filters.....	53
The Default Filtering Action – All Versions of BorderManager .....	55
FILTCFG Examples of the Default Filters.....	55

Default Filter Exceptions.....	58
BorderManager 3.0, 3.5 and 3.6 Exceptions .....	58
FILTCFG Examples of the BorderManager 3.0, 3.5 and 3.6 Default Filter Exceptions.....	59
BorderManager 3.7 Default Filter Exceptions.....	69
Filter Exceptions for Proxies You Might Have in BorderManager 3.7 .....	70
Security Considerations.....	71
<b>Chapter 3 - NetWare Tools Used in Filtering .....</b>	<b>73</b>
iManager .....	73
FILTSRV.NLM.....	73
BRDCFG.NLM.....	73
CONFIG (Not CONFIG.NLM).....	74
CONLOG.NLM .....	74
FILTCFG.NLM.....	74
IPFLT.NLM / IPFLT31.NLM.....	76
SET TCP IP DEBUG=1 .....	76
SET FILTER DEBUG=ON .....	77
TCPCON.NLM .....	77
VPNCFG.NLM .....	77
<b>Chapter 4 - Working with Filters.....</b>	<b>78</b>
Backing Up and Restoring Filters and Exceptions Prior to BorderManager 3.7 .....	78
Backing Up and Restoring BorderManager 3.7 IP Filters and Exceptions .....	78
The FILTSRV MIGRATE Procedure .....	80
Viewing Filters in Action (TCP IP DEBUG).....	81
TCP DEBUG PING & DNS Example.....	82
Browsing Example – No Proxy Configured.....	84
Browsing Example – Proxy Configured, Default Filter Exceptions.....	86
Filter Debug - An Alternative to TCP IP DEBUG .....	88
Filter Debug Example Output .....	90
NCF Files To Use With SET FILTER DEBUG=ON .....	91
T1.NCF (Turn On Debugging and Capture the Results).....	91
T0.NCF (Turn Off Debugging and Display the Results) .....	91
Making a Custom Filter Exception in FILTCFG.NLM.....	92
Part 1, Starting To Make A Filter Exception.....	92
Part 2, Defining a New Filter Definition.....	99
Part 3, Finishing the Filter Exception .....	107
Making a Custom Filter Exception in iManager .....	110
Start iManager .....	111
Adding a New Service Type .....	117
Adding a Filter Exception with the New Service Type Definition.....	122
ConsoleOne View of BorderManager 3.7 Filters .....	134
<b>Chapter 5 Example Outbound Filter Exceptions.....</b>	<b>136</b>
AIM (AOL Instant Messenger) / AOL.....	137
Cisco VPN Client .....	138
Citrix WinFrame / MetaFrame .....	140
Client-to-Site VPN over NAT .....	142
CLNTRUST.....	145
DNS .....	147
Outbound DNS Filter Exceptions for Internal Hosts.....	147
Outbound DNS Filter Exceptions for BorderManager 3.7 DNS Proxy (and Other Proxies).....	149
FTP .....	151
Outbound FTP Filter Exception for Internal Hosts .....	152

Outbound FTP Filter Exception for BorderManager 3.7 FTP Proxy.....	153
GroupWise Remote Client.....	154
ICQ (Version 2000b) .....	155
HTTP .....	157
Outbound HTTP Filter Exception for Internal Hosts .....	157
Outbound HTTP Filter Exception for BorderManager 3.7 HTTP and Transparent HTTP Proxy....	158
IMAP .....	159
Microsoft MSN Messenger.....	160
Microsoft Windows Media Player.....	161
NNTP.....	163
Outbound NNTP Filter Exception for Internal Hosts .....	163
Outbound NNTP Filter Exception for BorderManager 3.7 News Proxy .....	164
NTP/SNTP.....	165
pcANYWHERE .....	168
PING (ICMP).....	171
POP3.....	172
Outbound POP3 Filter Exception for Internal Hosts .....	172
Outbound POP3 Filter Exception for BorderManager 3.7 Mail Proxy.....	173
RDATE .....	174
RealAudio.....	176
Outbound RealAudio Filter Exception for Internal Hosts.....	176
Outbound RealAudio Filter Exception for BorderManager 3.7 RealAudio/RTSP Proxy .....	179
RTSP (Real Time Streaming Protocol) .....	180
Outbound RTSP Filter Exception for Internal Hosts .....	180
Outbound RTSP Filter Exception for BorderManager 3.7 RealAudio/RTSP Proxy .....	181
SMTP .....	182
Outbound SMTP Filter Exception for Internal Hosts .....	182
Outbound SMTP Filter Exception for BorderManager 3.7 Mail Proxy.....	183
SSL (HTTPS).....	184
Outbound SSL Filter Exception for Internal Hosts .....	184
Outbound SSL Filter Exception for BorderManager 3.7 HTTP Proxy.....	185
TELNET.....	186
Outbound TELNET Filter Exception for Internal Hosts.....	186
Outbound TELNET Filter Exception for BorderManager 3.7 Transparent Telnet Proxy.....	187
Terminal Server.....	188
VNC Viewer .....	189
VNC Browser Interface.....	190
<b>Chapter 6 - Example Inbound Filter Exceptions .....</b>	<b>191</b>
DHCP to a PC on the Public Subnet .....	192
DHCP to the BorderManager Server .....	195
Novell Remote Manager (NRM) on Generic TCP Proxy (on Secondary IP Address).....	197
Reverse HTTP Proxy (on Secondary IP Address).....	199
SSL to Reverse HTTP Proxy (on Secondary IP Address).....	201
FTP to Reverse FTP Proxy .....	203
SMTP to Mail Proxy or GWIA .....	206
POP3 to Mail Proxy.....	208
NNTP to News Proxy .....	210
RCONJ to Generic Proxy (on Secondary IP Address) .....	212
BorderManager 3.7 VPN.....	214
BorderManager 3.7 Default VPN Filter Exceptions.....	214
Client-to-Site Filter Exceptions .....	215
Client-to-Site VPN for Clients Not Behind NAT .....	215
Client-to-Site VPN for Clients Behind NAT .....	216
Site-to-Site VPN Exceptions.....	219

<b>Chapter 7 - Example Inbound Filter Exceptions Using Static NAT .....</b>	<b>224</b>
Citrix WinFrame .....	225
FTP .....	229
GroupWise Remote Client.....	232
GroupWise Web Access Spell Check .....	234
IMAP .....	236
Lotus Notes Clients .....	238
Microsoft Terminal Server .....	240
pcANYWHERE .....	242
Locating Internal pcANYWHERE Host with UDP port 5632 .....	243
Data Transfer Between pcANYWHERE Hosts using TCP port 5631 .....	245
Alternative - Locating Internal pcANYWHERE Host with UDP port 22 .....	247
POP3 .....	249
SMTP .....	251
VNC .....	255
Web Servers.....	257
HTTP to Internal Web Server.....	257
HTTPS/SSL to Internal Web Server .....	259
<b>Chapter 8 - BorderManager 2.1 – Stateful Filters Alternative.....</b>	<b>261</b>
Generic Exception for TCP Return Traffic .....	263
Generic Exception for UDP Return Traffic .....	264
<b>Chapter 9 - Advanced Topics.....</b>	<b>265</b>
Basic Improvement - Enhance the Security of the Default Exceptions .....	265
Creating a Custom Dyn/ACK/TCP Filter Exception.....	267
Customizing the Default Dynamic/TCP Default Filter Exception .....	268
More Security - A DMZ Scenario .....	270
Step 1 – Set Filters on the DMZ NIC .....	272
Step 2 – Open Filter Exceptions for Inbound Traffic from the Internet to the DMZ.....	273
Step 3 – Open Filter Exceptions for Outbound Traffic from the Internal LAN to the DMZ .....	274
Most Security - Completely Customized Filter Exceptions.....	277
Allow Outbound HTTP for the HTTP Proxy Only .....	278
Allow Outbound HTTPS / SSL for the HTTP Proxy Only .....	278
Allow Non-Standard Ports Outbound for the HTTP Proxy Only .....	279
Blocking Chat Programs .....	280
Blocking AOL Instant Messenger (as of 02/20/2002) .....	281
Blocking MSN Messenger (as of 11/29/2001) .....	281
Blocking ICQ (as of 2/20/2002) .....	282
Blocking Yahoo Messenger (as of 11/29/2001).....	282
Adding Dummy Static Routes.....	283
Entering a static route in NetWare.....	284
<b>Chapter 10 - Troubleshooting.....</b>	<b>285</b>
Is It A Filtering Problem? .....	285
Stateful Filter Exceptions Aren't Working.....	286
My Filter Exception Looks OK, But My Traffic Is Still Blocked .....	287
My Traffic is Blocked, But TCP/IP DEBUG Doesn't Show Any Discards.....	288
None of My Traffic is Blocked – Filters Are Not Working.....	288
My Filtering Doesn't Seem to Be Working Like this Book.....	289
BorderManager 3.7 Filtering Issues.....	290
No Filters or Exceptions Show Up in FILTCFG for BorderManager 3.7 .....	291
There is No NBMRuleContainer Object in NDS.....	291
Cannot Create the NBMRuleContainer Object in NDS.....	291

FILTSRV MIGRATE Process Gives –603 Errors .....	292
FILTSRV MIGRATE Process Gives –608 Errors .....	292
FILTSRV MIGRATE Process Gives –659 Errors .....	292
FILTSRV MIGRATE Completes With No Errors, FILTCFG Sees Nothing.....	292
Some Duplicate Filter Exceptions Exist in FILTCFG .....	292
I Have a Filter Exception with No Definition in FILTCFG .....	293
Filter Changes Not Showing Up At the BorderManager 3.7 Server .....	293
FILTSRV Returns a –6001 Error When Editing Filters in FILTCFG .....	293
FILTCFG Sees Filter Exceptions With Blank Packet Type .....	293
Deleted Filter Exceptions Keep Reappearing in FILTCFG.....	294
Server Console Shows “NWDSPutAttrVal returned with error –5” .....	294
Every Time My Server Starts, I See a FILTSRV Error .....	294
More Objects Appear in NBMRuleContainer Than I Have Filters or Exceptions .....	295
ACK Bit Filtering Not Working .....	295
My 3.7 Server Is Driving Me Nuts with Custom Filter Exceptions! How Can I Make It Like BorderManager 3.6? .....	296
Allow All Outbound IP from Public IP Address.....	296
Allow High Port Return Traffic to Public IP Address .....	296
Some BorderManager 3.7 Proxies Work, Some Don’t .....	297
No Option in iManager for NBM Access Management for BorderManager 3.7.....	297
Adding NBM Access Management Option to a Non-BorderManager Server .....	297
The BorderManager 3.7 HTTP Proxy Doesn’t Work for Some Web Sites .....	299
Server ABENDS if I Unload IPFLT Manually .....	300
Clustering Doesn’t Work With Filtering Enabled .....	300
Easy Fix – SET Statement to Change Filtering Behavior .....	300
Complex Fix – Change Clustering Address .....	301
APC PowerChute Software Doesn’t Work With Filtering .....	301
NAT Quit Working.....	301
BAD TCPIP.CFG FILE EXAMPLE.....	302
Fixing the Problem.....	304
NAT Works, but Intermittently, and Communications are Inconsistent or Strange.....	305
Some, or All of My Traffic Is Blocked, Even Proxies .....	305
BorderManager 3.7 .....	305
Any BorderManager Version .....	305
The Application Keeps Changing Port Numbers.....	306
Stateful Filters or TCP/IP Communications Work, But Quit Working or Are Inconsistent .....	306
My Port Numbers Are Really Weird! .....	307
FTP-PORT-PASV-ST Stateful Filter Doesn't Work in BorderManager 3.5.....	308
POP3-ST Stateful Filter Doesn't Work in BorderManager 3.5 .....	308
All IP Traffic Quits Working After Some Time.....	308
My Application Works For Me, But Not For My Friend Outside The Firewall.....	309
I Can't Filter Traffic That Brings Up My Dial-Up Connection! .....	309
Can't PING the Private IP Address of the BorderManager Server over Client-Site VPN .....	310
No VPTUNNEL Interface in INETCFG for Site-to-Site VPN Slave Server .....	311
<b>Chapter 11 - Odds &amp; Ends.....</b>	<b>312</b>
Other Useful Port Numbers .....	312
LDAP .....	312
SQL.....	312
SSH.....	312
SOCKS .....	312
NetWare NCP Over IP .....	312
NDPS .....	312
SNMP .....	313
SCMD .....	313

SLP .....	313
IPP .....	313
WEBMIN.....	313
Renaming Your Interfaces to Public and Private.....	314
Fixing the BorderManager 3.5 POP3-ST Definition.....	315
Novell's FILT01A.EXE File .....	316
Packet Filter Logging.....	317
NetWare for Small Business (SBS) .....	317
<b>Chapter 12 - Other References .....</b>	<b>318</b>
<b>Index .....</b>	<b>319</b>

# Table of Figures

---

Figure 1-1 - Network Addressing Scenario .....	23
Figure 2-1 - INETCFG, Protocols, TCP/IP .....	32
Figure 2-2 - INETCFG, Protocols, TCP/IP, LAN Static Route, <insert> .....	33
Figure 2-3 - INETCFG - Enter Next Hop for Default Route .....	34
Figure 2-4 - INETCFG - Reinitialize System Option .....	35
Figure 2-5 - INETCFG, Bindings, <public IP address>, Expert TCP/IP Bind Options, Network Address Translation .....	42
Figure 2-6 - INETCFG - Option to Disable NAT Implicit Filtering .....	43
Figure 2-7 - INETCFG - Network Address Translation .....	47
Figure 2-8 - INETCFG - Select Static and Dynamic NAT .....	48
Figure 2-9 - INETCFG - Entering Static NAT Mappings .....	49
Figure 2-10 - FILTCFG - Deny Packets in Filter List .....	55
Figure 2-11 - FILTCFG - Default Filter Blocking all IP Traffic to the Public Interface .....	56
Figure 2-12 - FILTCFG - Default Filter Blocking all IP Traffic from the Public Interface .....	57
Figure 2-13 - FILTCFG - Default Filter Exception Allowing all Outbound IP Traffic from the Public IP Address .....	60
Figure 2-14 - FILTCFG - Default Filter Exception Allowing Dynamic TCP to the Public IP Address .....	61
Figure 2-15 - FILTCFG - Default Filter Exception Allowing Dynamic UDP to the Public IP Address .....	62
Figure 2-16 - FILTCFG - Default Filter Exception Allowing VPN Master/Slave Traffic to the Public IP Address .....	63
Figure 2-17 - FILTCFG - Default Filter Exception Allowing VPN Client Authentication to the Public IP Address .....	64
Figure 2-18 - FILTCFG - Default Filter Exception Allowing VPN Client Keep-Alive Traffic to the Public IP Address .....	65
Figure 2-19 - FILTCFG - Default Filter Exception Allowing SKIP Protocol to the Public IP Address .....	66
Figure 2-20 - FILTCFG - Default Filter Exception Allowing Reverse Proxy HTTP Traffic to the Public IP Address .....	67
Figure 2-21 - FILTCFG - Default Filter Exception Allowing HTTPS (SSL) Traffic to the Public IP Address .....	68
Figure 3-1 - FILTCFG - Configure Interface Options .....	76
Figure 4-1 - Netscape Configured without Proxy settings .....	84
Figure 4-2 - Netscape Configured to Use HTTP Proxy .....	86
Figure 4-3 - SET FILTER DEBUG=ON .....	89
Figure 4-4 - FILTER DEBUG Capture Example .....	90
Figure 4-5 - FILTCFG - Main Menu .....	92
Figure 4-6 - FILTCFG - Select Packet Forwarding Filters .....	93
Figure 4-7 - FILTCFG - Select List of Packets Always Permitted .....	93
Figure 4-8 - FILTCFG - Filter Exception Menu .....	94
Figure 4-9 - FILTCFG - Select Source Interface .....	95
Figure 4-10 - FILTCFG - Select Destination Interface .....	96
Figure 4-11 - FILTCFG - Define Exception Packet Type .....	97
Figure 4-12 - FILTCFG - Create a New Packet Type .....	98
Figure 4-13 - FILTCFG - Enter Packet Type Name .....	99
Figure 4-14 - FILTCFG - Enter Packet Type Protocol .....	100
Figure 4-15 - FILTCFG - Select Protocol .....	101
Figure 4-16 - FILTCFG - Enter Source Port .....	102
Figure 4-17 - FILTCFG - Enter Destination Port .....	103
Figure 4-18 - FILTCFG - Specify Stateful Filtering .....	104
Figure 4-19 - FILTCFG - Comment the New Definition .....	105



Figure 4-20 - FILTCFG - Updated Packet Type List.....	106
Figure 4-21 - FILTCFG - Add Comment for New Exception.....	107
Figure 4-22 - FILTCFG - Save New Filter Option .....	108
Figure 4-23 - FILTCFG - New Filter Active in List of Packet Filter Exceptions.....	109
Figure 4-24 - NetWare Web Manager Main Menu .....	111
Figure 4-25 - iManager Login Screen.....	112
Figure 4-26 - iManager Main Menu, with NBM Access Management Option.....	113
Figure 4-27 - iManager Browse Screen for BorderManager Server Object.....	114
Figure 4-28 - BorderManager Server Selection in FilterConfiguration Menu .....	115
Figure 4-29 - BorderManager Filter Configuration Main Menu.....	116
Figure 4-30 - List of Configured Service Types (Filter Definitions).....	118
Figure 4-31 - Service Type Definition Menu.....	119
Figure 4-32 - Add Service Type Request Succeeded Menu .....	121
Figure 4-33 - New Service Type Added to List.....	121
Figure 4-34 - Select Packet Forwarding Filter.....	122
Figure 4-35 - Select Exception List.....	123
Figure 4-36 - List of Existing Filter Exceptions.....	124
Figure 4-37 - Configure Filter Exception - Add NDS Name .....	125
Figure 4-38 - Configure Filter Exception - Choose Custom Service Type Defined Earlier from List of Available Service Types.....	126
Figure 4-39 - Configure Filter Exception - Add Descriptive Comment .....	127
Figure 4-40 - Configure Filter Exception - Choose Source Type (Interface and/or Address) .....	128
Figure 4-41 - Configure Filter Exception - Select Source Interface .....	129
Figure 4-42 - Configure Filter Exception - Choose Destination Type (Interface and/or Address).....	130
Figure 4-43 - Configure Filter Exception - Select Public Interface as Destination.....	131
Figure 4-44 - Configure Filter Exception - Add Filter Request Succeeded Menu .....	132
Figure 4-45 - Configure Filter Exception - New Exception Appears in List.....	133
Figure 4-46 - ConsoleOne View of Filter Objects in NBMRuleContainer Object .....	134
Figure 4-47 - ConsoleOne View of BorderManager 3.7 Filter Object Attributes.....	135
Figure 5-1 - Filter Exception for Outbound AOL / AOL Instant Messenger / ICQ.....	137
Figure 5-2 - Filter Exception for Cisco VPN Client Connection, Part 1 of 2 .....	138
Figure 5-3 - Filter Exception for Cisco VPN Client Connection, Part 2 of 2 .....	139
Figure 5-4 - Filter Exception for Outbound Citrix ICA Client.....	140
Figure 5-5 - Filter Exception for Outbound Citrix Browser Client.....	141
Figure 5-6 - Filter Exception for Initial BorderManager Client-to-Site VPN Authentication over NAT...	142
Figure 5-7 - Filter Exception for Outbound BorderManager Client-Site VPN over NAT.....	143
Figure 5-8 - Filter Exception for BorderManager Client-to-Site VPN KeepAlive Packets over Dynamic NAT.....	144
Figure 5-9 - Filter Exception for Internal CLNTRUST Traffic to Public IP Address .....	145
Figure 5-10- Filter Exception for Outbound DNS Queries over UDP with Source Ports Specified.....	147
Figure 5-11 - Filter Exception for Outbound DNS Queries over TCP.....	148
Figure 5-12 - Filter Exception for DNS over TCP from BorderManager 3.7 Proxies .....	149
Figure 5-13 - Filter Exception for DNS over UDP from BorderManager 3.7 Proxies.....	150
Figure 5-14 - Filter Exception for Outbound FTP.....	152
Figure 5-15 - Filter Exception for Outbound FTP from BorderManager 3.7 FTP Proxy .....	153
Figure 5-16 - Filter Exception for Outbound GroupWise Remote Client .....	154
Figure 5-17 - ICQ 2000b Settings for AOL Port Number .....	155
Figure 5-18 - Filter Exception for Outbound ICQ 2000b .....	156
Figure 5-19 - Filter Exception for Outbound HTTP .....	157
Figure 5-20 - Filter Exception for Outbound HTTP from BorderManager 3.7 HTTP Proxy.....	158
Figure 5-21 - Filter Exception for Outbound IMAP.....	159
Figure 5-22 - Filter Exception for Outbound MSN Messenger.....	160
Figure 5-23 - Windows Media Player MMS Protocol Settings .....	161
Figure 5-24 - Filter Exception for Outbound Windows Media Player MMS Protocol .....	162
Figure 5-25- Filter Exception for Outbound NNTP.....	163

Figure 5-26 - Filter Exception for NNTP from BorderManager 3.7 News Proxy.....	164
Figure 5-27 - Filter Exception for Outbound NTP, Source Port=123 .....	165
Figure 5-28 - Filter Exception for Outbound NTP, Source Ports=1024-65535 .....	166
Figure 5-29 - Filter Exception for Outbound pcANYWHERE Location Protocol (Old).....	168
Figure 5-30 - Filter Exception for Outbound pcANYWHERE Location Protocol.....	169
Figure 5-31 - Filter Exception for Outbound pcANYWHERE Data.....	170
Figure 5-32 - Filter Exception for Outbound ICMP (PING & TRACERT).....	171
Figure 5-33 - Filter Exception for Outbound POP3 .....	172
Figure 5-34 - Filter Exception for POP3 from BorderManager 3.7 Mail Proxy.....	173
Figure 5-35 - Filter Exception for Outbound RDATE Time Protocol.....	174
Figure 5-36 - RealPlayer G2 Settings to Bypass PNA & RTSP Proxy.....	176
Figure 5-37 - Filter Exception for Outbound RealAudio (PNA) .....	178
Figure 5-38 - Filter Exception for RealAudio from BorderManager 3.7 RealAudio/RTSP Proxy.....	179
Figure 5-39 - Filter Exception for Outbound RTSP .....	180
Figure 5-40 - Filter Exception for RTSP from BorderManager RealAudio/RTSP Proxy .....	181
Figure 5-41 - Filter Exception for Outbound SMTP .....	182
Figure 5-42 - Filter Exception for SMTP from BorderManager 3.7 Mail Proxy.....	183
Figure 5-43 - Filter Exception for Outbound SSL / HTTPS.....	184
Figure 5-44 - Filter Exception for SSL (HTTPS) from BorderManager 3.7 HTTP Proxy.....	185
Figure 5-45 - Filter Exception for Outbound TELNET.....	186
Figure 5-46 - Filter Exception for TELNET from BorderManager 3.7 Transparent Telnet Proxy.....	187
Figure 5-47 - Filter Exception for Outbound Microsoft Terminal Server.....	188
Figure 5-48 - Filter Exception for Outbound VNC Viewer for 10 Console Sessions .....	189
Figure 5-49 - Filter Exception for Outbound VNC through a Web Browser for 10 Console Sessions .....	190
Figure 6-1 - Filter Exception for Initial DHCP Client Request to Broadcast Address on Public Interface.....	192
Figure 6-2 - Filter Exception for DHCP Client Responses from Public IP Address.....	193
Figure 6-3 - Filter Exception for Inbound DHCP Renewal Requests .....	194
Figure 6-4 - Filter Exception for Public Interface to get DHCP Address.....	196
Figure 6-5 - Filter Exception for Inbound NRM to Generic TCP Proxy on Secondary IP Address.....	197
Figure 6-6 - Filter Exception for Portal Responses from Generic TCP Proxy on Secondary Public IP Address.....	198
Figure 6-7 - Filter Exception for HTTP to Reverse HTTP Proxy on Secondary Public IP Address.....	199
Figure 6-8 - Filter Exception for Reverse HTTP Proxy Responses from Reverse HTTP Proxy on Secondary Public IP Address.....	200
Figure 6-9 - Filter Exception for Inbound HTTPS/SSL to Reverse HTTP Proxy on Secondary Public IP Address.....	201
Figure 6-10 - Filter Exception for Outbound HTTPS / SSL Responses from Reverse HTTP Proxy on Secondary Public IP Address .....	202
Figure 6-11 - Filter Exception for Inbound FTP Control and Data Ports to Reverse FTP Proxy.....	203
Figure 6-12 - Filter Exception for Outbound FTP Control Port Responses from Reverse FTP Proxy.....	204
Figure 6-13 - Filter Exception for Outbound FTP Data Port Responses from FTP Reverse Proxy.....	205
Figure 6-14 - Filter Exception for SMTP to Mail Proxy or GWIA.....	206
Figure 6-15 - Filter Exception for SMTP Responses from Mail Proxy or GWIA.....	207
Figure 6-16 - Filter Exception for POP3 to Mail Proxy or GWIA.....	208
Figure 6-17 - Filter Exception for POP3 Responses from Mail Proxy or GWIA.....	209
Figure 6-18 - Filter Exception for NNTP to News Proxy.....	210
Figure 6-19 - Filter Exception for NNTP Responses from News Proxy.....	211
Figure 6-20 - Filter Exception for Inbound RCONJ to Generic TCP Proxy on Secondary Public IP Address .....	212
Figure 6-21 - Filter Exception for Outbound Responses from RCONJ on Generic TCP Proxy .....	213
Figure 6-22 - Filter Exception for Client-to-Site VPN Authentication Responses .....	215
Figure 6-23 - Filter Exception for Inbound Client-to-Site VPN over NAT to BorderManager Server .....	217
Figure 6-24 - Filter Exception for Outbound Client-to-Site VPN over NAT Responses from BorderManager Server .....	218
Figure 6-25 - Filter Exception for Outbound SKIP Protocol.....	219

<i>Figure 6-26 - Filter Exception for Inbound Site-to-Site VPN Communications.....</i>	<i>220</i>
<i>Figure 6-27 - Filter Exception for Site-to-Site VPN Responses .....</i>	<i>221</i>
<i>Figure 6-28 - Filter Exception for Slave/Slave VPN Server Site-to-Site Communications Inbound .....</i>	<i>222</i>
<i>Figure 6-29 - Filter Exception for Slave/Slave VPN Server Site-to-Site Communications Outbound .....</i>	<i>223</i>
<i>Figure 7-1 - Filter Exception for Inbound Citrix ICA Client .....</i>	<i>225</i>
<i>Figure 7-2 - Filter Exception for Outbound Citrix ICA Client Responses .....</i>	<i>226</i>
<i>Figure 7-3 - Filter Exception for Inbound Citrix Browser-based Client.....</i>	<i>227</i>
<i>Figure 7-4 - Filter Exception for Outbound Citrix Browser-based Client Responses.....</i>	<i>228</i>
<i>Figure 7-5 - Filter Exception for Inbound FTP Control and Data Ports.....</i>	<i>229</i>
<i>Figure 7-6 - Filter Exception for Outbound FTP Control Port Responses .....</i>	<i>230</i>
<i>Figure 7-7 - Filter Exception for Outbound FTP Data Port Responses.....</i>	<i>231</i>
<i>Figure 7-8 - Filter Exception for Inbound GroupWise Remote Client .....</i>	<i>232</i>
<i>Figure 7-9 - Filter Exception for Outbound GroupWise Remote Client Responses.....</i>	<i>233</i>
<i>Figure 7-10 - Filter Exception for Inbound Collexion Spell Check Requests .....</i>	<i>234</i>
<i>Figure 7-11 - Filter Exception for Outbound Collexion Spell Check Responses .....</i>	<i>235</i>
<i>Figure 7-12 - Filter Exception for Inbound IMAP .....</i>	<i>236</i>
<i>Figure 7-13 - Filter Exception for Outbound IMAP Responses .....</i>	<i>237</i>
<i>Figure 7-14 - Filter Exception for Inbound Lotus Notes Client.....</i>	<i>238</i>
<i>Figure 7-15 - Filter Exception for Outbound Lotus Notes Client Responses .....</i>	<i>239</i>
<i>Figure 7-16 - Filter Exception for Inbound Microsoft Terminal Server .....</i>	<i>240</i>
<i>Figure 7-17 - Filter Exception for Outbound Terminal Server Responses.....</i>	<i>241</i>
<i>Figure 7-18 - Filter Exception for Inbound pcANYWHERE Location Protocol .....</i>	<i>243</i>
<i>Figure 7-19 - Filter Exception for Outbound pcANYWHERE Location Responses .....</i>	<i>244</i>
<i>Figure 7-20 - Filter Exception for Inbound pcANYWHERE Data .....</i>	<i>245</i>
<i>Figure 7-21 - Filter Exception for Outbound pcANYWHERE Data Responses .....</i>	<i>246</i>
<i>Figure 7-22 - Filter Exception for Inbound Older pcANYWHERE Location Protocol.....</i>	<i>247</i>
<i>Figure 7-23 - Filter Exception for Outbound Older pcANYWHERE Location Protocol Responses.....</i>	<i>248</i>
<i>Figure 7-24 - Filter Exception for Inbound POP3 Requests to Internal Mail Server .....</i>	<i>249</i>
<i>Figure 7-25 - Filter Exception for Outbound POP3 Responses from Internal Mail Server.....</i>	<i>250</i>
<i>Figure 7-26 - Filter Exception for Inbound SMTP.....</i>	<i>251</i>
<i>Figure 7-27 - Filter Exception for Outbound SMTP Responses.....</i>	<i>252</i>
<i>Figure 7-28 - Filter Exception for Outbound SMTP .....</i>	<i>253</i>
<i>Figure 7-29 - Filter Exception for Inbound SMTP Responses .....</i>	<i>254</i>
<i>Figure 7-30 - Filter Exception for Inbound VNC Console Connections 1-10.....</i>	<i>255</i>
<i>Figure 7-31 - Filter Exception for Outbound VNC Responses.....</i>	<i>256</i>
<i>Figure 7-32 - Filter Exceptions for Inbound HTTP to Web Server.....</i>	<i>257</i>
<i>Figure 7-33 - Filter Exception for Outbound HTTP Responses.....</i>	<i>258</i>
<i>Figure 7-34 - Filter Exception for Inbound HTTPS / SSL.....</i>	<i>259</i>
<i>Figure 7-35 - Filter Exception for Outbound HTTPS Responses.....</i>	<i>260</i>
<i>Figure 8-1 - Generic TCP Filter Exception to Allow All Return Traffic.....</i>	<i>263</i>
<i>Figure 8-2 - Generic UDP Filter Exception to Allow All Return Traffic.....</i>	<i>264</i>
<i>Figure 9-1 - Custom Dyn/ACK/TCP Filter Exception.....</i>	<i>267</i>
<i>Figure 9-2 - DMZ with Three Network Cards, IP Addressing Diagram.....</i>	<i>271</i>
<i>Figure 9-3 - Filters Applied for PUBLIC and DMZ Interfaces.....</i>	<i>272</i>
<i>Figure 9-4 - Filter Exception to Allow Inbound HTTP to DMZ Web Server from the Internet.....</i>	<i>273</i>
<i>Figure 9-5 - Filter Exception to Allow Outbound HTTP Responses from DMZ Web Server to the Internet .....</i>	<i>274</i>
<i>Figure 9-6 - Filter Exception to Allow HTTP to DMZ Web Server from Internal LAN .....</i>	<i>275</i>
<i>Figure 9-7 - Filter Exception to Allow FTP to DMZ Web Server from Internal LAN.....</i>	<i>276</i>
<i>Figure 9-8 - Dummy Static Route to Redirect MSN Messenger .....</i>	<i>284</i>

# Acknowledgements

---

The author would like to acknowledge the following people who have contributed significantly to the creation of this book.

**Caterina Luppi**, who tirelessly proofread many revisions of this book and contributed many suggestions.

**Marcus Williamson**, and the other **Novell Support Connection Sysops** who have contributed suggestions and caught errors in various revisions.

**Shane Rogers, Steven Meier, Mark Smith, Lance Haig, Steven Coutts, Colin Mair, David Coyte, Barbara Myers, Sérgio M. F. Valeriano, Dolf Krikke, Rob Blomjous** and **Mike Sixsmith** who helped proofread various drafts of the book and gave feedback and suggestions.

**Frank Berzau and Gonzalo Morera, and Ross Irvine, Novell Engineers**, who contributed valuable technical advice and corrections to this book.

**Danita Zanrè**, Novell Support Connection Sysop and nationally renowned GroupWise consultant, who helped get this book on the market.

**John Ryan**, whose encouragement convinced me to write a book on the subject of BorderManager.

# About the Author

---

Craig Johnson has been working with computers since he wrote his first program in college at Purdue University in 1971. Currently Craig owns his own consulting business based in Phoenix, Arizona and working on projects around the continent (and beyond). Many of Craig's clients became familiar with him through his forum work or books.

Craig has been a Novell Support Connection Sysop for over four years, and he specializes in (naturally) the BorderManager forums at [forums.novell.com](http://forums.novell.com) (NNTP). Craig has been working with BorderManager since before the official release of BorderManager version 2.1. Through his consulting business and the Novell Support Connection forums, Craig has provided advice on numerous BorderManager installations.

Craig has also presented sessions on BorderManager packet filtering, troubleshooting and VPN seminars, with Caterina Luppi, at Novell's BrainShare seminar in Salt Lake City.

As of this writing (August, 2002), Craig is the only non-Novell employee on Novell's BorderManager Core Development Team.

When not spending 12 hours per day at a computer, Craig likes to work out in Taekwondo, where he holds the rank of Black Belt, third degree and is a certified instructor.

Most days, Craig can be reached via the Novell Support Connection Public Forums, in the BorderManager sections. His web site is <http://nscsysop.hypermart.net>. Craig is available for hire, and does the majority of his BorderManager consulting work over the Internet, with clients all over the world.

# Licensing

---

This book is distributed in Adobe Acrobat PDF format. Why? Because publishing it in printed and bound format would take so long that it would be obsolete before it hit the market, or it would never be published at all due to the small size of the target market! This does not mean that just because you can make copies of the book that you are allowed to. This book is sold with the understanding that each purchaser may make ONE printed copy of the book, and keeps TWO electronic copies (in PDF format). You may not electronically or otherwise reproduce (copy) or make multiple copies of this book. You also may not put a copy of this book on a network server where multiple people can reference it without purchasing it, unless you buy one copy of this book for each BorderManager server you have running.

**This book is being sold online at <http://www.caledonia.net/>.**

Volume purchase agreements are available. Contact the author at [craigsj@ix.netcom.com](mailto:craigsj@ix.netcom.com) for details.

# Official Disclaimer

---

The author and publisher have made their best efforts to prepare this book. The author and the publisher make no representation or warranties of any kind with regard to the completeness or accuracy of the contents herein and accept no liability of any kind including but not limited to performance, merchantability, fitness for any particular purpose, or any losses or damages or any kind caused or alleged to be caused directly or indirectly from this book.

# What This Book is About

---

The purpose of this book is to help readers configure packet filter exceptions in Novell BorderManager 2.1 and 3.x. I wrote this book after spending over three years answering questions on Novell's BorderManager products in the Novell Support Connection forums and setting up numerous BorderManager servers myself. After answering many of the same types of questions day after day, I could see a clear need for a book that explains how packet filters work and how to set up filter exceptions.

I also gained some insight into the level of experience of the typical BorderManager administrator who frequents the Novell Support Connection public forums. Most have some knowledge of TCP/IP, routing, proxies, and filters, but do not have the breadth and depth of knowledge to feel comfortable in dealing with packet filtering. Even those public forum users who were comfortable with packet filtering frequently need a little help in understanding how all the parts fit together, or simply want a quick explanation for a particular filter exception. This book is written to the level of understanding of that 'average' forum user. Despite the title, this book is not limited to just the 'beginner', and it will prove a useful reference to even quite advanced users. I often consult it when answering questions online.

One of the frequent complaints that most public forum users have about documentation on Novell products is that there are not enough examples. I have tried to address that concern in this book by providing many examples. As is true with most people, I find it easier to understand the theory behind a complex networking function when I can see an example. Therefore, I provide explanations of how packet filters operate and examples of working packet filter exceptions. Readers can take the examples provided, in most cases simply substitute their interface names or IP addresses, and have their own custom filter exceptions working in a very short amount of time. In particular, I discuss and provide examples of packet filter exceptions for:

- Outbound traffic for AOL Instant Messenger (AIM), Cisco VPN Client, Client-to-Site Novell VPN Client, Citrix, DNS, FTP, GroupWise Remote Client, ICQ, IMAP, Microsoft MSN Messenger, Microsoft Windows Media Player, NNTP, NTP/SNTP, pcANYWHERE, PING, POP3, RDATE, RealAudio, RTSP SMTP, SSL, TELNET, Terminal Server and VNC.
- Inbound traffic to reverse proxy acceleration of internal web servers on secondary IP addresses, generic TCP proxy for Portal Web Manager and RCONJ, and DHCP for PC's on the public subnet, and for the BorderManager server acting as a DHCP client.
- Inbound traffic through static NAT configurations for Citrix WinFrame, FTP, GroupWise Remote Client, GroupWise Web Access Spell Check, IMAP, Lotus Notes Client, Microsoft Terminal Server, pcANYWHERE, POP3, SMTP, VNC and Web Servers.



Most of the discussion and examples focus on the filtering capabilities provided with BorderManager 3.x (such as stateful filtering), but mention is also made of the limitations of BorderManager 2.1 and how to work around them.

A good source of information on BorderManager in general is the web-based Novell Support Connection Public Forums at <http://support.novell.com/>, or support-forums.novell.com (NNTP). I highly recommend using an NNTP reader to check out the forums.

I have written a book on configuring BorderManager 3.x that covers BorderManager comprehensively. You can buy that book at the same place as this one – <http://www.caledonia.net/>. That book only touches on packet filtering, but covers proxies, gateways, access rules, patches, logging and usage.

BorderManager documentation from Novell is also available at Novell's web site at the following URL:

<http://www.novell.com/documentation>

# What's New

---

## Please Read This!

If you have experience with how filtering works in BorderManager versions prior to version 3.7, I advise you to at least read this section, and any section with '3.7' in the table of contents. There are radical differences in how BorderManager 3.7 treats the default filter exceptions on a fresh install, and unless you understand them, you could be quite frustrated in trying to get the proxies to function.

This book is a modification of the previous edition. I have tried to leave most of the old text intact, except for particular changes related to BorderManager 3.7. The basic concepts of filtering, NAT, and routing are unchanged with any firewall, and BorderManager 3.7 is no different. However, I had to make a choice of how to show filter exceptions in the examples – iManager or FILTCFG screenshots –, and I chose to use FILTCFG, partly because it is so much more efficient in showing all the settings in one screen, and partly because I don't think iManager offers any advantages over using FILTCFG. I did include a section on how to use the iManager interface, and some troubleshooting information on getting the BorderManager 3.7 plugins working with iManager. I have also tried to show the BorderManager default exceptions for the outbound proxy usage within the outbound filter exceptions section. It seemed to me the most logical place to show and discuss them. For example, in the second edition, I showed how to bypass the proxy for HTTP with a stateful filter exception. I have now added another example showing how BorderManager 3.7 uses a stateful filter exception for the HTTP Proxy itself to work, and discussed some particular ramifications of that philosophy.

I think that the changes to how BorderManager 3.7 handles default filter exceptions will confuse a lot of administrators, and I am open to suggestions for how to clarify things for the readers.

## What's Different About BorderManager 3.7?

In terms of filtering, BorderManager 3.7 introduces **significant differences** not only in how the filters can be managed (using a browser), but also a major change in the default filter exceptions. Here is a quick summary:

- BorderManager 3.7 stores the IP filtering information in NDS. The IPX and Appletalk filtering information is still stored in the FILTERS.CFG file.
- A fresh install of BorderManager 3.7 creates only **very specific** exceptions during installation, **for outbound traffic only**. I show these exceptions in the examples for outbound traffic.
- If you upgrade an existing BorderManager 3.x server in-place, the old filter exceptions are not changed at all, and no new filter exceptions are added.
- The first time you run BorderManager 3.7, particularly after an in-place upgrade, you need to perform a migration of your old (IP) filters and filter exceptions to NDS with FILTSRV MIGRATE.
- You can use a browser to manage filters and exceptions, if you have at least one NetWare 6.0 server in your tree by using iManager. You can still use FILTCFG as well.
- Although the initial release of BorderManager 3.7 has a browser-based GUI interface (iManager) for managing IP filtering information, the design is essentially a GUI-ized version of FILTCFG. FILTCFG is both faster and more intuitive to use. I expect a later version of the iManager plugin to appear in a patch or an Enhancement Pack, which should make the GUI more useable.
- Because the IP filters are stored in NDS, there are **significant issues** related to managing filters, especially if you have been used to modifying or copying the FILTERS.CFG file.
- If NDS is unavailable for some reason, **BorderManager 3.7 could boot up without any filters at all**. This behavior is modified with the BM37FLT.EXE patch, which will block all traffic to the public interface until NDS is initialized. You will need to correctly define the public interface within FILTCFG!

## What's New in the Third Edition, Revision 2

Soon after I finished the Third Edition, to get it out of beta status, I realized that I had forgotten to update the Advanced chapter with a simpler procedure to correct the deficiencies of the default Dynamic/TCP filter exception. Also, I wanted to put more tips for troubleshooting BorderManager 3.7 filtering issues into the troubleshooting section and update a couple of other tips in there.

The major changes are all in the Troubleshooting chapter, plus a bit in the Advanced chapter. I have added new tips (almost all for BorderManager 3.7), and modified some of the existing tips. In the Advanced chapter, I added the example to create a Dyn/ACK/TCP custom filter exception.

I also added a new section called The FILTSRV MIGRATE Procedure, and called that out in the Index, because it was too hard to find the old procedure.

## What's New in the Third Edition

The main purpose of producing a Third Edition was to add new text to cover the differences between BorderManager 3.7 and all previous versions in how filtering is configured and managed.

- Because BorderManager 3.7 does not have default exceptions allowing high ports inbound, I also added several new exceptions for inbound traffic, some of which would not have been necessary with versions prior to 3.7.
- Because the default exceptions for BorderManager 3.7 only allow specific traffic outbound, I have shown a number of other exceptions that used to not be required for BorderManager VPN to function.
- In this edition, I show examples for outbound exceptions *for the proxies themselves* – something that did not have to be done in many cases for BorderManager versions prior to 3.7. For completeness, I show the BorderManager 3.7 default exceptions for each proxy, in case they might need to be recreated.
- I show the FILTSRV MIGRATE process for migrating filter information into NDS. I also show how to export filter information from NDS into a backup file with the FILTSRV\_BACKUP\_FILTERS command, available with the BM37FLT.EXE patch.
- I show how to manually extend the schema for filtering information to be stored in NDS.

- I show how to use iManager to add a new filter exception. I also provided some iManager troubleshooting tips.
- I have added a number of troubleshooting tips that are unique to BorderManager 3.7 filtering issues.

## What's New in the Second Edition

Since the First Edition came out in 1999, I have been looking forward to revising it someday with additional examples and more information on securing your servers. The biggest differences between the Second Edition and the First Edition are:

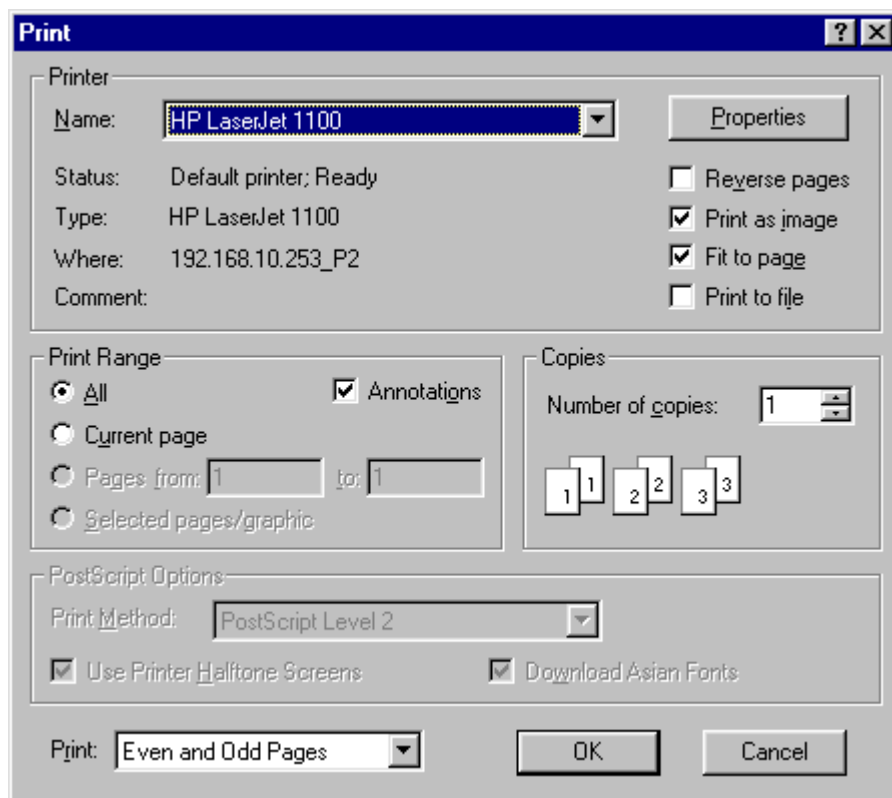
- Every screenshot has been redone, is (in general) larger, in color, and should print more clearly on a wider range of printers.
- Many more filter exception examples, and *almost every filter exception example uses a **custom definition***, instead of the built-in definitions. This was done to specify source ports and/or ACK bit filtering for better security.
- ACK Bit filtering discussed, and used in almost all non-stateful examples.
- Advanced section added discussing DMZ scenarios, complete customization of the filter exceptions, and blocking chat programs.
- Enhancements to the formatting of the book to improve readability, including chapter headers, different spacing and formatting of the table of contents, listing the parameters of filter examples in bulleted lists, and cross-references to figures, headings and page numbers.

# Printing This Book

---

This book is sold in PDF format. While it should display beautifully on your monitor, you may find that you have problems getting the graphics to print well. The symptom is very “blocky” looking graphics. If you have problems printing the graphics, be sure to print as follows:

1. Use at least Acrobat Reader 4.0, Acrobat Reader 3.0 does not have the option shown below. Or, use Acrobat Reader 5.0, which doesn't seem to have a problem printing this book.
2. When you print, you might want to select the option '**Print As Image**' in the print dialog as shown below. This option made a huge difference for me when printing to a 600 dpi HP LaserJet PCL printer.



3. If the above settings do not help, see the troubleshooting guide at the following URL:

<http://www.adobe.com/support/techdocs/150d6.htm>

# Chapter 1 - The Network Configuration

This book works under the following assumptions, and most of the examples provided are based on this network configuration, which is shown in Figure 1-1.

**Note** Some BorderManager 3.7 filtering examples show a server that uses a public IP address of 192.168.1.253 and a private IP address of 192.168.10.3.

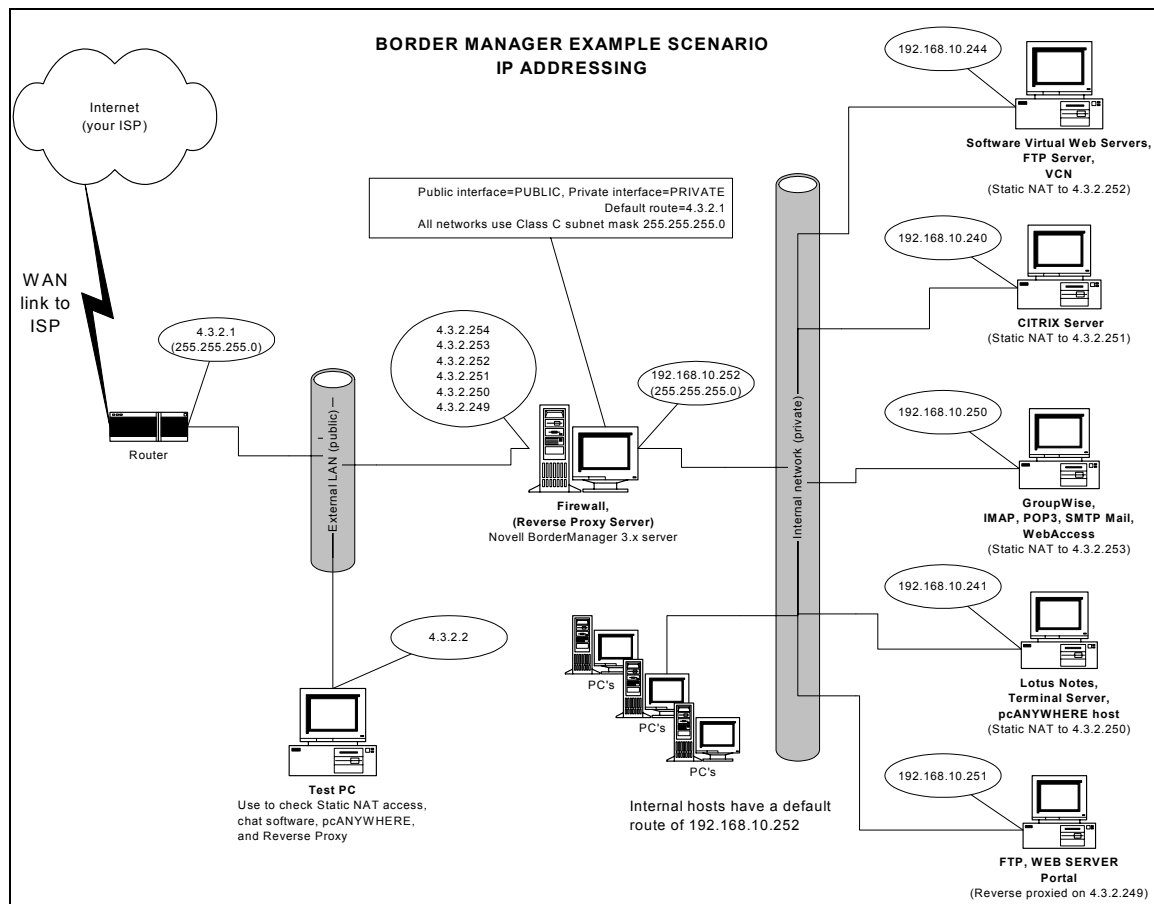


Figure 1-1 - Network Addressing Scenario

A dedicated Internet connection is provided through a WAN link to a router with a small public IP segment. On that public IP segment, there is a LAN connection to the router, a LAN connection to a

BorderManager server, and potentially a LAN connection to a PC, which can be used for testing connections 'outside' the internal LAN.

A BorderManager server is set up with two network interfaces, one connected to the external LAN (Internet/public side) and one connected to the internal LAN (private side). Multiple publicly-registered IP addresses are assumed to be available on the external IP segment, meaning at least a 255.255.255.248 subnet mask is in use on the external IP segment to allow up to six useable IP addresses on that segment.

---

*With a .248 subnet mask, one of the six available publicly-registered IP addresses will be assigned to the router LAN port, one could be reserved for a test PC, and the remaining four are then available to assign to the BorderManager public interface. More IP addresses can be assigned to the BorderManager server if you have, for instance, a full Class-C public address range available to you.*

---

The internal (private side) LAN consists of a private address range, in this book 192.168.10.0.

Dynamic and Static NAT is enabled on the primary public IP address of the BorderManager server (4.3.2.254).

In the scenario shown in, all the internal hosts have a default gateway set to 192.168.10.252. The BorderManager server has a default route set to the LAN port of a router at 4.3.2.1, as does the test PC. The BorderManager public side network interface is named PUBLIC, and the private side network interface is named PRIVATE.

At least one secondary IP address is assigned for use for reverse proxy acceleration of an internal web server. Several secondary IP addresses have been assigned for static NAT to an internal mail server, an FTP server, a web server, a pcANYWHERE host, and several other services. The BorderManager default filters are applied.



# Chapter 2 - The Basics

---

Before you start trying to set up filter exceptions, you need to know basic information like:

- How packet filtering works
- How port numbers are used
- How routing works
- How to add additional IP addresses to one network interface card
- How to view what is happening to IP packets at a NetWare server

This section covers much of what you need.

## How Packet Filtering Works

Packet filtering works by examining fields in TCP/IP data packets and allowing them to be routed or not depending on the value of certain fields. This book primarily is concerned with filtering as it pertains to the source and destination port numbers, and somewhat with source and destination IP Address fields.

Generally, ALL traffic is filtered at some point, and certain exceptions are allowed through. In the case of Novell's BorderManager, the default filters stop all inbound IP traffic from the public interface, and outbound IP traffic going to the public interface. In addition, the default filters include certain filter exceptions that are needed for HTTP Proxy and VPN to function.

---

**Note** The default filters also include IPX filters, but this book is entirely concerned with IP filters and does not make any other mention of IPX filtering.

---

The filters, and filter exceptions, do not just stop data from getting 'through', but also actually stop traffic from either entering or leaving a certain IP address or a certain network interface. This is an important point, because you may run into situations where traffic is allowed to an internal (private side) interface on your BorderManager server, but is dropped when it tries to leave an external (public side) IP address. Thus, the direction that traffic is flowing is also considered in packet filtering.

## Stateful Filter Exceptions

Stateful filter exceptions automatically keep track of a 'conversation' between the originating host and the receiving host. A stateful filter

exception not only allows a certain port number in one direction, but also allows the necessary return traffic back from the other end (without having to set up a second filter exception for the return traffic). A stateful filter exception is best applied to source and destination interfaces. A stateful filter exception automatically includes ACK bit filtering for TCP packets.

BorderManager versions 3.0 and later provide the option of stateful filtering, and come with several predefined stateful filter definitions. In fact, for some time now, the filtering modules used in BorderManager have been included in the latest NetWare Support Packs, so the same filtering can be done even without BorderManager installed.

Stateful filter exceptions are very convenient as they are easy to set up and are more secure than non-stateful filters. However, they are (theoretically) a bit slower as there is more CPU and memory overhead involved. In this book, most of the outbound filter examples shown use stateful filters but the inbound exceptions (to proxy or static NAT hosts) do not use stateful filters in an attempt to maximize performance, and because of a potential security problem (not discussed in this book). Stateful filters can be applied to TCP, UDP and ICMP traffic.

## ACK Bit Filters

ACK bit filters check for the presence of the ACK bit (acknowledge bit) in a TCP packet. (UDP packets don't have an ACK bit). When a TCP 'conversation' (session) is initiated, the first TCP packet doesn't have the ACK bit set. However, the return traffic and subsequent packets between source and destination for that session do have the ACK bit set. Since you generally cannot initiate a TCP session to a host with the ACK bit already set, filtering incoming traffic for the presence of an ACK bit (not set = filter the packet) is better for security than not checking the ACK bit. The idea here is to not allow the packets in unless the ACK bit is already set.

Novell provides the ability to use ACK bit filtering in BorderManager 3.0 and later. This book also applies ACK bit filtering in all of the outbound response filter exceptions for static NAT, in order to increase security and control of the server.

If the default Dynamic/TCP filter exception is changed to call out ACK bit filtering, (see "Basic Improvement - Enhance the Security of the Default Exceptions", page 265), then inbound high port traffic is allowed, but only if it is a **response** to an earlier outbound request that set the ACK bit.

UDP does not use ACK bits, therefore you cannot set ACK bit filtering on the default Dynamic/UDP filter exception. However, there are not many applications that listen on UDP high ports on a

NetWare server, so there is not very much exposure to problems with the default Dynamic/UDP exception.

## Filters and the Relationship to NAT and Routing

Filters can take place before or after a packet is routed to the next hop. It is more useful to think of filters and filter exceptions as applying when a packet either arrives at or leaves an interface or an IP address. In the vast majority of cases, filters are applied on one, (or more), interfaces, and so filtering takes place when a packet tries to enter or leave an interface. However, filter exceptions are set up to apply when packets enter or leave an interface or to or from certain IP addresses.

It is more confusing for most people to see how filtering is related to NAT. The important concept here is that NAT itself takes place 'at the edge'. That is, NAT is either the first or the last process to take place, depending on the direction of the packet and where NAT has been applied.

The normal case for NAT is that it is applied on the public IP binding. Therefore, NAT is applied on the 'outside edge' of the network. Packets coming IN to the network from the Internet FIRST undergo NAT, then packet filtering, and then are routed. Packets going OUT of the network are filtered, and THEN undergo a NAT translation. This is why the static NAT filter exceptions shown in this book call out internal IP addresses rather than secondary public IP addresses. If NAT is applied to the private IP address (as is sometimes done with Client-Site VPN), inbound packets are filtered before undergoing NAT, and outbound packets undergo NAT before filtering.

One last point: NAT defaults to performing its own type of filtering, called NAT Implicit Filtering, by simply dropping packets not undergoing a NAT translation. This is normally seen when dynamic NAT is configured on the public interface, and inbound traffic to a reverse proxy is dropped. Unless there are no reverse proxies (or static NAT mappings) configured on the server, you want to disable NAT Implicit filtering (INETCFG, Protocols, TCP/IP, if you have the server patched properly). Before the NAT Implicit Filtering menu entry was added to INETCFG in a NetWare support pack, you could only use the SET NAT DYNAMIC MODE TO PASS THRU=ON command. NAT Implicit Filtering drops packets without showing any reason in TCP IP DEBUG traces, unlike filters, which show DISCARD.

## What Are Port Numbers?

Port numbers can be thought of as describing the type of data contained within a packet. (Strictly speaking, port numbers just represent the endpoint of a connection, and not the type of data, but it is useful in this book to consider certain port numbers as meaning certain types of data). Both the sending and receiving program must agree on what port numbers are being used for the program to function. Typically, a 'well-known' port number is used for the destination port when trying to initiate a conversation over IP. Most well known programs use port numbers below 1024.

---

**Note** A search of the Internet for 'well-known port numbers' should turn up a number of sites that list many well-known port numbers. See the Other References chapter for some web sites to check.

---

For example, a TELNET session would typically send out the first packet with a destination port of 23. The receiving program would effectively be looking for port 23 inside incoming data packets, and when one was found with that port, the data would be processed further.

---

**Note** Strictly speaking, the application registers a listener port with the stack. The stack itself then monitors incoming packets on the port and puts it into a queue (where there is one queue per listener). The application monitors this queue, picks up any packets found in there, and processes them. This is the technical explanation (from a Novell engineer), but I find it easier to just think of an application listening for certain port numbers!

---

On the other hand, the source port is typically assigned at random, from a range of 1024 to 65535. Return traffic to the originating host would send back data packets using the original source port as a destination port. Consider the following example:

A Simple Mail Transfer Protocol (SMTP) packet is sent from 4.3.2.1 to 192.168.10.100 using source port 1059 (chosen at random from a range of numbers between 1024 and 65535) and destination port 25. The originating SMTP program expects to receive return traffic on a destination port of 1059 or it will not recognize the traffic as a response to this packet.

The SMTP mail server at 192.168.10.100 receives the packet, recognizes that it is an SMTP mail packet (because the destination port number is 25) and processes it.

The SMTP mail server at 192.168.10.100 then sends out a return packet using the original source and destination ports, except that it switches the two. A new packet goes back to 4.3.2.1 using source port

25 and destination port 1059. The host at 4.3.2.1 receives the packet from 192.168.10.100, and it recognizes the source port 1059 as being part of a conversation it was trying to have with the host at 192.168.10.100, and it processes the data accordingly.

## How Routing Works

A discussion of routing protocols is well beyond the scope of this book, but some discussion must be made *since packet filter issues often turn out to be routing issues instead!* Generally, you need to be sure that a routing protocol such as IP RIP is enabled on your internal routers, **or** you have entered static routes for all your internal networks.

There are two basic points to be made here:

In order to move a packet inside your LAN, all the routers (including BorderManager) must know all the interior subnets and what interface/IP address is used to forward any packets destined for that subnet. You may think that data is not getting 'through' the firewall because you are not seeing a response, but the problem may simply be that the BorderManager server did not know how to get the packet back to your host.

**All** routers and hosts on both internal and external IP *segments must have a default route set up!* Once you try to communicate with hosts on the Internet, it is not possible to build a routing table in which the location of every subnet is included. Instead, a default route (default gateway) is used to forward any packet with an unknown network address. The default route/gateway is always an IP address on the same network as the host, and points to the next hop to be used to get an unknown packet out to the Internet. Typically, on a BorderManager server, the default route is set to the external IP segment address (this would be a LAN port, not a WAN port) of a router connected to the Internet. NetWare server default routes are configured in INETCFG.NLM, and stored in the SYS:\ETC\GATEWAYS file. They can be verified using TCPCON.NLM.

The first thing to try when you have a communications issue is to disable packet filtering. (This is, of course, an obvious security hole, but it is the easiest way to confirm that filtering is an issue). If the traffic still doesn't flow, you are likely to be having a routing issue, and experience says that most routing issues involve incorrect or missing default routes.

Here is an analogy of what a default route is. Say you live in a house with several other people, and you want to send one of them a little letter. You write the letter and are ready to deliver it. Since you live in the house, you not only know the address of everyone in the house (master bedroom, kid's bedroom, etc.) but you know how to find the room. Therefore, you go to the room and slide the letter under the door. Now, let's say you want to mail a letter to someone else, and that person lives in another city. You have the address, but you have no clue how to deliver the letter yourself! However, you DO have a

'default route' - the mailbox. Therefore, you drop the letter in the mailbox, and trust the postman to deliver the letter. The postman of course doesn't have any idea how to deliver that letter either, but he/she DOES have another default route - the letter-sorting bin for letters going out of town. Along various steps of the way, the letter keeps getting delivered not directly to the end destination, but instead to the next 'hop' along the way to the destination. Finally, a postman on the final step of the letter's journey gets the mail, and since he/she actually knows where the house is, the letter can be put into a final destination (mailbox). Even then, someone at the house may pick up the letter and forward it onto the intended recipient.

Now suppose that the reader of the letter wants to send a reply. All of the same steps have to occur in the reverse direction, or the reply does not get through.

There is even a measure of filtering involved here - if the letter does not have enough postage, it does not get through.

What is the point of all of this? If ANY step in the process IN BOTH DIRECTIONS does not have a default route, the mail (your TCP/IP packets) will not get through, unless the end address is a local address (inside your house = inside your LAN). Remember that the first step a packet takes toward a host outside your LAN is the next router on your LAN, and your own PC needs a default route to it in order to start the packet on its way.

## Setting up the Default Route

**Note** A fresh installation of BorderManager 3.7 will prompt you to enter a default route. However, you may as well configure the default route when you install NetWare, before installing BorderManager, using the following procedure.

Your BorderManager server will need a default route configured in order to function. This is most easily done at the server console using INETCFG.NLM as shown in Figure 2-1.

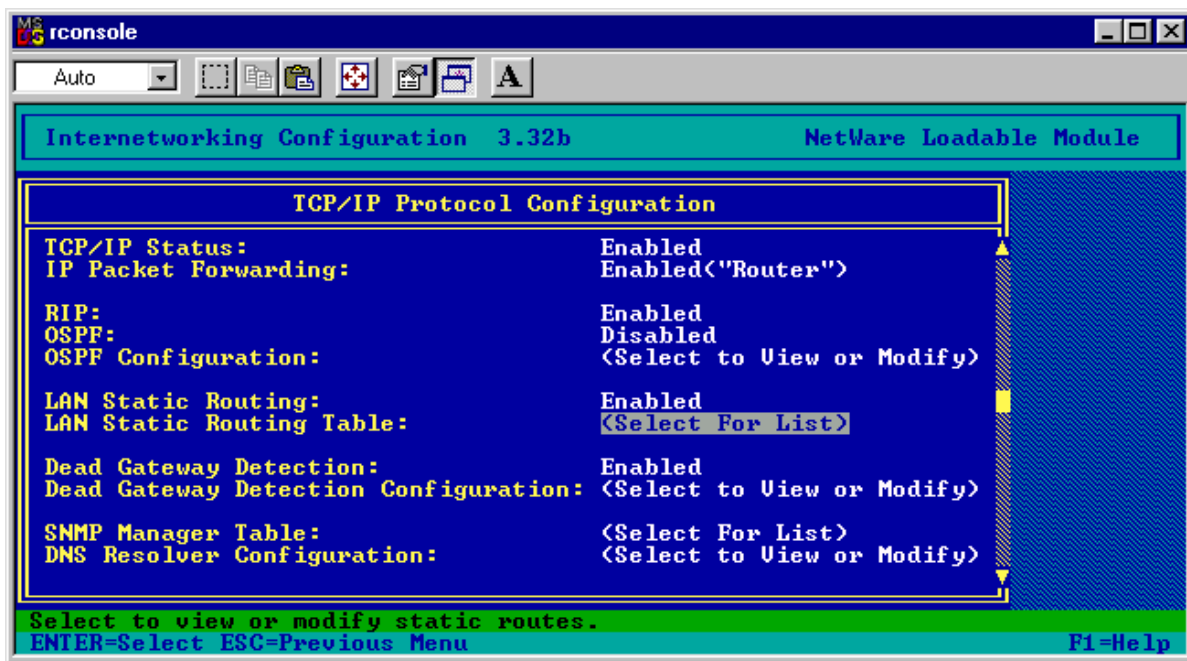


Figure 2-1 - INETCFG, Protocols, TCP/IP

To set up a default route, type **LOAD INETCFG** at the server console prompt, select **Protocols**, **TCP/IP**, enable static routing, and select **LAN Static Routing Table**.

**Note** The screenshot shown in Figure 2-1 was taken from a NetWare 5.1 server that has the proper configuration files for Dead Gateway Detection. Your server may or may not have that option, which is related to the version of TCPIP.NLM that is installed by patches. Dead Gateway Detection is a method of determining if one default route is active and switching to another one if it is not.



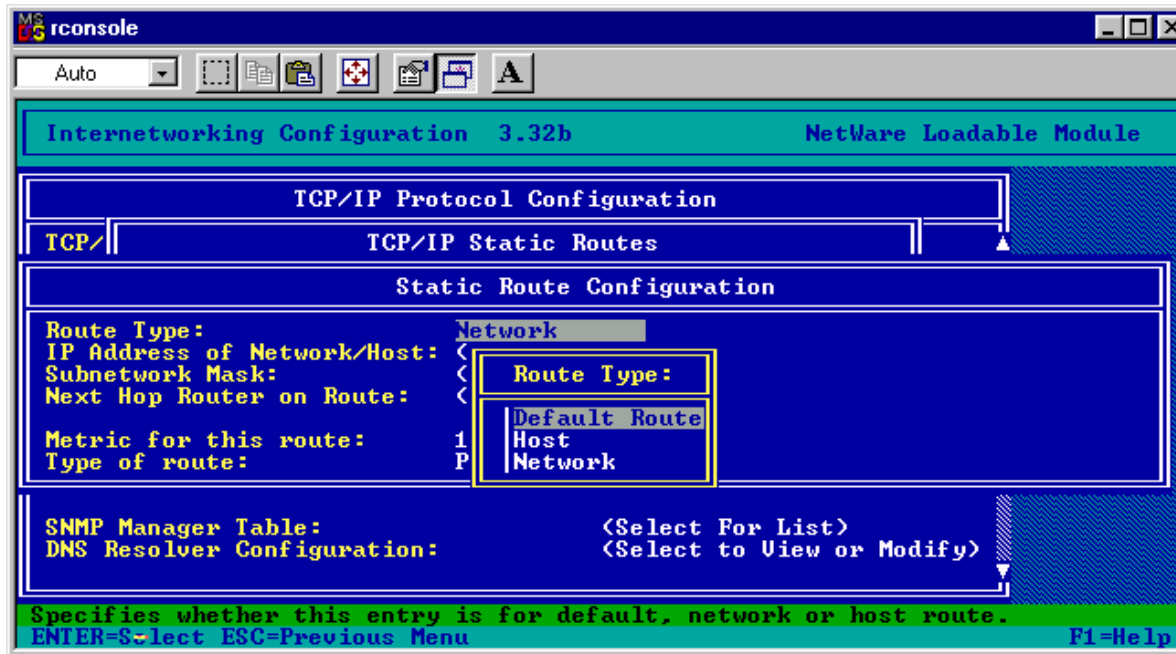


Figure 2-2 – INETCFG, Protocols, TCP/IP, LAN Static Route, <insert>

Once you select **LAN Static Routing Table**, press **Insert**, and then select **Default Route**.

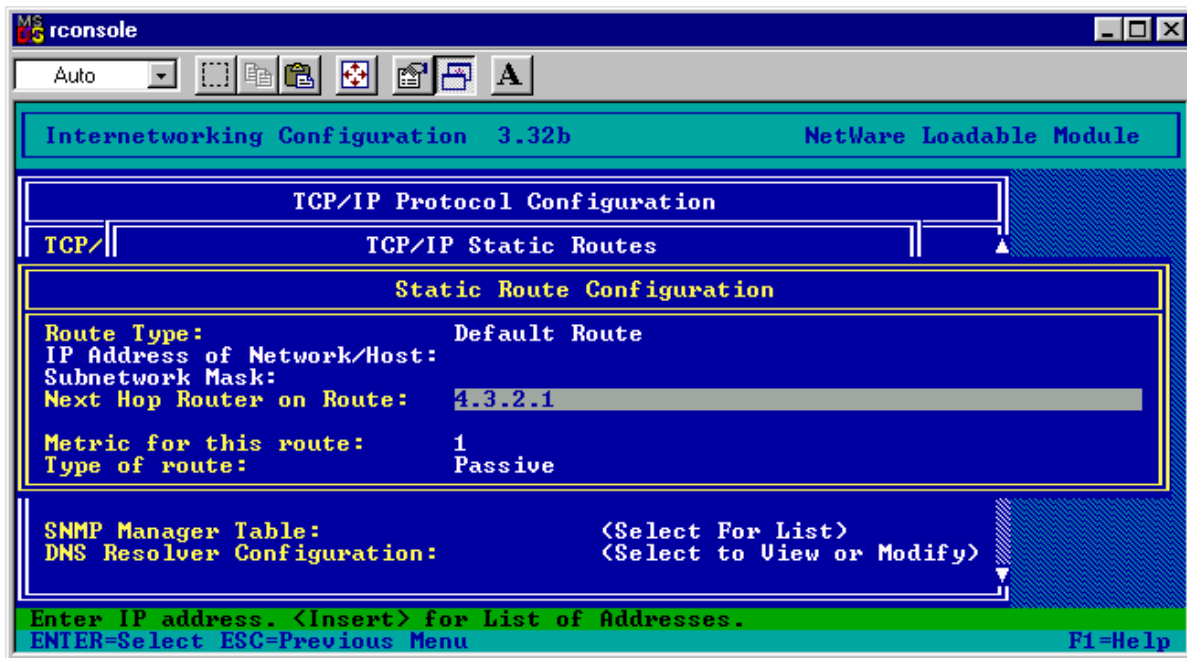


Figure 2-3 - INETCFG - Enter Next Hop for Default Route

Enter the next hop for the default route for your server. This should be the local LAN address of the router connecting your BorderManager server to the Internet, or the router that is the next hop towards the Internet. Refer to Figure 2-3.

When done entering data, accept the changes, and go back to the main menu.

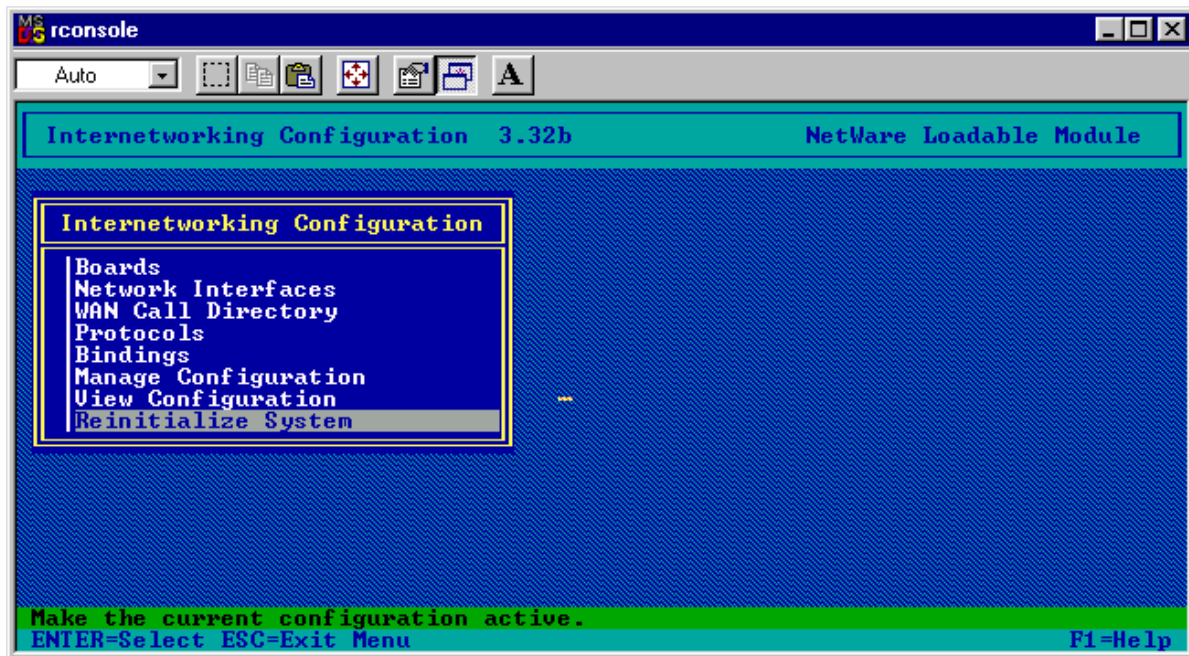


Figure 2-4 - INETCFG - Reinitialize System Option

Select **Reinitialize System** to put the changes in to effect. (Or type in **reinitialize system** at the console prompt). Exit INETCFG when done.

**Note** The default route entry should appear in the **SYS:\ETC\GATEWAYS** file, and look something like this:

Net 0	Gateway 4.3.2.1 Metric 1 Passive
-------	----------------------------------

You can also edit this file manually if you wish, but remember to type **reinitialize system** when done editing.

The Net 0 (sometimes shown as 0.0.0.0) indicates 'default route', by convention. Gateway is a key word indicating that all packets going to network 0 (which means all packets going to some address not otherwise present in the routing tables on the server) will be sent to the IP address following (4.3.2.1). Metric 1 means that the cost of this route is '1' (which is as low as Novell allows and takes precedence over higher cost routes). Passive effectively means that the route is considered to always be available.

## Public and Private IP Address Networks

In order to route IP traffic to the proper host on the Internet, each host must be configured with a globally unique IP address that is **registered** with Internic. Such an IP address is called a *public* IP address. A company will normally purchase an IP address range from an Internet Service Provider (ISP) and pay a yearly maintenance fee based partly on the number of IP addresses they are reserving. The ISP will take care of ensuring that all incoming Internet traffic to a host within that IP address range knows how to get there. It is essential to have at least one properly registered public IP address configured on the public interface of your BorderManager server for it to communicate to the Internet (unless using Network Address Translation on an 'upstream' router).

Partly because of the cost involved, and partly because the world is running out of publicly available IP address ranges, not everyone has public IP addresses assigned **inside** their private LAN's. In some cases (not recommended), an address range registered to a different company is in use on a private LAN. To avoid the situation where registered addresses are being used on different networks, three different IP address networks have been set aside for anyone to use. These special IP networks are called *private* IP addresses. **Internet routers are programmed to drop packets with a private IP destination address.** The three private address ranges set aside for use are:

- 10.x.x.x (a full class A range)
- 172.16.x.x to 172.31.x.x (15 Class B ranges)
- 192.168.x.x (254 Class C ranges)

You can use these IP networks as you wish within your internal network and subnet them as needed, but they **MUST** be used with either dynamic NAT (Network Address Translation) or proxy services or both. Most people find the 192.168.x.x network to be the easiest to work with, as it is easier to understand Class C subnetting than other classes. The use of these IP networks is discussed in the following document:

RFC 1918 - Address Allocation for Private Internets. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot & E. Lear. February 1996. (Format: TXT=22270 bytes) (Obsoletes RFC1627, RFC1597) (Also BCP0005) (Status: BEST CURRENT PRACTICE)

---

**Note** Use this URL for a link to RFC 1918: <ftp://ftp.isi.edu/in-notes/rfc1918.txt>

---

Since these ranges cannot 'talk' to the Internet (packets from these addresses will be dropped at some point on an Internet router), some special techniques must be used to let internal hosts using these addresses communicate to the Internet. The two techniques in use are NAT (Network Address Translation) and Proxy. BorderManager provides both capabilities. Both techniques effectively substitute the public IP address of the BorderManager server for the actual (private) source address of the host originating the traffic.

There is not a problem using NAT and proxy services at the same time. You can easily have some types of traffic using a proxy and other types of traffic using NAT. For the security-conscious, consider that using a proxy is considered to be more secure than using NAT, if you have the option of using either.

---

*Remember – if you use the private IP addresses, you will not get a response back from the Internet to your PC unless you are using a Proxy, a Gateway service or have dynamic NAT enabled! This has nothing to do with filtering! The routers on the Internet will drop packets with private addresses.*

---

## Secondary IP Addresses

Once you wish to provide a service to users on the Internet (like a public web server), you will often find that you need to assign more than one IP address to the public network interface card in a firewall, in this case BorderManager. You will usually need a dedicated IP address for each service, such as a web server or a mail server that you want to host. The characteristic involved is whether or not you need to allow incoming traffic – traffic going from the internal LAN to the outside (Internet) is usually sent out a single IP address and doesn't require any additional addresses on the BorderManager server.

With NetWare, it is possible to assign many (at least 254) IP addresses to each network interface in a server, though it isn't so easy to see more than one assigned address.

---

**Note** You can assign addresses in different networks to a single network card, and NetWare will route between them as if they were assigned to two different network cards. Assigning addresses from different networks is done in INETCFG by simply binding a new address to an interface. An example would be to assign 192.168.10.254 and 172.16.31.254 to an interface. This book does not cover such an assignment, as it is not normally needed in a BorderManager configuration. This is NOT the same as a secondary IP address.

---

A typical way to assign multiple addresses to a network interface is to add IP addresses from within the same IP network to an interface. An example would be to assign 192.168.10.253 and 192.168.10.252 to an interface that already has IP address 192.168.10.254 bound (from INETCFG). These types of addresses on a NetWare server are called *secondary IP addresses*. Assign a secondary IP address to an interface with the **ADD SECONDARY IPADDRESS** command, as in this example, which adds IP address 192.168.10.253 to an existing interface.

```
ADD SECONDARY IPADDRESS 192.168.10.253
```

---

**Note** IPADDRESS is all one word!

---

NetWare will look at the addresses already assigned to the interfaces and add the secondary IP address to the interface that is already configured for that network range. Again, an example would be to

have a current binding of 192.168.10.254 on an interface (configured with INETCFG under Bindings, TCP/IP), and ADD SECONDARY IPADDRESS 192.168.10.253 at the server console. You would then have two addresses assigned to the same interface. Once you have executed the ADD command, the IP address is instantly available – you do not have to reinitialize or reboot the server.

Secondary IP addresses do not show up when typing CONFIG at the server, and they do not show up in the Bindings menu of INETCFG. You display the secondary IP addresses with the command

```
DISPLAY SECONDARY IPADDRESS
```

If you wish to remove a secondary IP address, use the command **DELETE SECONDARY IPADDRESS** as in this example that removes the previously defined secondary address of 192.168.10.253.

```
DELETE SECONDARY IPADDRESS 192.168.10.253
```

---

*Caution! Secondary IP addresses **are not permanent** – you need to put the ADD SECONDARY IPADDRESS 129.168.10.253 command in AUTOEXEC.NCF (after the primary bindings are made) so that the addresses will be available after a server reboot.*

---

## NAT (Routing) versus Proxy

BorderManager provides more than one means of getting to the Internet – using a Gateway service, using a Proxy service (only HTTP proxy in BorderManager 2.1, and several proxies in BorderManager 3.0 or later), or using simple routing. For the purposes of this book, the difference is that simple routing requires more setup on the part of the administrator in terms of filter exceptions, NAT and DNS.

When using a Proxy service on BorderManager, the originating program at the PC (a browser, for example) is normally configured with a special proxy setting that points to the BorderManager private IP address and designated listening port number. When that PC sends out traffic, packets are handed off to the BorderManager proxy, and the proxy then **regenerates** the packets onto its public interface (with the BorderManager public IP address as the packet's source address). A proxy does not route the packet between interfaces, it regenerates it - this is an essential difference between using a proxy and using NAT, because NAT will require filter exceptions to allow the packet to be routed between interfaces while a proxy simply skips that step. When the return traffic comes back to the BorderManager server, the proxy regenerates the reply onto the private interface with the original PC's IP address as the return packet destination address. Because the proxy is doing all the work for the PC, the PC doesn't have to be configured with DNS (at least not in the case of HTTP proxy), nor does dynamic NAT have to be enabled at the server. However, the server itself must be properly configured to resolve DNS queries.

A fresh install of BorderManager 3.7 (not an in-place upgrade), will only have filter exceptions created to allow specific outbound proxy access for proxies chosen during the install procedure. The administrator will have to manually add filter exceptions for any proxies enabled after the initial installation, and may very well have to add custom exceptions in some cases. The administrator will have to manually add filter exceptions for all inbound traffic desired for reverse acceleration. This is a major difference between a fresh installation of BorderManager 3.7, and an in-place upgrade, or previous versions of BorderManager. Prior to BorderManager 3.7, certain default filter exceptions were created which automatically allowed virtually all outbound traffic through any proxy, as well as inbound traffic for HTTP Acceleration (reverse HTTP Proxy).

You control traffic through proxies by setting up access rules in the BorderManager configuration.

If routing (as with NAT) is used instead of Proxy services, you will need to

- Define a DNS entry on the originating host PC, at least if DNS hostname queries are required for the service (such as HTTP)



- Enable Dynamic NAT on the BorderManager server if a private IP network address is used on the internal LAN

In addition, some type of filter exception must be configured on the BorderManager server to allow the desired traffic to go out and to allow the return traffic to get back in. The only control over outbound traffic is to set up the filter exceptions allowing the traffic. These exceptions can allow every host in the internal LAN to get out, or only selected IP network ranges, or only selected IP addresses (hosts).

## Dynamic NAT - for Outbound Traffic

Dynamic NAT is used to automatically translate (and 'hide') internal IP addresses to a public IP address on the BorderManager server. Dynamic NAT keeps track of the conversations taking place and dynamically couples the return traffic to the original requester. Dynamic NAT is usually set up on the primary public IP address only (in INETCFG, under Bindings, select the public IP address, then select Expert Options). With dynamic NAT, all the IP packets sent out will have the same source IP address.

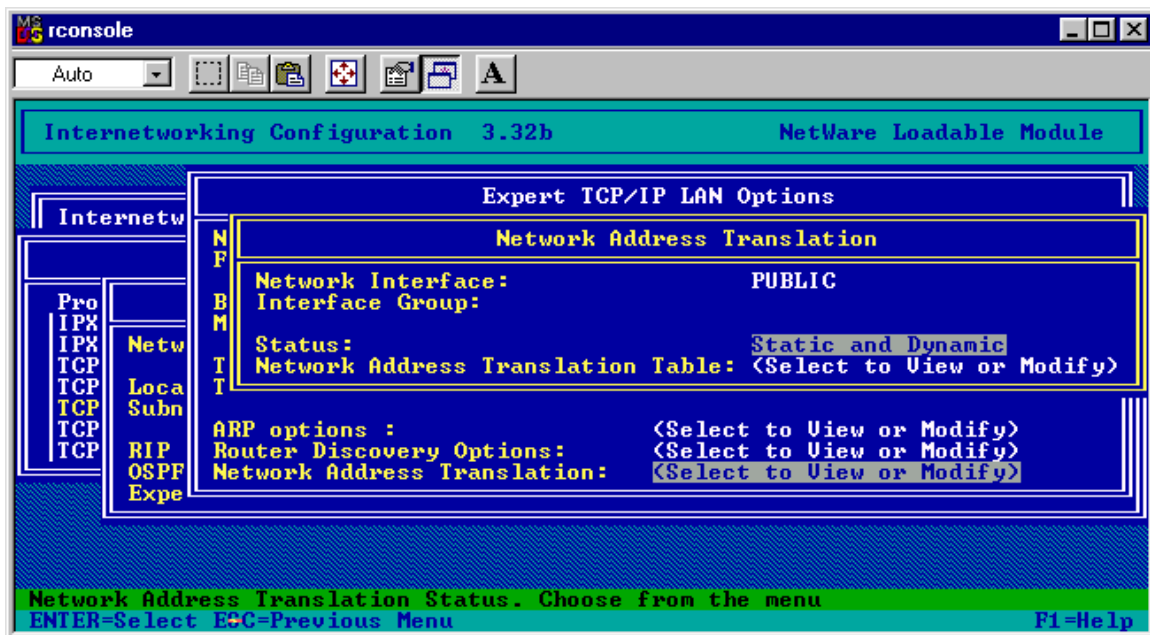


Figure 2-5 - INETCFG, Bindings, <public IP address>, Expert TCP/IP Bind Options, Network Address Translation

Figure 2-5 above shows an entry in INETCFG for both static and dynamic NAT enabled.

Some points in regard to dynamic NAT:

- Dynamic NAT may not be as secure against Internet 'hacks' as using proxies.
- Dynamic NAT still requires filter exceptions to allow traffic through from the internal LAN to the Internet.
- Dynamic NAT is used to allow **outbound** traffic – traffic originating from a host on your internal LAN.

## NAT Implicit Filtering

If you have a service running directly on the BorderManager server that you need to access from the Internet, you need to disable NAT Implicit Filtering when you enable Dynamic NAT. NAT Implicit Filtering drops inbound packets for connections that did not originate from the public IP address. If NAT Implicit Filtering is enabled – and it is enabled by default – some of inbound packets are simply dropped, and nothing will be seen in TCP/IP DEBUG=1. (Filtering does not discard the packets, so no DISCARD data will be seen).

In some cases, NAT Implicit Filtering might also block inbound traffic to a static NAT address, although it is not supposed to.

Before some of the later NetWare support packs were released, only a command entered at the server prompt (or in AUTOEXEC.NCF) could be used to disable NAT Implicit Filtering. (See below).

## Disabling NAT Implicit Filtering in INETCFG

If you have the latest NetWare Support Pack installed, you should have an option in INETCFG, Protocols, TCP/IP for enabling or disabling NAT Implicit Filtering.

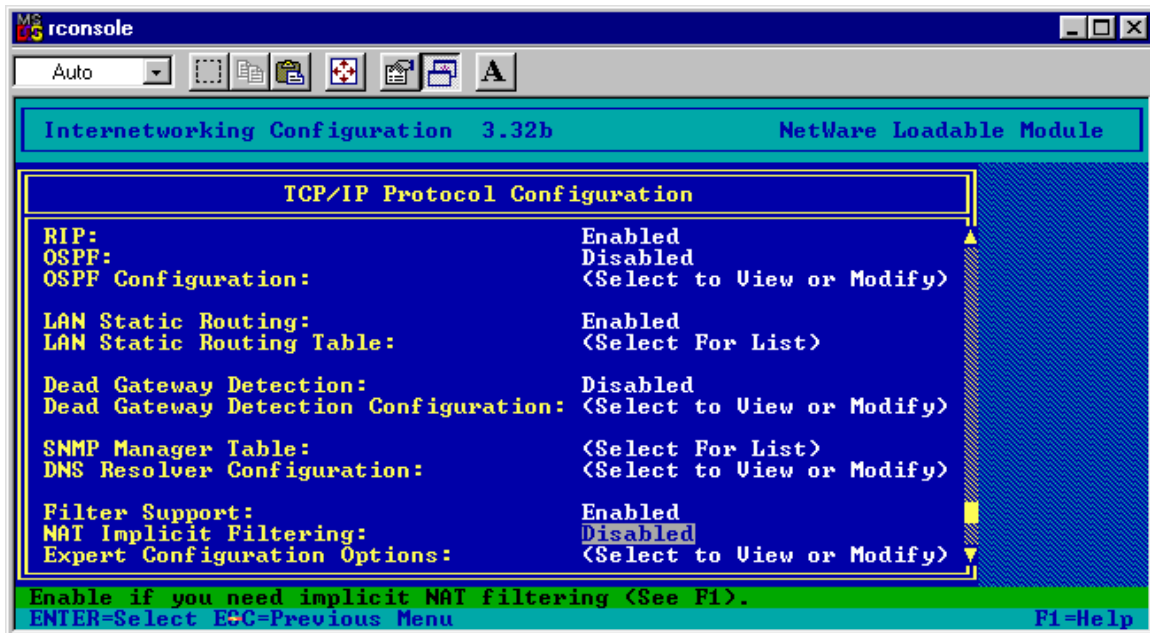


Figure 2-6 - INETCFG - Option to Disable NAT Implicit Filtering

The setting shown in Figure 2-6 was taken from a NetWare 5.1 server, with TCP/IP version 5.52u and the NW51SP3 support pack installed. The option for dead gateway detection is related to a TCP/IP patch, and you may not see that option on your server.

## Disabling NAT Implicit Filtering at the Server Console Prompt

Add the following command to AUTOEXEC.NCF so that traffic is

```
SET NAT DYNAMIC MODE TO PASS THRU=ON
```

allowed to 'get into' the BorderManager server:

In general, if you have dynamic NAT enabled, and something on the server isn't working, try the above SET command.

---

**CAUTION** *There has been at least one patch released which cause NAT implicit filtering to always get set back to Enabled each time INETCFG is launched. Using the SET NAT DYNAMIC MODE TO PASS THRU=ON is a good idea to have in AUTOEXEC.NCF as it would override the INETCFG setting upon a server reboot should an administrator not notice that setting got changed. If you have the latest NetWare support pack installed, this problem should no longer occur*

---

## Security Implications for Disabling NAT Implicit Filtering

In general, there are no implications for disabling NAT implicit filtering, because if you need to disable it, you are running some service on the BorderManager server itself, or through static NAT. Without disabling NAT implicit filtering, you would not have access to those services.

However, there is an alternative that makes use of how static NAT functions. Should you need to allow access to a reverse proxy, or generic proxy, on BorderManager, and wish to disable NAT Implicit Filtering for just that IP address, you can static NAT the public IP address to itself. What will happen then is that the static NAT assignment should preempt the NAT filtering, and allow traffic through to the service listening on that IP address.

## Static NAT - For Inbound Traffic

Static NAT is used to allow **inbound** traffic through a BorderManager firewall to a specific internal host IP address. If you want to make an internal host available to the Internet with BorderManager, your options are to set up static NAT (and appropriate filter exceptions) or Reverse Proxy. Static NAT involves pairing an address on the public side of the BorderManager server with the IP address of any internal host on your network.

Static NAT is usually done with a secondary IP address assigned to the public interface in a BorderManager server. Using Static NAT with the primary public IP address on the BorderManager server will result in almost all BorderManager services failing.

Static NAT requires filter exceptions to work. Generally, you set up one filter exception to allow desired traffic TO the private internal IP address, and a second filter exception to allow traffic FROM the secondary IP address.

---

**Note** That's right - I said the filter exceptions for static NAT use the internal IP address of the host, not the IP address assigned on the BorderManager server!

---

Static NAT offers less security than Reverse Proxy does. Reverse proxy for later versions of patches for BorderManager 3.5 and later also provides for virus pattern detection in the PROXY.CFG file.

You can only configure one static NAT address pair for any IP address. If you want more than one internal host to be available to the Internet through static NAT, you will have to have more than one public-side IP address assigned to the server.

For instance, you have two internal FTP servers you wish to get to from the Internet, 192.168.10.100 and 192.168.10.101. You have assigned a secondary IP address of 4.3.2.253 to your BorderManager public IP interface. You set up a static NAT address pair of public=4.3.2.253 and private=192.168.10.100. You cannot now assign the 192.168.10.101 address as a static NAT pair unless you add one more public IP address, such as 4.3.2.252.

Static NAT can be (and usually is) configured in addition to dynamic NAT.

## Static NAT and Filtering

The reader will observe later in this book that filter exceptions for static NAT are applied differently than filter exceptions for a reverse proxy. Specifically, the filter exceptions for static NAT are applied to the internal IP address of the host in the static NAT configuration, while the filter exceptions for reverse proxy are applied to the secondary IP address on the BorderManager server (and not the internal IP address of the web server being reverse accelerated). The reason for this has to do with how filtering works in relation to NAT.

When you enable static NAT on an interface, (usually the public interface, via the public IP address binding) visualize NAT as being on the interface, but filtering as happening in the 'middle' of the server. When packets come into the server from the public side, they will first be acted upon by NAT, and then by filtering. Thus, the filter exceptions for static NAT (which is primarily used to allow inbound traffic) have to be set up for the traffic after the address conversion has occurred.

Conversely, for the outbound traffic, filtering happens first, and then the NAT translation.

## Setting up Static NAT

The following instructions show how to use INETCFG.NLM at the server console to configure a static NAT address pair. You must first have the public IP address configured on the server.

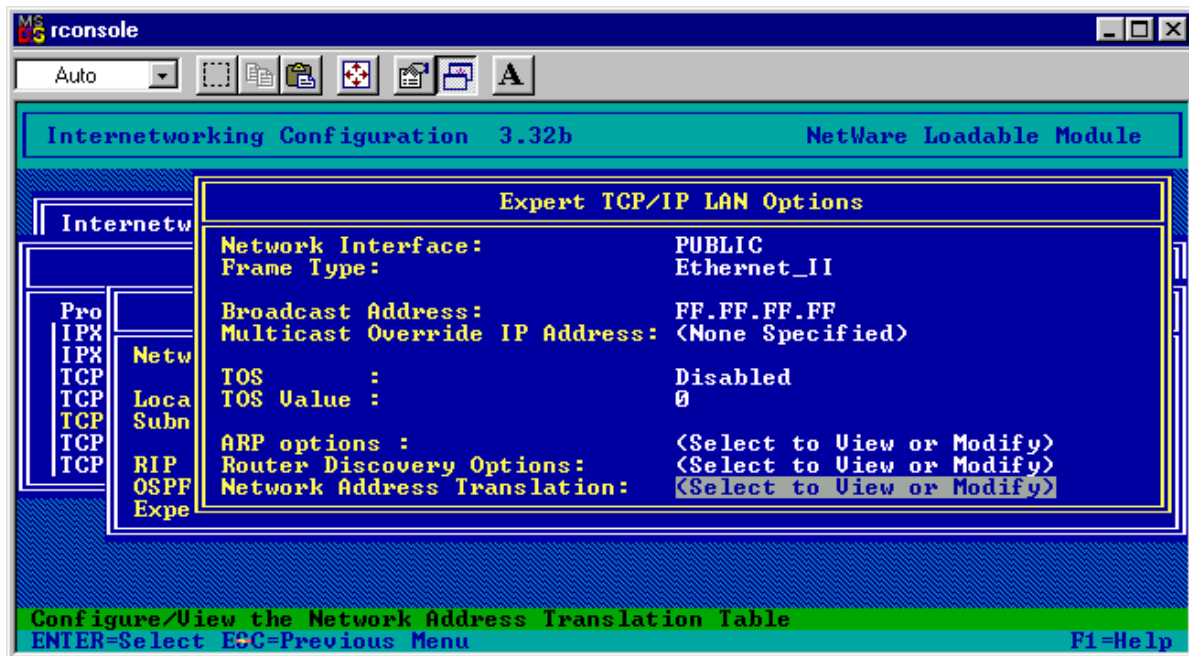


Figure 2-7 - INETCFG - Network Address Translation

At the BorderManager server console prompt, type **LOAD INETCFG**, select **Bindings**, select <your public IP address>, select **Expert TCP/IP Bind Options**, and you will be able to select the menu entry for **Network Address Translation**.

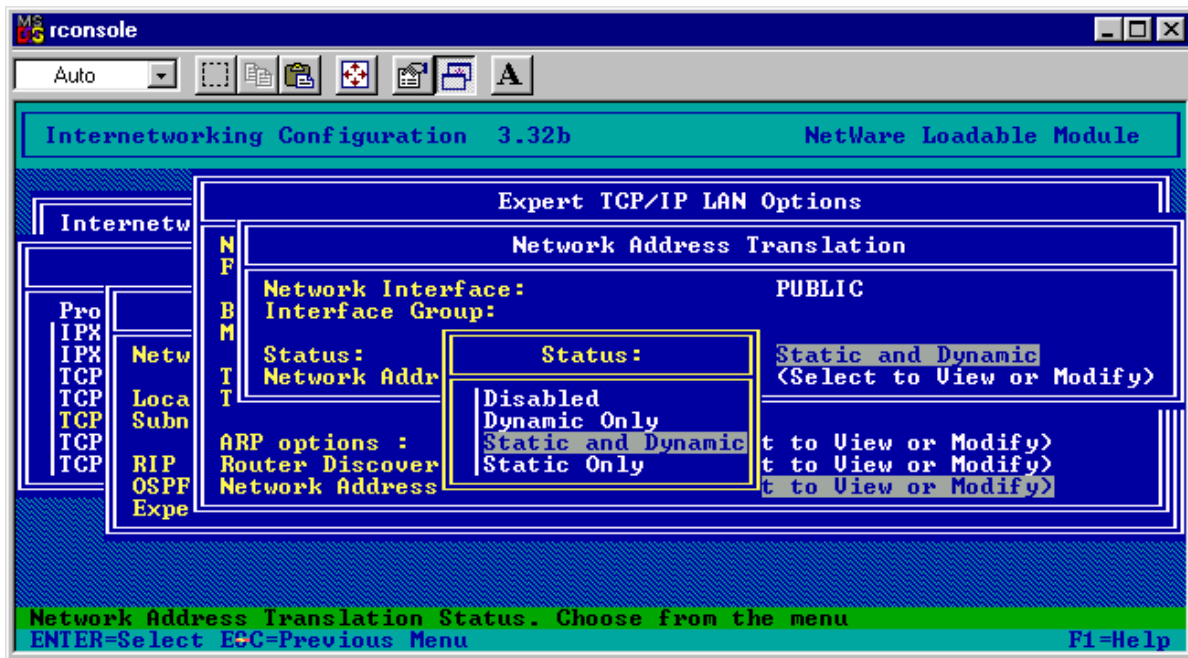


Figure 2-8 - INETCFG - Select Static and Dynamic NAT

If you have any secondary IP addresses set up, and you want to use static NAT as well as dynamic NAT, select **Static and Dynamic**.



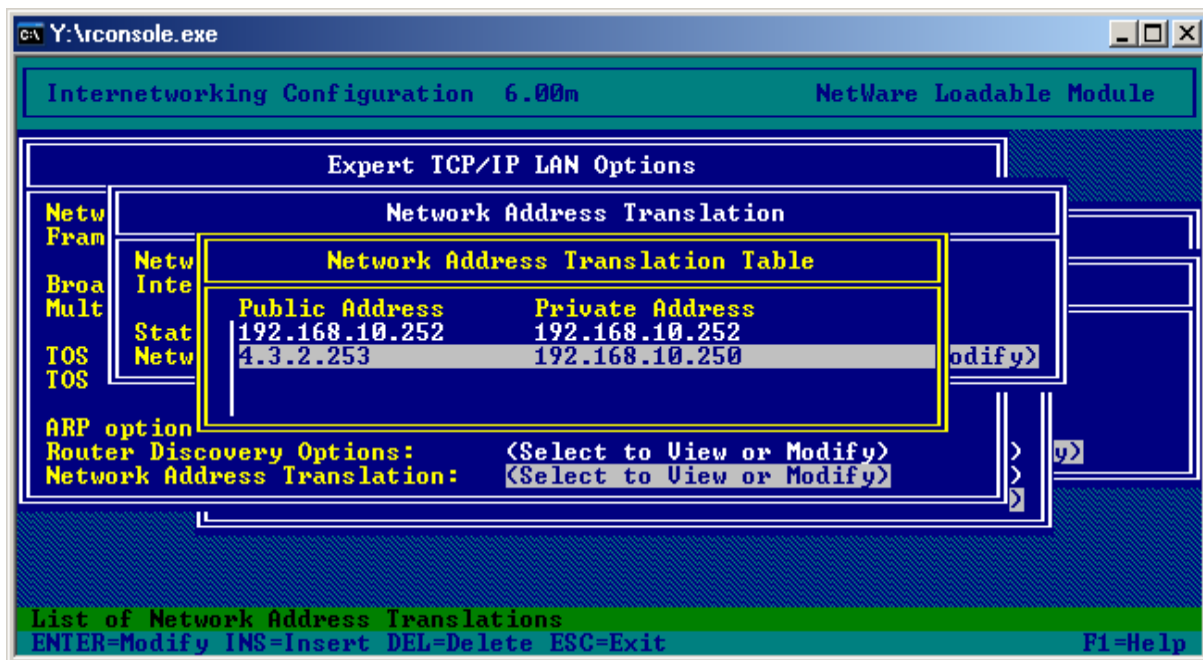


Figure 2-9 - INETCFG - Entering Static NAT Mappings

If you select **static NAT**, or **Static and Dynamic** as an option, you must enter a public/private address pair. Enter the public IP address you are using for static NAT and the internal IP address of the server being accessed via static NAT. Save the changes, exit to the main menu, and select Reinitialize System to put the new static NAT table into effect.

You may notice that the private IP address is mapped to itself in the example in Figure 2-9 above. This mapping was put in to work around a NAT issue with certain versions of BorderManager and VPN. If you can ping the private IP address of the BorderManager server over a VPN connection, you do not need this mapping.

---

**Note** Should you at any time delete the private interface network card setting in INETCFG and recreate it, or if you first set up the public interface before setting up the private interface, you may find you have a problem. In at least some versions of NetWare (4.11 and 5.0 have been seen to do this under various patch levels), static NAT will not retain the address pairs following a reboot. The cause is that the public interface gets loaded first, and for some reason that wipes out the static NAT settings. The cure is to go into INETCFG, remove the public interface definition and reinitialize system (possibly even reboot the server). Then go back into INETCFG and re-enter the public interface definition and bindings. An even better solution might be to simply rename the existing SYS:\ETC\NETINFO.CFG and SYS:\ETC\TCPIP.CFG files and recreate all the settings in INETCFG.

---

## Static NAT versus Reverse Proxy Acceleration

All versions of BorderManager offer Reverse Proxy Acceleration, (for HTTP), sometimes just called Acceleration. This capability can greatly enhance the performance and security of an internal web server by making it available to the Internet only through a proxy. Performance enhancement comes about by using the BorderManager HTTP caching capability to offload most/all static HTTP requests from the actual web server and serving those requests from disk/memory cache.

Static NAT simply passes the inbound traffic through BorderManager directly to the internal web server, as long as filter exceptions are in place to allow the traffic through.

Reverse proxy, if using a secondary IP address, requires a filter exception. Reverse proxy configured to use the primary public BorderManager IP address does not require a (custom) filter exception because one is added for you when the default filters are set up.

Both static NAT and reverse proxy acceleration automatically pass CGI script data through without caching the data.

Dynamic pages are usually generated using CGI (Common Gateway Interface) using languages such as PERL or ASP. Reverse proxy acceleration passes CGI script data through without caching the data. However, elements of dynamic pages, such as graphics, can still take advantage of the reverse proxy cache.

## Viewing & Capturing TCP/IP Traffic

You will almost certainly want to debug a filter exception at some point (or you just may be curious to see the actual IP traffic on a NetWare server). Currently, the best tool supplied with NetWare is a set command that allows you to see all the IP packets hitting the server in real time. The command to enable viewing of the traffic is:

```
SET TCP IP DEBUG=1
```

And the command to turn off the viewing is:

```
SET TCP IP DEBUG=0
```

You may find it convenient to make an NCF file to turn this debug on and off without having to type it out each time. For instance, set up a D1.NCF file in SYS:SYSTEM with the =1 statement and a D0.NCF file with the =0 statement. Then simply type D1 at the console to enable debugging, and D0 at the console to disable it. You could also add LOAD CONLOG MAX=100 at the beginning of D1.NCF and UNLOAD CONLOG at the end of D0.NCF to more easily capture the data changes to a log file. Remember that leaving TCP IP DEBUG=1 on for a period of time can create huge SYS:\ETC\console.log files if you do not use the MAX= parameter to

```
REM D1.NCF  
LOAD CONLOG  
SET TCP IP DEBUG=1
```

limit the size of the log file. Here are some examples:

This is a very handy method for seeing what ports and addresses are

```
REM D0.NCF  
SET TCP IP DEBUG=0  
UNLOAD CONLOG
```

in use and what is being filtered, but a production server can have so much traffic on it that it can be nearly impossible to catch the traffic of interest. Best to use this command when little or no other traffic

exists on the server than your test traffic. You may need to set up an isolated, non-production BorderManager server just for testing (always a good idea when modifying filters).

You could see a great deal of extraneous data from TCP IP DEBUG, and much of it will be normal. For example, loopback packets on NetWare 5 servers, as well as multicast traffic, are typical, and should be ignored. See the Odds & Ends section for a brief explanation of the SET FILTER DEBUG=ON statement (available on BorderManager 3.0 or later) as an alternative to SET TCP IP DEBUG=1.

## Static NAT Example Debug Trace

Here is an example of what a PING test looks like with SET TCP IP DEBUG=1 when sent out through a static NAT connection. You will see a packet going from the host (192.168.10.251) then being regenerated with a new source address. The static NAT configuration on the BorderManager server has 4.3.2.253 as the public NAT address and 192.168.10.251 as the private NAT address. The host is pinging IP address 4.3.2.100, and the trace is taken from the BorderManager server.

```
RECEIVE:pktid:38936 192.168.10.251->4.3.2.100 ttl:128 (ICMP)Echo Request  
FORWARD:pktid:38936 4.3.2.253->4.3.2.100 ttl:127 (ICMP)Echo Request
```

The originating host (192.168.10.251) sends a PING packet to 4.3.2.100. NAT regenerates the packet and forwards the packet as if it came from the public side of the static NAT address (4.3.2.253).

```
RECEIVE:pktid:38936 4.3.2.100->4.3.2.253 ttl:255 (ICMP)Echo Reply  
FORWARD:pktid:38936 4.3.2.100->192.168.10.251 ttl:254 (ICMP)Echo Reply
```

Here is the reply traffic. Host 4.3.2.100 sends its reply to 4.3.2.253, and static NAT regenerates it and forwards the packet to the NAT private address 192.168.10.251.

---

**Note** As of this writing, unpatched versions of NetWare 6.0 redirect TCP IP DEBUG data to a special screen that cannot be captured to a file. This problem should be fixed in a TCP patch later than NW6SP1 (which has TCPIP version 6.03) patch. The data is supposed to be sent to the console prompt, where it can be captured again with CONLOG. The same issue occurred with the NW51SP4.EXE patch, and was fixed in the TCP590N.EXE patch.

---

# Default BorderManager Filters

## What are Default Filters?

The default packet forwarding filters (not filter exceptions) are what stop traffic from being routed through BorderManager, whether or not you have public IP addresses internally, static NAT, dynamic NAT, whatever. The concept is simple: block all traffic TO the public interface, and block all traffic FROM the public interface.

It is important to distinguish a difference between the default filters and default filter exceptions. The default filters cover the entire public interface, while the default filter exceptions all call out the public IP address, as either source or destination. Because of this, the default exceptions do not allow traffic to or from secondary public IP addresses, while the default filters block such traffic.

Also, note that the default filters are based on the interface name, not an IP address or interface number. **If you should rename your public interface, you will no longer be filtering any packets** – until you update the default filters using BRDCFG!

The default filters do not block traffic to or from the private interface(s) - except from private to public interface. By cutting off traffic between the public and private interfaces, BorderManager controls both incoming and outgoing traffic.

Filter exceptions always override filters – you cannot override an exception with an additional filter.

## The BorderManager 3.x Default Filters

The following is a list of all of the default filters (not exceptions) set up by BRDCFG.NLM for BorderManager versions prior to 3.7 with VPN configured. Should you see additional filters using FILTCFG.NLM, you may have accidentally used BRDCFG.NLM twice - once on the public interface and once on the private interface. (You would need to delete the incorrect entries to get BorderManager to function). These filter definitions are based on the example configuration shown earlier in this book. IP Network 192.168.100.0 is the virtual IP network assigned for the VPN. The name of the interface connected to the Internet side of the BorderManager server is PUBLIC. No AppleTalk protocol was enabled on the BorderManager server, or some filters pertaining to AppleTalk would also have shown up.

### Outgoing RIP Filters:

- Filtered Route: Route to Network or Host: Network, IP address of Network/Host: 0.0.0.0, Subnetwork mask: 0.0.0.0, Do Not

Advertise Route To: Destination Type: Interface, Destination: VPTUNNEL

- Filtered Route: Route to Network or Host: Network, IP address 192.168.100.0, Subnetwork mask: 255.255.255.0, Do Not Advertise Route To: Destination type: Interface, Destination: <All Interfaces>
- Filtered Route: Route to Network or Host: Network, IP address of Network/Host: 4.0.0.0, Subnetwork mask: 255.0.0.0, Do Not Advertise Route To: Destination type: Interface, Destination: VPTUNNEL
- Filtered Route: Route to Network or Host: Network, IP address of Network/Host: 4.3.2.0, Subnetwork mask: 255.255.255.0, Do Not Advertise Route To: Destination type: Interface, Destination: VPTUNNEL

#### **Incoming RIP Filters**

- Filtered Route: Route to Network or Host: All Routes, IP address of Network/Host: <blank>, Subnetwork mask: <blank>, Do Not Accept Route From: Source Type: Interface, Source: PUBLIC

#### **Outgoing EGP Filters:**

- Filtered Route: Route to Network or Host: All Routes, IP address of Network/Host: <blank>, Subnetwork mask: <blank>, Do Not Advertise Route To: Destination Type: Interface, Destination: PUBLIC

#### **Incoming EGP Filters**

- Filtered Route: Route to Network or Host: All Routes, IP address of Network/Host: <blank>, Subnetwork mask: <blank>, Do Not Accept Route From: Source Type: Interface, Source: PUBLIC

#### **OSPF External Route Filters**

Routes denied: All Routes

## The Default Filtering Action – All Versions of BorderManager

It is important that your filters are set up to 'Deny Packets in Filter List', so that the filters block traffic and the exceptions allow traffic.

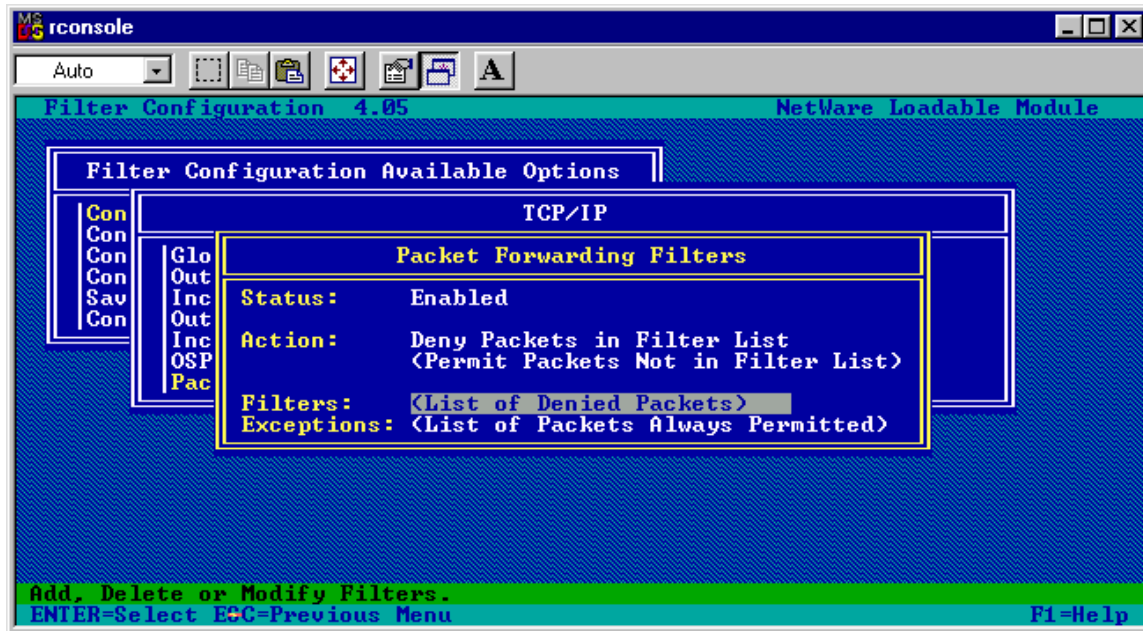


Figure 2-10 - FILTCFG - Deny Packets in Filter List

The screenshot shown in Figure 2-10 is how you should have your filters configured.

---

**Note** As of this writing, Novell has a patch for BorderManager 3.7 FILTSRV.NLM that will block all traffic on an interface defined as Public in FILTCFG if NDS is not available when the filters are initialized. This action is necessary, because if NDS is not available for some reason, the filters cannot be read, and all traffic would be allowed.

---

## FILTCFG Examples of the Default Filters

The following screenshots show what the default filters should look like. It is extremely important that the interface name called out on your BorderManager server matches the current interface name shown in INETCFG. In my servers, I immediately delete the old interface names and reconfigure them using PUBLIC and PRIVATE. (PRIVATE1 and PRIVATE2 if I happen to have a server with multiple private IP addresses). Not only does this make it much simpler when adding custom filter exceptions, it also is more flexible in that I can replace a network card and not be tied to its old name, such as PRO100\_1 or 3C9X5\_1. I can also copy the FILTERS.CFG file from one server to another without having to make changes, except where public IP addresses are called out.

**Note** You cannot simply change the FILTERS.CFG file between servers with BorderManager 3.7 since the IP filtering information is stored in NDS. You can make the changes described here, but afterwards you need to delete the old filter exceptions in NDS, and repeat the FILSRV MIGRATE process.

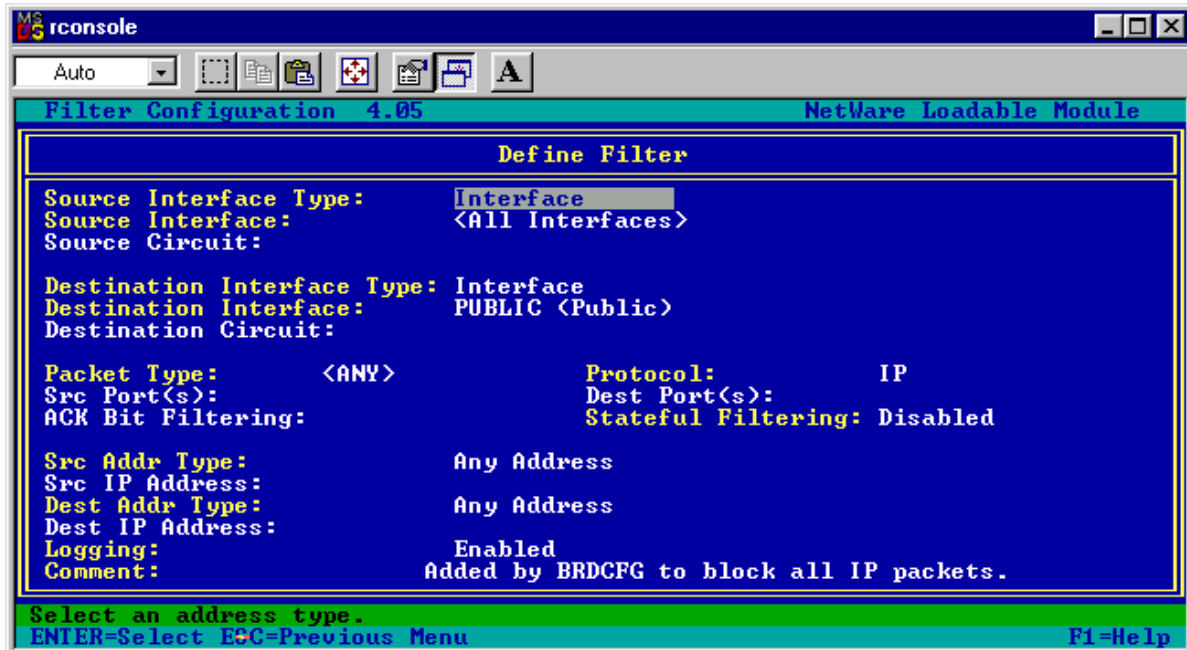


Figure 2-11 - FILTCFG - Default Filter Blocking all IP Traffic to the Public Interface

The default filter shown in Figure 2-11 blocks all traffic to the public interface, whether it comes from the Internet or a private IP address. Without filter exceptions, BorderManager cannot receive any traffic from the Internet.





Figure 2-12 - FILTCFG - Default Filter Blocking all IP Traffic from the Public Interface

The default filter shown in Figure 2-12 blocks all traffic from the public interface. Without additional filter exceptions, the proxy cannot send any traffic to the Internet.

## Default Filter Exceptions

With all traffic blocked both to and from the public interface, you might as well not have an Internet connection at all – no traffic would move in or out of the server on that interface. It is necessary to allow at least some traffic in order to have any connectivity to the Internet. The BorderManager proxies, gateways and VPN will not function without having some filter exceptions in place that override the filters for selected ports and protocols.

BorderManager 3.7 is significantly different in how the exceptions are handled by default. BorderManager 3.7 is covered after the sections on previous versions of BorderManager.

### BorderManager 3.0, 3.5 and 3.6 Exceptions

When you install BorderManager versions prior to 3.7, one of the steps involved is to run the BRDCFG.NLM to set up the default filters, and several filter exceptions. By design, the default filters block **all traffic to and from** the public IP address on the BorderManager server, (and you have to specify the address). BRDCFG sets up default filter exceptions to allow the BorderManager proxies to work, as well as VPN connections.

You can always add the default filters and default filter exceptions back into the BorderManager server by running BRDCFG again, for versions prior to 3.7. It will only add filters and filter exceptions, and it will not delete any filters or exceptions that already exists. If you have set up a filter exception to allow too much traffic through, adding back the default filters will not modify your exception, and your server will still not be secure.

BorderManager 3.0, 3.5 and 3.6 have following default filter exceptions, designed to allow the proxy services and VPN to function. These exceptions are put into place when the BRDCFG.NLM program is run at the server console. These are the filter exceptions as shown in FILTCFG. Each is shown in the following section as well as described here.

1. Allow all outbound IP packets from the BorderManager public IP address to the public interface.
2. Allow all inbound dynamic TCP ports (1024-65535) from the public interface to the public IP address of the BorderManager server.
3. Allow all inbound dynamic UDP ports (1024-65535) from the public interface to the public IP address of the BorderManager server.

4. Allow TCP port 213 from the public interface to the public IP address of the BorderManager server in order to allow VPN server-server communications.
5. Allow TCP port 353 from the public interface to the public IP address of the BorderManager server in order to allow VPN client authentication to the server.
6. Allow UDP port 353 from the public interface to the public IP address of the BorderManager server in order to allow VPN client to send periodic keep-alive packets to the server.
7. Allow the SKIP protocol (protocol 57) from the public interface to the public IP address of the BorderManager server. The SKIP protocol is necessary for Novell VPN to function.
8. Allow TCP port 80 (HTTP) traffic from the public interface to the BorderManager public IP address in order for the web server accelerator to function.
9. Allow TCP port 443 (HTTPS/SSL) traffic from the public interface to the BorderManager public IP address in order for proxy authentication to a reverse web proxy accelerator to function.

### **FILTCFG Examples of the BorderManager 3.0, 3.5 and 3.6 Default Filter Exceptions**

A careful study of the default filters shows that all IP traffic is blocked between the public interface and all other interfaces with two filters. Traffic is denied FROM the public interface to other interfaces and traffic is denied from all interfaces TO the PUBLIC interface. In both cases, the default filters apply to any IP address. These default filters have the effect not only of cutting off traffic between the public and private interfaces, but also of cutting off both incoming and outgoing traffic from the public interface to or from any external IP address. Certain exceptions are required in order for the BorderManager proxies to function as well as to allow the BorderManager VPN to function. Without these exceptions, one would be forced to manually add specific (or general) filter exceptions in order for the proxies to work.

The following graphics show what FILTCFG should show for the default filter exceptions that are produced by BRDCFG.NLM in BorderManager 3.x.

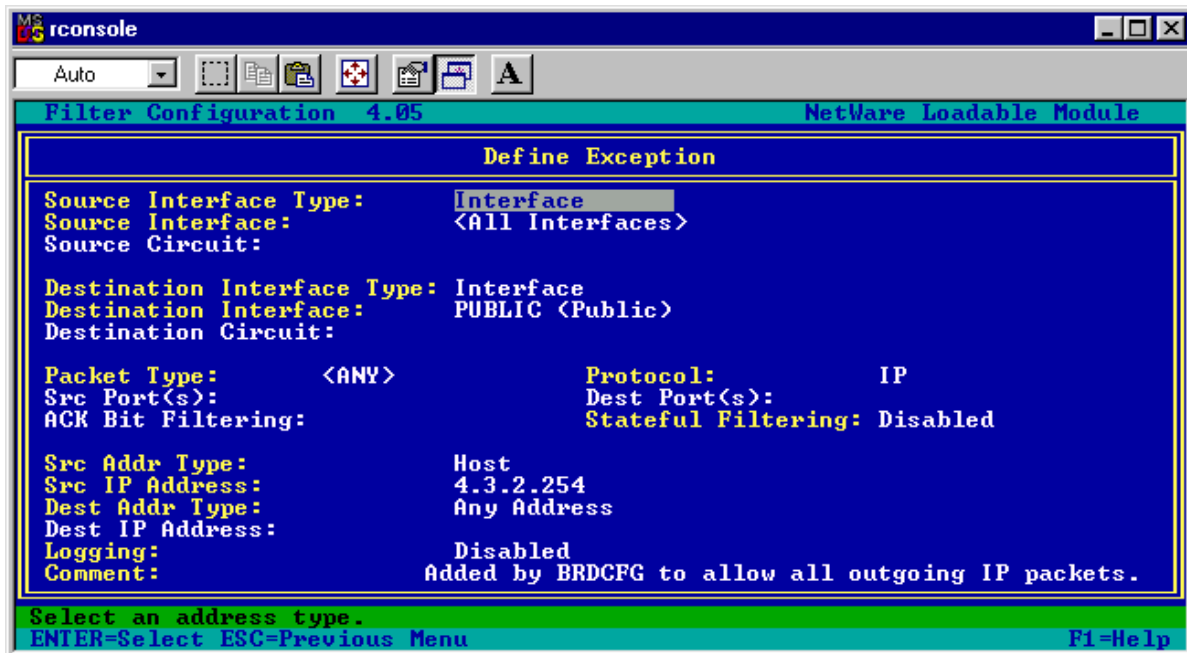


Figure 2-13 - FILTCFG - Default Filter Exception Allowing all Outbound IP Traffic from the Public IP Address

The filter exception shown in Figure 2-13 allows **all** outgoing IP packets from the **public IP address** of the BorderManager server. In some cases, this may allow more traffic out than desired, such as SLP packets that can bring up an ISDN dial-up link. Without this exception, the BorderManager proxies would not be able to send any packets out.

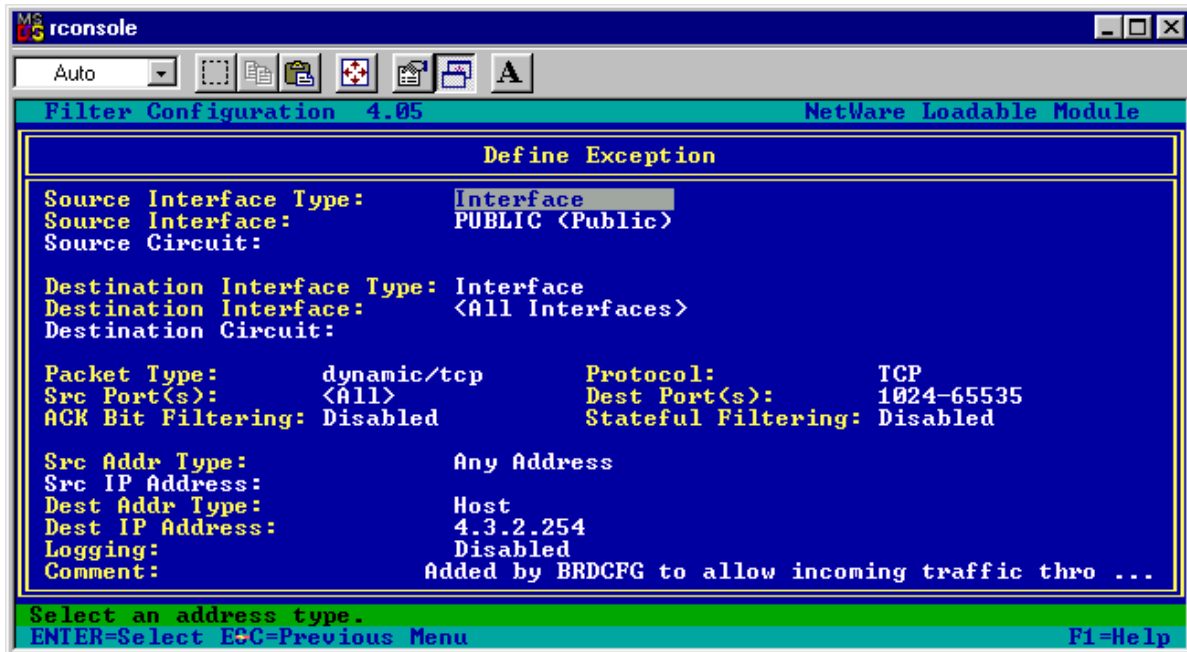


Figure 2-14 - FILTCFG - Default Filter Exception Allowing Dynamic TCP to the Public IP Address

The default filter exception shown in Figure 2-14 allows incoming TCP traffic with a destination port of the 'high' TCP port numbers into the public IP address of the BorderManager server. Without this filter exception, the BorderManager proxies would not see a response to their TCP requests.

---

**Note** This default exception is probably the single biggest security hole on a typical BorderManager server. It allows inbound traffic to certain services that might be listening on the public IP address. See the chapter on advanced topics later in this book for ways to deal with this issue.

---

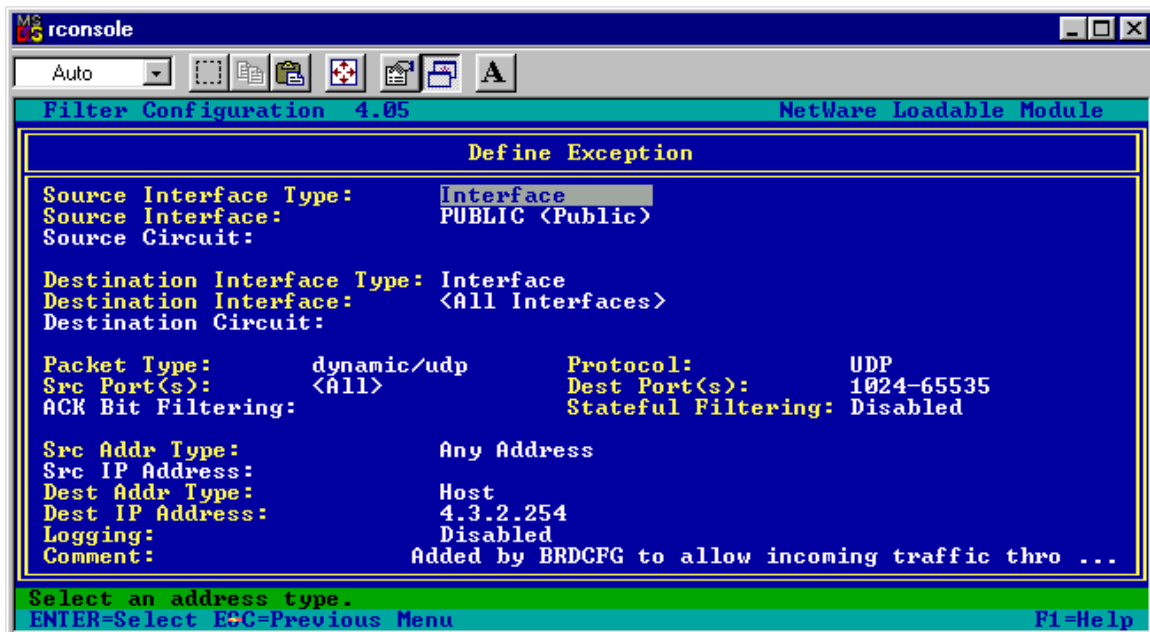


Figure 2-15 - FILTCFG - Default Filter Exception Allowing Dynamic UDP to the Public IP Address

The default filter exception shown in Figure 2-15 allows incoming UDP traffic with a destination port of the 'high' UDP port numbers into the public IP address of the BorderManager server. Without this filter exception, the BorderManager proxies would not see a response to their UDP requests

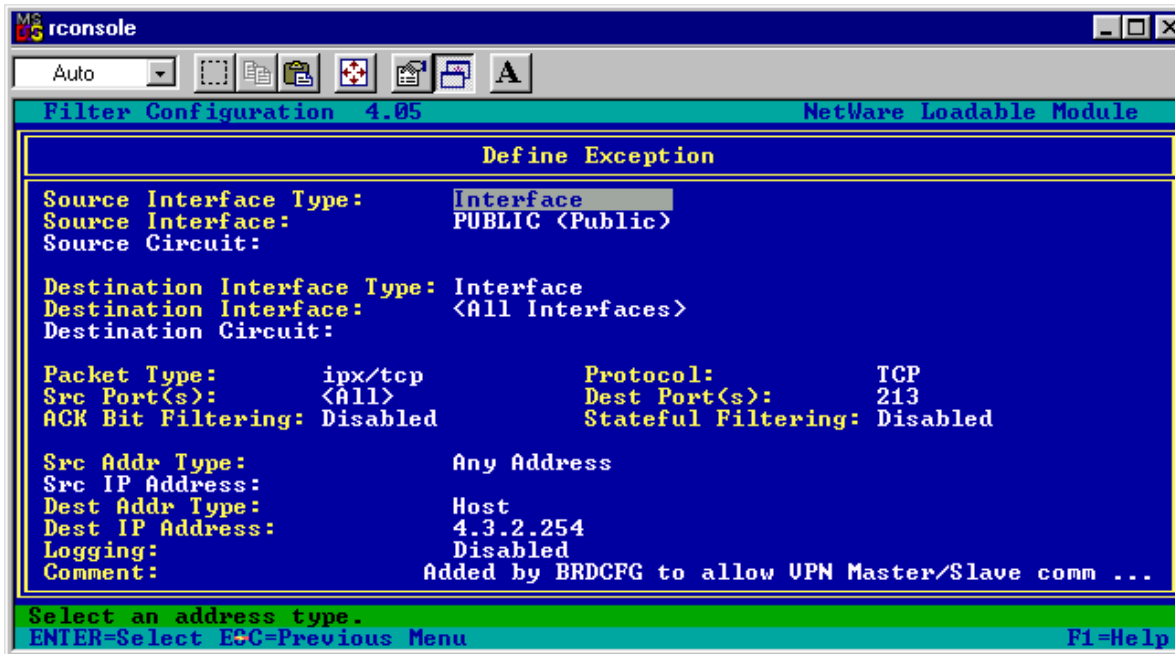


Figure 2-16 - FILTCFG - Default Filter Exception Allowing VPN Master/Slave Traffic to the Public IP Address

The default filter exception shown in Figure 2-16, and most of the following, is used to allow VPN communications to the public IP address of the BorderManager server. In this case, TCP destination port 213 is allowed inbound for VPN Master/Slave communications.

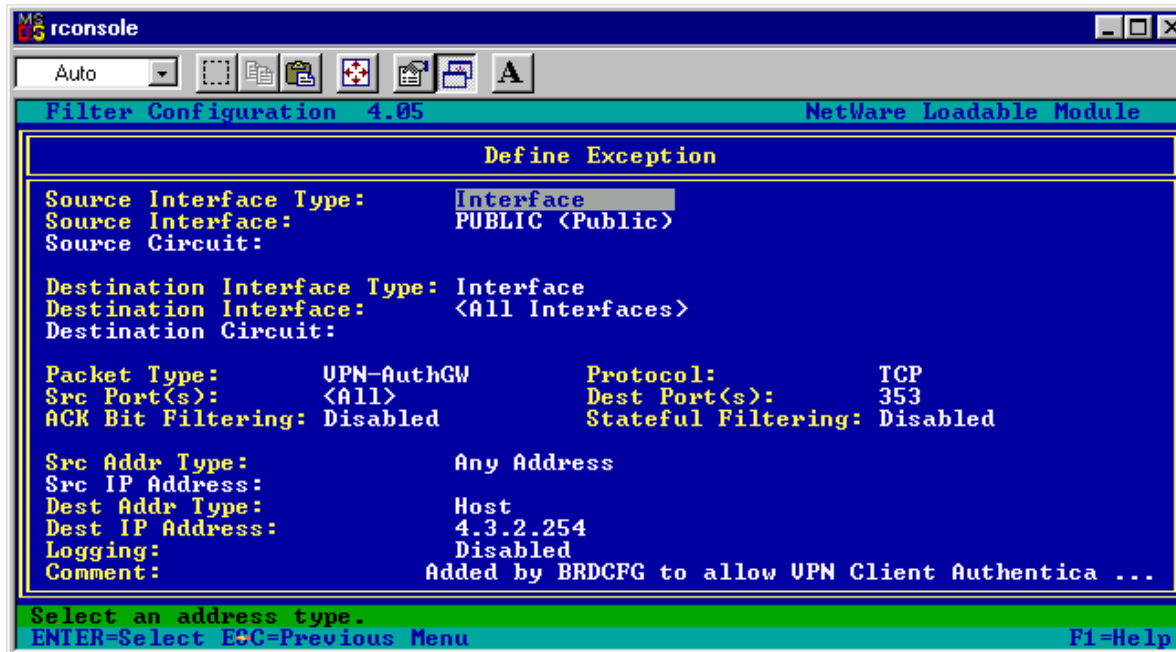


Figure 2-17 - FILTCFG - Default Filter Exception Allowing VPN Client Authentication to the Public IP Address

The default filter exception shown in Figure 2-17 is used to allow a VPN client to authenticate to the BorderManager VPN server using TCP destination port 353.



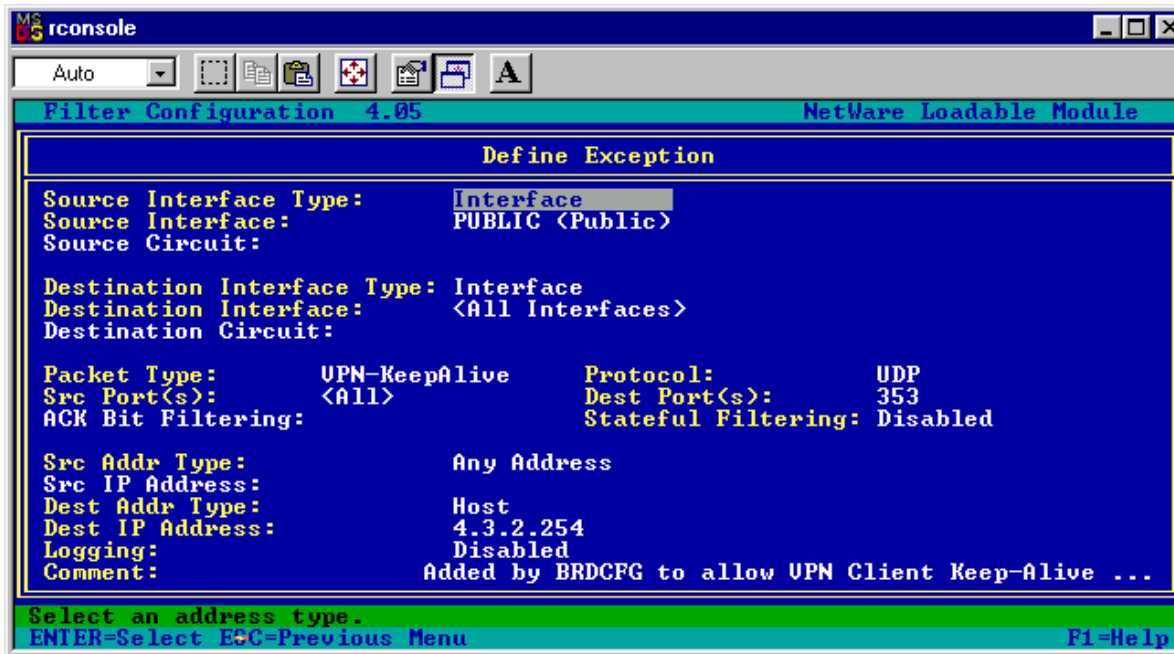


Figure 2-18 - FILTCFG - Default Filter Exception Allowing VPN Client Keep-Alive Traffic to the Public IP Address

The default filter exception shown in Figure 2-18 allows a VPN client to send periodic keep-alive packets to the VPN server using UDP destination port 353 so that the VPN server realizes that the VPN client is still connected.

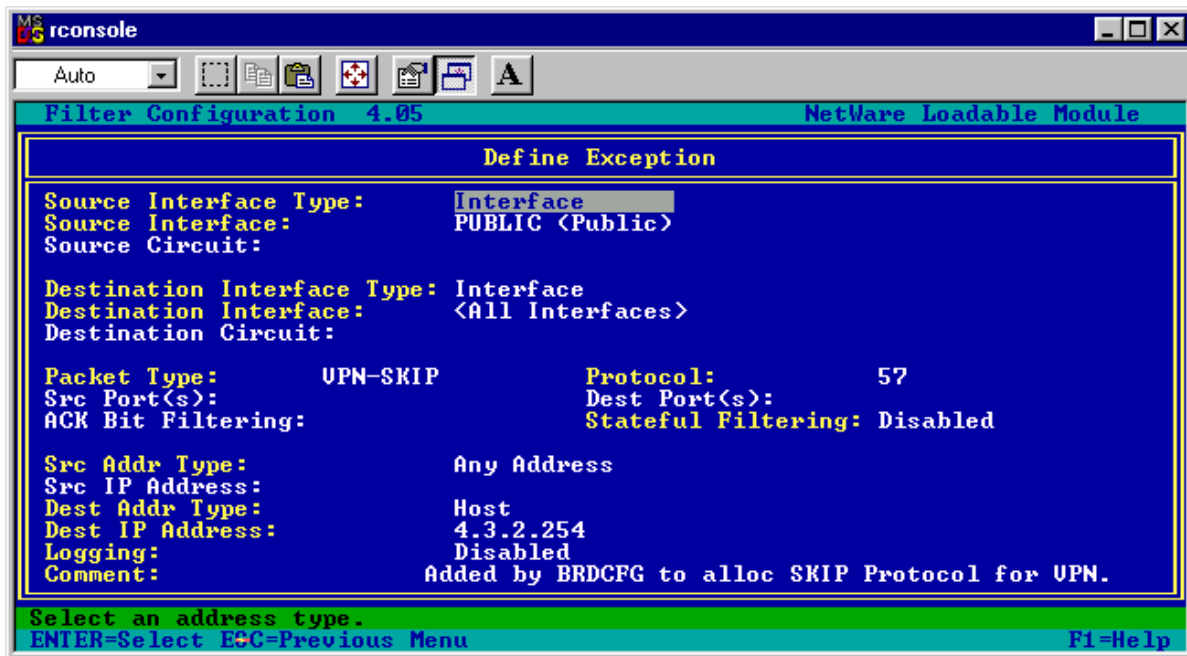


Figure 2-19 - FILTCFG - Default Filter Exception Allowing SKIP Protocol to the Public IP Address

The default filter exception shown in Figure 2-19 allows the VPN SKIP protocol inbound to the BorderManager VPN server public IP address.

---

**Note** SKIP is neither TCP nor UDP, but simply another protocol with protocol ID 57. The protocol ID is a field in the IP header of a packet, and unlike TCP (which has protocol ID 6) or UDP (which has protocol ID 17), SKIP has protocol ID 57 which identifies it for a packet filtering router.

---

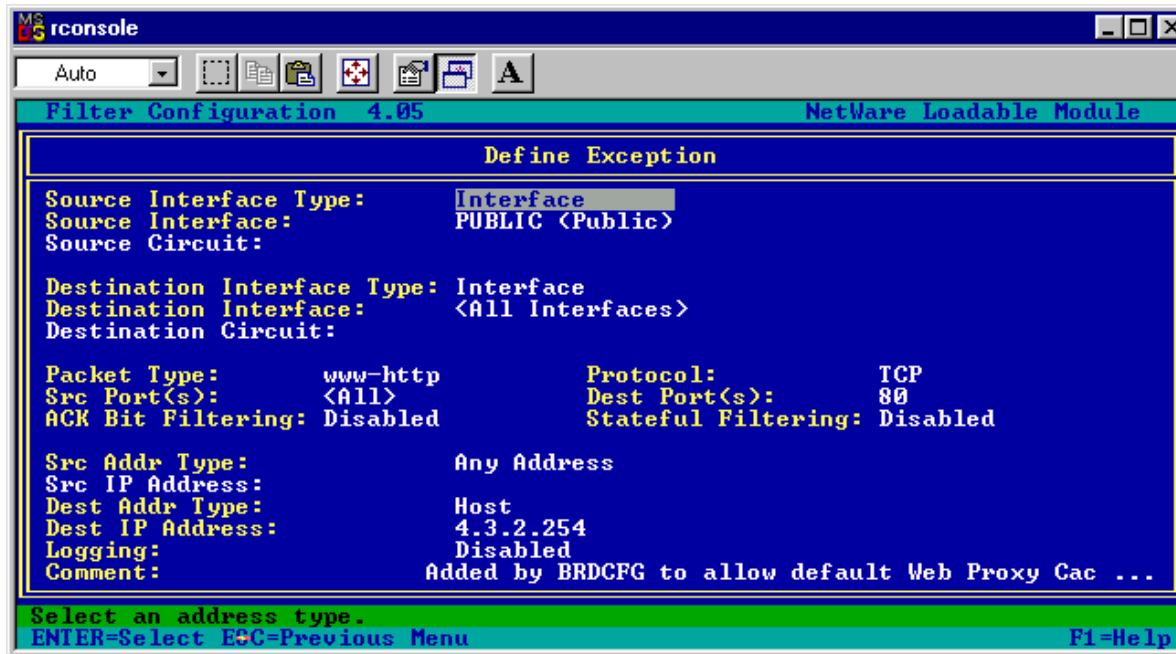


Figure 2-20 - FILTCFG - Default Filter Exception Allowing Reverse Proxy HTTP Traffic to the Public IP Address

The default filter exception shown in Figure 2-20 allows TCP destination port 80 (HTTP) traffic to flow from the public interface to the BorderManager public IP address in order for a reverse HTTP Proxy to function. You will not have a reverse HTTP Proxy set up by default, but this exception allows you to configure one on the public IP address without having to add a filter exception.

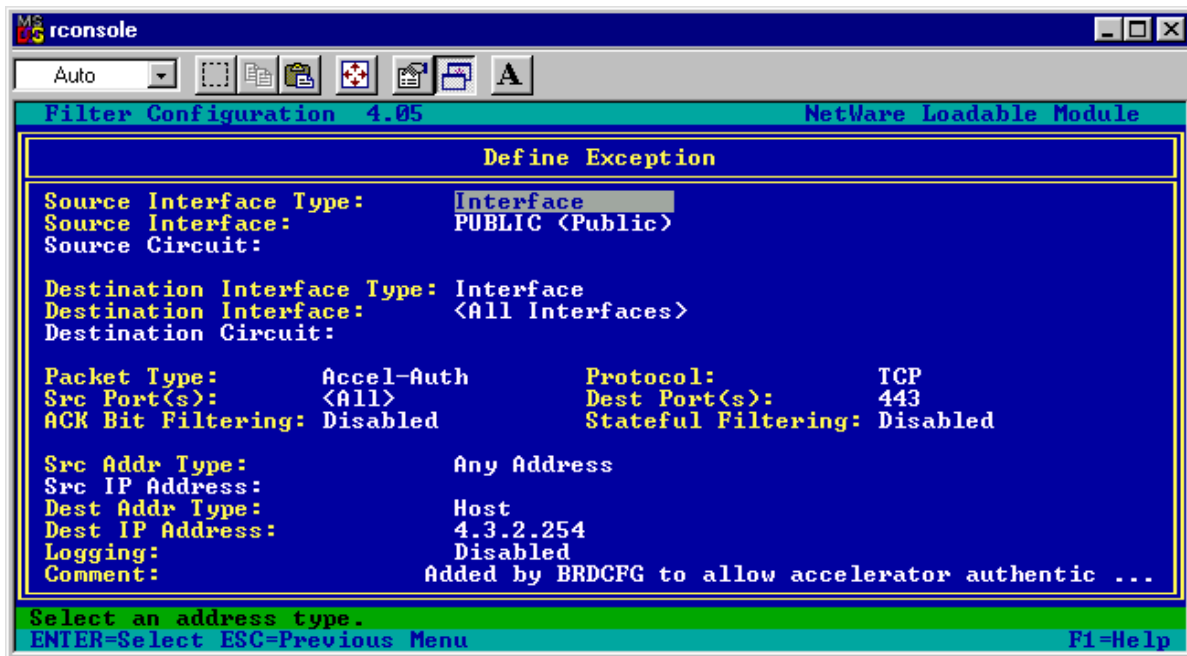


Figure 2-21 - FILTCFG - Default Filter Exception Allowing HTTPS (SSL) Traffic to the Public IP Address

The default filter exception shown in Figure 2-21 allows TCP destination port 443 (HTTPS/SSL) from the public interface to the BorderManager public IP address to allow proxy authentication to function for a reverse proxy. (It also allows any other inbound SSL/HTTPS traffic to the public IP address). You do not have a reverse HTTP proxy configured by default, but if you should add one on the public IP address, and require proxy authentication on it, this filter exception will allow it to work without you having to do more work.

## BorderManager 3.7 Default Filter Exceptions

Novell's default filter exceptions for BorderManager 3.7 are quite different from previous versions, and only include those exceptions necessary to allow outbound traffic from the proxies. Novell took this stance in order to increase the security of a default BorderManager installation.

A quick summary of the points described below:

1. If you install BorderManager 3.7 over an existing BorderManager installation, the old filters and exceptions will be used.
2. If you make a fresh installation of BorderManager 3.7 on a new server, and do not select any proxies when installing, you will not have any filters or exceptions.
3. If you make a fresh installation of BorderManager 3.7 on a new server, and select any of the proxies, you will have filters, and stateful outbound filter exceptions for all of the proxies.

Prior to BorderManager 3.7, BRDCFG.NLM was used to create a known, consistent set of filter exceptions that allowed all outbound traffic for all proxies, inbound traffic for a reverse HTTP proxy, and VPN. With BorderManager 3.7, only the basic proxy exceptions and a limited number of VPN exceptions will be added. That part of the installation routine only runs if no previous version of BorderManager has been installed on the server.

If you are upgrading a previous BorderManager server, the existing filters and exceptions will not be changed, and you will need to migrate them into NDS with the FILTSRV MIGRATE process.

---

**Note** This section assumes that you have a server with at least two network interface cards. Setting up a server with a single interface (as in a dedicated HTTP Proxy server behind an existing firewall) will not see an option to select a proxy, nor set any default filter exceptions. You should not enable filtering on a single interface server anyway.

---

When installing BorderManager 3.7, and not upgrading a previous version in place, you will have the option to select proxies to be used. Regardless of the proxies you select, the BorderManager 3.7 installation will configure a stateful filter exception to allow basic outbound access for every proxy. No inbound traffic will be allowed – if you intend to configure a reverse HTTP proxy, you must set up your own exceptions after the BorderManager installation is completed. Examples of these exceptions are shown later in this book.

VPN default filter exceptions in the past relied on the BRDCFG filter exceptions for outbound requests, and responses. Because the default

VPN exceptions have not made allowance for the lack of these default exceptions, you will have to add a number of additional exceptions yourself. Examples of these exceptions are shown later in this book.

It is very important that you understand how BorderManager 3.7 handles exceptions as they are quite different from previous versions!

## Filter Exceptions for Proxies You Might Have in BorderManager 3.7

---

**Note** The HTTP Proxy may require additional custom exceptions for every non-standard port number used by a web site that your users encounter on the Internet. Only ports 80 and 443 will be configured for outbound traffic for the HTTP Proxy by the BorderManager 3.7 installation routine.

---

Here is a list of the filter exceptions that are created by a fresh installation of BorderManager 3.7. The source IP address in each case is the BorderManager public IP address. The source interface is also set to the public interface.

- HTTP Proxy – TCP destination ports 80, 443
- FTP Proxy - TCP destination port 21, with port & pasv capabilities enabled
- DNS Proxy – TCP and UDP destination port 53
- News Proxy – TCP destination port 119
- RealAudio & RTSP Proxy – TCP destination ports 7070 & 554
- Mail Proxy – TCP destination port 25
- Transparent Telnet Proxy – TCP destination port 23
- Transparent HTTP Proxy – TCP destination port 80

An example of each of these exceptions is shown in the chapter showing outbound filter exceptions.

Inbound exceptions are not created for any of the proxies, and so you must manually create your own exceptions in every case (except VPN), unlike previous versions of BorderManager, which created exceptions for reverse HTTP Proxy AND allowed all inbound high ports by default. Examples for inbound exceptions to proxies are shown in this book, and will be almost the same as in previous versions of BorderManager when configuring exceptions for proxies listening on secondary IP addresses.

---

**Note** As of this writing, Novell is planning on releasing an updated version of BRDCFG.NLM for BorderManager 3.7 that will add some inbound filter exceptions for certain proxies.

---

## Security Considerations

This book shows how to set up specific filter exceptions for various software programs to operate in either an outbound or an inbound direction. This book does not delve deeply into the security aspects of setting up these exceptions. In general, the more exceptions that are allowed, particularly inbound, the more risk one must assume for a break-in or a denial of service attack. Packet filters can be effective in stopping many attempts at compromising the security of a network, but they may not stop all attacks. Use of the BorderManager proxy services is more secure than using packet filter exceptions to do the same function. However, stateful filter exceptions for outbound traffic, available in BorderManager 3.0 and later versions, are very secure, and should not be cause for major worry.

It is always a good idea for the network administrator to monitor Internet sites related to computer security, and keep a close eye on the Novell Minimum Patch list for bug fixes. Some sites of interest are:

<http://www.cert.org/>

<http://www.nessus.org/>

<http://www.iss.net/>

<http://www.rootshell.org/>

<http://www.icsa.net/>

The default filters applied to BorderManager by the BRDCFG.NLM program do not restrict IP packets from being sent out from the public interface. That is, if the server itself generates outgoing traffic on the public interface, it will be sent out. For NetWare 5.0, this includes SLP multicasts. The most security-conscious administrator may want to consider deleting the default filter exceptions and manually implementing exceptions allowing only specific outbound traffic as needed. (See the chapter on advanced topics). This does not mean that **inbound** IP traffic is allowed, only that in some cases NetWare may be advertising its presence unnecessarily on the public LAN side.

---

**CAUTION**      *DISCLAIMER! The author has written this book with the best of intentions and has done testing and proofreading to find typographical errors. The filter exceptions given in this book should be workable, with minimal security impact, given the technology available in the version of BorderManager used. However, there are no guarantees that a filter exception or setting shown here does not provide some means for an intrusion or denial of service attack. On the contrary, each filter exception used may decrease the security of a network. You must make a tradeoff between functionality and security. You are warned to use caution, common sense and firewall analysis techniques and tools to secure your network. This book is provided 'as-is'. The author is not responsible for any losses, network intrusions, or other problems resulting from using the advice or examples in this book, whether such problems are caused by typographical errors, or mistakes on the part of the author. In short - check your work carefully, and do not rely 100% on this book!*

---



# Chapter 3 - NetWare Tools Used in Filtering

---

Several NLM's that ship with NetWare are commonly used in helping you to set up filters or filter exceptions. Some simply allow you to see what is happening, while others help you to make changes to the configuration. The following utilities are very useful or essential to working with BorderManager packet filtering.

## iManager

Actually, iManager is not a file, but a complete web-based management system based on the Apache web server running Java and XML code. With BorderManager 3.7, you can use iManager to add, modify or delete IP filters and exceptions. You can run iManager only from a NetWare 6.0 or later server, and you do not have to be running iManager from the BorderManager server.

## FILTSRV.NLM

FILTSRV.NLM is a module that works in conjunction with IPFLT31.NLM to provide filtering services. For BorderManager versions prior to 3.7, you simply don't worry about it. For BorderManager 3.7, there are some FILTSRV command line options you need to use in order to migrate data into NDS, and to back up filters to a text file.

## BRDCFG.NLM

For versions prior to 3.7, when you first set up BorderManager, you are asked at one point if you want to set up the default filters to block all traffic to the public IP address. **This is essential to set up BorderManager as a secure firewall!** If you ever need to add the default filters again, just LOAD BRDCFG at the file server console and follow the prompts. To reset your server to have ONLY the default filters, you should use FILTCFG to delete every filter and filter exception entered, and then run BRDCFG as it will not delete exceptions already present.

---

**CAUTION** *If you accidentally apply the default filters to the private (internal) IP address, you must manually delete the filters and filter exceptions that are configured or BorderManager will not function. Running the BRDCFG program will not remove those filters, and all your traffic will be blocked.*

---

## CONFIG (Not CONFIG.NLM)

Typing **CONFIG** at the server console will show the configured LAN interfaces and addresses (and default route). It is a quick way to see what is set up on the server. It does not show secondary IP addresses or additional non-secondary IP addresses bound to an interface. (Typing **LOAD CONFIG** or **LOAD CONFIG /S** at the console produces a `SYS:SYSTEM/CONFIG.TXT` file that helps to document your server).

## CONLOG.NLM

CONLOG is used to capture all traffic on the server console to a text file. **LOAD CONLOG** starts saving data to a file in `SYS:ETC` called `CONSOLE.LOG`. Unloading CONLOG stops the capture and allows you to view/edit the text file. `CONSOLE.LOG` is very useful in conjunction with `SET TCP IP DEBUG=1` to capture IP packets when you are testing filter exceptions.

---

**Note** As of this writing, unpatched versions of NetWare 6.0 redirect TCP IP DEBUG data to a special screen that cannot be captured to a file. This problem should be fixed in NW6SP1. The data is supposed to be captured to the Logger screen, which can be saved to a file on the DOS partition with the F2 key.

---

## FILTCFG.NLM

To view, export or configure filters and exceptions, type **LOAD FILTCFG** at the server console.

---

**Note** Until BorderManager 3.7, the filters and filter exceptions are stored in the `SYS:\ETC\FILTERS.CFG` file. If you want to experiment with creating new filter exceptions, it is a good idea to make a backup copy of this file first. The filters are also saved in server memory, until a server reboot, and creating a new filter exception will bring back the old filter definitions if you accidentally delete the `FILTERS.CFG` file.

---

FILTCFG.NLM utility has a feature called **Configure Interface Options** that allows you to define one of the interfaces as public and another as private. Once you do this, the words (public) and (private) are added to the network interface names when applying filters to help you remember which interface is which.

If the interface names are incorrect, select an interface, and press the Tab key to toggle the title 'Public' or 'Private' as needed.

---

**Note** With BorderManager 3.7, it can be critical to correctly define the interfaces as Public or Private. All data to and from the Public interface as defined here will be blocked if NDS is not available when the filters are initialized.

---

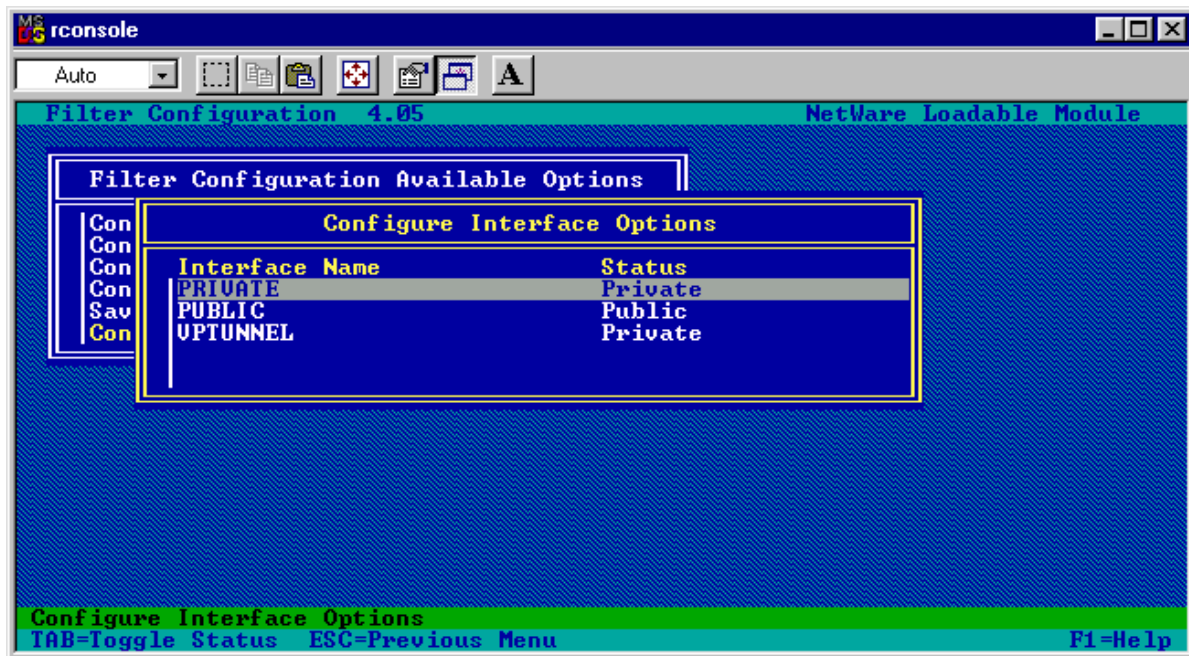


Figure 3-1 - FILTCFG - Configure Interface Options

The screenshot in Figure 3-1 shows FILTCFG.NLM on a BorderManager 3.6 server being used to conveniently define interfaces as Public or Private so that they are more easily identified when setting up filters and filter exceptions. *Even better is to name the interfaces as PUBLIC and PRIVATE when setting up the server!*

## IPFLT.NLM / IPFLT31.NLM

IPFLT31.NLM is the module used to perform IP packet filtering. It is automatically loaded by IPFLT.NLM if filtering is enabled in INETCFG for IP. If you wish to see immediately if a communication problem is being caused by IP filtering, type **UNLOAD IPFLT** at the server console to disable all IP packet filtering. If communications start working, then you have a filtering issue; if not – you have at least some other issue (and may still have a filtering issue as well). **Unloading IPFLT opens up your BorderManager server completely to hacking, so do this only as a quick test.** After testing, remember to **LOAD IPFLT** again.

## SET TCP IP DEBUG=1

To view all IP packets hitting the server, type **SET TCP IP DEBUG=1** at the server console.

To stop viewing all IP packets, type **SET TCP IP DEBUG=0** at the server console.

## SET FILTER DEBUG=ON

This command, which works at the server console on BorderManager 3.x, brings up a menu of various TCP filter debugging commands which can then be used to see a much more specific display of filtering activity than the SET TCP IP DEBUG=1 command. On the other hand, you don't see all the other things going on with IP packets that the TCP IP DEBUG command can show. It is most useful when trying to see specific filter discards on a production server where there is so much traffic zooming by that you cannot easily find the packets of interest if using TCP IP DEBUG. Use SET FILTER DEBUG=OFF to disable the filter debug function. See Figure 4-4 in the next chapter for an example of the filter debug output.

## TCPCON.NLM

While not actually a filtering tool, this utility does let you view IP routing information. Problems thought to be filter-related often end up being routing issues, such as lacking a proper default route. Type LOAD TCPCON at the file server console to start this utility.

Some useful features of TCPCON:

- View the current routing table
- See what ports on the server are 'open'. (Some service is listening on the port). Look in Protocol Information, TCP, TCP Connections. Similar for UDP.
- View the ARP table. Look in Protocol Information, IP, IP Address Translations.

## VPNCFG.NLM

VPNCFG is used at the server console to begin configuring VPN services. One of the options in VPNCFG is to create the filter exceptions required for both Site-to-Site and Client-to-Site VPN's to function.

The default filter exceptions created with VPNCFG in BorderManager 3.7 do not create the exception needed for Client-to-Site VPN over NAT to function. Nor do the filters necessary for the response traffic get created, or even the VPN Site-to-Site communications. You have to create the exception yourself.

# Chapter 4 - Working with Filters

---

This book would be far less useful without real-world examples to view. The bulk of the remainder of this book provides various examples that can be used easily by anyone with only an address or interface change.

## Backing Up and Restoring Filters and Exceptions Prior to BorderManager 3.7

Until BorderManager 3.7, all filters and exceptions are stored in the SYS:\ETC\FILTERS.CFG file. All custom filter definitions (service types) are also stored in that file. However, the definitions for the built-in filter exceptions supplied by NetWare are stored in the SYS:\ETC\BUILTINS.CFG file. With rare exception, one of which is shown in the Advanced chapter, you will not need to modify the BUILTINS.CFG file. However, it is safest to treat both the FILTERS.CFG and BUILTINS.CFG files as a matched pair.

Before making changes to the filters or filter exceptions, make a backup copy of the FILTERS.CFG and BUILTINS.CFG file. Should you need to put those filters and exceptions back in place, use the following procedure:

1. UNLOAD IPFLT (disables IP packet filtering)
2. Copy back the FILTERS.CFG and BUILTINS.CFG files to SYS:ETC.
3. REINITIALIZE SYSTEM (assuming you have filtering enabled in INETCFG).

## Backing Up and Restoring BorderManager 3.7 IP Filters and Exceptions

Because BorderManager 3.7 stores IP filtering information in NDS, you cannot simply make copies of the BUILTINS.CFG and FILTERS.CFG files. You must also export the NDS filters to a text file. Once you have the filters backed up in a text file, you can re-import them into NDS with the FILTSRV MIGRATE command.

As of this writing, Novell is to provide a patched version of FILTSRV that will provide the function shown below. This patched

version might be on the International version of the BorderManager 3.7 CD, but it was not on the US version that was released on April 17, 2002.

The FILTSRV\_BACKUP\_FILTERS command should be available with FILTSRV version 1.60i, April 11, 2002, or later.

FILTSRV\_BACKUP\_FILTERS TEST.TXT

Type the command FILTSRV\_BACKUP\_FILTERS at the server console while FILTSRV is loaded, and the filters will be exported to a text file in SYS:ETC called FILTERS.BAK. If you add a filename to the end of the command, the filters will be exported to a file by that name. The file will always be located in the SYS:ETC directory.

In order to restore the filters, copy the backup file to SYS:\ETC\FILTERS.CFG, unload IPFLT, unload IPXFLT, and Reinitialize System. The filtering information in FILTERS.CFG will be added back into NDS. Note that if you add a filter exception with iManager, you specify an NDS name for the filter object. However, if you delete that object, and then restore it with FILTSRV MIGRATE, a new, generic name will be created for the NDS object. (In case you are looking at the NDS objects in ConsoleOne, NWADMN32, etc.)

## The FILTSRV MIGRATE Procedure

You must perform a FILTSRV MIGRATE procedure after you install BorderManager 3.7 before you see any filters or filter exceptions in FILTCFG or iManager. You may also need to perform the FILTSRV MIGRATE procedure in order to fix filtering issues, by manually deleting the NDS objects in the NBMRuleContainer with NWADMN32 or ConsoleOne.

The FILTSRV MIGRATE procedure relies on having a good SYS:\ETC\FILTERS.CFG file as the information in that file will be migrated into NDS. Once the data is migrated into NDS, FILTCFG or iManager should be able to see the filters and filter exceptions.

It is essential that FILTERS.CFG is correct. I suggest you back up FILTERS.CFG each time you have modified filters in case you need to remigrate the filters later.

1. **UNLOAD IPXFLT**
2. **UNLOAD IPFLT.** Both IPFLT.NLM and IPFLT31.NLM should unload. If IPFLT31 does not unload automatically, you need to patch your server, but you can work around the issue for now by disabling filtering in INETCFG, Protocols, TCP/IP and Reinitialize System. Do not force FILTSRV to unload if IPFLT31 is not unloaded or you will ABEND the server.
3. **UNLOAD FILTSRV**
4. **LOAD FILTSRV MIGRATE.** There should be a slight delay as filtering information is written into NDS. There should be no NDS errors on the server console or Logger screen if the procedure was successful!
5. **UNLOAD FILTSRV**
6. **LOAD FILTSRV**
7. **REINITIALIZE SYSTEM.** Be sure IP and IPX filtering is enabled in INETCFG.
8. Use FILTCFG to check that your filters and exceptions look correct. If everything looks good, back up the FILTERS.CFG file now.



## Viewing Filters in Action (TCP IP DEBUG)

Here is an example of some filtering being applied to packets at a BorderManager server. The data was viewed using SET TCP IP DEBUG=1 and captured using CONLOG.NLM.

```
LOCAL:pktid:39517 4.3.2.100->4.3.2.255 ttl:128 (UDP)
UDP:Source Port:137(NETBIOS-NS) Destination Port:137(NETBIOS-NS)

Discard Incoming: cause(FILTERING), reason(5)
```

---

**Note** See Novell TID 2953403 for some explanation of the reason codes for filtering in the TCP IP DEBUG trace. Reason 5 simply means the data was discarded due to a filter.

---

The above example shows a local broadcast from a PC with a host address of 4.3.2.100. (4.3.2.255 is the broadcast address for that subnet). The source and destination ports are 137, and the type of packet is a NETBIOS name search request resulting from having the Microsoft Client for Microsoft Networks installed on the PC without it using WINS to locate services. The packet was filtered as it came into the public interface on the BorderManager server ('Discard Incoming').

## TCP DEBUG PING & DNS Example

Following are some TCP IP DEBUG traces captured with CONLOG.NLM showing what happens with default filters enabled when trying to ping WWW.NOVELL.COM. This configuration involved using dynamic NAT on BorderManager and no proxies. The workstation at 192.168.10.114 was configured with a default gateway pointing to the BorderManager private IP address, and a DNS server entry of 199.182.120.203.

```
RECEIVE:pktid:162 192.168.10.114->199.182.120.203 ttl:128 (UDP)
UDP:Source Port:1034 Destination Port:53(DOMAIN)

Discard Outgoing: cause(FILTERING), reason(1)
```

The first thing that happened was that PING needed to resolve WWW.NOVELL.COM to an IP address, and it failed because the default filters don't allow DNS requests through. The DNS packets (UDP port 53) were dropped as they left the BorderManager server (Discard Outgoing).

FILTCFG was loaded and a stateful filter exception for DNS over UDP was configured and applied. The test was then repeated (in this case using two DNS server entries – 199.182.120.203 and 4.3.4.1)

```
RECEIVE:pktid:192 192.168.10.114->199.182.120.203 ttl:128 (UDP)
UDP:Source Port:1039Destination Port:53(DOMAIN)

FORWARD:pktid:192 4.3.2.254->199.182.120.203 ttl:127 (UDP)
UDP:Source Port:59878Destination Port:53(DOMAIN)

RECEIVE:pktid:193 192.168.10.114->4.3.4.1 ttl:128 (UDP)
UDP:Source Port:1040Destination Port:53(DOMAIN)

FORWARD:pktid:193 4.3.2.254->4.3.4.1 ttl:127 (UDP)
UDP:Source Port:59877Destination Port:53(DOMAIN)

RECEIVE:pktid:19565 4.3.4.1->4.3.2.254 ttl:126 (UDP)
UDP:Source Port:53(DOMAIN) Destination Port:59877

FORWARD:pktid:19565 4.3.4.1->192.168.10.114 ttl:125 (UDP)
UDP:Source Port:53(DOMAIN) Destination Port:1040
```

(The DNS server at 4.3.4.1 responded before the one at 199.182.120.203 did, and the DNS information was passed back to the PC at 192.168.10.114. Now the PC knows that the IP address of WWW.NOVELL.COM is 137.65.2.11, and it begins to ping it).

```
RECEIVE:pktid:194 192.168.10.114->137.65.2.11 ttl:32 (ICMP)Echo Request
Discard Outgoing: cause(FILTERING), reason(1)

RECEIVE:pktid:196 192.168.10.114->137.65.2.11 ttl:32 (ICMP)Echo Request
Discard Outgoing: cause(FILTERING), reason(1)

RECEIVE:pktid:197 192.168.10.114->137.65.2.11 ttl:32 (ICMP)Echo Request
Discard Outgoing: cause(FILTERING), reason(1)

RECEIVE:pktid:198 192.168.10.114->137.65.2.11 ttl:32 (ICMP)Echo Request
Discard Outgoing: cause(FILTERING), reason(1)
```

ICMP (PING) packets now go to the BorderManager server, but they are filtered on the way out of the private interface because there is no filter exception for ICMP packets with the default filters in place.

Next, a filter exception was set up to allow ICMP through, and the test was repeated. Multiple PING packets were sent to a host at 4.3.2.1. Only some of the traffic is shown.

```
RECEIVE:pktid:296 192.168.10.114->4.3.2.1 ttl:32 (ICMP)Echo Request
FORWARD:pktid:296 4.3.2.254->4.3.2.1 ttl:31 (ICMP)Echo Request

RECEIVE:pktid:296 4.3.2.1->4.3.2.254 ttl:255 (ICMP)Echo Reply
FORWARD:pktid:296 4.3.2.1->192.168.10.114 ttl:254 (ICMP)Echo Reply
```

After adding a filter exception for ICMP, the PING traffic looks normal. The PC at 192.168.10.114 sends an ICMP packet to 4.3.2.1. Dynamic NAT regenerates the packet as coming from its public IP address 4.3.2.254 and sends it on. The host at 4.3.2.1 responds and sends a reply to 4.3.2.254, and dynamic NAT returns that response to the original requester by regenerating the packet with the destination address of 192.168.10.114.

## Browsing Example – No Proxy Configured

Here is an example of what the default filters do when someone tries to browse the Internet without using the HTTP Proxy. The (Netscape) browser at host 192.168.10.114 was configured for a “Direct connection to Internet” (no proxy) as shown in Figure 4-1.

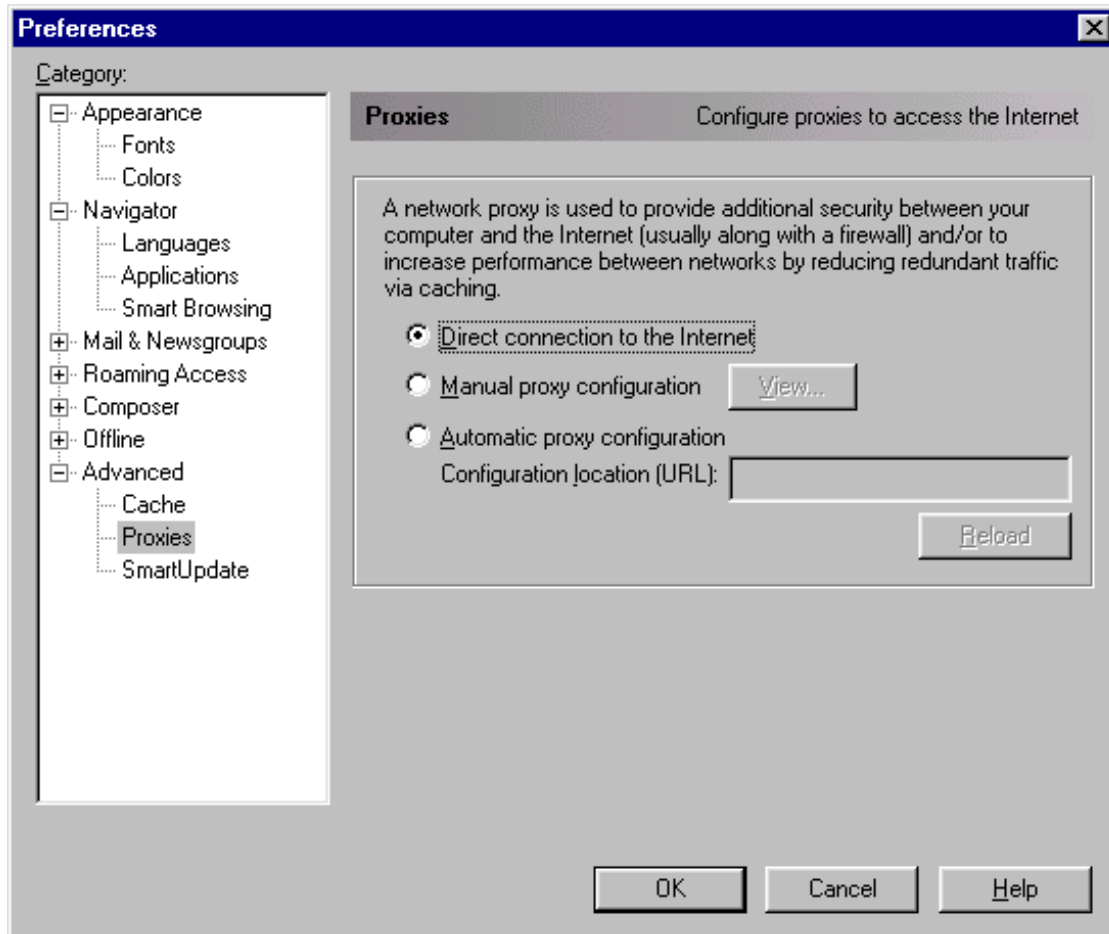


Figure 4-1 - Netscape Configured without Proxy settings

The same output could have been generated using any other browser set for direct connection to the Internet, as long as Transparent Proxy was disabled on the BorderManager server.

```
FORWARD:pktid:247 4.3.2.254->4.3.4.1 ttl:127 (UDP)
UDP:Source Port:59873Destination Port:53(DOMAIN)

RECEIVE:pktid:24136 4.3.4.1->4.3.2.254 ttl:126 (UDP)
UDP:Source Port:53(DOMAIN) Destination Port:59873

FORWARD:pktid:24136 4.3.4.1->192.168.10.114 ttl:125 (UDP)
UDP:Source Port:53(DOMAIN) Destination Port:1049

RECEIVE:pktid:248 192.168.10.114->137.65.2.118 ttl:128 (TCP)
TCP:SYN Source Port:1050, Dest Port:80 Sequence No.:3202996 Ack No:0 Window:8192
UrgPtr:0
Discard Outgoing: cause(FILTERING), reason(1)
```

The DNS exception set up earlier allows WWW.NOVELL.COM to be resolved to an IP address, but then the browser times out because the default filters block HTTP (port 80). The default filters do not allow traffic to automatically go from the private IP address to the public IP address.

The HTTP Proxy works differently by regenerating its HTTP port 80 traffic directly onto the public IP address, where it is allowed out by the default filter exceptions.

## Browsing Example – Proxy Configured, Default Filter Exceptions

In this example, the proxy is configured to use the HTTP Proxy, instead of trying to bypass it.

Type	Address of proxy server to use	Port
HTTP:	192.168.10.252	8080
Security:	192.168.10.252	8080
ETP:	192.168.10.252	8080
Socks:		1080
Gopher:	192.168.10.252	8080
WAIS:		0

Exceptions

Do not use proxy servers for domains beginning with:

127.0.0.1, 192.168.10.252

Use commas (,) to separate entries.

OK Cancel

Figure 4-2 - Netscape Configured to Use HTTP Proxy

The screenshot shown in Figure 4-2 shows the browser proxy settings used for the following trace.

```
RECEIVE:pktid:254 192.168.10.114->192.168.10.252 ttl:128 (TCP)
TCP:ACK Source Port:1046, Dest Port:8080 Sequence No.:3185555 Ack No:2873913276
Window:8208 UrgPtr:0
LOCAL:pktid:254 192.168.10.114->192.168.10.252 ttl:128 (TCP)
TCP:ACK Source Port:1046, Dest Port:8080 Sequence No.:3185555 Ack No:2873913276
Window:8208 UrgPtr:0
```

<some text deleted>

```
RECEIVE:pktid:9490 137.65.2.118->4.3.2.254 ttl:114 (TCP)
TCP:ACK Source Port:80, Dest Port:2422 Sequence No.:727352340 Ack No:2878541653
Window:64494 UrgPtr:0
LOCAL:pktid:9490 137.65.2.118->4.3.2.254 ttl:114 (TCP)
TCP:ACK Source Port:80, Dest Port:2422 Sequence No.:727352340 Ack No:2878541653
Window:64494 UrgPtr:0
```

The browser makes requests to the HTTP Proxy at 192.168.10.252 using port 8080. The HTTP Proxy **regenerates** the requests on its public IP address of 4.3.2.254 and receives responses on that port. Eventually it builds up a complete 'node' (HTTP entity) in its cache and then sends all that data back to the browser on port 8080 (not shown). Alternatively, the data is not retrieved from the origin host and is instead retrieved from cache.

Notice in the example above how there is no routing of port 8080 to the Internet. Traffic between the originating PC is using port 8080, but the HTTP Proxy uses standard HTTP port 80 when it talks to the web server at 137.65.2.118.

## Filter Debug - An Alternative to TCP IP DEBUG

If you have ever used SET TCP IP DEBUG=1 on a production server with a lot of traffic crossing it, you know how much information can fly by in a few seconds, obscuring the packet or two of interest to you. There is a way to view individual packets being filtered as they occur without seeing all the non-filtered traffic. Use the following command to enable the filter debug options, and choose the option of interest. As with any debug option, this option should **not** be left enabled on a production server.

```
SET FILTER DEBUG=ON
```

---

**Note** The Filter Debug setting is a feature of the IPFLT31.NLM filtering module, and therefore filtering must be enabled to use the command. BorderManager 2.1 servers did not contain a version of IPFLT31.NLM that provided this capability.

---



```

Warning: Do NOT enable debug on production server ...
Set Filter Debug=on/off
Set IP Forward Filter Debug=1/0
Set IP Discard Filter Debug=1/0
Set TCP Forward Filter Debug=1/0
Set TCP Discard Filter Debug=1/0
Set UDP Forward Filter Debug=1/0
Set UDP Discard Filter Debug=1/0
Set ICMP Forward Filter Debug=1/0
Set ICMP Discard Filter Debug=1/0
Set SKIP Forward Filter Debug=1/0
Set SKIP Discard Filter Debug=1/0
Set Others Forward Filter Debug=1/0
Set Others Discard Filter Debug=1/0
Set Route Filter Debug=1/0
Set Route Allow Filter Debug=1/0
Set Route Deny Filter Debug=1/0

Or type one these keywords below at the server console:
ipflt_debug_help      <Display help messages>
ipflt_debug_off       <Disable all filter debug>
ipflt_debug_on        <Enable packet forwarding filter debug>
ipflt_route_debug     <Enable routing filter debug>

Filter Debug set to ON
BORDER1:

```

Figure 4-3 - SET FILTER DEBUG=ON

This command should bring up a list of options as shown in Figure 4-3. Enable the option of interest with the command shown. With these commands, you can have more control over debug traffic reported. I personally found the SET TCP DISCARD FILTER DEBUG=1 to be the most useful of the options, though occasionally it is useful to use the ICMP discard or forward option.

You can turn several options on at the same time.

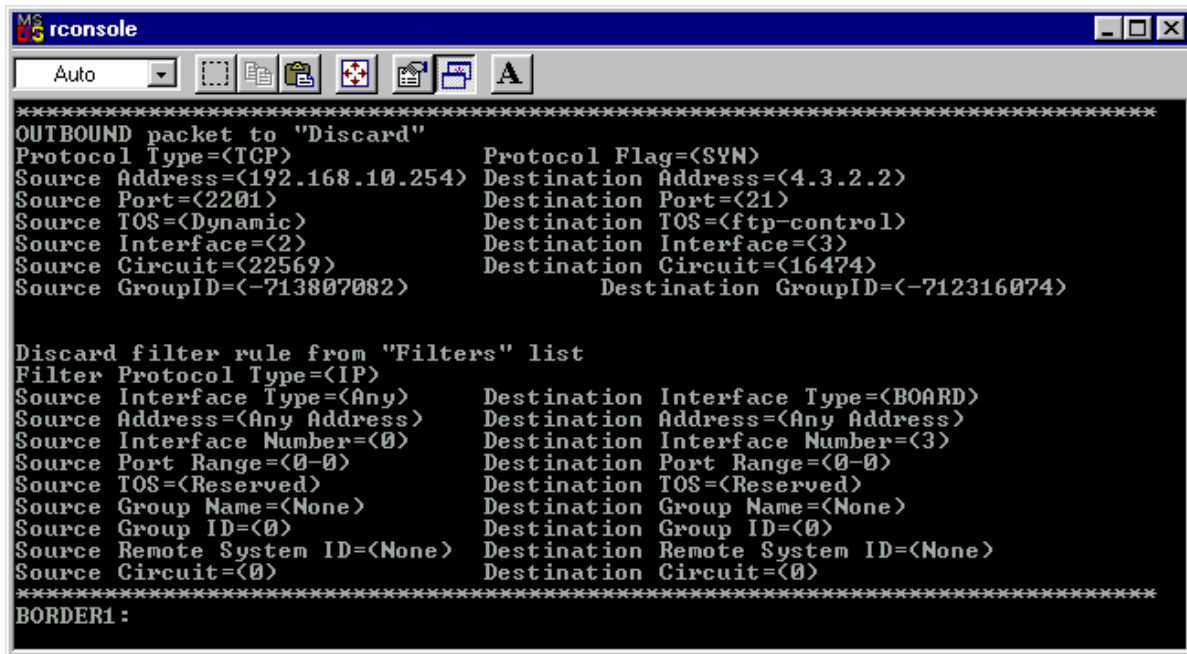
The value of the FILTER DEBUG output is that you have a chance of concentrating on a particular type of discard, while being able to ignore all the rest of the traffic flowing through a busy server. This value is also a potential problem – you can miss other traffic that might be useful for troubleshooting. Still, if you must debug a new filter exception during production hours on a busy server, this option may be the only reasonable way to see what the filters are stopping.

Be sure to disable the command when done using the SET FILTER DEBUG=OFF command.

Some experimentation will be necessary to understand the Filter Debug options.

## Filter Debug Example Output

The following example was generated by trying to FTP from an internal host to an FTP server at 4.3.2.2, without having a filter exception to allow FTP outbound.



```

*****
OUTBOUND packet to "Discard"
Protocol Type=(TCP)          Protocol Flag=(SYN)
Source Address=(192.168.10.254) Destination Address=(4.3.2.2)
Source Port=(2201)           Destination Port=(21)
Source TOS=(Dynamic)         Destination TOS=(ftp-control)
Source Interface=(2)         Destination Interface=(3)
Source Circuit=(22569)        Destination Circuit=(16474)
Source GroupID=(-71307082)    Destination GroupID=(-712316074)

Discard filter rule from "Filters" list
Filter Protocol Type=(IP)
Source Interface Type=(Any)   Destination Interface Type=(BOARD)
Source Address=(Any Address) Destination Address=(Any Address)
Source Interface Number=(0)  Destination Interface Number=(3)
Source Port Range=(0-0)      Destination Port Range=(0-0)
Source TOS=(Reserved)        Destination TOS=(Reserved)
Source Group Name=(None)     Destination Group Name=(None)
Source Group ID=(0)          Destination Group ID=(0)
Source Remote System ID=(None) Destination Remote System ID=(None)
Source Circuit=(0)           Destination Circuit=(0)
*****
BORDER1 :
  
```

Figure 4-4 - FILTER DEBUG Capture Example

The example shown in Figure 4-4 shows a single TCP packet being filtered, after using the commands

```
SET FILTER DEBUG=ON
```

```
SET TCP DISCARD FILTER DEBUG=1
```

This example shows an FTP request (note destination port number=21), being filtered in the outbound direction (Note source IP address of 192.168.10.254 is inside the LAN, while destination IP address 4.3.2.2 is outside the LAN). The source port was 2201, which is 'randomly' assigned as a high port. The source interface was 2, (which is the private interface), and the destination interface was 3, (which is the public interface).

## NCF Files To Use With SET FILTER DEBUG=ON

I have provided these examples for your use. I think you will find them useful to capture filter debug information and display it easily.

### T1.NCF (Turn On Debugging and Capture the Results)

```
Rem This NCF file starts IP filter debugging and logs the screen results
Rem to a file with CONLOG. Type in T1 to start the debug and T0 to stop it.
Rem Uncomment the lines below to start the desired debug options.
Unload CONLOG
LOAD CONLOG MAX=100
SET FILTER DEBUG=ON
SET TCP DISCARD FILTER DEBUG=1
rem SET UDP DISCARD FILTER DEBUG=1
rem SET ICMP DISCARD FILTER DEBUG=1
```

### T0.NCF (Turn Off Debugging and Display the Results)

```
Rem This NCF file stops IP filter debugging and displays the logged results
Rem by using EDIT. Type in T1 to start the debug and T0 to stop it.
Rem If you want to use CONLOG after running this NCF file, you must
Rem restart it manually. (LOAD CONLOG MAX=100)
Unload CONLOG
SET FILTER DEBUG=Off
SET TCP DISCARD FILTER DEBUG=0
SET UDP DISCARD FILTER DEBUG=0
SET ICMP DISCARD FILTER DEBUG=0
LOAD EDIT SYS:ETC\CONSOLE.LOG
```

Be sure to remember to reload CONLOG after using the T0.NCF commands if you normally use CONLOG and wish to continue console logging.

---

**Note** NetWare 6.0 redirects FILTER DEBUG data to the Logger screen, and the Logger screen cannot be captured with Conlog. However, the Logger screen can be captured to a DOS file (C:\NWSERVER\LOGGER.TXT) use the F2 key. The Logger screen can also be scrolled with the PageUp and PageDown keys. Press F1 while viewing the Logger screen for help with the Logger screen options. If you need to copy the LOGGER.TXT file, you can use TOOLBOX.NLM. You could even view the file (and cut-and-paste the contents to another file) with EDIT.NLM. TOOLBOX is a utility supplied by Novell, and it can be downloaded from <http://support.novell.com>.

---

# Making a Custom Filter Exception in FILTCFG.NLM

## Part 1, Starting To Make A Filter Exception

This example shows how to set up a custom filter definition. For the purpose of this book, the example shows a meaningless stateful TCP filter exception being defined for all source ports 1024 through 65535 and destination port 999. This exception is simply being used as an example of how to create a filter exception where you also have to define a custom filter definition because it doesn't exist in the list of predefined filters supplied by Novell with BorderManager.

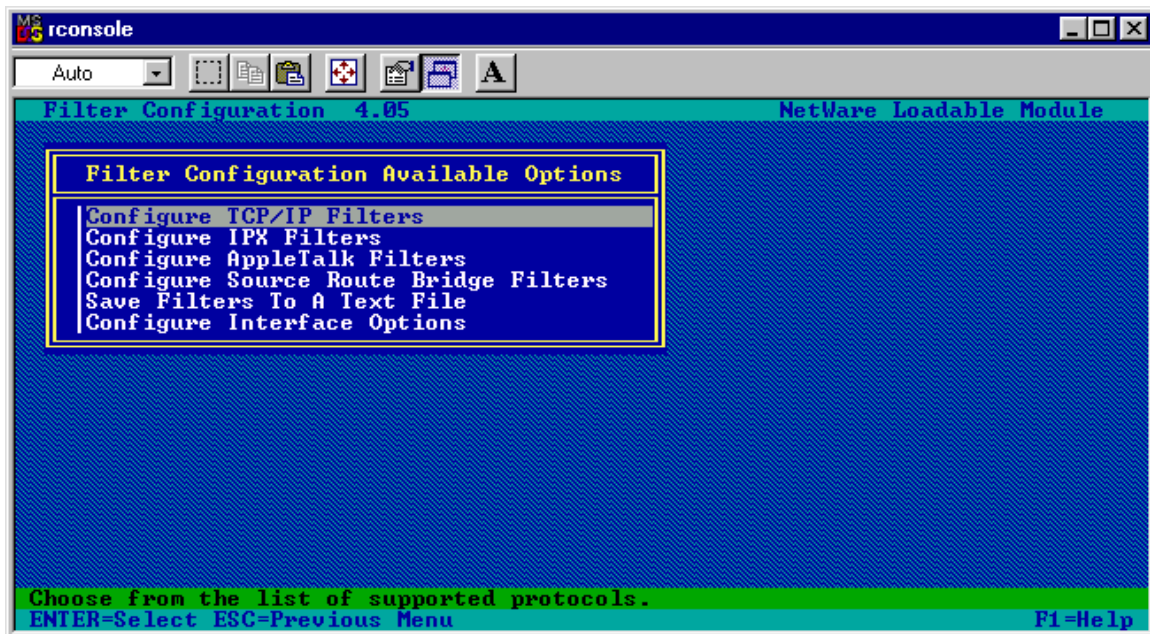


Figure 4-5 - FILTCFG - Main Menu

At the server console, type **LOAD FILTCFG.**

Select **Configure TCP/IP Filters**

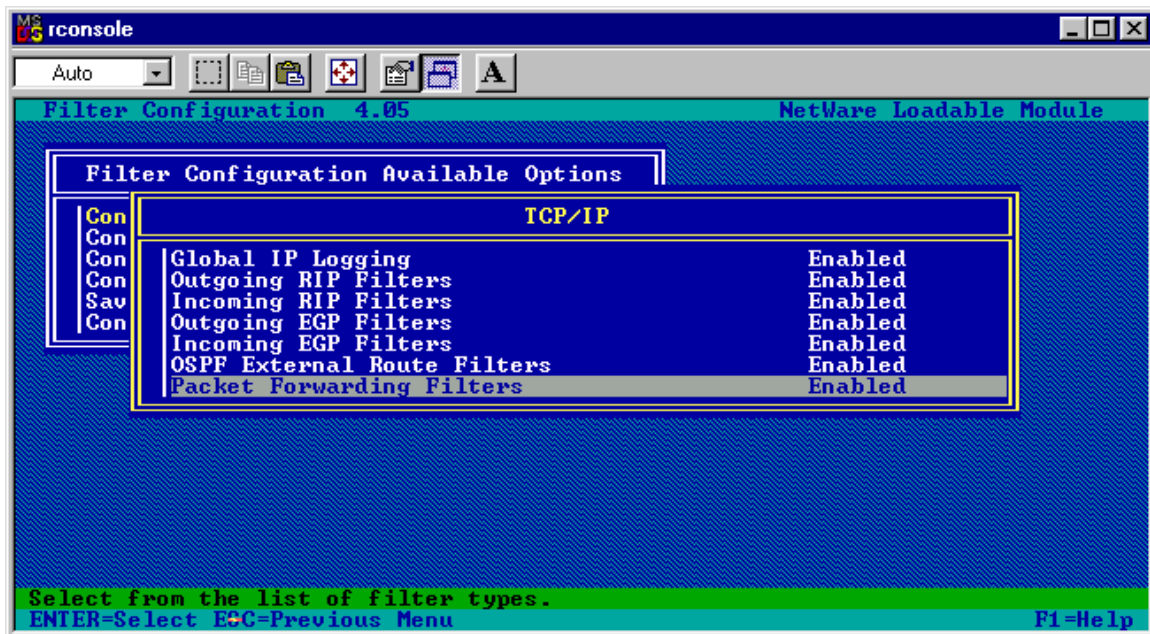


Figure 4-6 - FILTCFG - Select Packet Forwarding Filters

### Select Packet Forwarding Filters

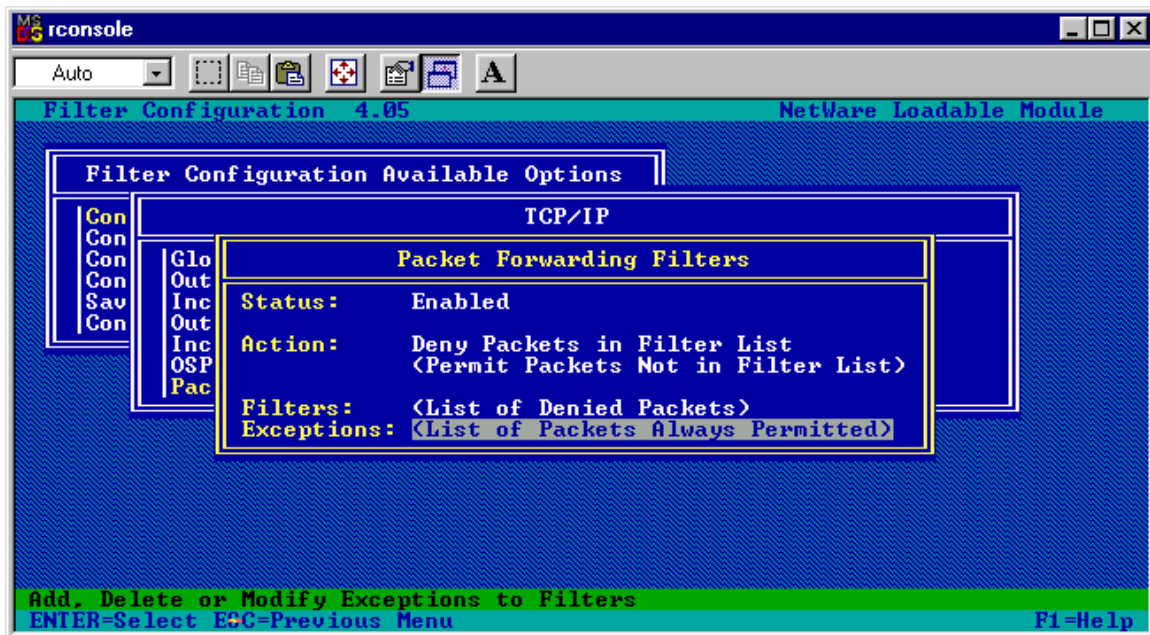


Figure 4-7 - FILTCFG - Select List of Packets Always Permitted

Select **List of Packets Always Permitted** to create a new filter exception.

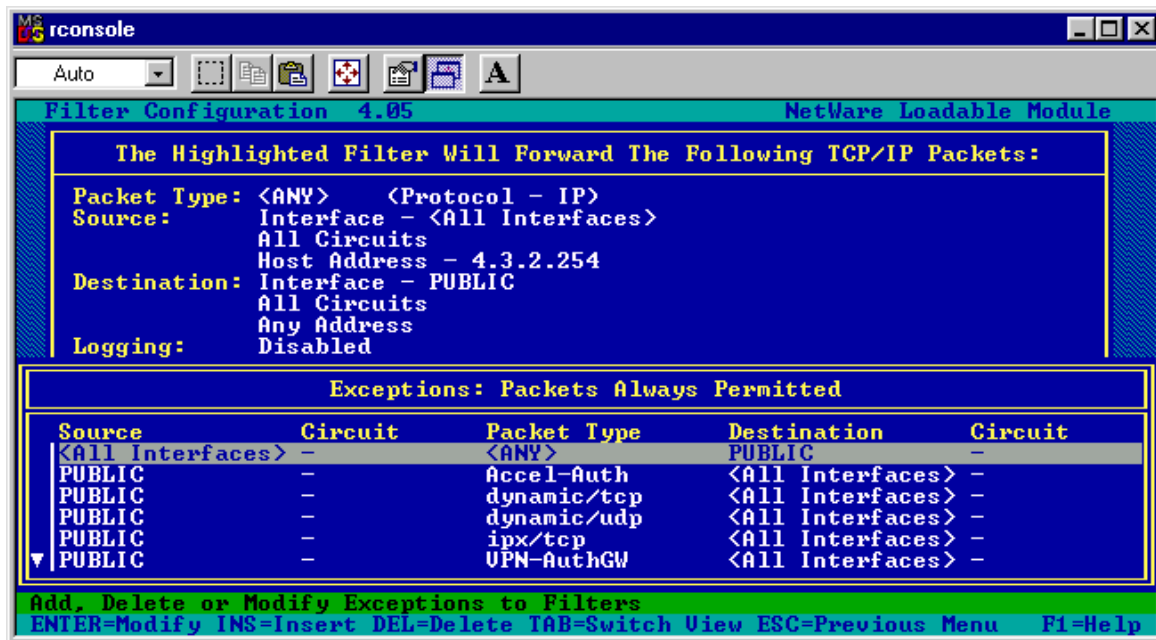


Figure 4-8 - FILTCFG - Filter Exception Menu

The screenshot shown in Figure 4-8 shows the first of several filter exceptions. From this menu, you can create, delete and modify existing filter exceptions, except that you cannot directly modify the definitions for the 'built-in' definitions.

Press the **Insert** key to create a new filter exception.

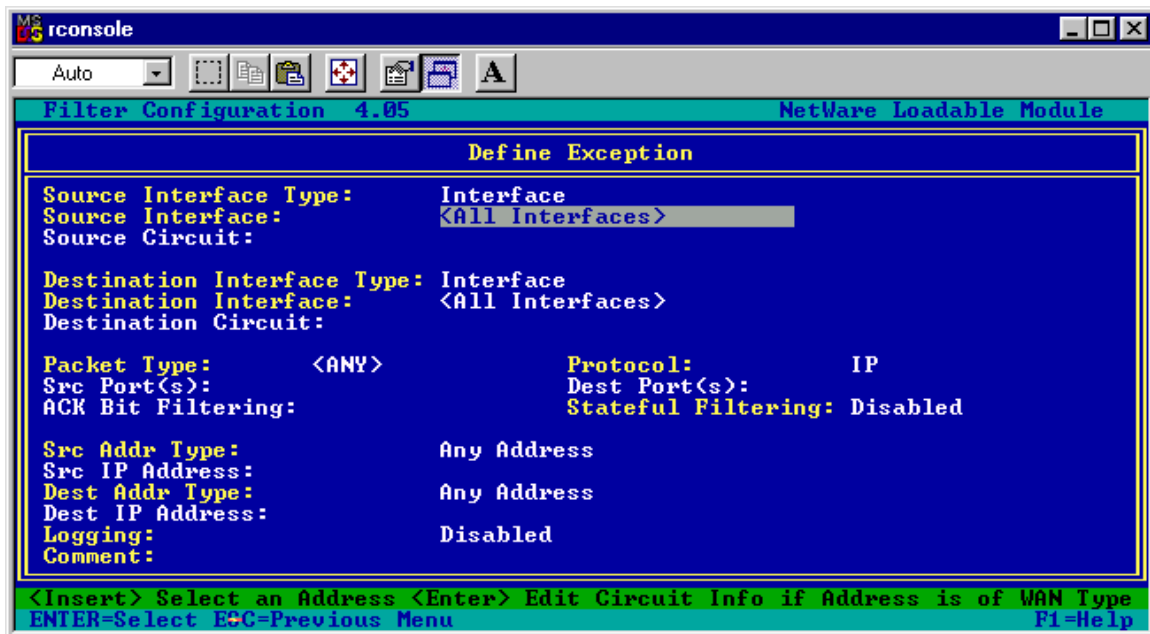


Figure 4-9 - FILTCFG - Select Source Interface

Select **Source Interface**, and choose your internal (private) network interface card. (Stateful filter exceptions for outbound traffic are best applied from the internal network interface to the external network interface).

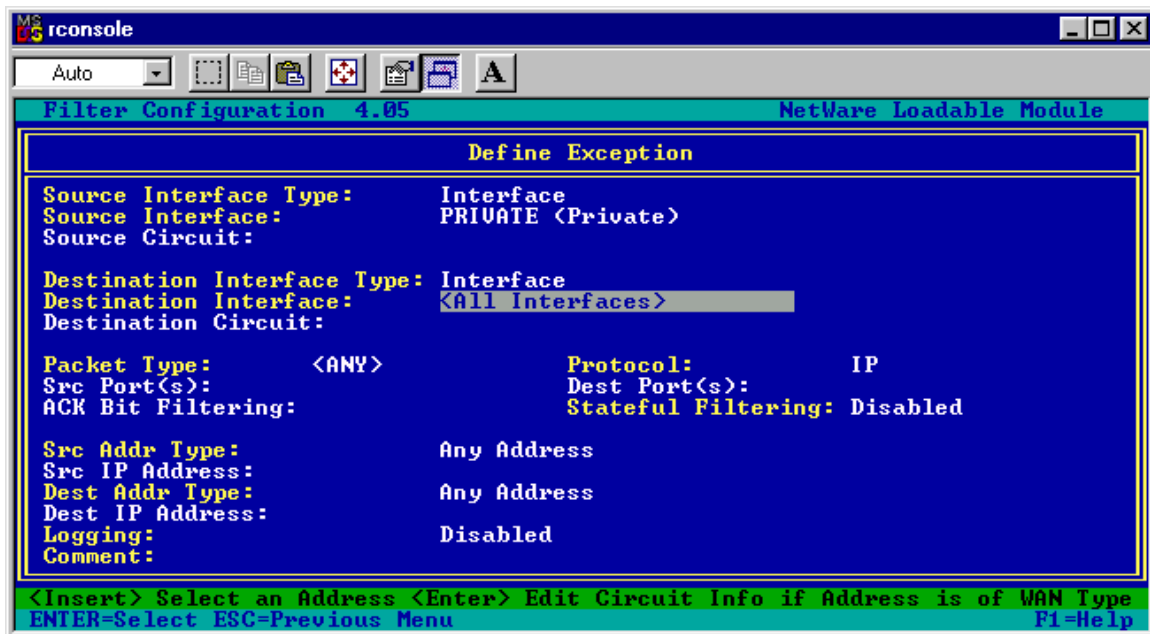


Figure 4-10 - FILTCFG - Select Destination Interface

Next, select **Destination Interface**, and choose your external (public) network interface card



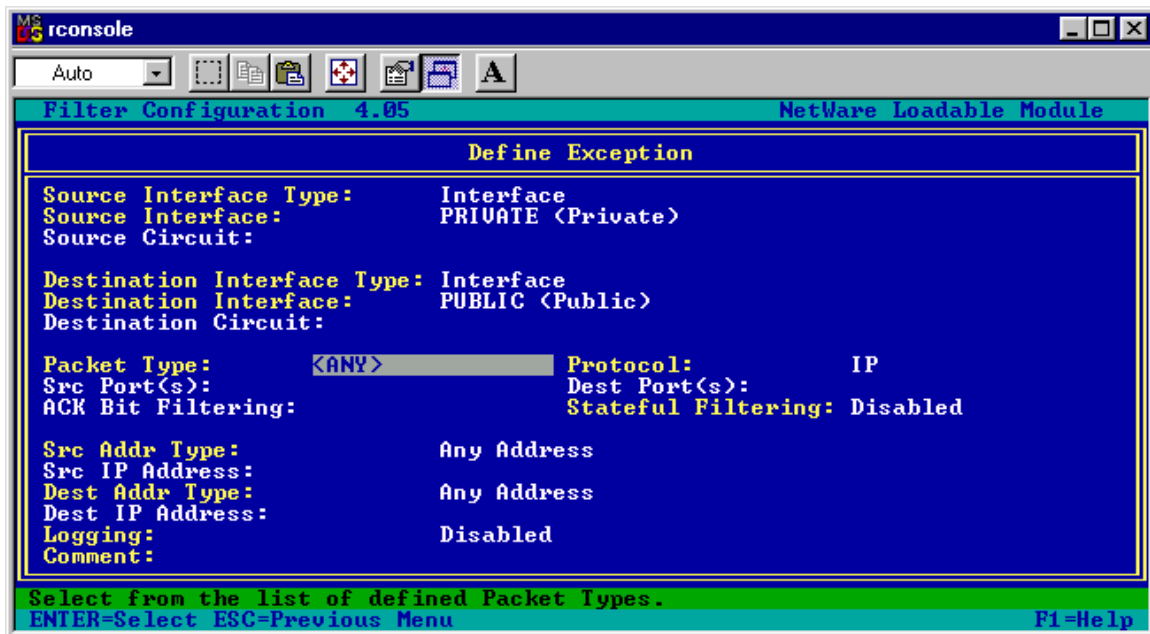


Figure 4-11 - FILTCFG - Define Exception Packet Type

You should now have your private and public interfaces configured and are ready to define the type of exception to apply to them.

---

**Note** The steps for selecting source and destination interface are important for stateful Filters. What this example does is set up the filter exception to be applied to any packets coming from the private network interface card to the public network interface card. This way you don't have to worry about IP address changes on the interfaces themselves, and the filter (in this case a filter exception) will only function in the outbound direction. Because the filter exception is to be defined as stateful, BorderManager will automatically keep track of the return traffic and allow it in, without having to set up an additional filter exception to allow Dynamic TCP or Dynamic UDP ports (essentially any port number from 1024 up) through the firewall.

---

Select **Packet Type** and press Enter.

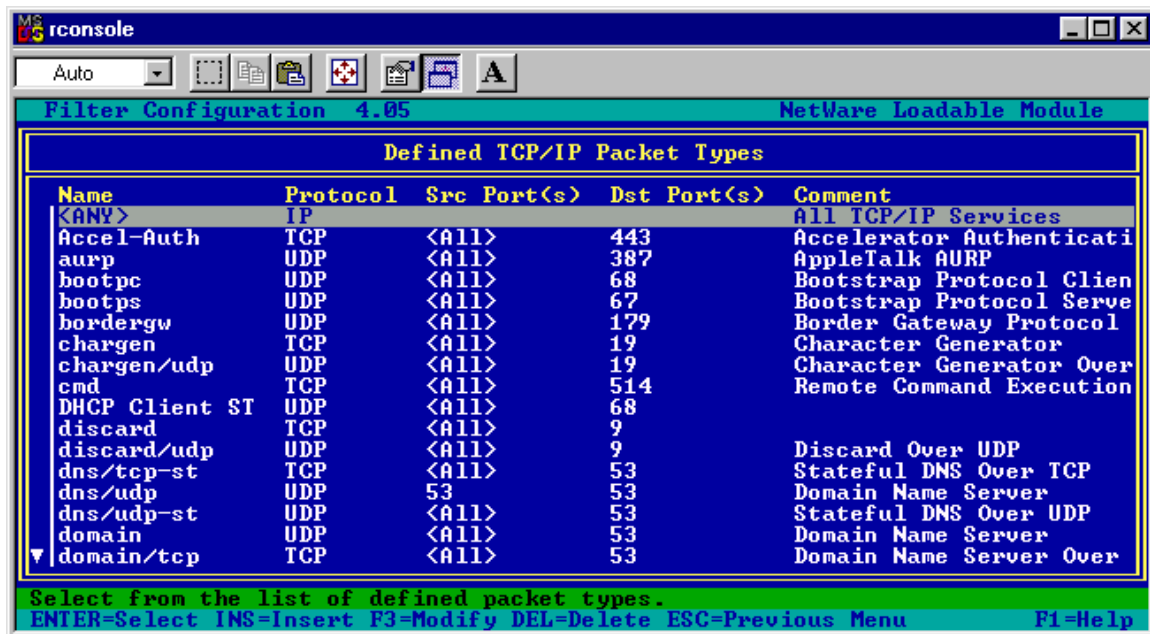


Figure 4-12 - FILTCFG - Create a New Packet Type

The figure above shows the point where you should not find any pre-defined filter definitions matching your requirements, and so you must create your own definition.

**Note** Your list of packet types will probably not match the one shown in Figure 4-12. The example shown is from a test server where many custom exceptions have already been added.

It is at this point that you have the choices seen at the bottom of the FILTCFG menu – Select an existing packet type, Create a new one, or Modify an existing packet type. Note that you are not allowed to modify the pre-defined packet types ('built-ins') supplied with BorderManager. (However, you can manually modify the SYS:\ETC\BUILTINS.CFG file if you need to).

## Part 2, Defining a New Filter Definition

Starting from the last point in Part 1 above (Figure 4-12), you should be at the list of defined TCP/IP packet types in FILTCFG.NLM.

Press the **Insert** key to add a new filter definition.

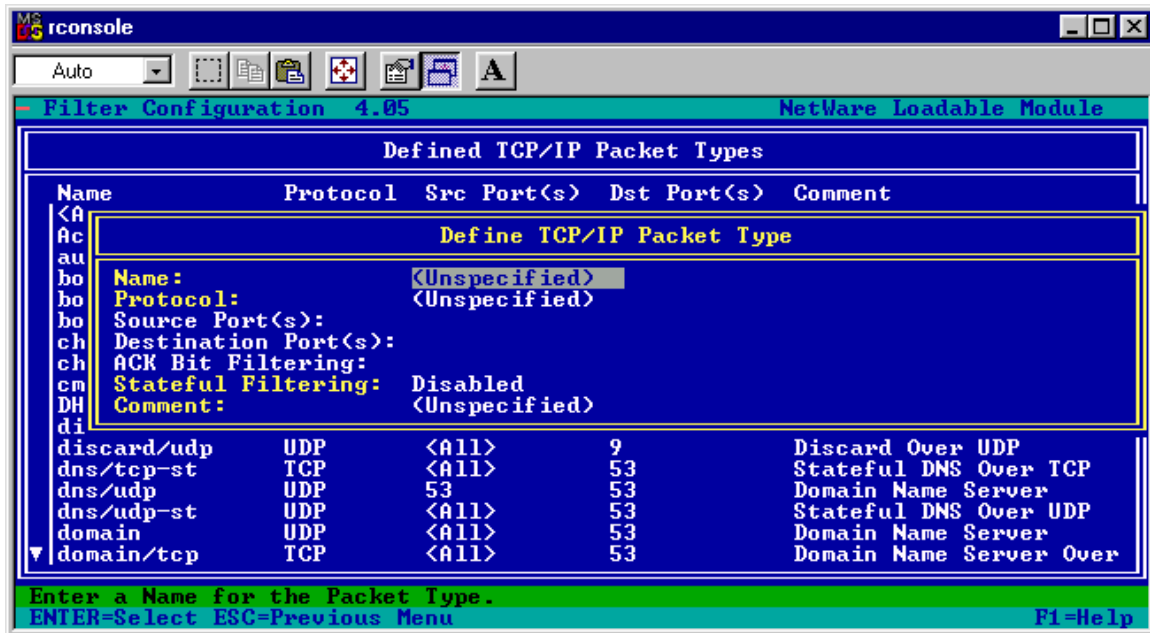


Figure 4-13 - FILTCFG - Enter Packet Type Name

The menu for defining your own filter definition comes up. Select **Name**, and enter a descriptive title. (You can edit this name later by re-selecting the filter definition and pressing F3).

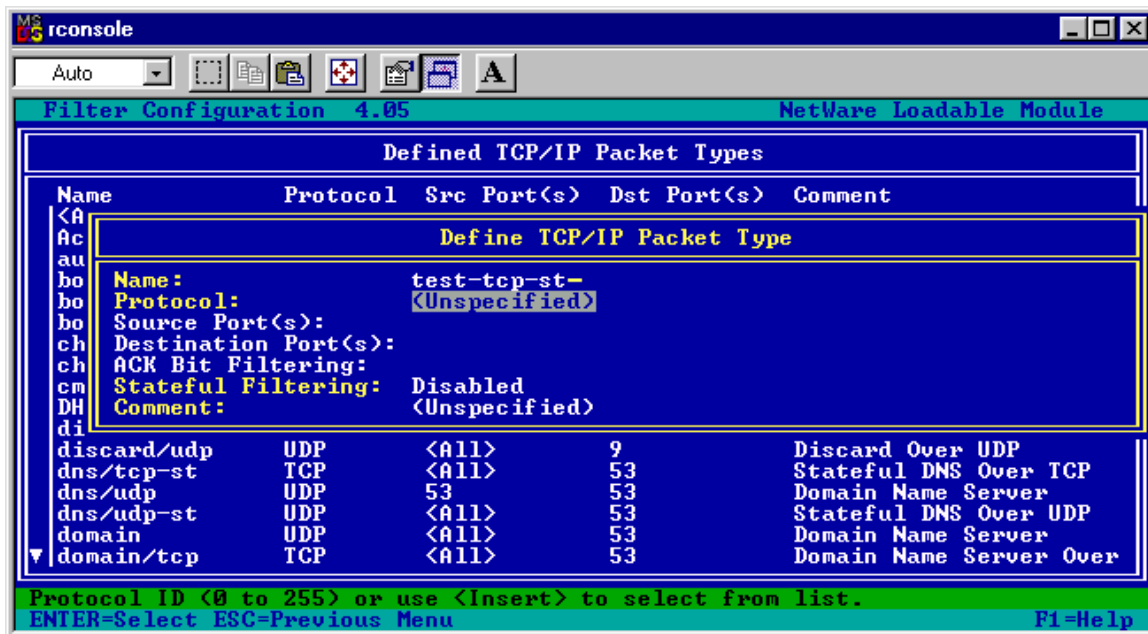


Figure 4-14 - FILTCFG - Enter Packet Type Protocol

After entering a descriptive name for the filter definition, select **Protocol** and press **Insert**.

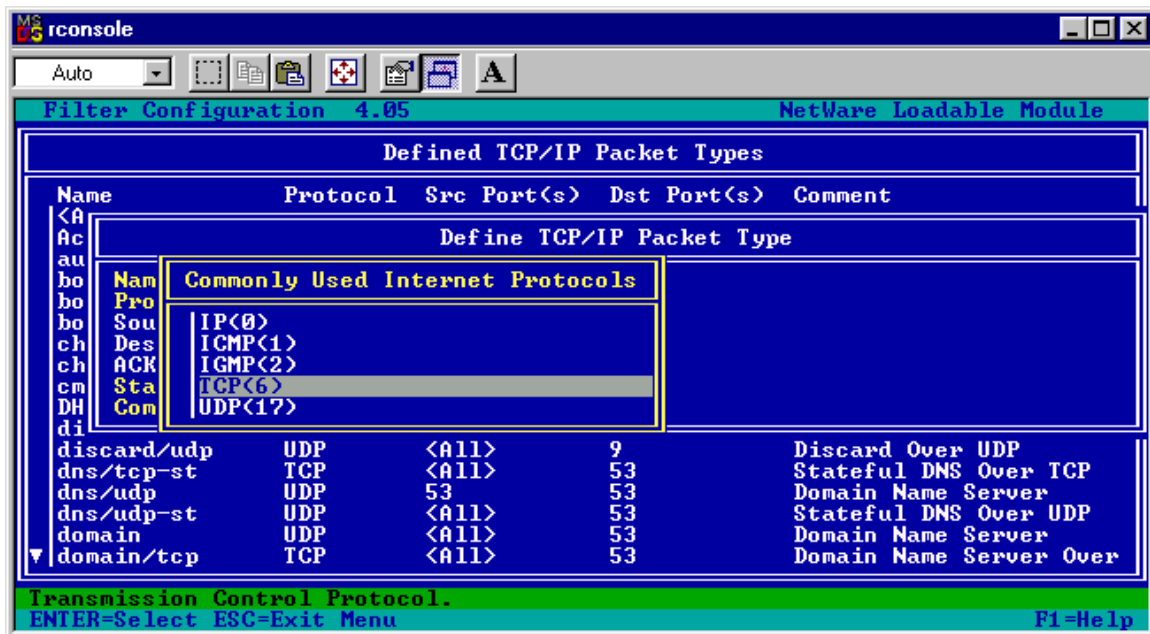


Figure 4-15 - FILTCFG - Select Protocol

Select the desired protocol, in this case **TCP(6)** and press **Enter**.

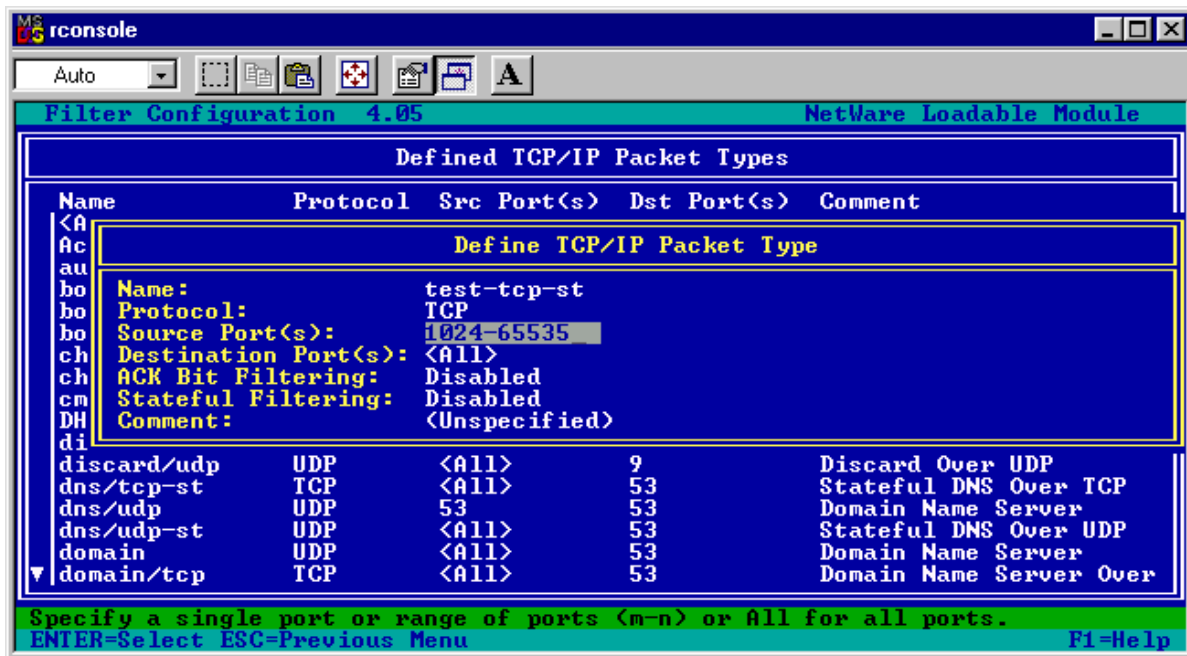


Figure 4-16 - FILTCFG - Enter Source Port

Select **Source Port(s)**, and enter one port number or a range of port numbers. In this example, all the port numbers between 1024 and 65535 are to be allowed as source ports, so enter **1024-65535**.

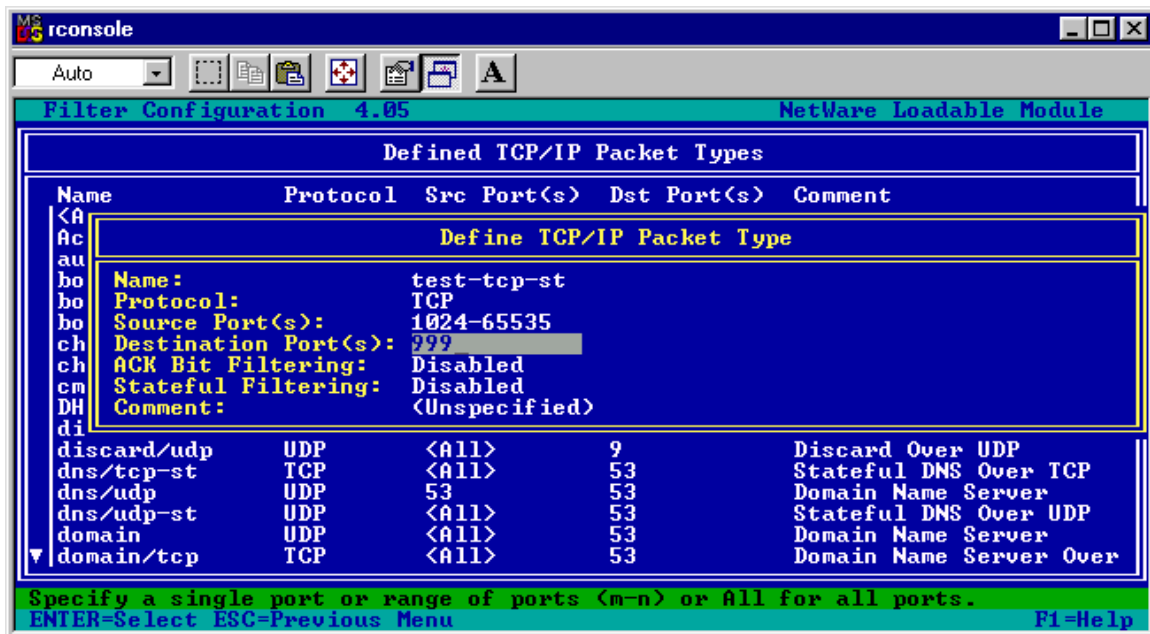


Figure 4-17 - FILTCFG - Enter Destination Port

After entering the source port or port range, select **Destination Port(s)** and enter a value of **999**.

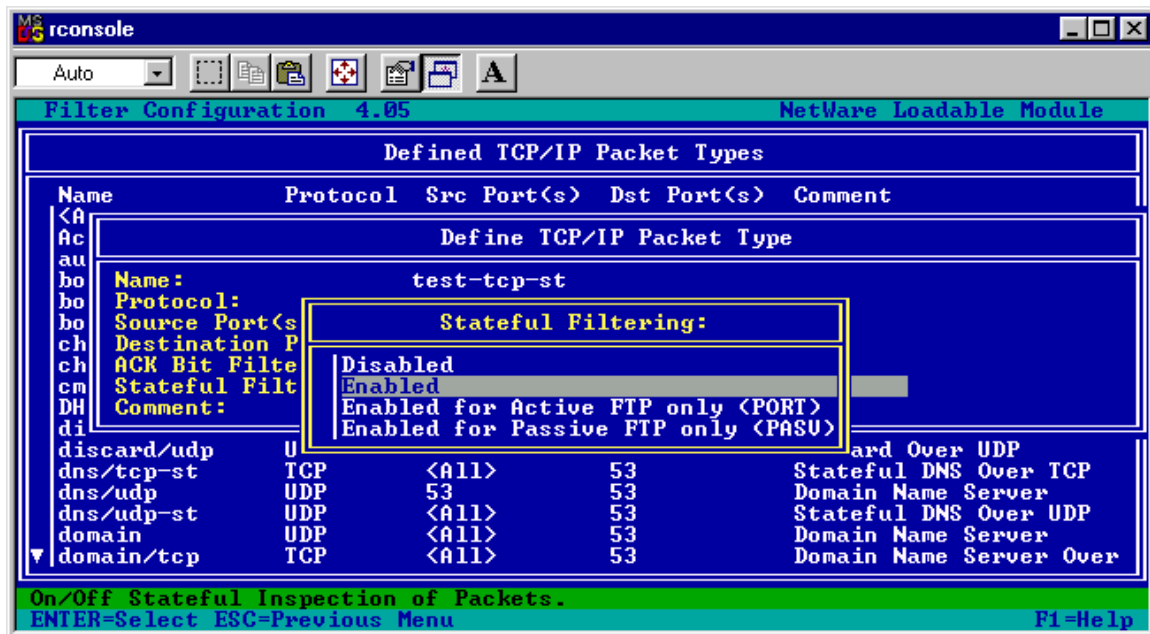


Figure 4-18 - FILTCFG - Specify Stateful Filtering

Next, select **Stateful Filtering**, and then select **Enabled** from the menu option.



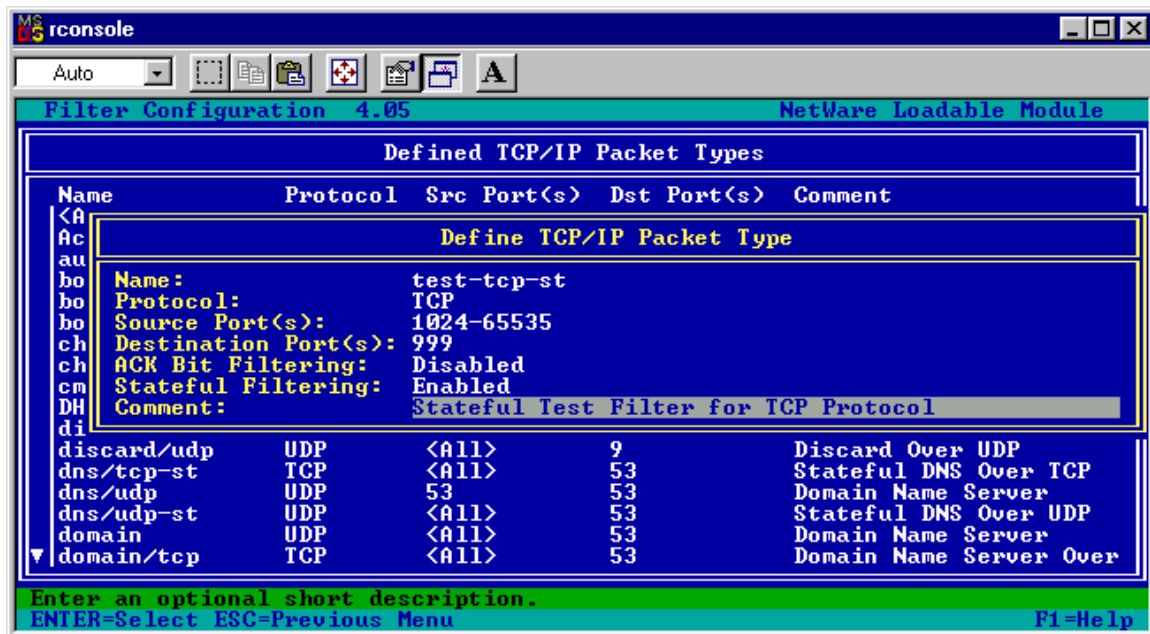


Figure 4-19 - FILTCFG - Comment the New Definition

Finally, select **Comment**, and enter a good description of the filter definition. It might be a good idea to enter a date and your initials to make custom filter definitions easier to track. You can edit a definition later by selecting it and pressing the F3 key.

Press **Escape** to save the new definition.

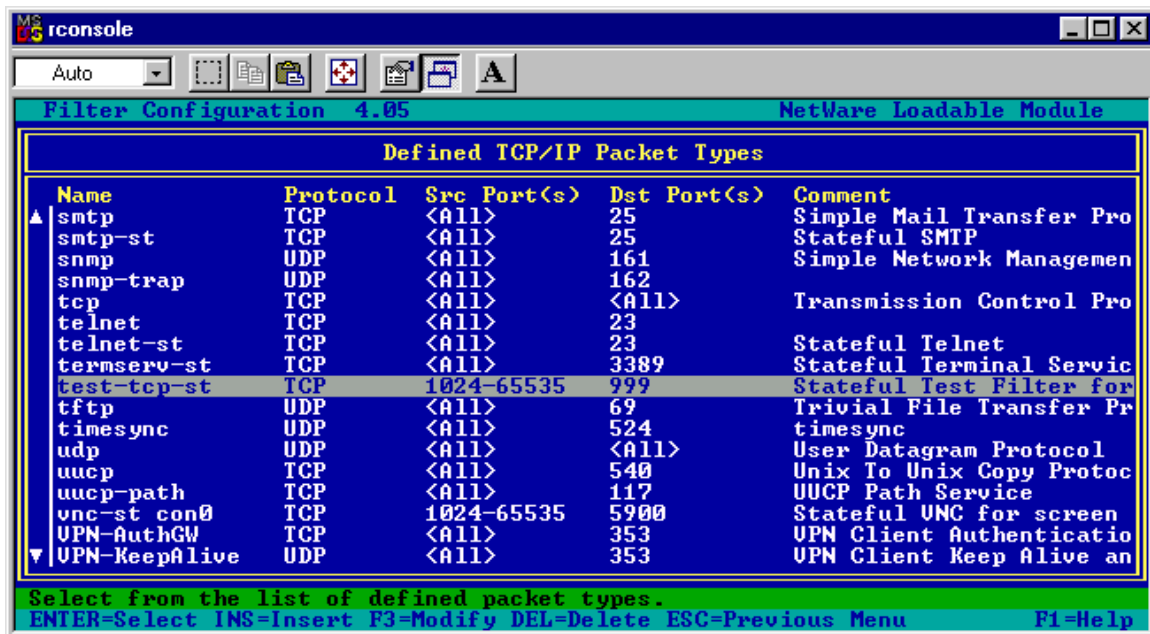


Figure 4-20 - FILTCFG - Updated Packet Type List

The new definition appears in the list of available filter definitions.

Press **Enter** to select this new definition to insert it into your filter exception.

### Part 3, Finishing the Filter Exception

After creating the new filter definition in Part 2 above, pressing Enter actually applies it to the filter exception being created. All you need to do now is to set source and destination IP addresses, and, if desired, add a useful comment, and save the changes. In this example, no source or destination IP address is used.

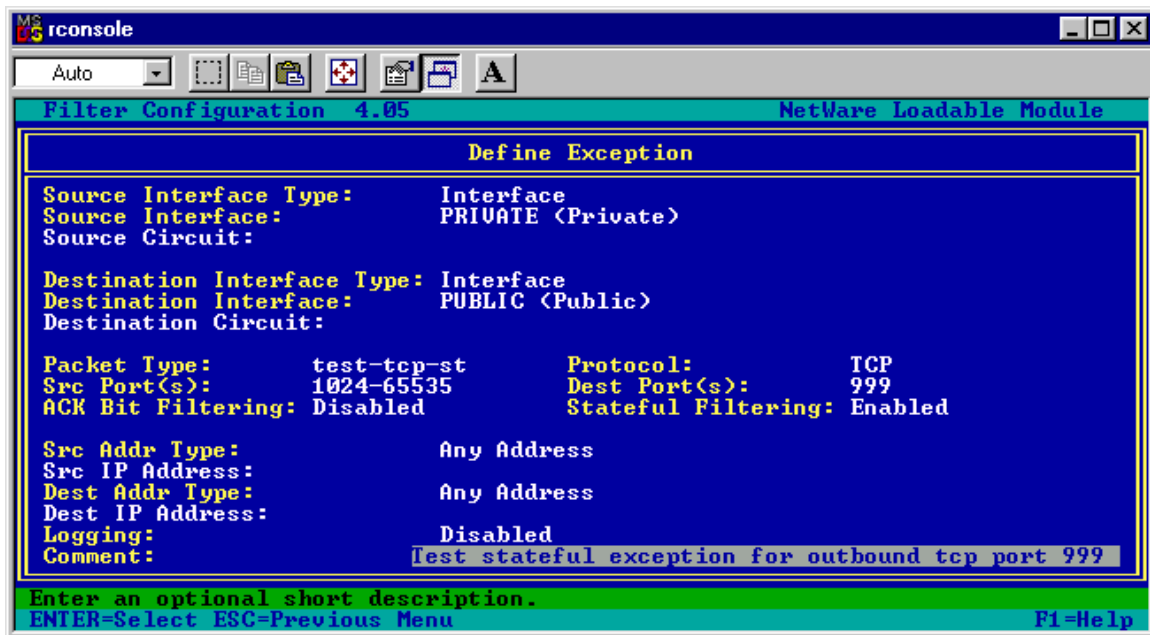


Figure 4-21 - FILTCFG - Add Comment for New Exception

Select **Comment**, and add as descriptive a comment for this filter exception as possible. This is *important* as you can easily lose track of what an exception was intended to accomplish. Be sure to press **Enter** when done typing.

---

**Note** Unfortunately, FILTCFG does not let you specify a range of IP addresses for source or destination IP address. If you cannot use a subnet to define a range, you need to set up individual filter exceptions for each IP address you need.

---

Press **Enter** to save the comment.

Press **Escape**.

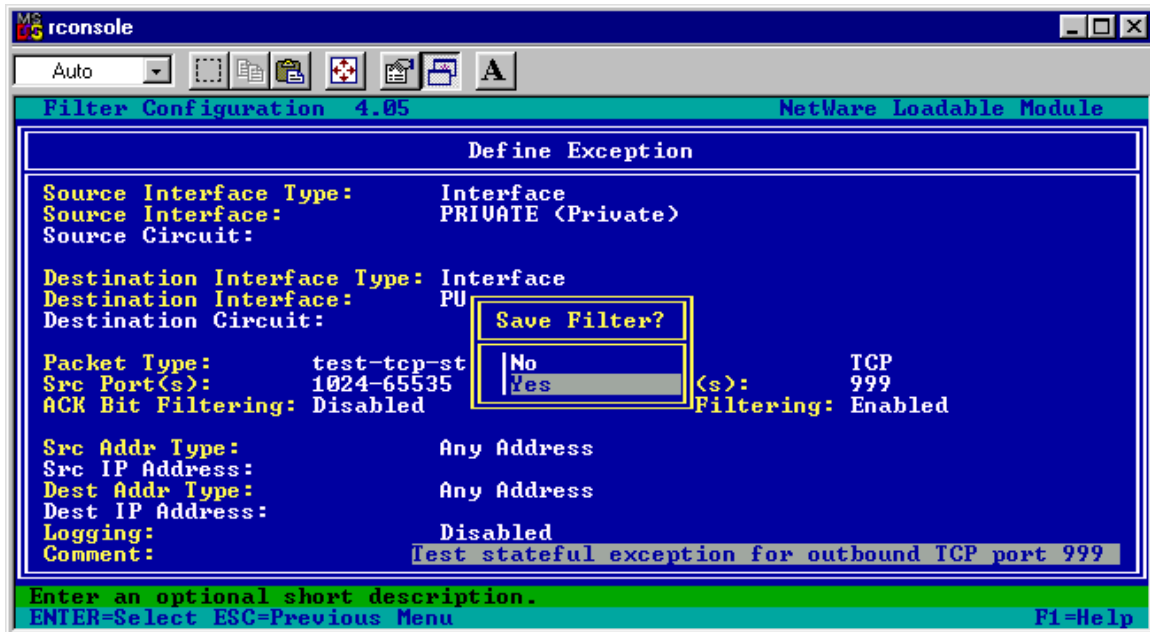


Figure 4-22 - FILTCFG - Save New Filter Option

If you want to save this filter exception, select **Yes** at the **Save Filter?** Prompt.

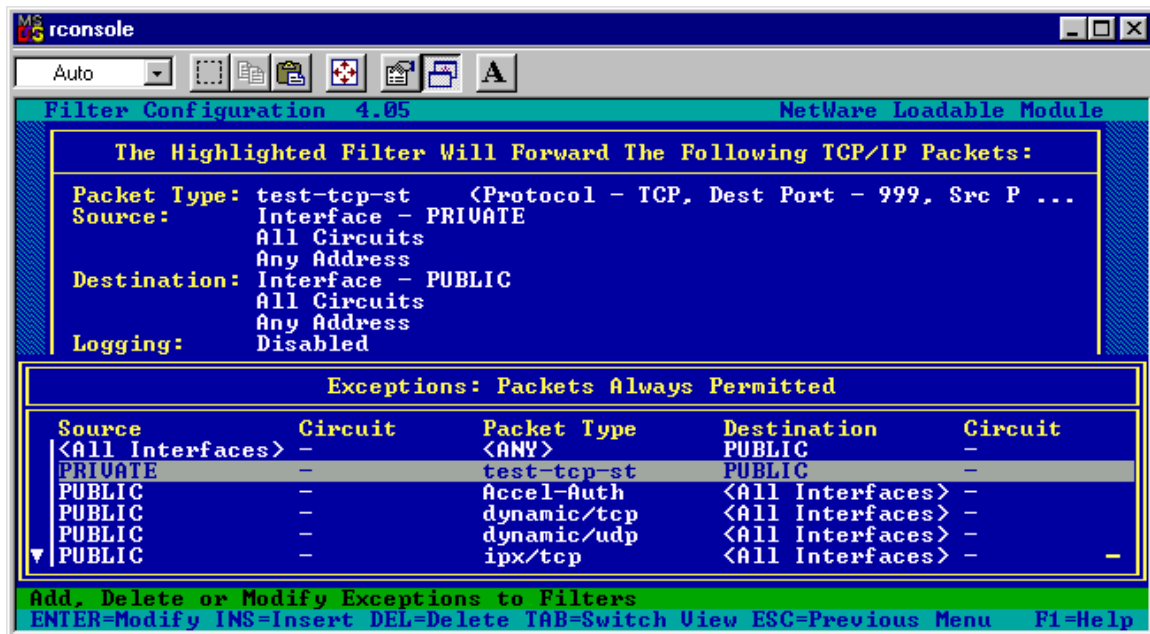


Figure 4-23 - FILTCFG - New Filter Active in List of Packet Filter Exceptions

The new filter exception should appear in the filter exception list, and should go into effect **immediately**.

**Note** It might sometimes be necessary to reinitialize system, or possibly UNLOAD IPFLT, and then LOAD IPFLT, but this is not normally needed.

Should you wish to make changes to this custom exception, select it, select Packet Type, and press F3 to modify the definition.

## Making a Custom Filter Exception in iManager

BorderManager 3.7 can make use of the iManager browser-based GUI interface, available on NetWare 6.0 or later server, to manage filters. Only IP (not IPX or Appletalk) filtering information can be managed with iManager, because only IP filtering information is migrated into NDS in BorderManager 3.7.

A BorderManager 3.7 installation should automatically extend the NDS schema and produce a new NDS container called NBMRuleContainer. After installing BorderManager 3.7, you should use the following command (once) at the server console to migrate existing filters and filter definitions from the FILTERS.CFG and BUILTINS.CFG files into NDS, into the NBMRuleContainer

```
LOAD FILTSRV MIGRATE
```

ainer.

T

When BorderManager is installed on a NetWare 6 server, iManager should show a new option called NBM Access Management.

---

**Note** Troubleshooting and configuring iManager if you do not see the option for is not within the scope of this book, but is covered in my book "A Beginner's Guide to BorderManager 3.x". See details at <http://nscsysop.hypermart.net> or <http://www.caledonia.net>. That book also describes how to configure a non-BorderManager NetWare 6.0 server to show the new filter management interface.

---

## Start iManager

Go into the iManager interface by pointing your browser at the NetWare 6 server and entering the following URL. Substitute your BorderManager server's private IP address for the one shown in the example below.

```
https://192.168.10.3:2200
```

---

**Note** 2200 is the default iManager port number. Yours could be different.

---

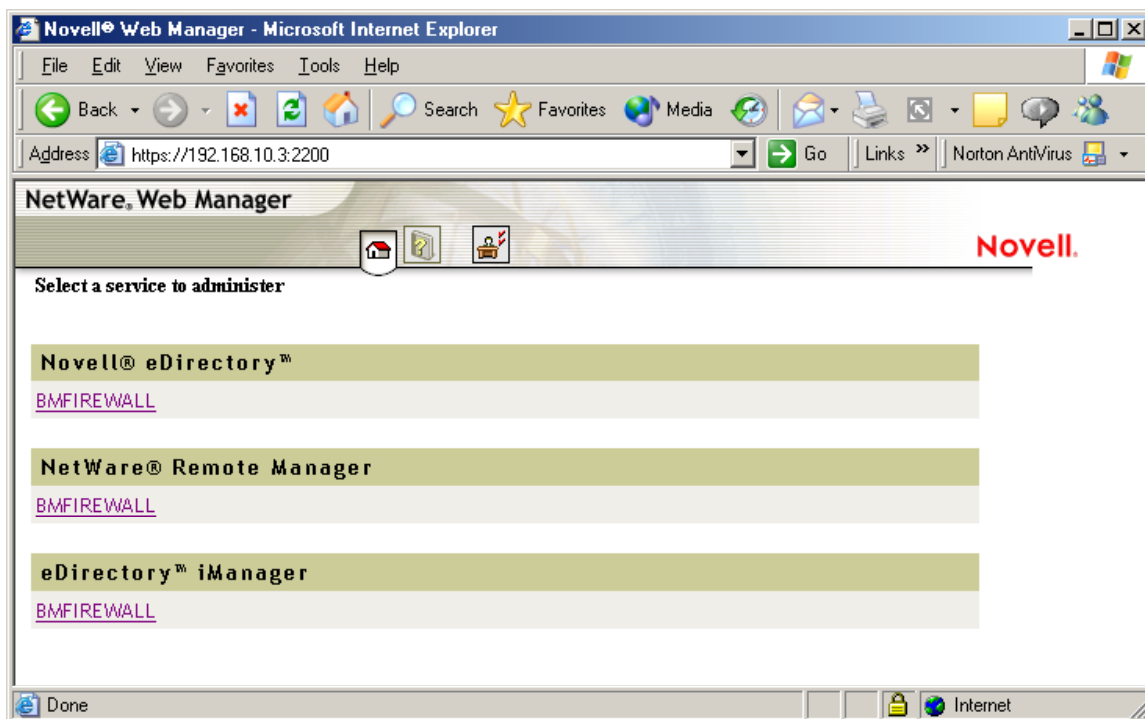


Figure 4-24 - NetWare Web Manager Main Menu

You should have the option to enter iManager on the server. Select that option.

Next, you should be prompted to log in to iManager.

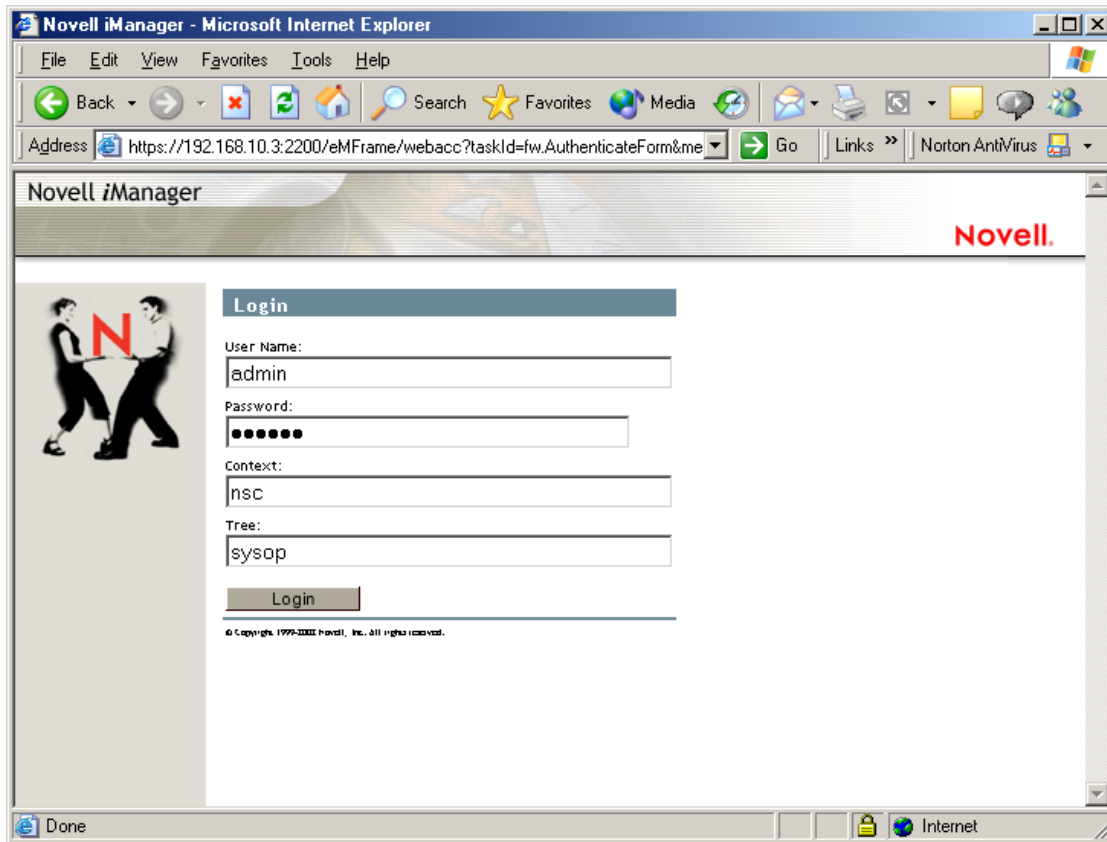


Figure 4-25 - iManager Login Screen

Log into the server, and you should then see various iManager options.



If the BorderManager Java files were correctly installed on the server, and the schema was properly extended, you should see an option for **NBM Access Management**.

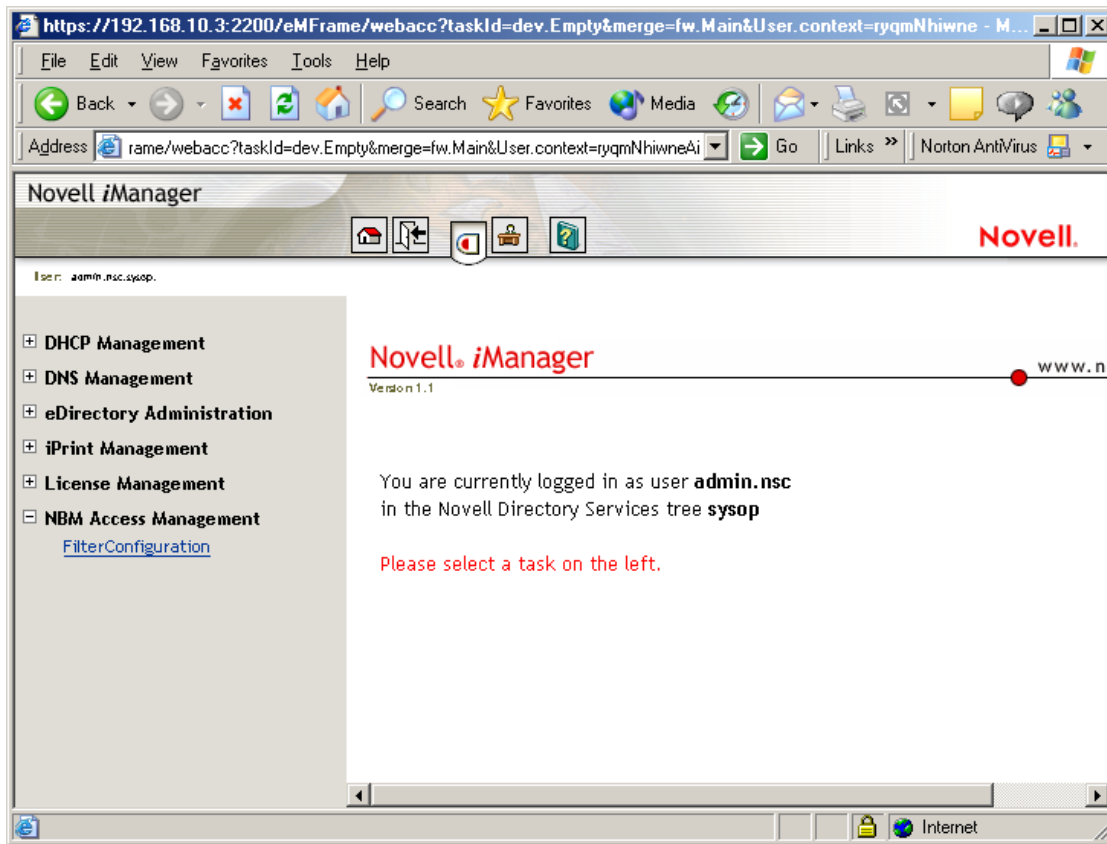


Figure 4-26 - iManager Main Menu, with NBM Access Management Option

Select the **NBM Access Management** link on the left to expand it. You should have a link called **FilterConfiguration**.

Select the **FilterConfiguration** option to begin using iManager to configure IP filters in BorderManager 3.7.

You will have to browse to a BorderManager 3.7 server and select it to continue.

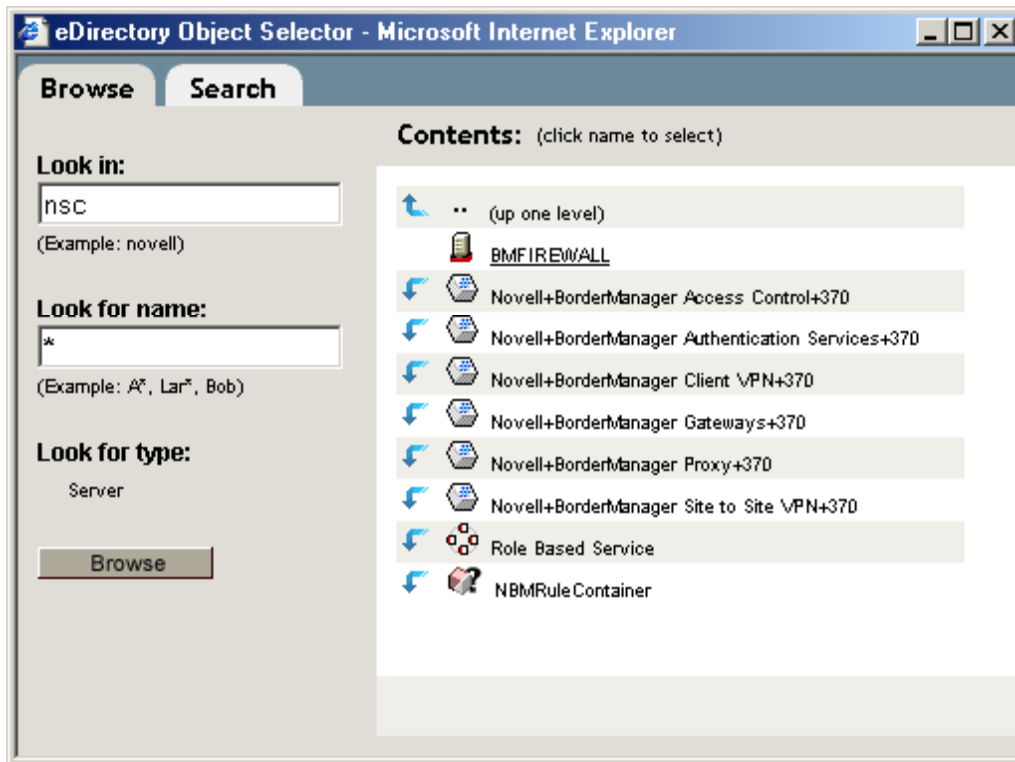


Figure 4-27 - iManager Browse Screen for BorderManager Server Object

The screenshot shown in Figure 4-27 above shows the browser applet being used in Browse mode to find a BorderManager server. In this example, a server called **BMFIREWALL** will be selected in a container called NSC.

You can choose any BorderManager 3.7 server in your NDS tree.

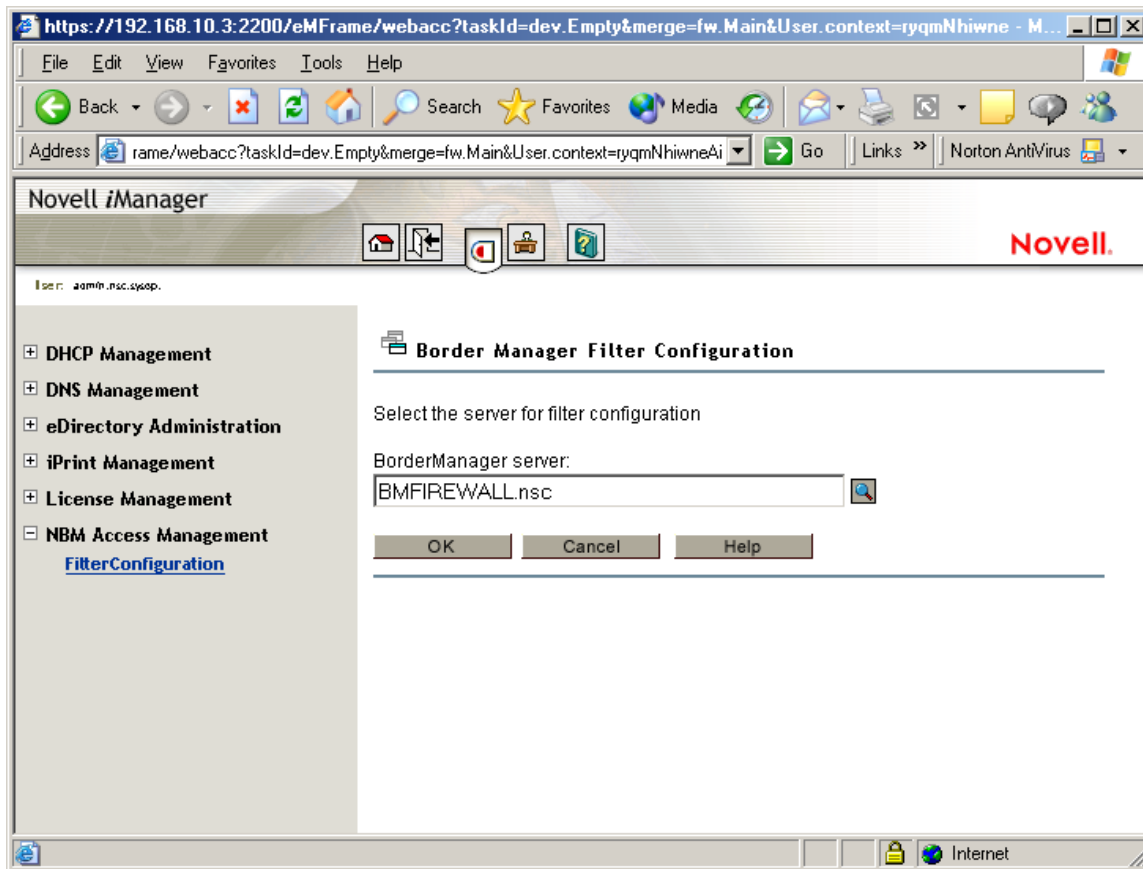


Figure 4-28 - BorderManager Server Selection in FilterConfiguration Menu

Once you have browsed to a BorderManager 3.7 server, it should show up in the selection menu as shown above in Figure 4-28.

Click on **OK** to continue.

Once you have selected a BorderManager 3.7 server, you should see a menu option allowing you to configure various types of filters or exceptions.

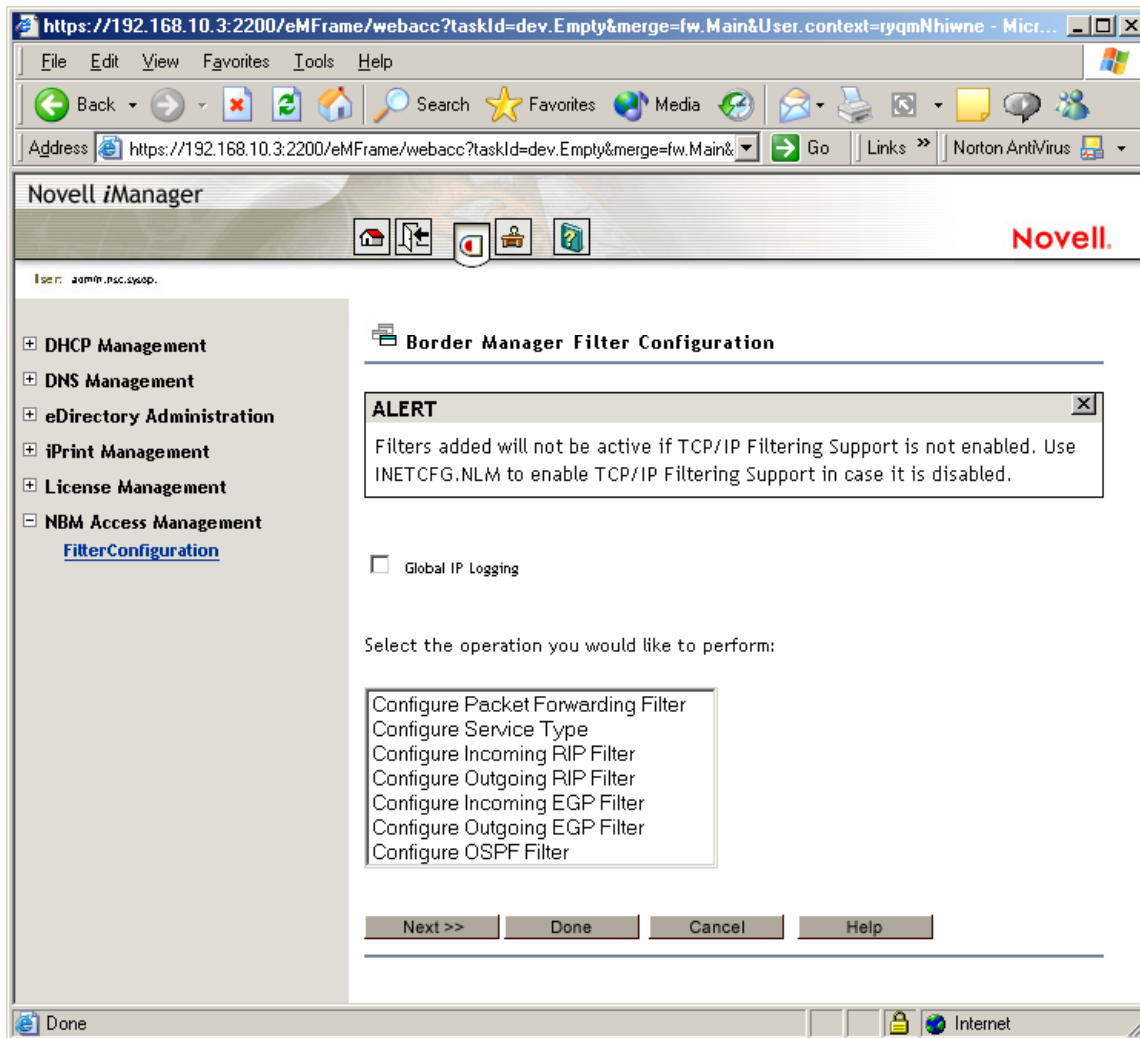


Figure 4-29 - BorderManager Filter Configuration Main Menu

Normally you will work within only two of the choices shown on the main menu:

- Configure Packet Forwarding Filter
- Configure Service Type

Configure Service Type is the only place within iManager that you can add, delete or modify packet type definitions. You **MUST** configure custom exceptions here **BEFORE** you can select them when adding a new filter exception. This is much different from

FILTCFG, which will allow you enter a service type definition menu as you are configuring a new filter exception.

Configure Packet Forwarding Filter is the option to select when you want to view, modify, add or delete either a filter or an exception.

### **Adding a New Service Type**

In the example which follows, a filter exception will be added using iManager to allow a fictitious application that wants to use TCP destination port 666 in order to function.

Because iManager only allows you to select a service type (packet filter definition) already defined when configuring a new exception, you must first add the service type.

Select **Configure Service Type** and click on **Next>>**.

---

**Note** The screenshot shown in Figure 4-30 contains some custom filter exceptions not included in BorderManager 3.7. Your Service Type Configuration list should look similar, but not exactly the same.

---

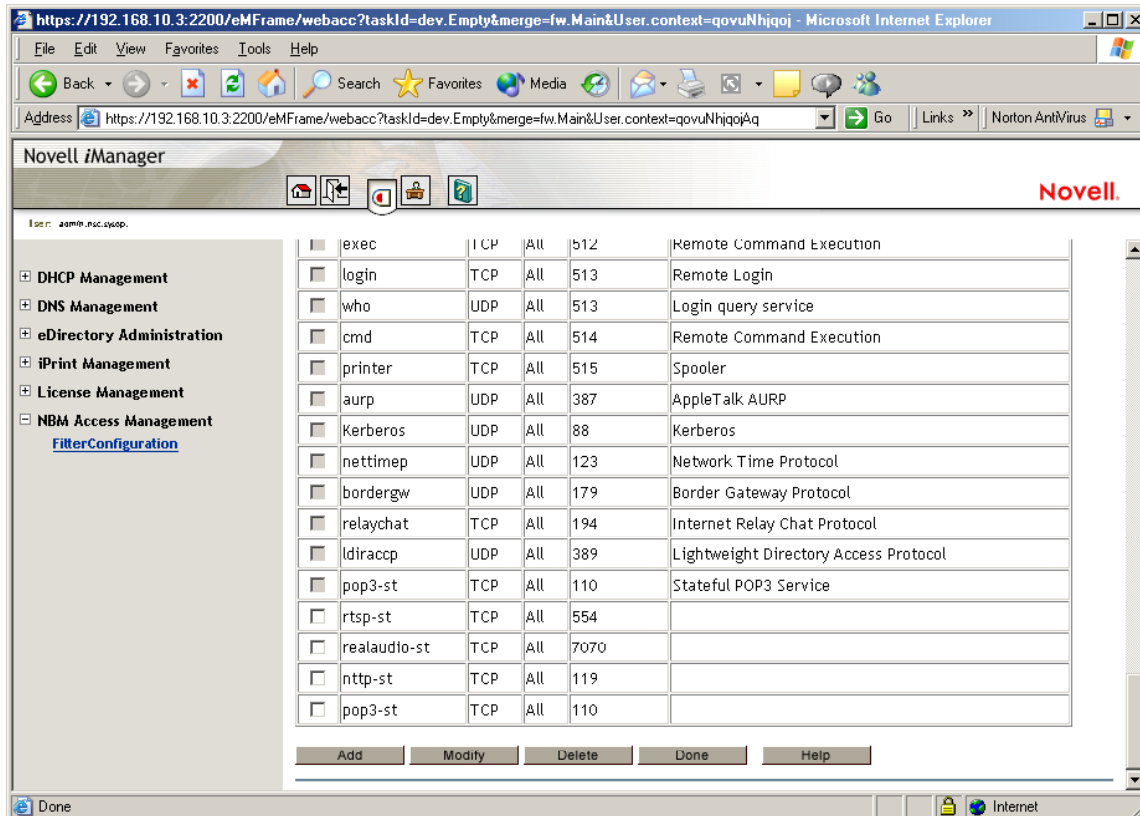


Figure 4-30 - List of Configured Service Types (Filter Definitions)

Some service types cannot be modified, and the check boxes for them will be grayed out.

Click on **Add** to add a new service type.

The screenshot shows a web browser window with the address bar displaying `https://192.168.10.3:2200/eMFrame/webacc/taskId=dev.Empty&merge=fw.Main&User.context=qov...`. The browser's address bar also shows `https://192.168.10.3:2200/eMFrame/webacc?taskId=dev.Empty`. The page title is "Novell iManager". The left sidebar contains a navigation menu with the following items: DHCP Management, DNS Management, eDirectory Administration, iPrint Management, License Management, NBM Access Management, and FilterConfiguration (highlighted). The main content area is titled "Service Type Configuration" and contains the "Add New Service Type" form. The form fields are: Name (666app-st), Protocol (Select from list, TCP(6)), Source Port (1024-65535), Destination Port (666), ACK Bit Filtering (Disabled), Stateful Filtering (Enabled), and Comment (Stateful 666 Application). At the bottom of the form are buttons for OK, Cancel, and Help.

Figure 4-31 - Service Type Definition Menu

The **Add Service Type** form appears. In this example, we want to configure a stateful filter exception for outbound TCP destination port 666, so we enter the following data into the HTML form.

**Name: 666app-st** – This is the name of the service type that will appear later when we select it when creating a filter exception.

**Protocol: TCP** – This protocol is selected from a list.

**Source Port: 1024-65535** – The range of source ports includes the high port range, and is typical for many, but not all, TCP applications. (Some types of FTP and NTP exceptions are notable exceptions). You must know the proper source port range for an application before defining the service type. If you do not know the source ports that the application makes use of, you can leave the selection blank, and all source ports will be enabled.

**Destination Port: 666** – The destination port typically defines the application when making a connection to a server. You must know the destination port before configuring the service type.

**ACK Bit Filtering: disabled** – For this example, we will not enable ACK Bit Filtering.

**Stateful Filtering: enabled** – For this example, we will enable stateful filtering.

**Comment: Stateful 666 Application** – Always add some sort of descriptive comment for the service type.

Click on **OK** to add the definition.



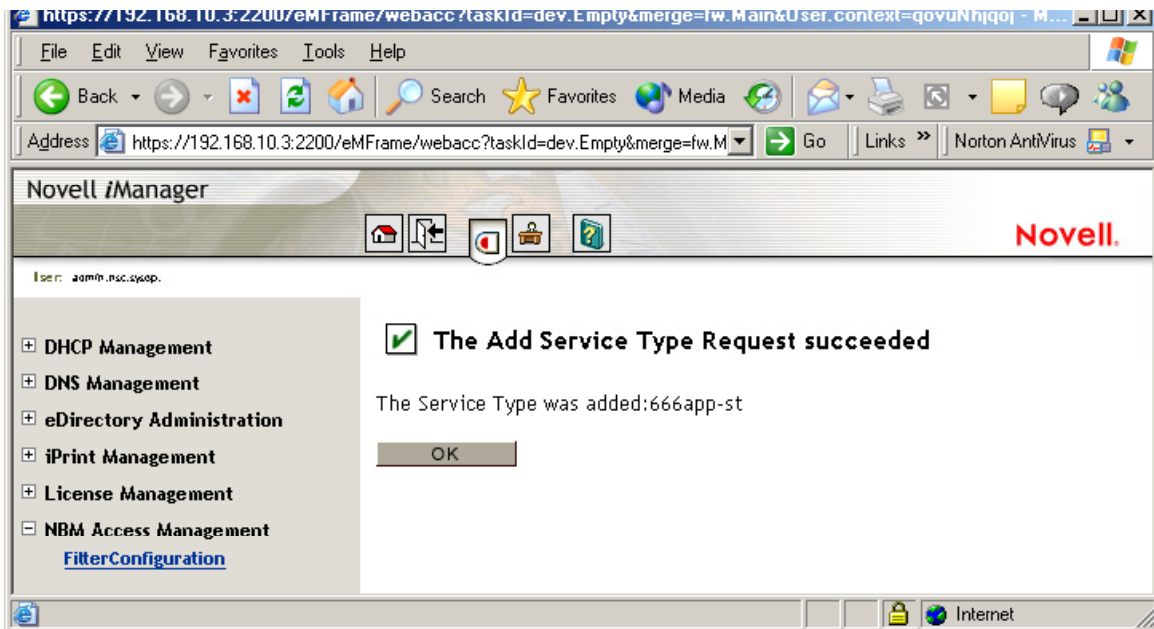


Figure 4-32 - Add Service Type Request Succeeded Menu

You should see a web page indicating that the add service type request succeeded. Click on **OK** to return to the main menu.

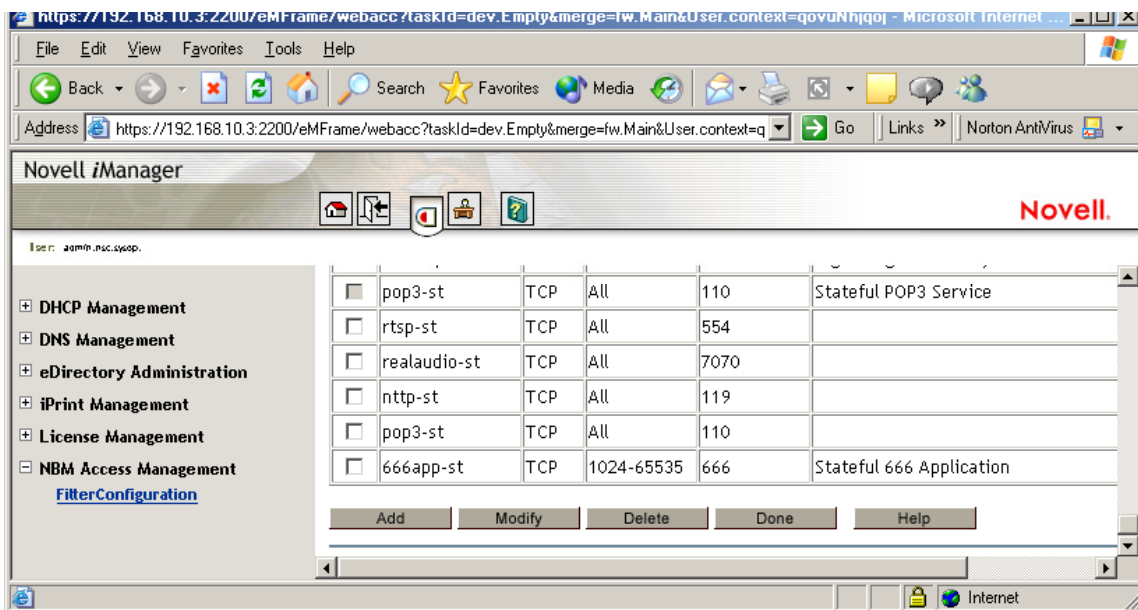


Figure 4-33 - New Service Type Added to List

The new service type definition should show up at the bottom of the list of defined service types.

Click on **Done** to return to the main menu.

## Adding a Filter Exception with the New Service Type Definition

Now that the new service type definition for our 666 application has been added, you can begin to use it in a filter exception.

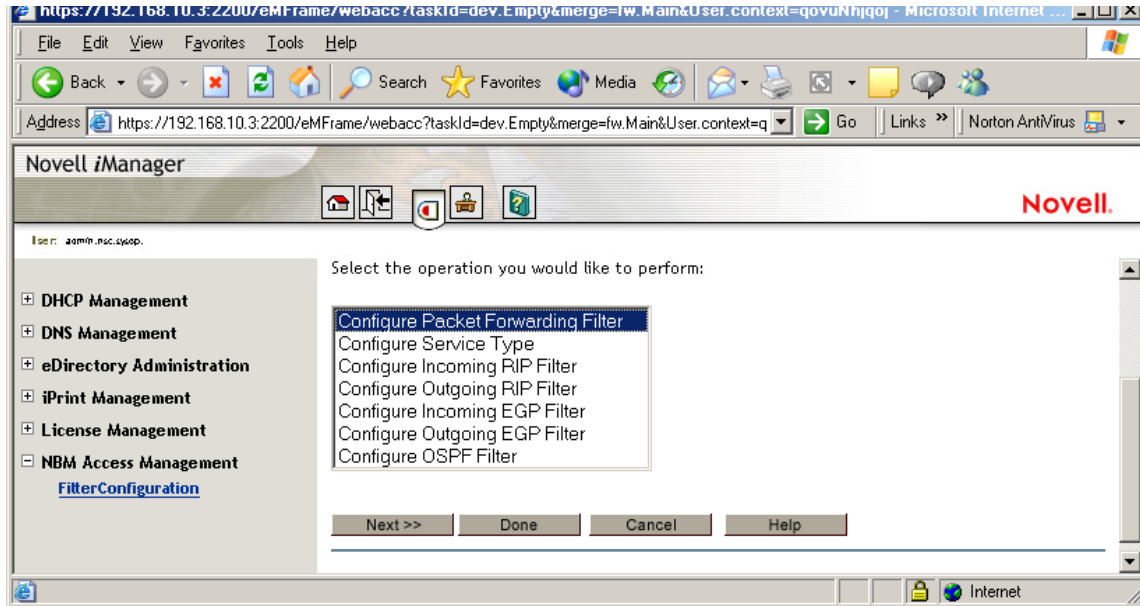


Figure 4-34 - Select Packet Forwarding Filter

From the FilterConfiguration main menu, select **Configure Packet Forwarding Filter** and then click on **Next>>** to begin adding a new filter exception.

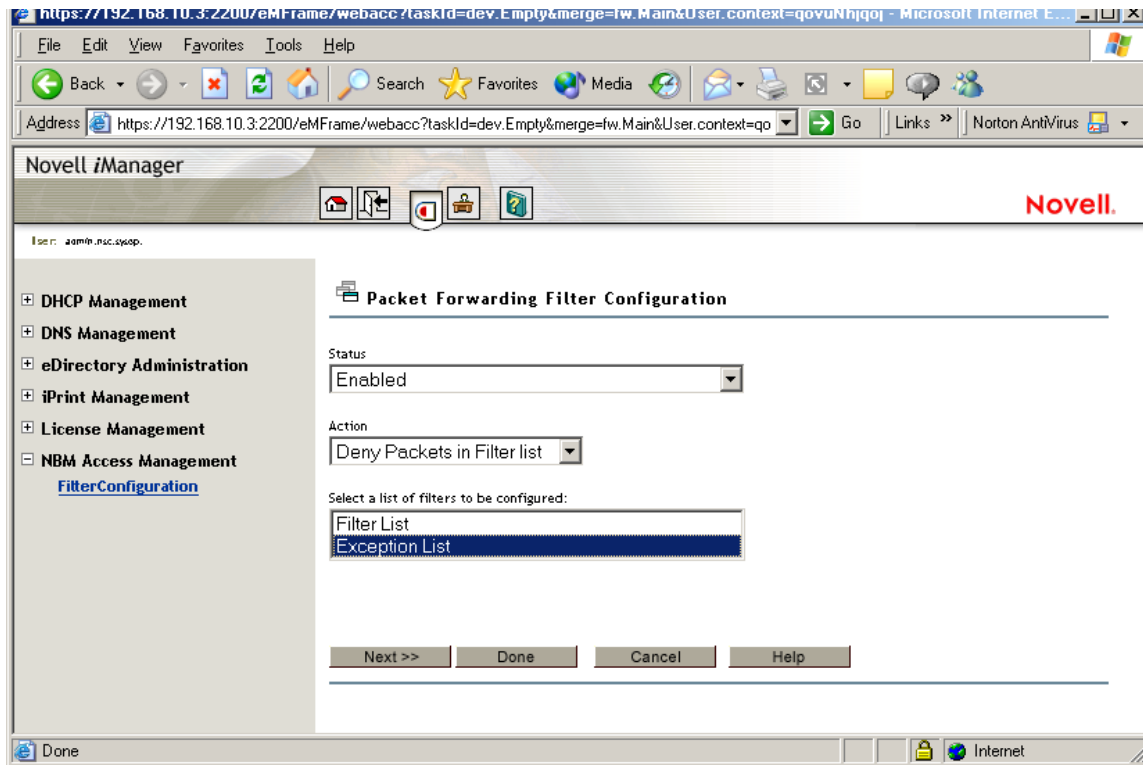


Figure 4-35 - Select Exception List

Select **Exception List** and then click on **Next>>** to configure a filter exception.

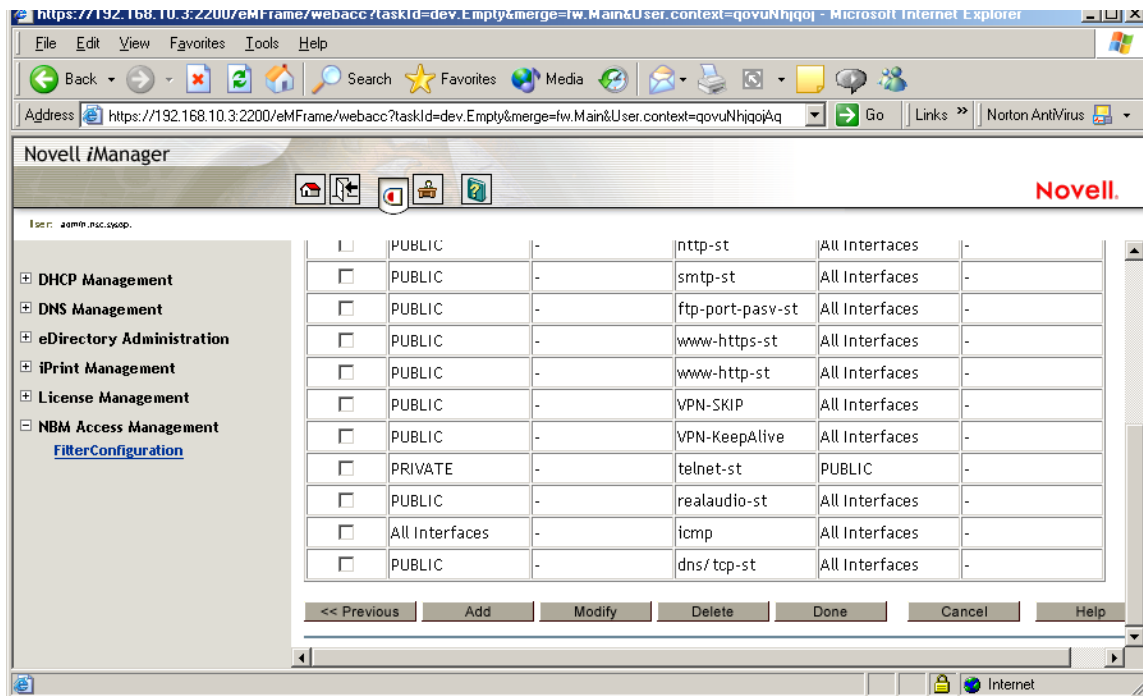


Figure 4-36 - List of Existing Filter Exceptions

A list of filter exceptions currently in place should appear.

Scroll to the bottom of the page, and click on **Add** to add a new filter exception.

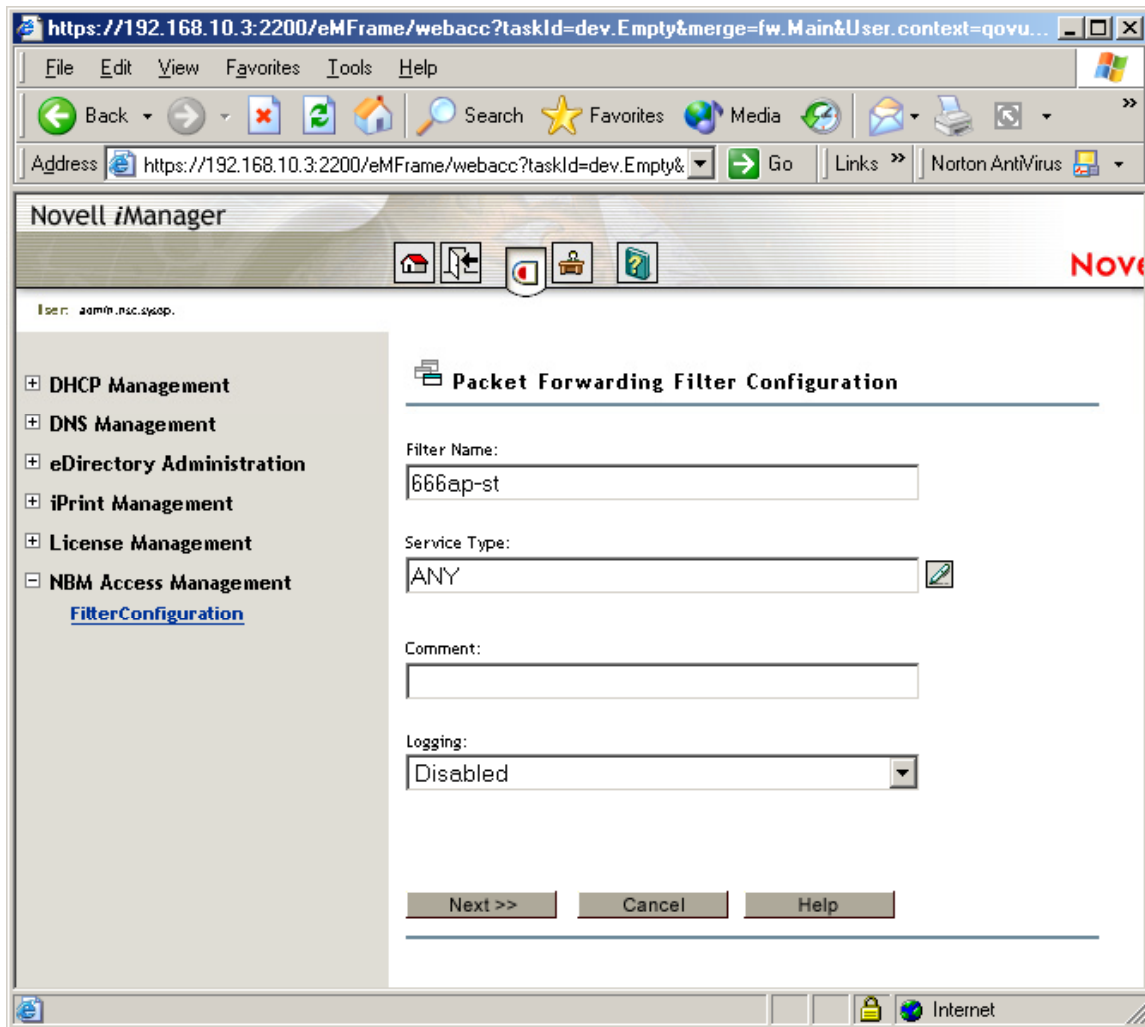


Figure 4-37 – Configure Filter Exception - Add NDS Name

The first entry to be made in the Packet Forwarding Filter Configuration dialog is **Filter Name**.

The Filter Name is the name of the NDS object that will be created in the NBMRuleContainer container that will hold the filter definition data in NDS. If you create a filter exception using FILTCFG, a name will be created automatically. You cannot enter a name for a filter or exception that already exists in the current NBMRuleContainer object.

In this example, a name of **666ap-st** was chosen.

Select the **browse** button for the **Service Type** entry to choose from the list of defined service types. This will take you to a list where you can choose the service type definition set up previously.

You cannot add a new service type at this point! You must have already made your service type definitions before getting to this menu option.

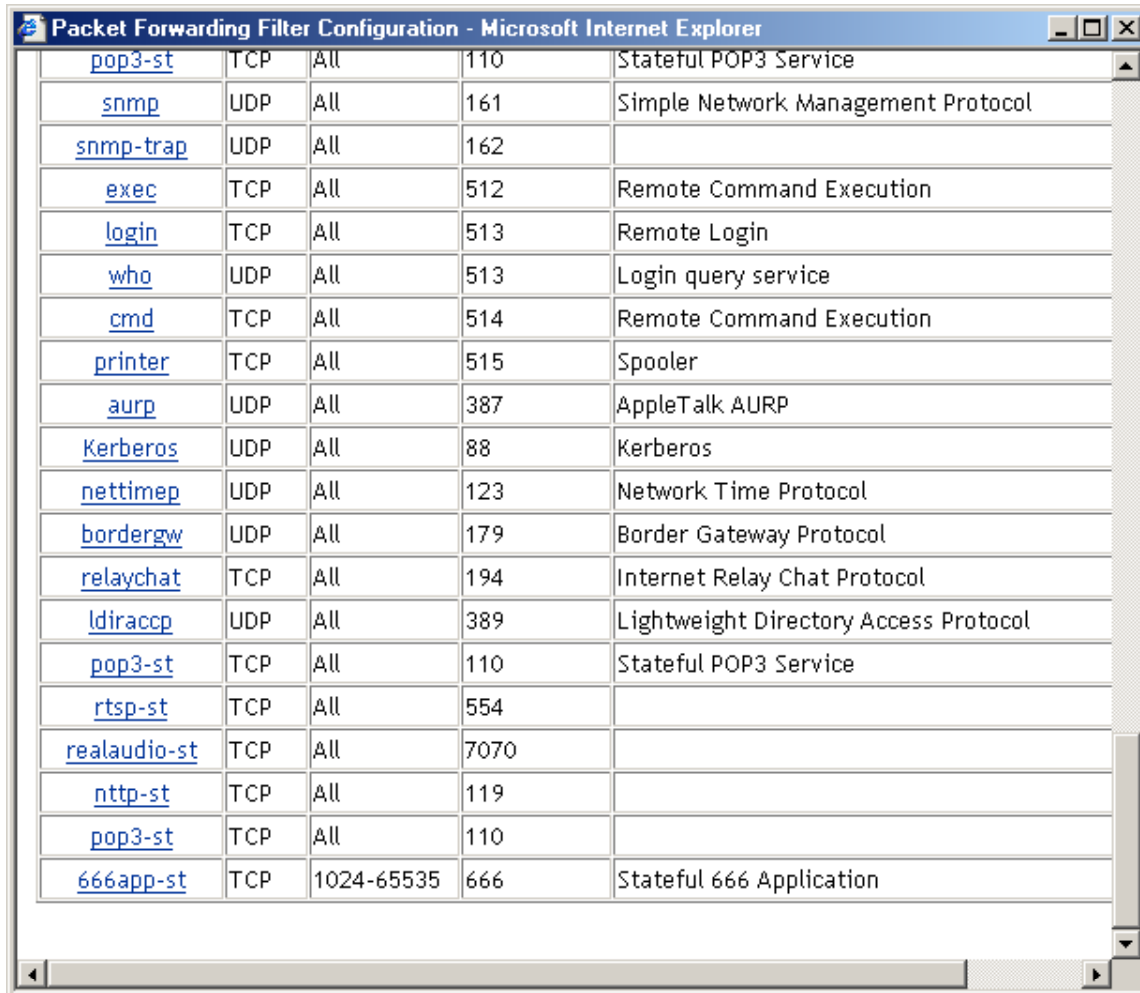


Figure 4-38 – Configure Filter Exception – Choose Custom Service Type Defined Earlier from List of Available Service Types

You should now see a list of all available Service Types available for use in a filter or filter exception.

Scroll to the **666ap-st** Service Type created early, and click on the link to select it.

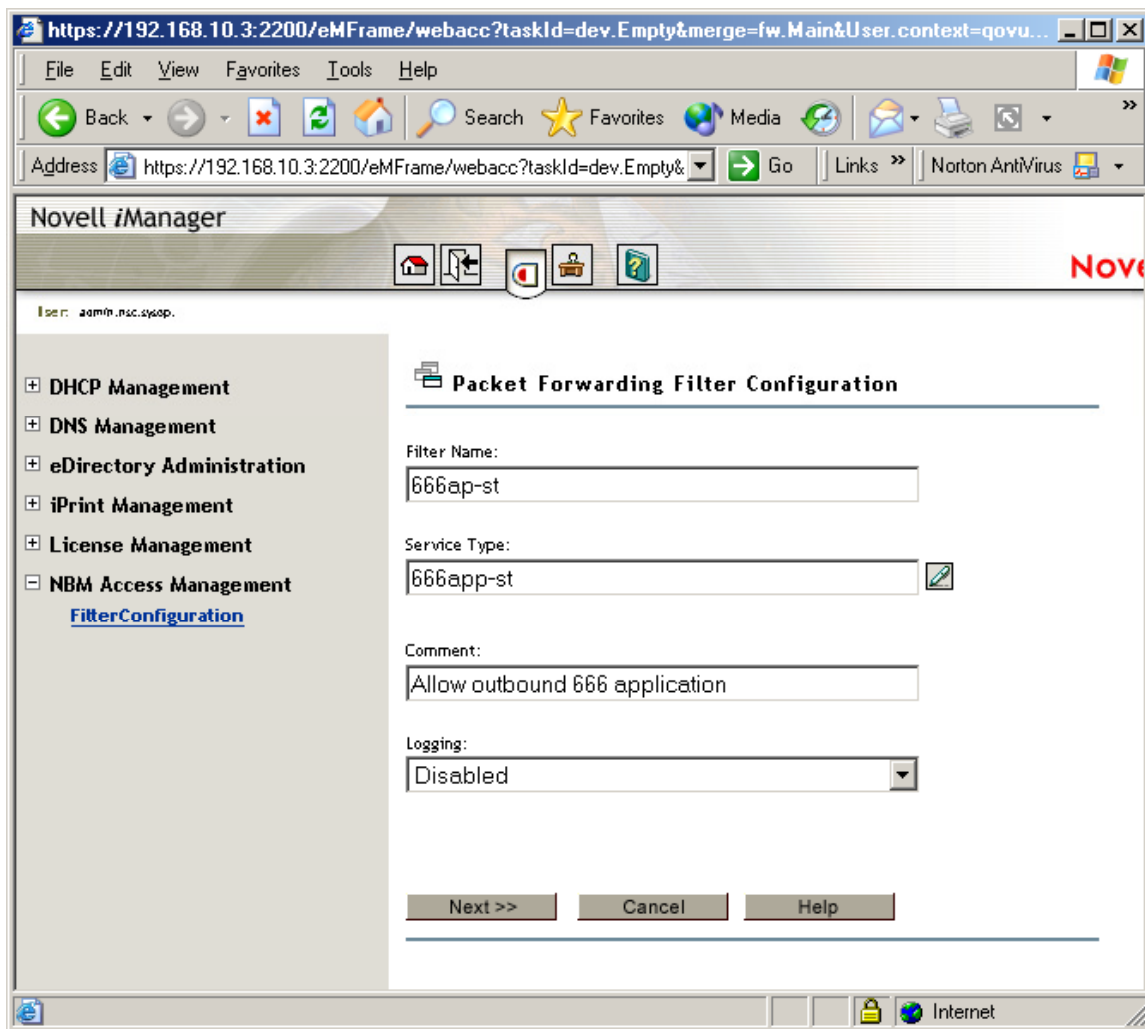


Figure 4-39 - Configure Filter Exception - Add Descriptive Comment

The **666app-st** Service Type appears in the configuration menu.

Add a description comment for the filter exception.

**NEVER ADD A FILTER OR FILTER EXCEPTION WITHOUT ADDING SOME DESCRIPTIVE COMMENT!**

Click **Next>>** to continue.

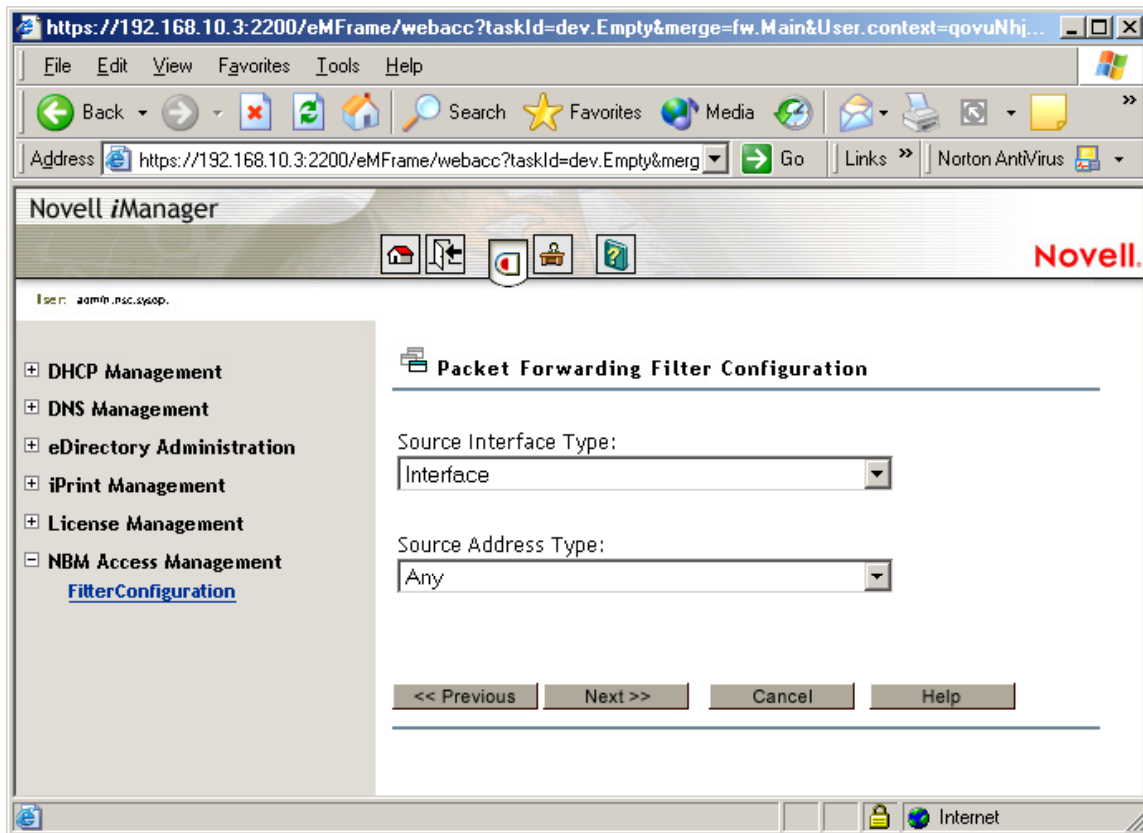


Figure 4-40 - Configure Filter Exception - Choose Source Type (Interface and/or Address)

The first of several pages appears which define the direction of the filter exception. On the page shown in Figure 4-40, leave the choices at the default in order to create a filter exception with a source that will be the Private interface.

You cannot select the actual source interface or address on this menu. That has to be done on the following menu. (FILTCFG is much faster and more intuitive...)

Click on **Next>>** to continue.



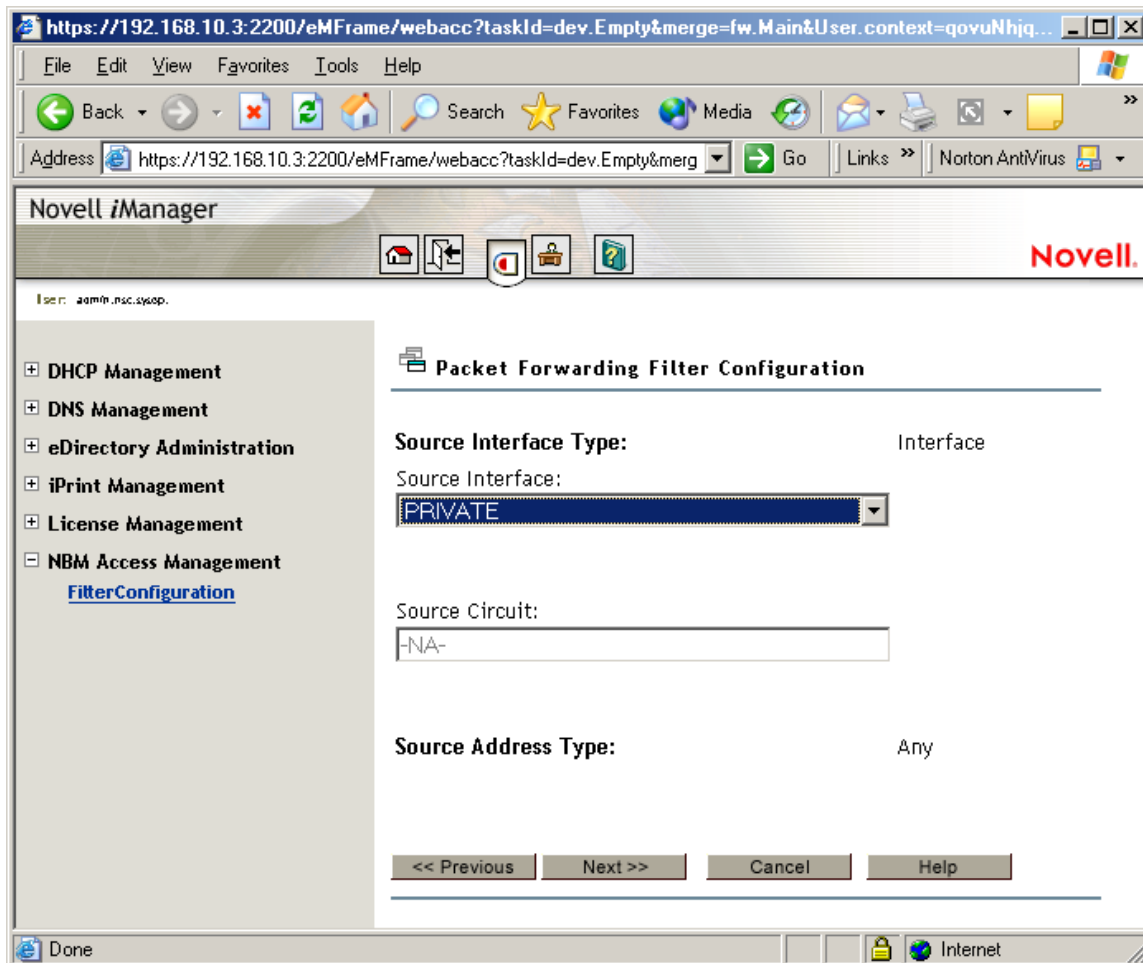


Figure 4-41 - Configure Filter Exception - Select Source Interface

Select the **Private** interface in order for the filter exception to allow an outbound filter exception from the internal LAN to the Internet.

(You might select a source interface of Public to allow only a proxy to make a connection to the Internet).

Click on **Next>>** to continue.

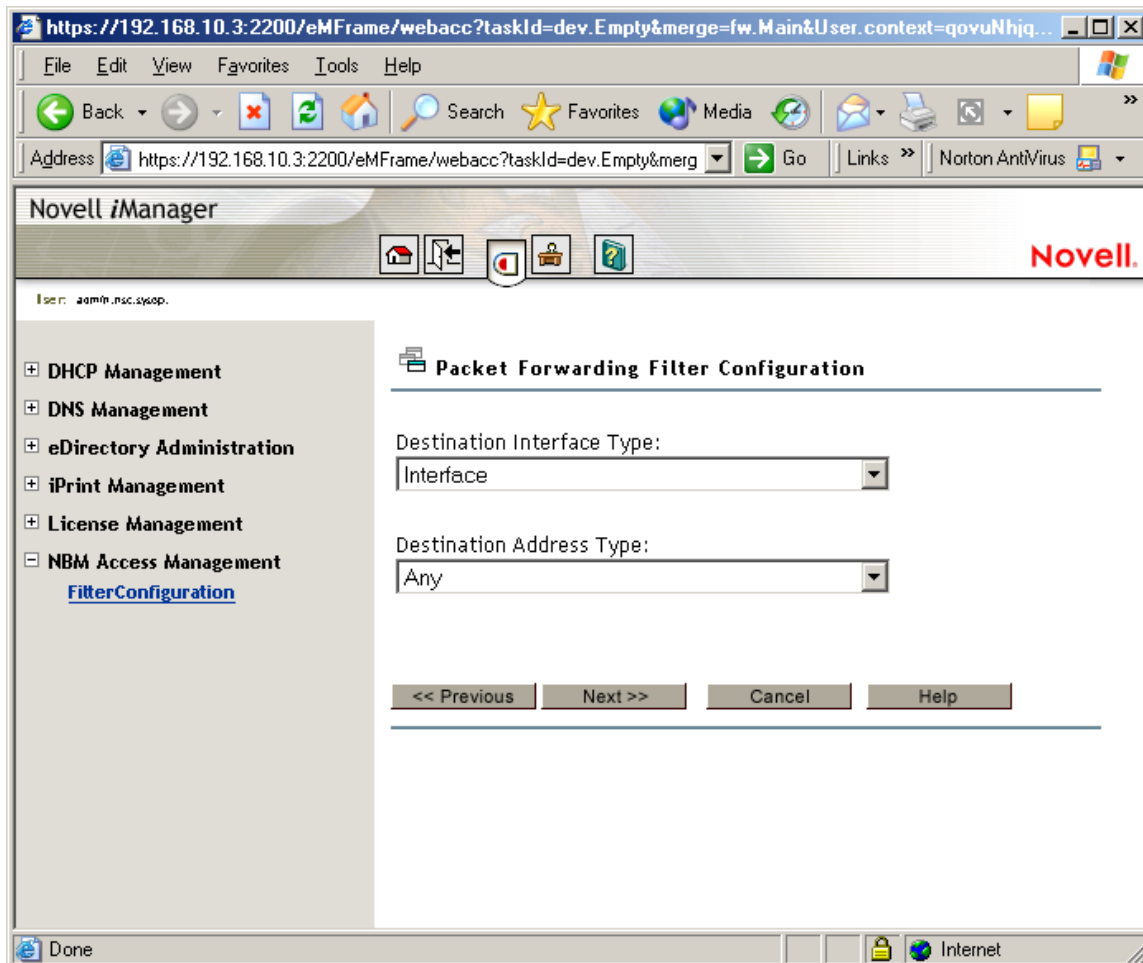


Figure 4-42 - Configure Filter Exception - Choose Destination Type (Interface and/or Address)

A menu selection allowing you to choose a destination type of interface or address. For this example (outbound stateful filter for our mythical 666 application), leave the entry at the default, which will allow you to specify a destination interface of Public on the next menu.

(If you never liked FILTCFG, you may be changing your mind at this point...)

Click **Next>>** to continue.

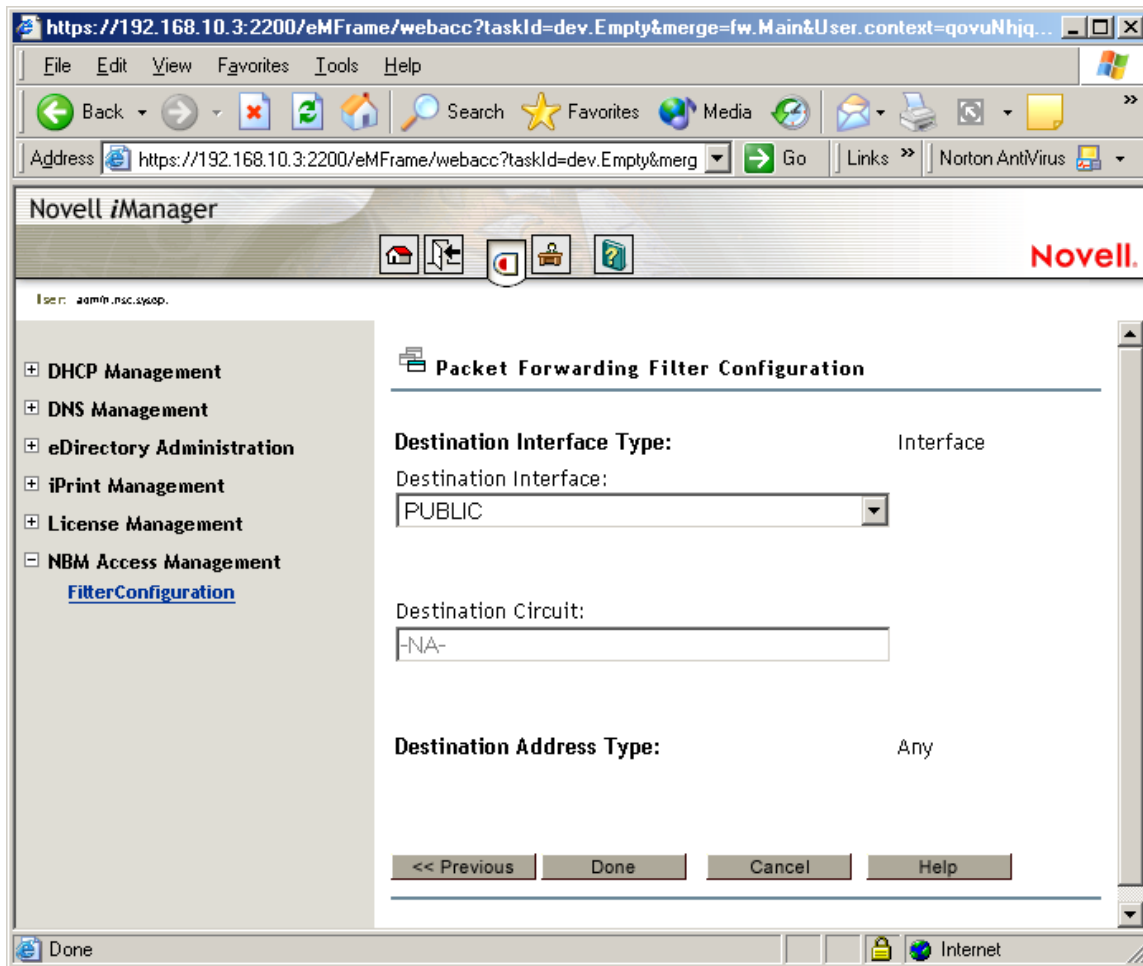


Figure 4-43 - Configure Filter Exception - Select Public Interface as Destination

Choose the **Public** destination interface.

Click **Done** to save the new filter exception.

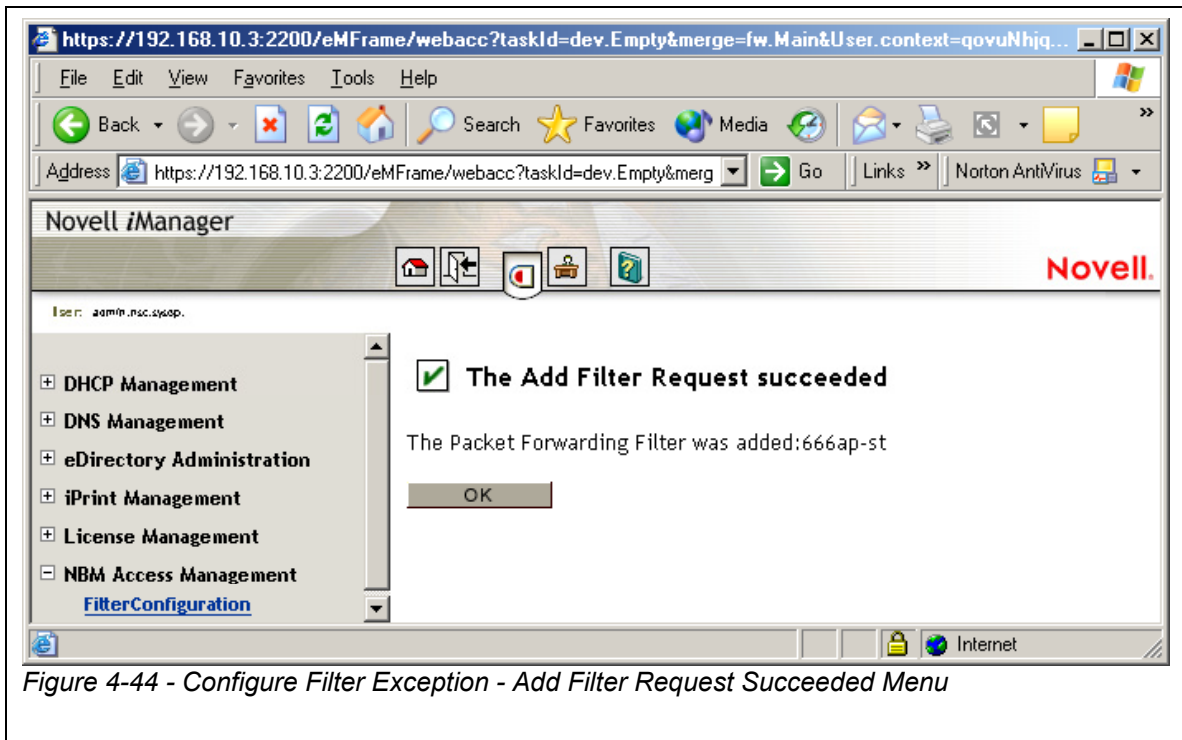


Figure 4-44 - Configure Filter Exception - Add Filter Request Succeeded Menu

You should see a page telling you that the add filter request succeeded.

Click on **OK** to return to the **Packet Forwarding Filter Configuration** menu.

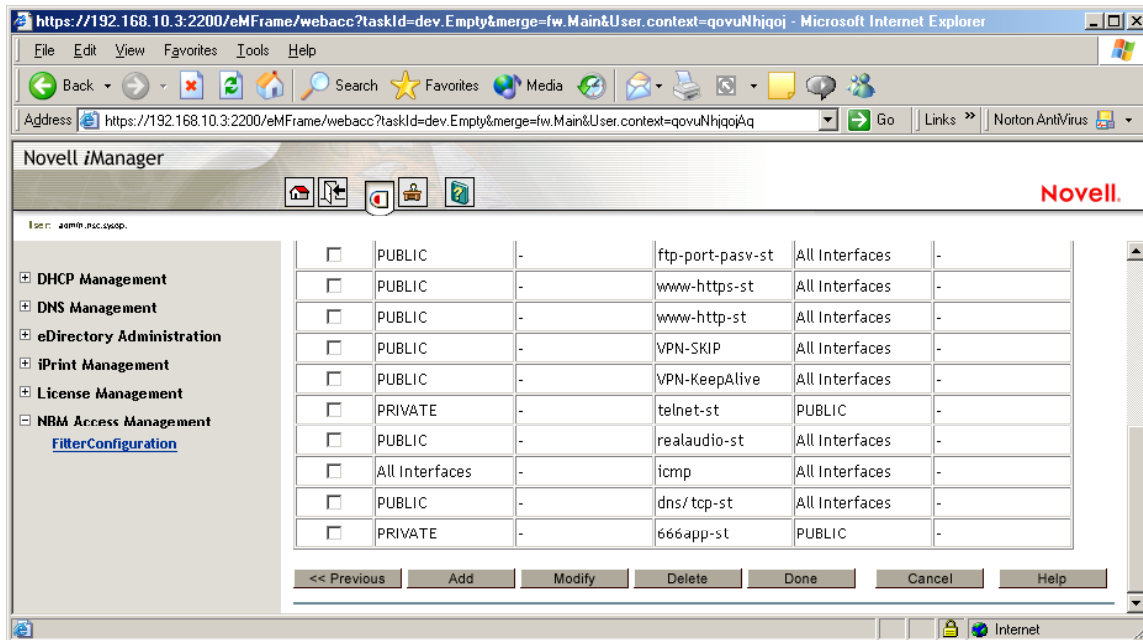


Figure 4-45 - Configure Filter Exception - New Exception Appears in List

You should see your new filter exception in the list of filter exceptions. Click **Done** to exit the filter configuration menus.

## ConsoleOne View of BorderManager 3.7 Filters

BorderManager 3.7 stores filters and filter exceptions as NDS objects. The following screenshot shows what the filter objects configured in the BMFIREWALL server looked like following the addition of the 666app-st filter exception shown in the last section.

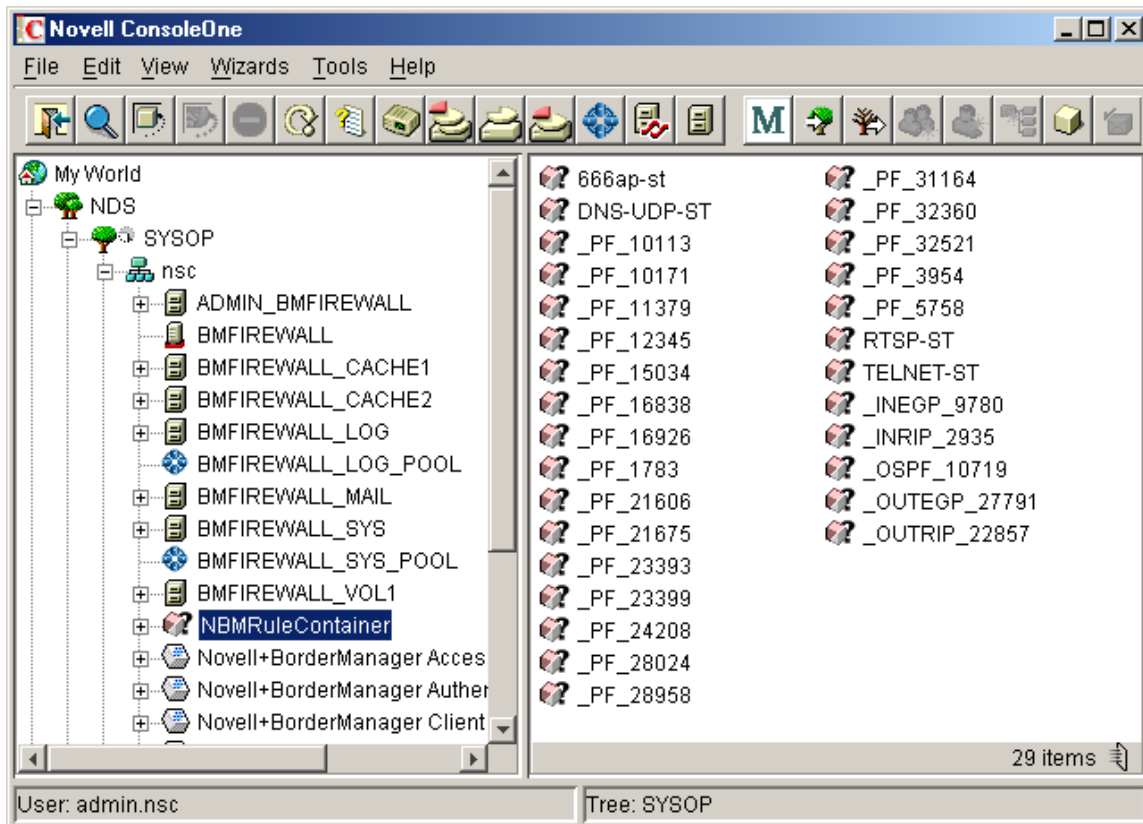


Figure 4-46 - ConsoleOne View of Filter Objects in NBMRuleContainer Object

The exceptions configured either in FILTCFG or by the BorderManager installation process all start with an underscore character.

You can manually edit selected characteristics of the filter exceptions within ConsoleOne, though it is not advisable due to the critical nature of the syntax used.

Selecting the **666ap-st** object leads to the following screenshot.

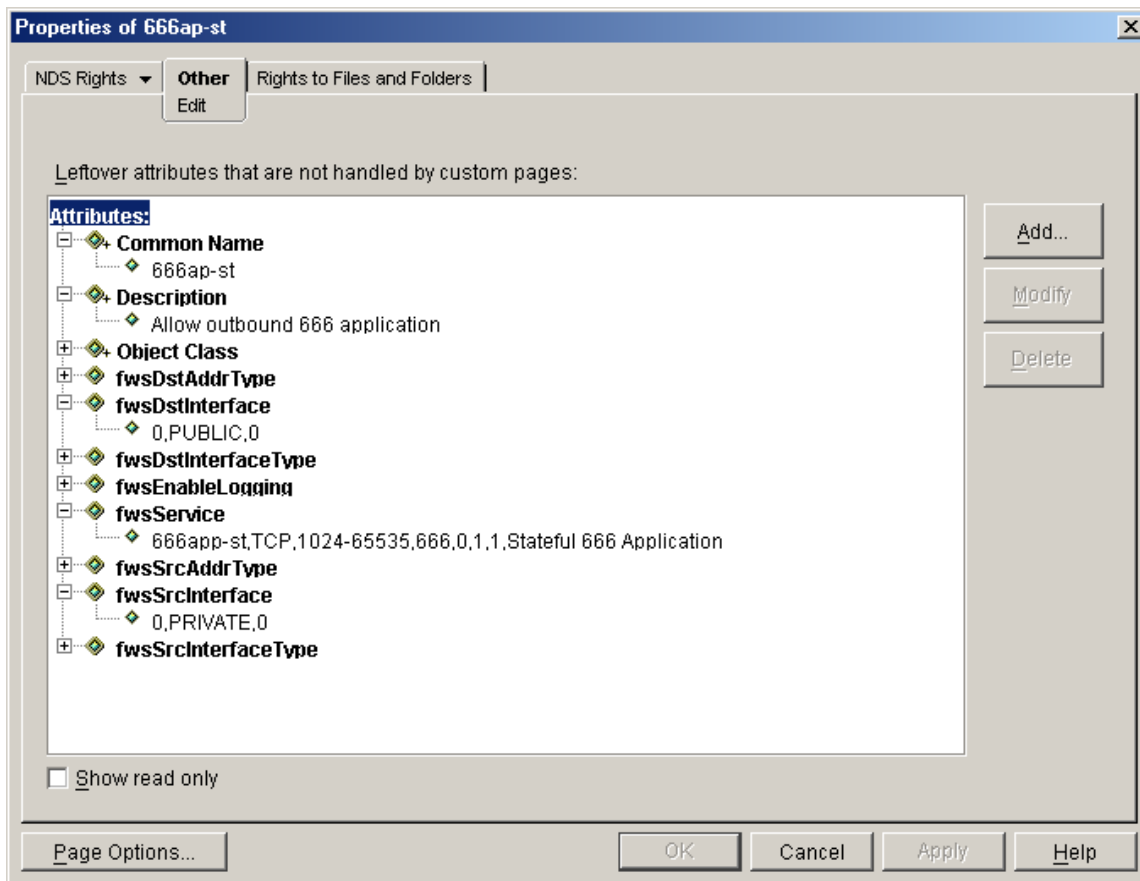


Figure 4-47 - ConsoleOne View of BorderManager 3.7 Filter Object Attributes

The screenshot shown in Figure 4-47 above shows some of the new 666app-st filter exception attributes in ConsoleOne.

# Chapter 5 Example Outbound Filter Exceptions

---

All of the examples in this chapter are for connections *initiated by a client on the internal LAN*. The first packet is sent from the inside of the BorderManager server to the outside, hence the term 'outbound'.

The examples shown show only how the filter exceptions appear in FILTCFG, since it is so much more efficient at displaying the information than iManager.

BorderManager 3.7 filters configured as part of a fresh installation are also shown.



## AIM (AOL Instant Messenger) / AOL

AOL Instant Messenger is something like ICQ, but more limited in features. As you don't really use 'real-time' chat with AOL, you do not have to set up inbound TCP connectivity for a range of listening ports. All you have to do is to set up a stateful filter exception that opens up TCP port 5190 as shown. The same exception will work to allow AOL as well.

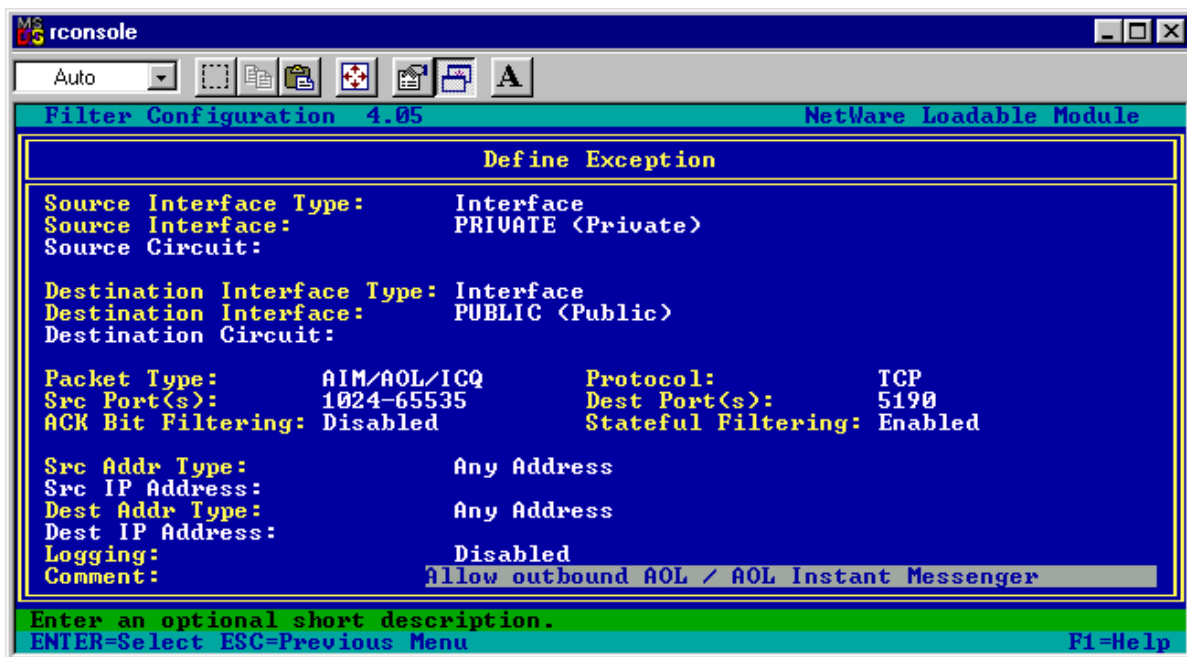


Figure 5-1 - Filter Exception for Outbound AOL / AOL Instant Messenger / ICQ

Figure 5-1 shows a stateful filter exception that will allow AIM or AOL. Later versions of ICQ can also be configured to use destination port 5190.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 5190
- Stateful filtering: Enabled

---

**Note** DNS must also be functional for AIM to work, whether by a DNS filter exception, internal DNS server, or DNS proxy on BorderManager.

---

## Cisco VPN Client

Cisco has a fairly new (as of this writing) version of their VPN client that does work behind a NAT connection. Older versions of the Cisco VPN client do not work through NAT, and I am not sure what Cisco components are required to make it work. (You might try a test of the client with filters disabled – if the client VPN doesn't work then, filter exceptions won't help you!)

The Cisco VPN client that does work over NAT uses only two ports – UDP port 10000 and UDP port 500.

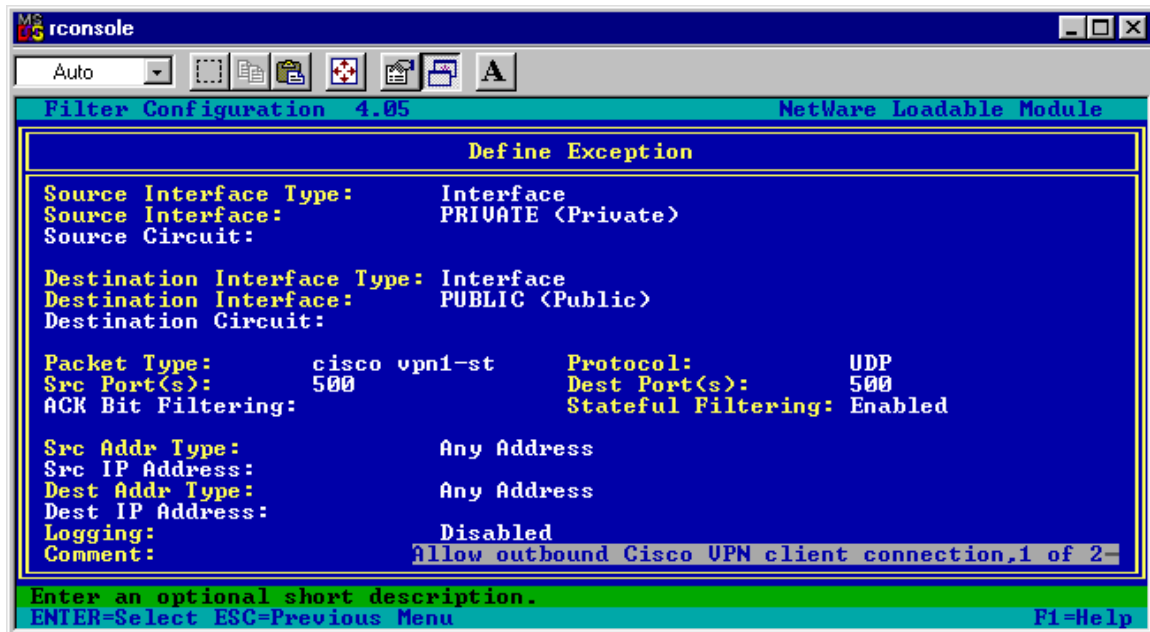


Figure 5-2 - Filter Exception for Cisco VPN Client Connection, Part 1 of 2

The filter exception shown in Figure 5-2 is one of two filter exceptions necessary to allow Cisco's VPN client to work through a dynamic NAT connection behind a BorderManager firewall.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source port: 500
- Destination port: 500
- Stateful filtering: Enabled

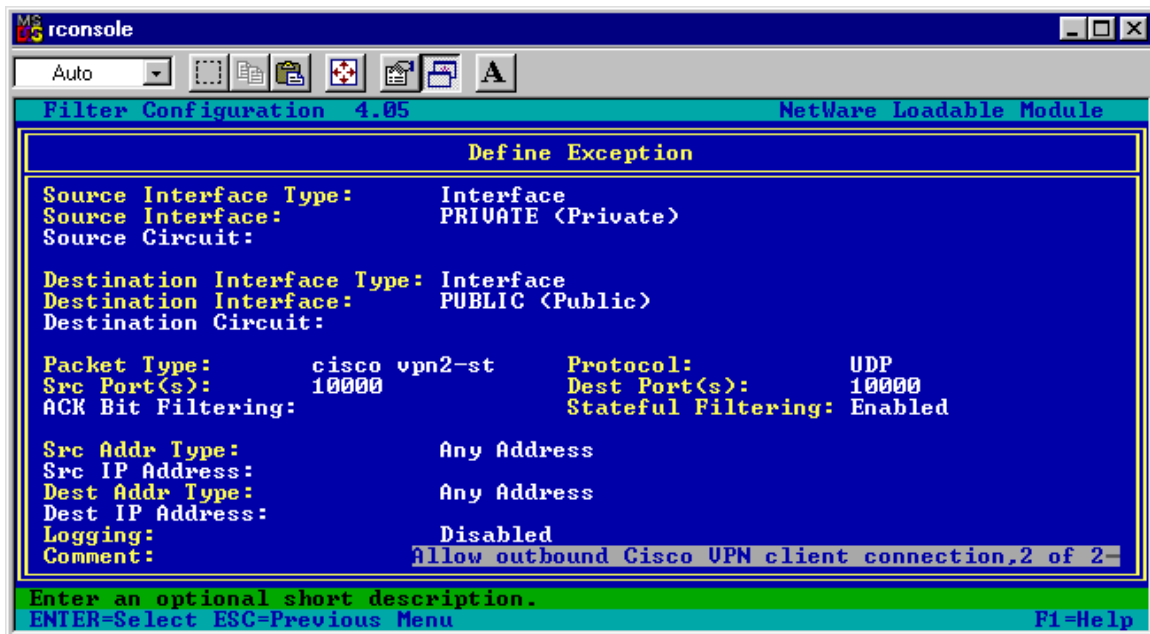


Figure 5-3 - Filter Exception for Cisco VPN Client Connection, Part 2 of 2

The filter exception shown in Figure 5-3 is the second of two filter exceptions required to allow a Cisco VPN client to connect through a dynamic NAT connection behind a BorderManager firewall.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source port: 10000
- Destination port: 10000
- Stateful filtering: Enabled

## Citrix WinFrame / MetaFrame

These filter exceptions will allow the Citrix ICA client traffic and the Citrix browser-based client traffic **out** of the BorderManager firewall.

Because Citrix has used two different client technologies, one a stand-alone based client (ICA) and the other a snap-in component of a web browser, different filter exceptions may be required.

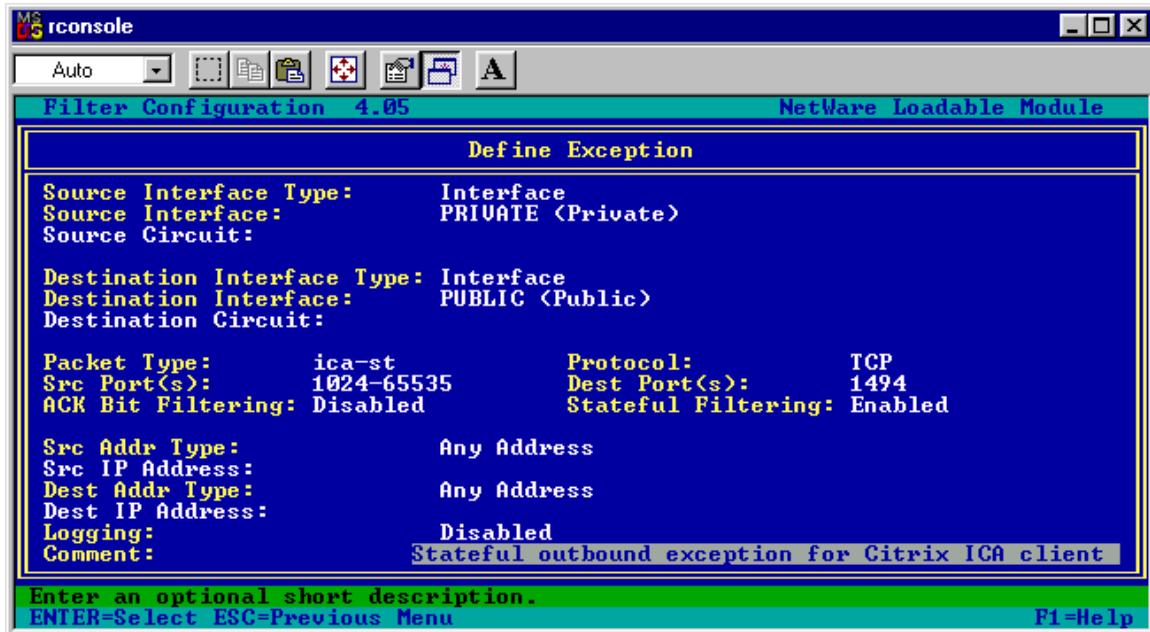


Figure 5-4 - Filter Exception for Outbound Citrix ICA Client

The filter exception shown in Figure 5-4 allows the stand-alone ICA client to communicate with a remote Citrix WinFrame / MetaFrame host outside the BorderManager firewall.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 1494
- Stateful filtering: Enabled

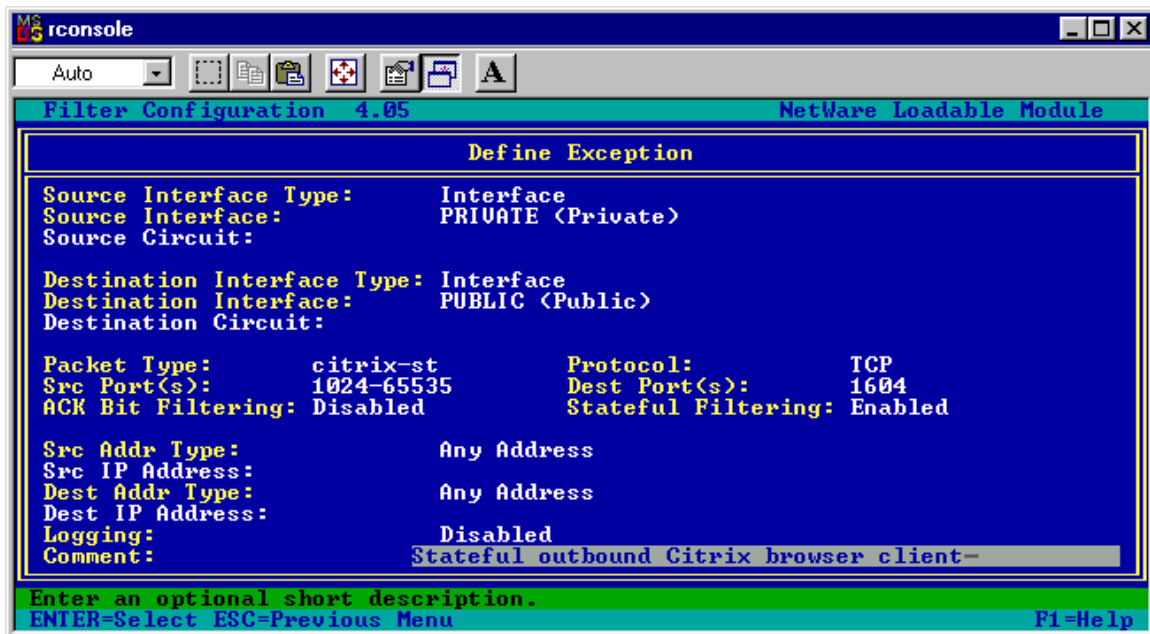


Figure 5-5 - Filter Exception for Outbound Citrix Browser Client

The filter exception shown in Figure 5-5 allows the browser-based (and later versions of the stand-alone ICA) client to communicate with a remote Citrix WinFrame / MetaFrame host outside the BorderManager firewall.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 1604
- Stateful filtering: Enabled

## Client-to-Site VPN over NAT

These filter exceptions are needed to allow a host to make an outbound Novell BorderManager Client-to-Site VPN connection over dynamic NAT.

---

**Note** Only BorderManager version 3.6 (or later) can accept a VPN client connection when the client is behind a NAT router hop. This will not work for BorderManager 2.1, 3.0 or 3.5 Client-to-Site VPN, or BorderManager 3.6 if an earlier VPN client is installed on the remote PC.

---

The BorderManager VPN server must allow inbound UDP port 2010 to the public IP address, in addition to the filter exceptions configured by VPNCFG.NLM.

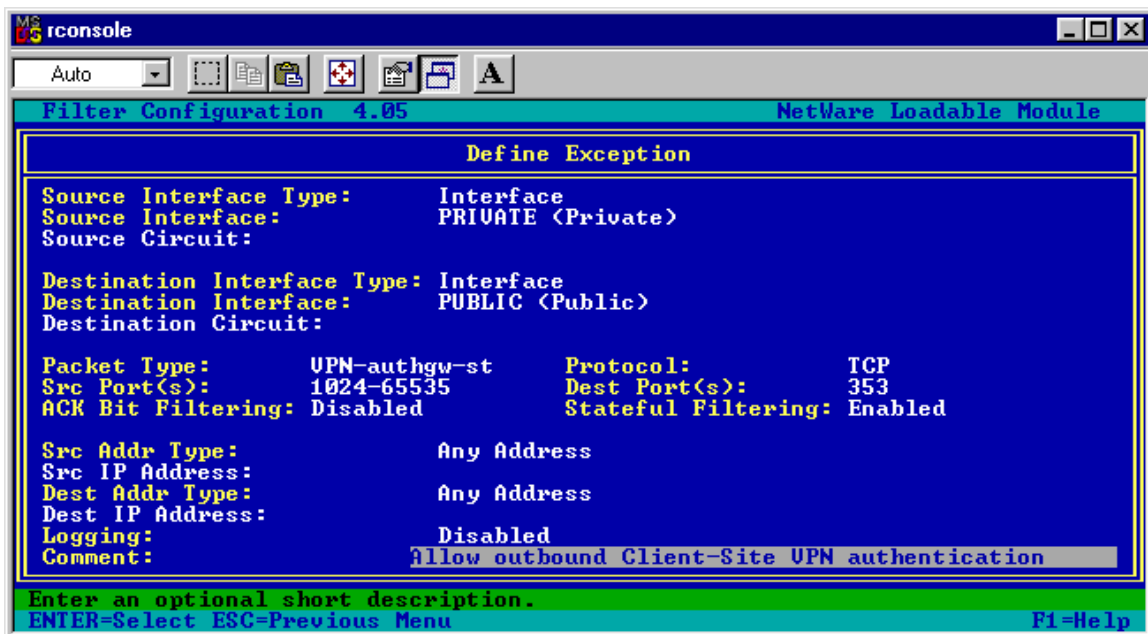


Figure 5-6 - Filter Exception for Initial BorderManager Client-to-Site VPN Authentication over NAT

The filter exception shown in Figure 5-6 allows the initial Client-to-Site VPN connection to be made by allowing the authentication information to pass through.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 353
- Stateful filtering: Enabled

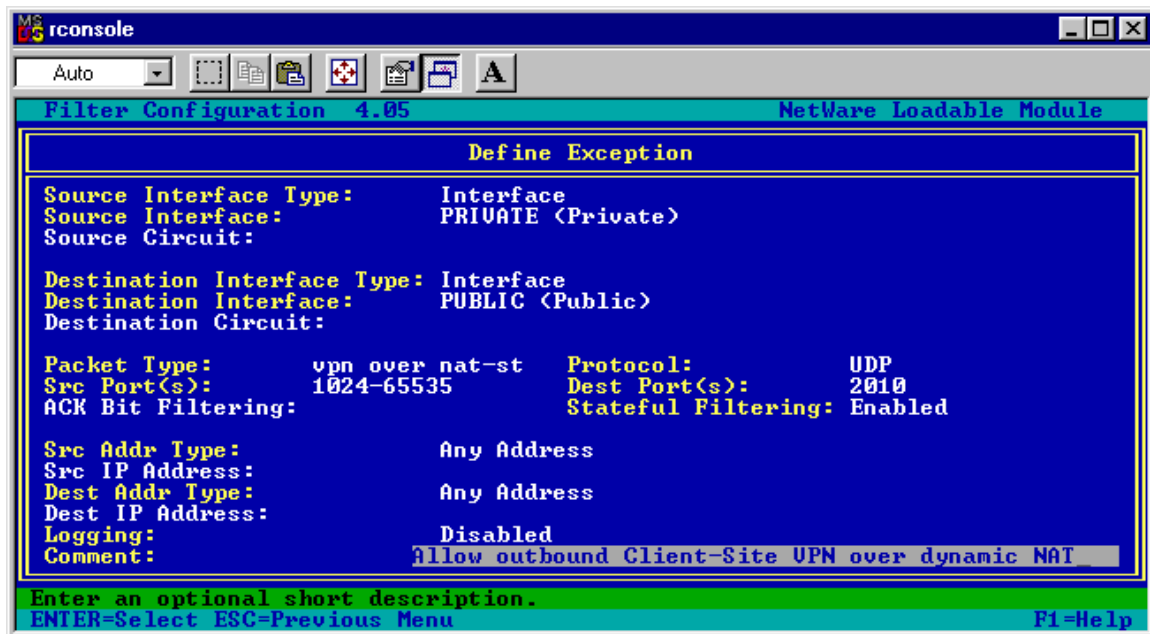


Figure 5-7 - Filter Exception for Outbound BorderManager Client-Site VPN over NAT

The filter exception shown in Figure 5-7 allows the Client-to-Site VPN data to be passed through NAT using UDP port 2010.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source ports: 1024-65535
- Destination port: 2010
- Stateful filtering: Enabled

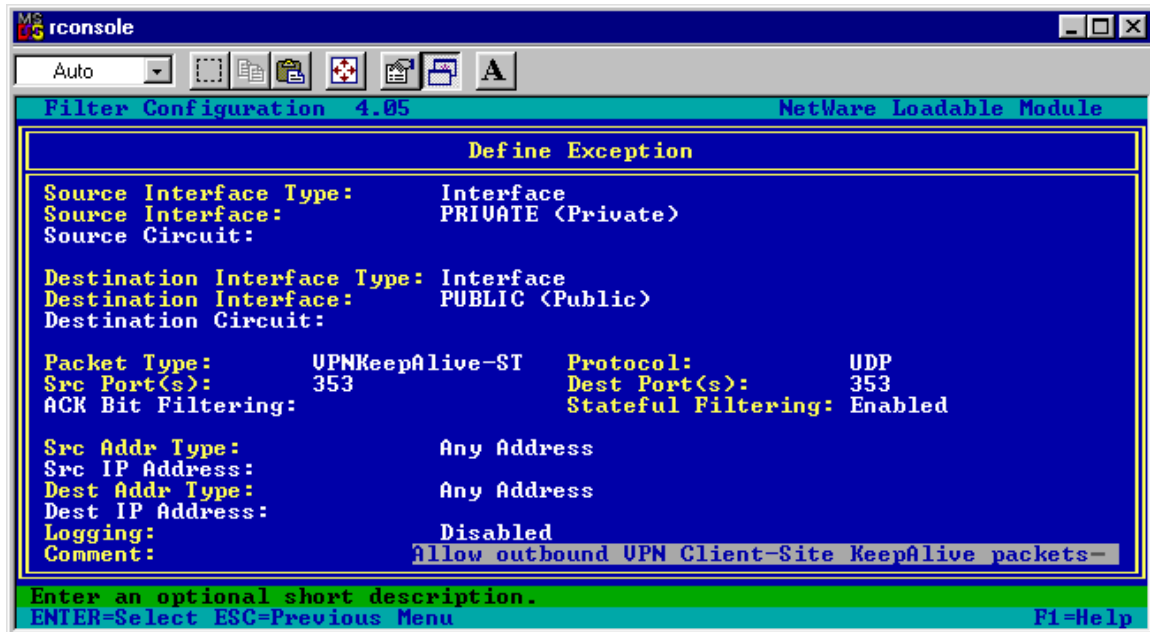


Figure 5-8 - Filter Exception for BorderManager Client-to-Site VPN KeepAlive Packets over Dynamic NAT

The filter exception shown in Figure 5-8 allows for the VPN keep-alive packets necessary to maintain a BorderManager Client-to-Site VPN connection, once established.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source ports: 353
- Destination port: 353
- Stateful filtering: Enabled



# CLNTRUST

The CLNTRUST utility supplied with BorderManager 3.x is extremely useful when you have enabled Proxy Authentication. Unfortunately, it sometimes tries to communicate with the BorderManager public IP address, where it is then blocked by the default filters.

The following filter exception allowing TCP port 524 to the public IP address seems to allow CLNTRUST to work more reliably.

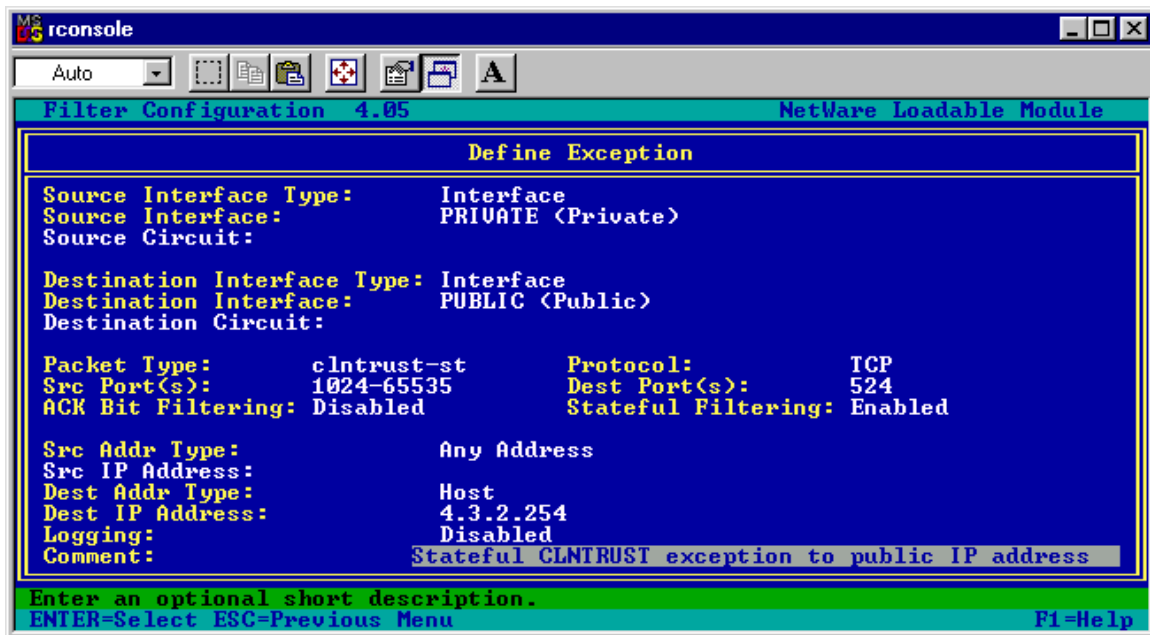


Figure 5-9 - Filter Exception for Internal CLNTRUST Traffic to Public IP Address

The filter exception shown in Figure 5-9 allows often fixes random problems with CLNTRUST not working.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 524
- Stateful filtering: Enabled
- Destination IP Address: <your BorderManager server public IP Address>

This problem may be prevented in the first place by doing one of the following:

- 1) In Monitor, Server Parameters, NCP, put the BorderManager private IP address(es) in the **NCP Include IP Address** field. The NCP parameters may not be present until you apply the latest NetWare patch.
- 2) In SYS:\ETC\TCPIP.CFG, the first TCP/IP binding should always be the private IP address.

# DNS

## Outbound DNS Filter Exceptions for Internal Hosts

You can also use the DNS proxy in BorderManager 3.x, but if you simply want to pass DNS port 53 requests out through the BorderManager servers so that internal hosts can access external DNS servers, here is how to do it with a stateful filter exception. (I do not recommend using the DNS Proxy if you have an internal DNS server.)

You want to create a filter definition called new dns/udp-st, and then possibly another one called new dns/tcp-st. The first exception will be to allow DNS queries over UDP (commonly used), and the second will be to allow DNS queries over TCP (not so commonly used).

You should be aware that DNS zone transfers are done using TCP, while (most) DNS lookup queries are done using UDP.

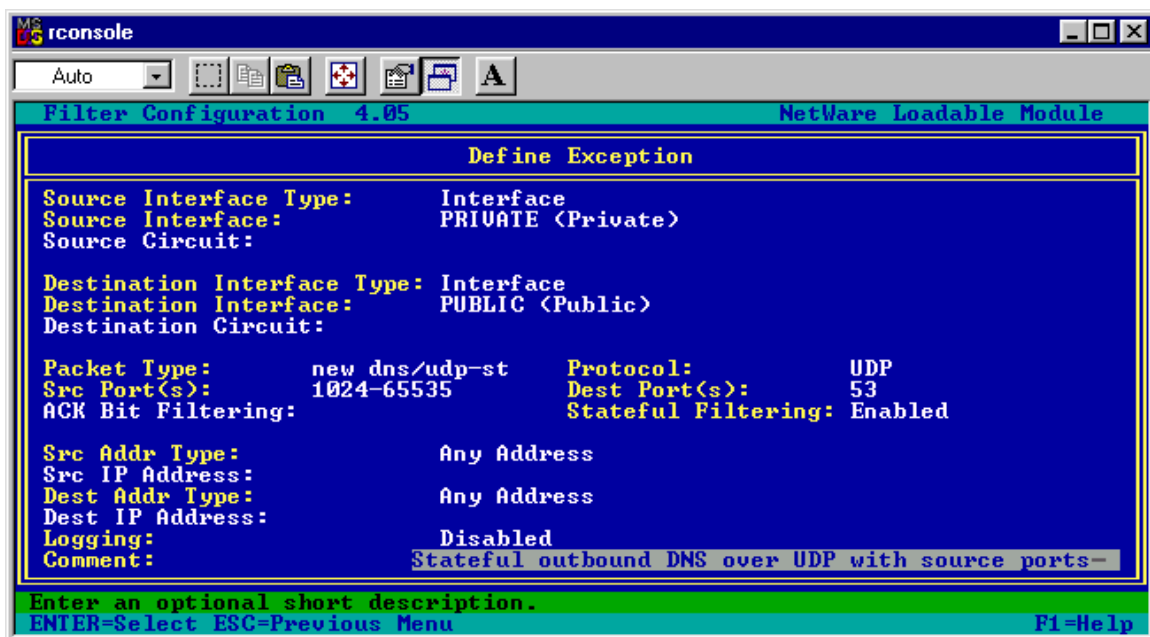


Figure 5-10- Filter Exception for Outbound DNS Queries over UDP with Source Ports Specified

The filter exception shown in Figure 5-10 allows typical outbound DNS lookup queries, over UDP.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source ports: 1024-65535
- Destination port: 53
- Stateful filtering: Enabled

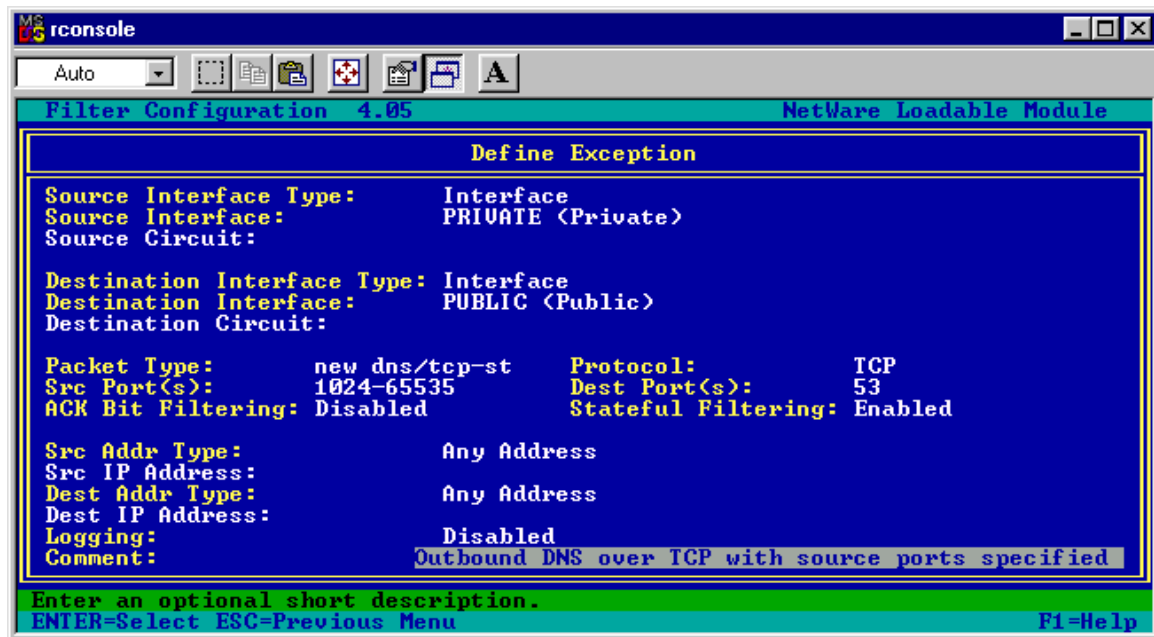


Figure 5-11 - Filter Exception for Outbound DNS Queries over TCP

The filter exception shown in Figure 5-11 allows outbound DNS lookup queries over TCP, which is not generally done. DNS responses may be required to use TCP if the data in the response does not fit within a single UDP packet. However, a more typical use of DNS over TCP for lookup queries is within NSLOOKUP tools (such as Cyberkit), which can specify UDP or TCP protocol to be used.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 53
- Stateful filtering: Enabled

## Outbound DNS Filter Exceptions for BorderManager 3.7 DNS Proxy (and Other Proxies)

The following two stateful exceptions should be created during a BorderManager 3.7 installation. This filter exception would also be used by any of the other BorderManager proxies that require DNS services.

The first exception shown allows DNS queries using TCP protocol. The second exception shown allows DNS queries using UDP protocol. BorderManager will use the DNS servers configured in INETCFG, Protocols, TCP/IP, DNS Resolver Configuration, and will use the protocol (TCP or UDP) specified in NWADMN32, BorderManager Setup, DNS.

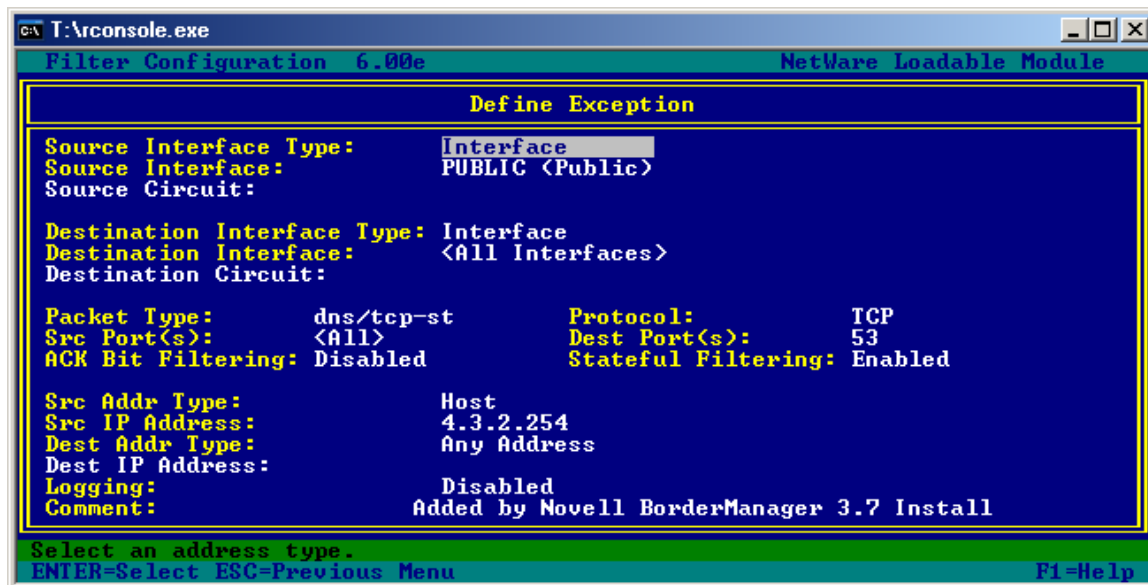


Figure 5-12 – Filter Exception for DNS over TCP from BorderManager 3.7 Proxies

The filter exception shown in Figure 5-12 allows any of the BorderManager proxies to make DNS requests using the TCP protocol.

- Source interface: Public
- Destination Interface: All interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 53
- Stateful filtering: Enabled
- Source IP Address: <BorderManager public IP address>

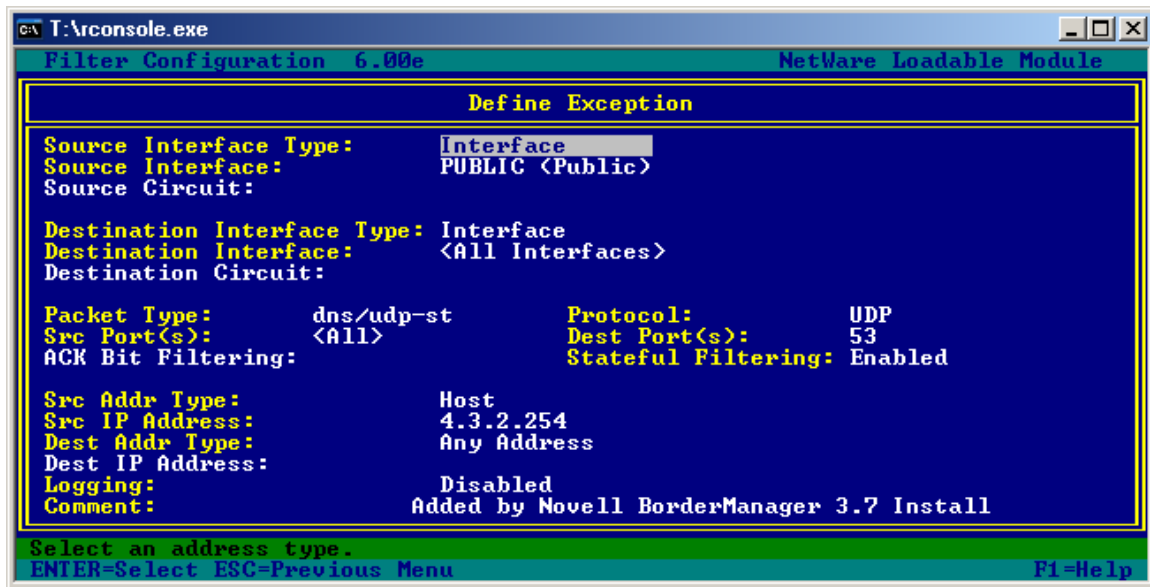


Figure 5-13 - Filter Exception for DNS over UDP from BorderManager 3.7 Proxies

The filter exception shown in Figure 5-13 allows any of the BorderManager proxies to make DNS requests using the UDP protocol.

- Source interface: Public
- Destination Interface: All interfaces
- Protocol: UDP
- Source ports: All
- Destination port: 53
- Stateful filtering: Enabled
- Source IP Address: <BorderManager public IP address>

## FTP

FTP filter exceptions are also a bit tricky. If you use a browser to go to an FTP server, you may actually be using the HTTP protocol to retrieve a file, and this filter exception isn't required. (That occurs when the FTP server is being proxied). You definitely will need an FTP filter exception when trying to download FTP files using a true FTP client. You also need to be aware that an FTP client (and server) may allow only Active or Passive FTP requests.

In Passive mode the FTP server tells the client to initiate the FTP request to the FTP server at a specific high port number. In Active mode, the client selects the high port number and tells the FTP server to respond on that port.

BorderManager 3.x contains a number of additional filtering features for FTP that are not available in BorderManager 2.1. Thus, it is possible that you will be able to accomplish some FTP-related tasks using BorderManager 3.0 and later versions that are simply not possible with BorderManager 2.1.

FTP itself uses two well-known port numbers. FTP control is done on port 21 to establish sessions, change directories, etc. Port 20 is used when transferring data. A stateful exception for both ports will be required to make FTP file transfers functional.

## Outbound FTP Filter Exception for Internal Hosts

A number of different FTP filter exceptions might be required, but using the Novell-supplied ftp-port-pasv-st filter is a good one to try.

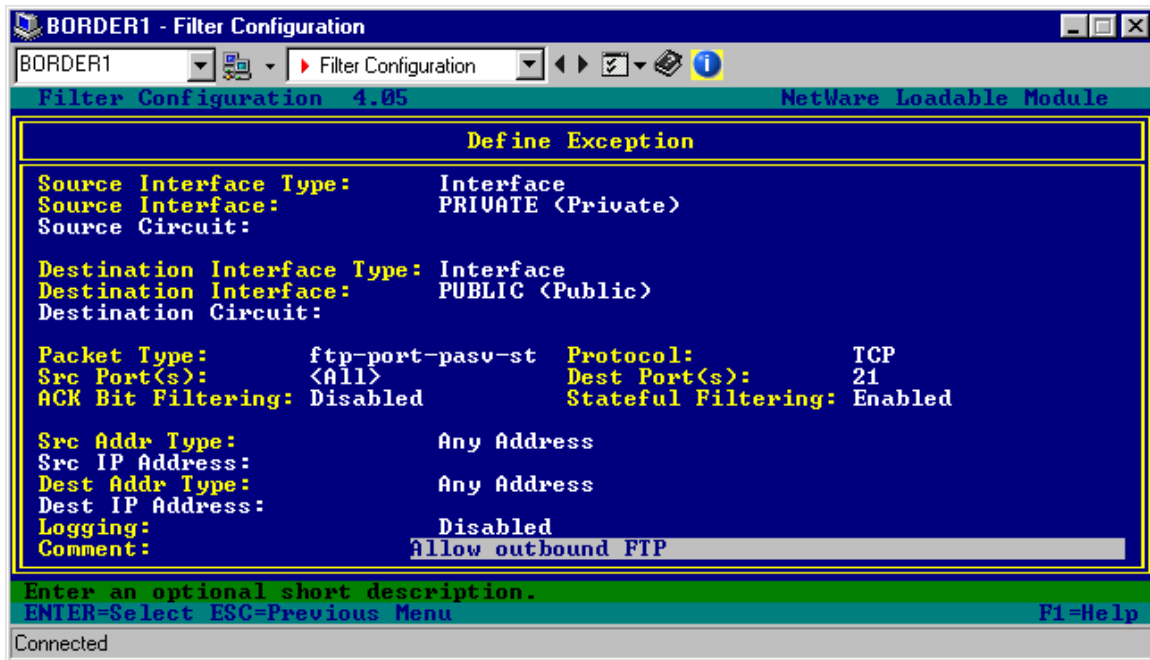


Figure 5-14 - Filter Exception for Outbound FTP

The filter exception shown in Figure 5-14 should allow FTP clients to establish an FTP session with an external host.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: <All>
- Destination port: 21
- Stateful filtering: Enabled. For port and pasv modes

---

**Note** When using the ftp-port-pasv-st filter definition, port 20 traffic (used for FTP data transfers) is automatically allowed, and a separate filter exception for port 20 is not required. (A very smart filter exception, that ftp-port-pasv-st!)

---



## Outbound FTP Filter Exception for BorderManager 3.7 FTP Proxy

The following example shows the stateful filter exception that a fresh BorderManager 3.7 installation provides.

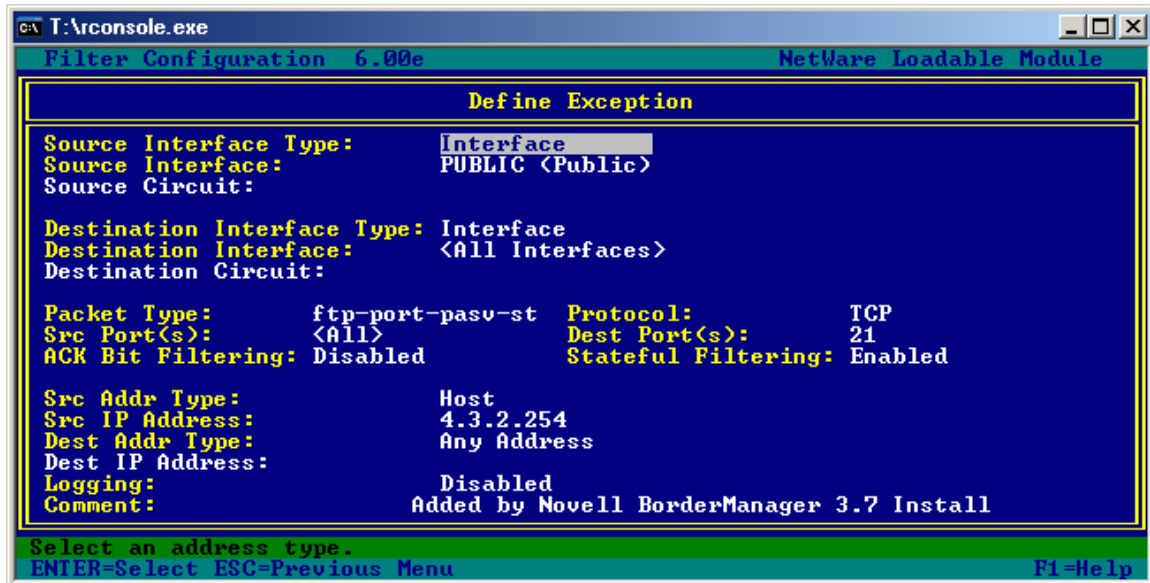


Figure 5-15 - Filter Exception for Outbound FTP from BorderManager 3.7 FTP Proxy

The filter exception shown in Figure 5-15 allows the FTP proxy to establish an FTP session with an external host.

- Source interface: Public
- Destination Interface: All interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 21
- Stateful filtering: Enabled, for port and pasv modes
- Source IP Address: <BorderManager public IP address>

## GroupWise Remote Client

Should you need to connect to a GroupWise server on the Internet, you can set up the following stateful filter exception. (It is more likely that you will need to set up inbound capability for GroupWise remote, and an example of that is shown in the section on inbound exceptions for static NAT).

The standard GroupWise client port number is 1677, though any port number could be configured by the GroupWise administrator.

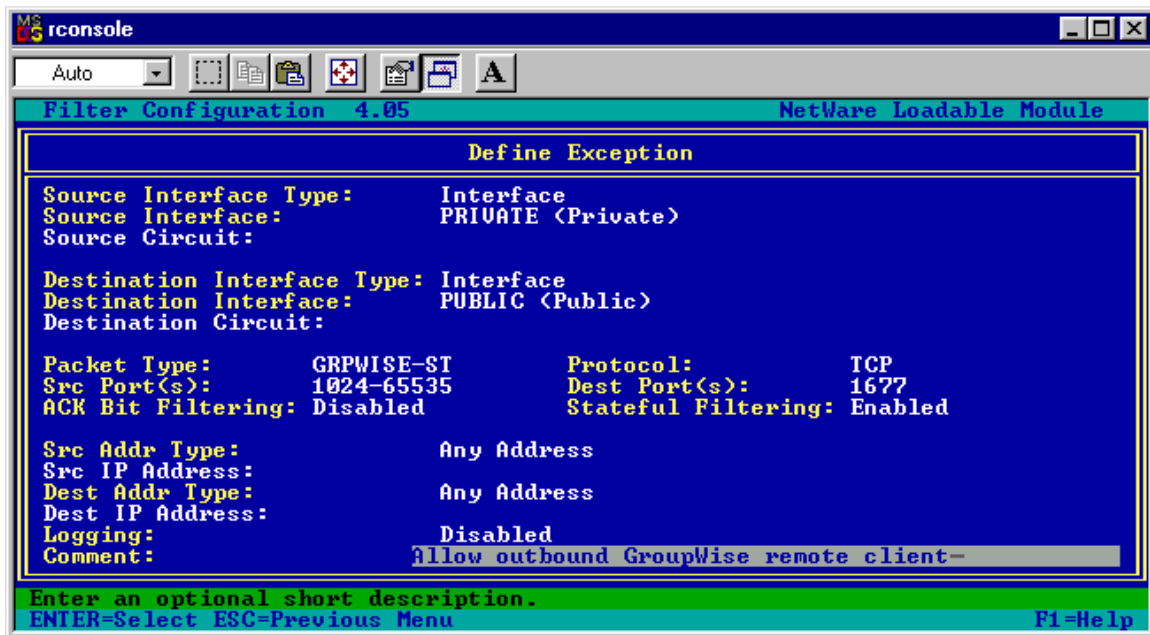


Figure 5-16 - Filter Exception for Outbound GroupWise Remote Client

The filter exception shown in Figure 5-16 allows internal hosts to access a GroupWise on the Internet using the standard GroupWise port number.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 1677
- Stateful filtering: Enabled

## ICQ (Version 2000b)

ICQ 2000b, (and I assume later versions), might default to using the same port number as AOL (TCP destination port 5190). If in ICQ, Preferences, Server, you see port 5190 entered for the server login.icq.com, use the filter exception for AOL Instant Messenger.

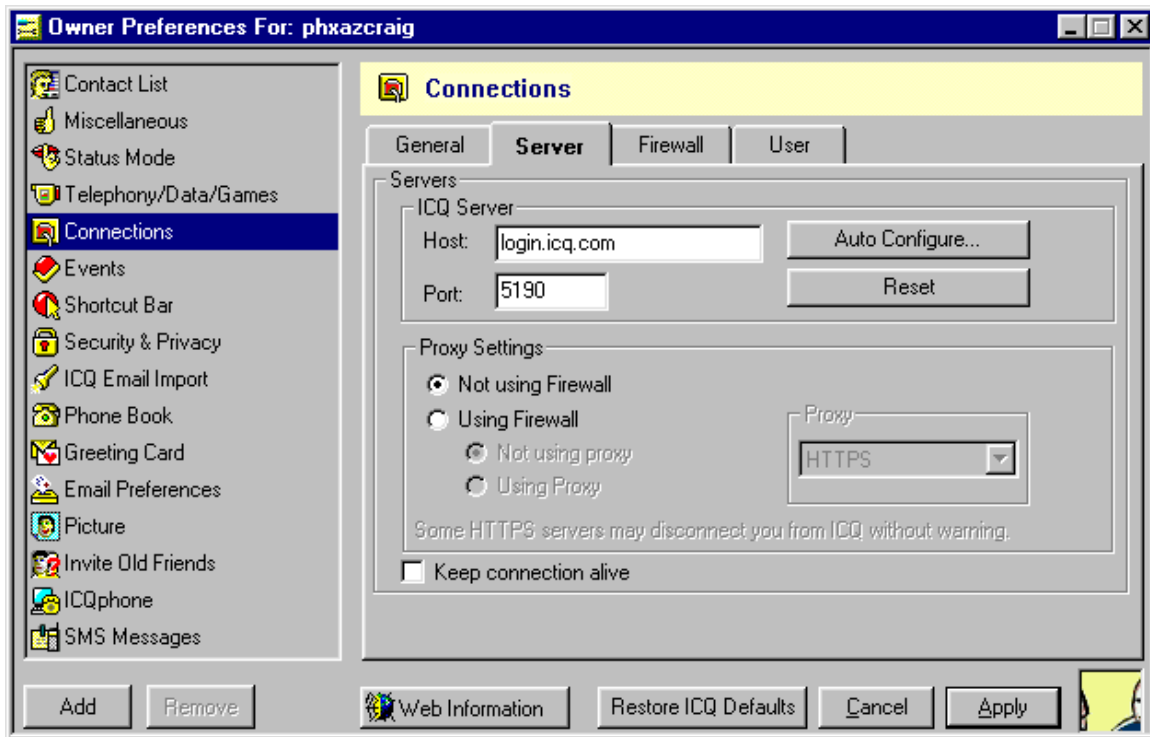


Figure 5-17 - ICQ 2000b Settings for AOL Port Number

Figure 5-17 shows settings for ICQ 2000b, set up for the same port number as for AOL Instant Messenger.

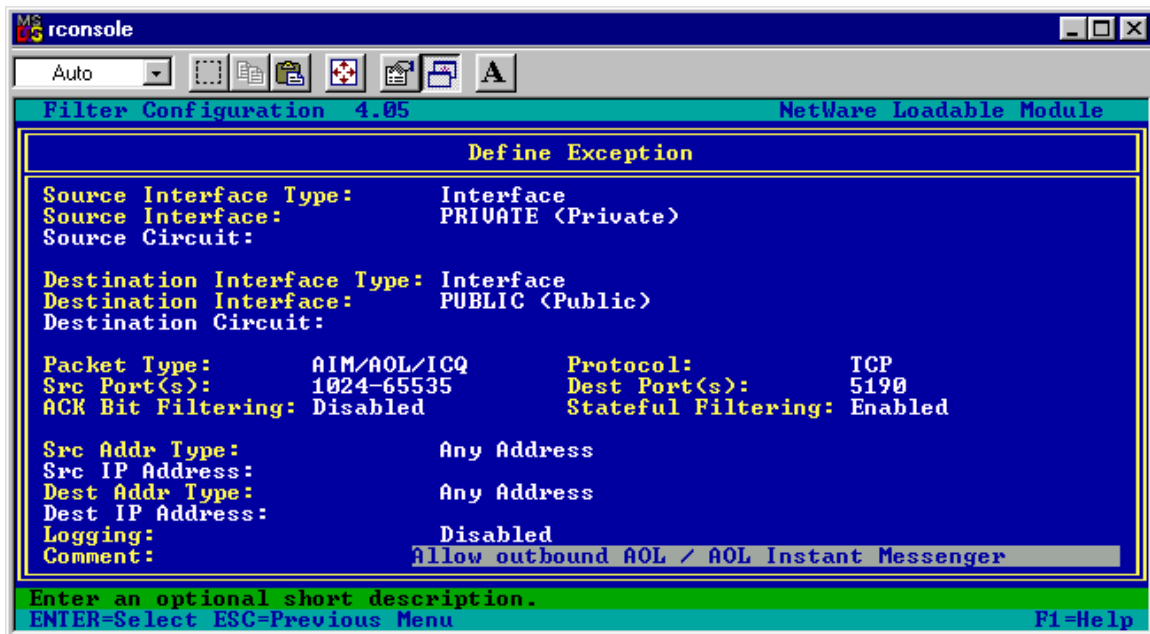


Figure 5-18 - Filter Exception for Outbound ICQ 2000b

The filter exception shown in Figure 5-18 allows in an internal ICQ 2000b client configured for port 5190 to establish an ICQ connection.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 5190
- Stateful filtering: Enabled

If you want to BLOCK Chat programs like this, refer to the discussion “Blocking Chat Programs”, Page 280.

# HTTP

Should you wish to bypass the HTTP Proxy, you can set up the following stateful filter exception to allow the standard web browsing port 80 through the filters. This will not allow bypass of the Transparent HTTP Proxy unless you also configure exceptions within the Transparent HTTP Proxy settings.

## Outbound HTTP Filter Exception for Internal Hosts

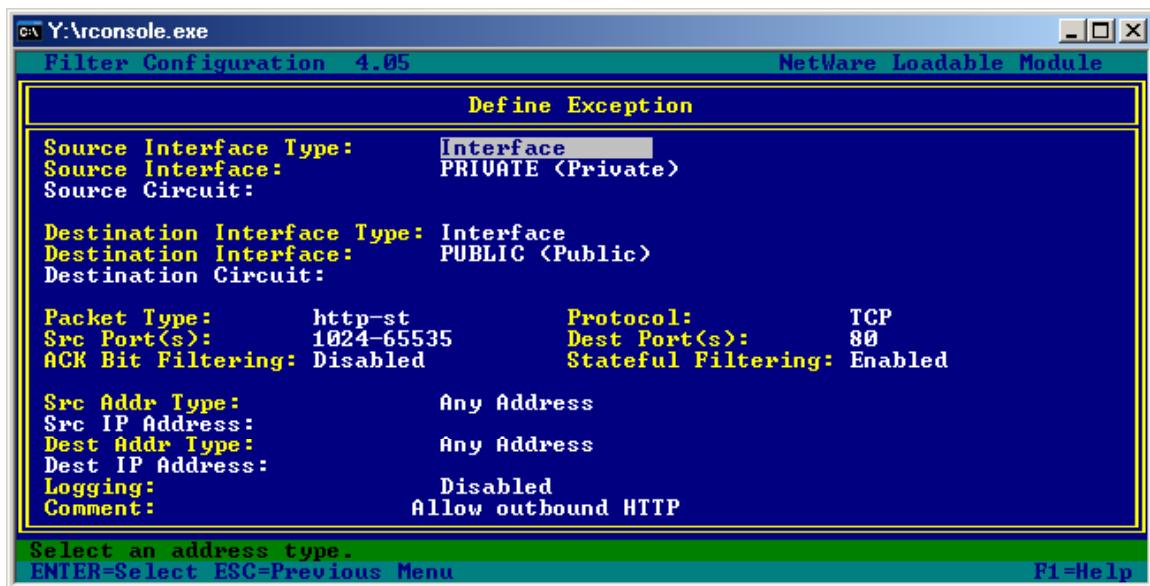


Figure 5-19 - Filter Exception for Outbound HTTP

The example shown in Figure 5-19 allows internal hosts to bypass the HTTP Proxy for web browsing to most web sites. This exception tends to make more sense when allowing proxy bypass only for specific web sites, since bypassing the HTTP Proxy also allows bypassing Access Rules.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 80
- Stateful filtering: Enabled.

## Outbound HTTP Filter Exception for BorderManager 3.7 HTTP and Transparent HTTP Proxy

The following example shows the stateful HTTP filter exception that a fresh BorderManager 3.7 installation.

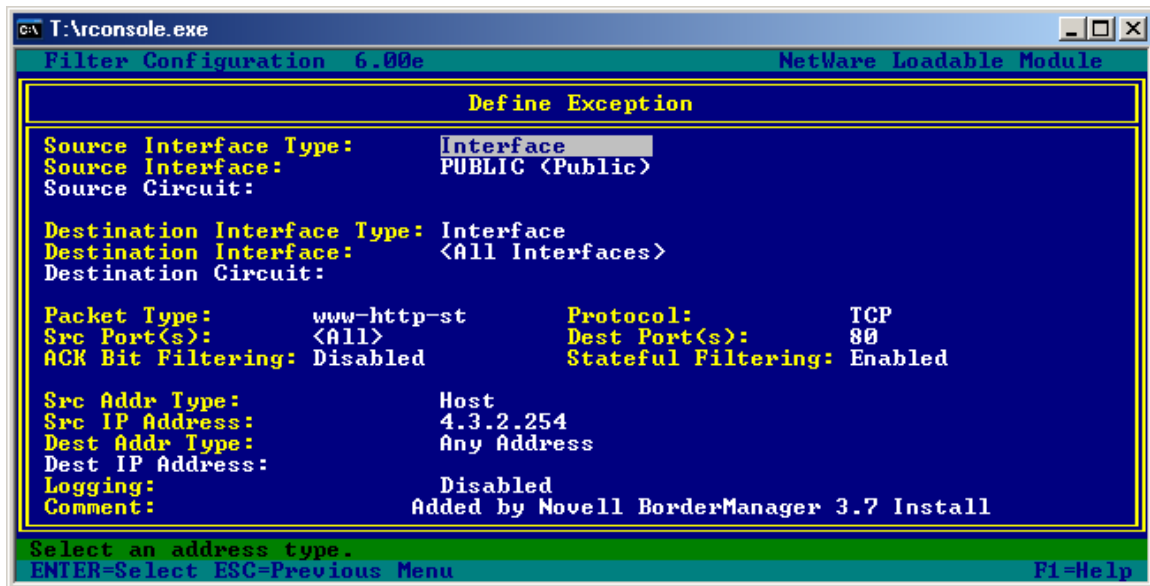


Figure 5-20 - Filter Exception for Outbound HTTP from BorderManager 3.7 HTTP Proxy

The example shown in Figure 5-20 allows the HTTP Proxy and HTTP Transparent Proxy to establish connections with most web servers on the Internet. This exception does not allow connections to be made on non-standard port numbers.

- Source interface: Public
- Destination Interface: All Interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 80
- Stateful filtering: Enabled
- Source IP Address: <BorderManager Public IP address>

# IMAP

IMAP is a mail protocol that might be used instead of POP3. This filter exception allows an internal host to check mail on an Internet host using IMAP protocol.

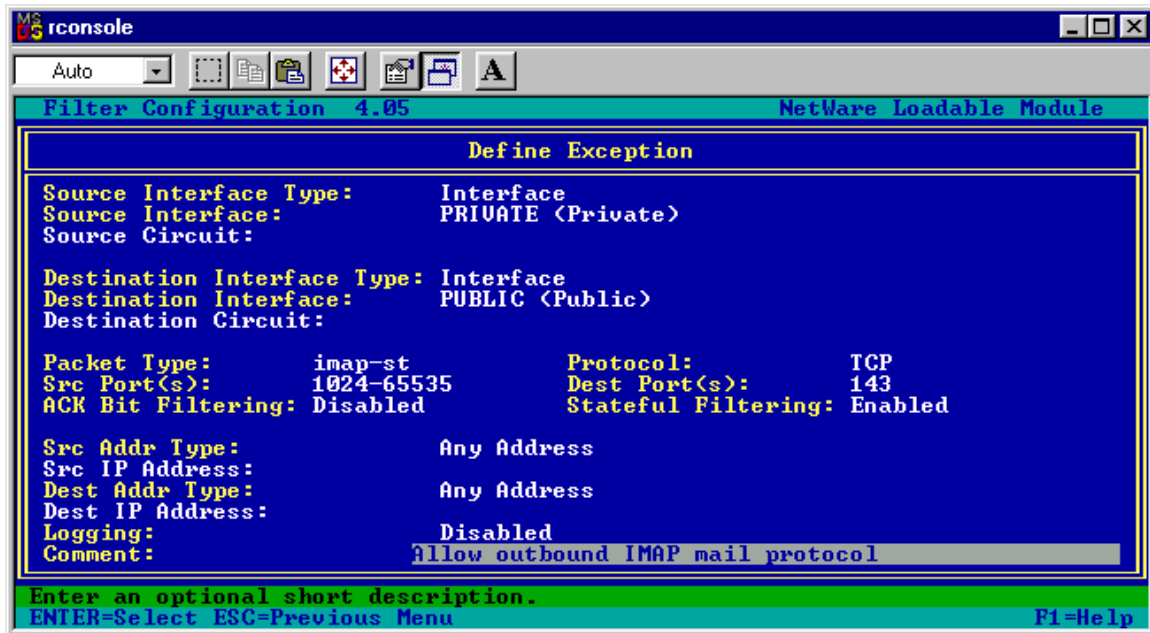


Figure 5-21 - Filter Exception for Outbound IMAP

This filter exception shown in Figure 5-21 allows an internal host to check email on an Internet host using IMAP.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 143
- Stateful filtering: Enabled

## Microsoft MSN Messenger

Microsoft MSN Messenger, (version 3.6.0039 tested for this example), has an option to work through an HTTP Proxy. However, even when the application is configured to use HTTP Proxy, it still attempted to make a direct connection on TCP destination port 1863. Only after timing out on port 1863 did MSN Messenger try to use the HTTP Proxy settings. If you want to simply allow this application to work without using the HTTP Proxy, you can use the following stateful filter exception.

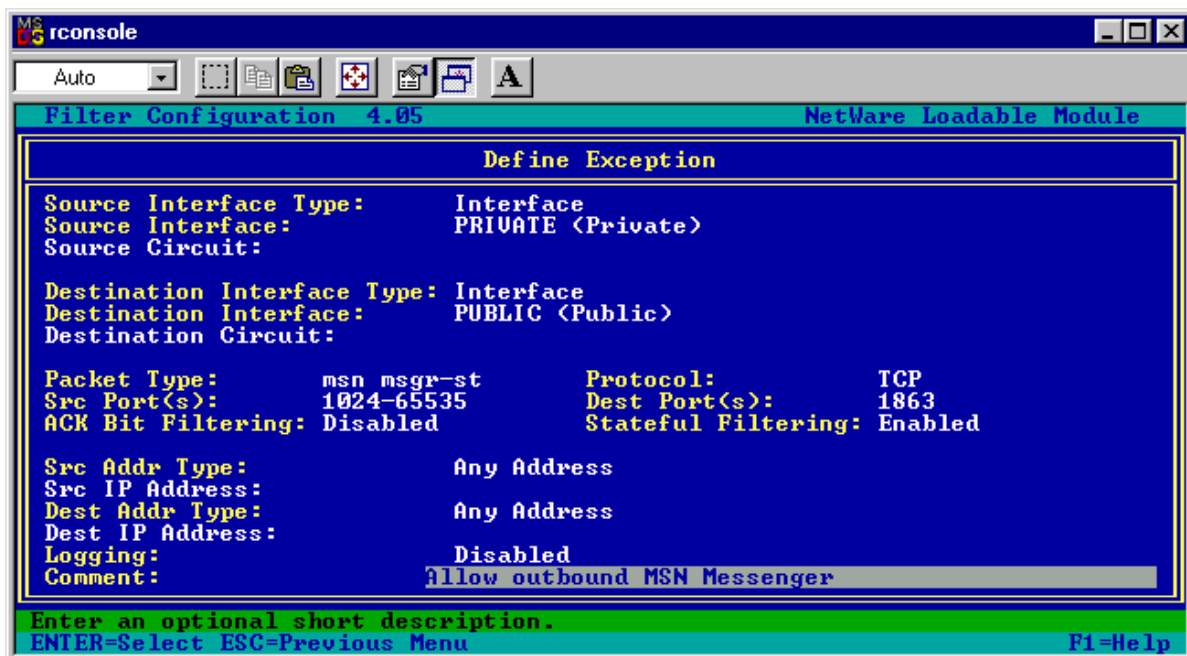


Figure 5-22 - Filter Exception for Outbound MSN Messenger

The filter exception shown in Figure 5-22 allows an internal host using Microsoft MSN Messenger to directly access MSN messaging services without configuring a proxy.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 1863
- Stateful filtering: Enabled



## Microsoft Windows Media Player

Microsoft Windows Media Player can connect to the Internet to access files in MMS format. Most Internet connections can make use of the HTTP Proxy, if set in Media Player, but the MMS streaming format should be configured to use TCP destination port 1755.

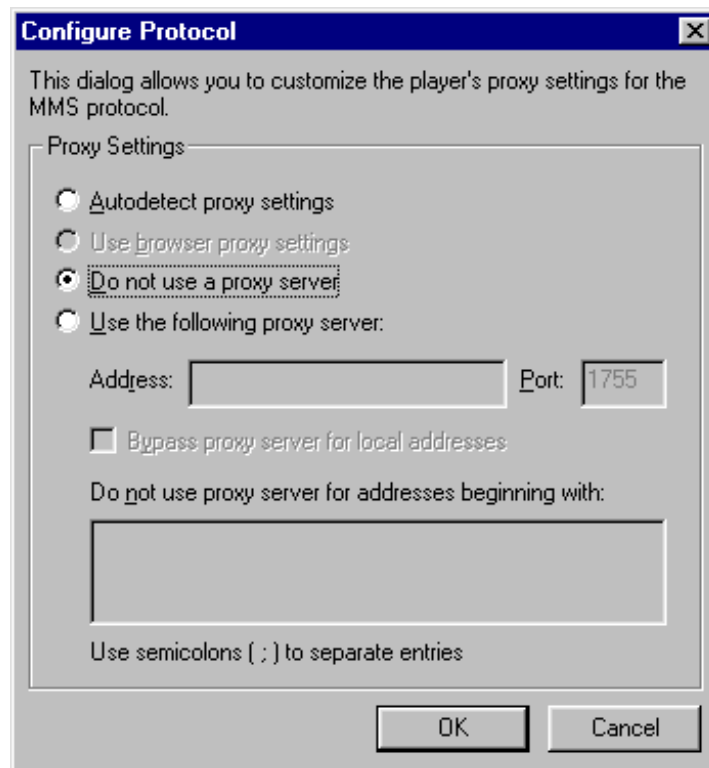


Figure 5-23 - Windows Media Player MMS Protocol Settings

The screenshot shown in Figure 5-23 shows Media Player configured not to use a proxy server for the MMS Protocol.

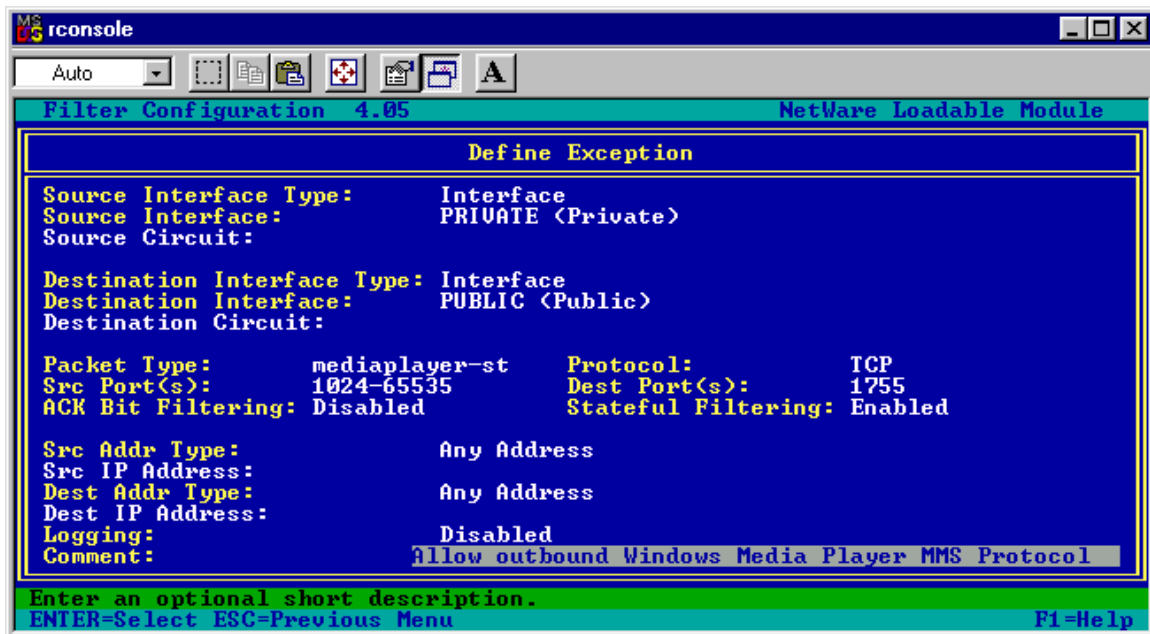


Figure 5-24 - Filter Exception for Outbound Windows Media Player MMS Protocol

The filter exception shown in Figure 5-24 allows an internal host using Windows Media Player to access MMS streaming sources on the Internet.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 1755
- Stateful filtering: Enabled

# NNTP

## Outbound NNTP Filter Exception for Internal Hosts

Since the BorderManager 3.x NNTP Proxy service only allows you to proxy one NNTP server for port 119, it is often much easier to just set up a stateful filter exception to allow any NNTP server to be accessed across BorderManager from inside the network.

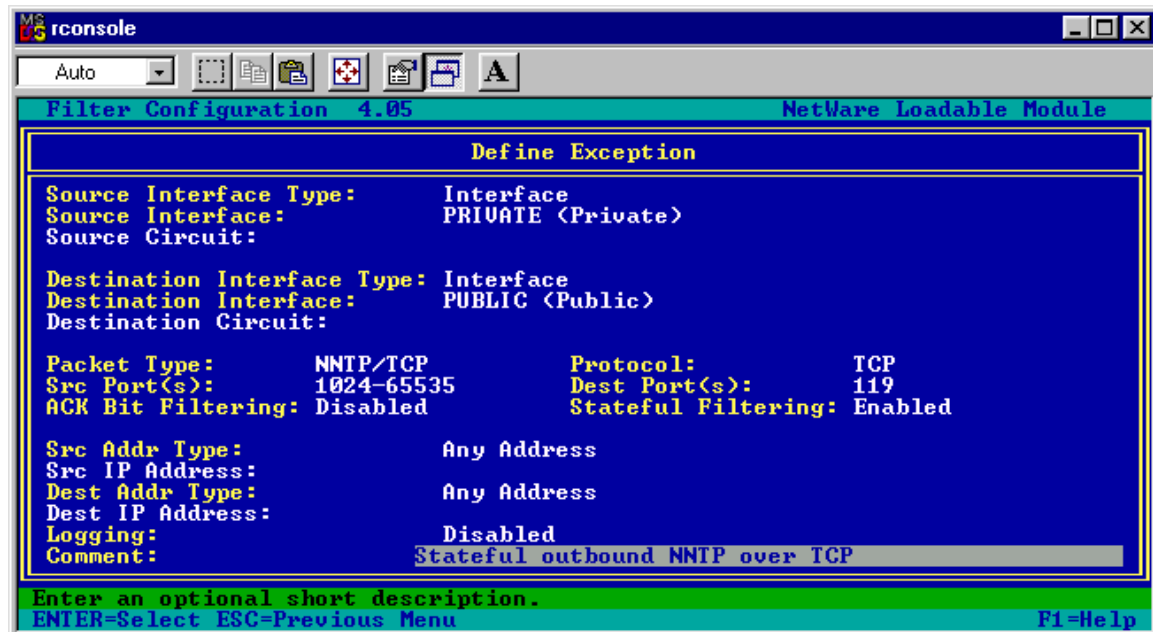


Figure 5-25- Filter Exception for Outbound NNTP

The filter exception shown in Figure 5-25 allows internal hosts to make NNTP connections to a Usenet server on the Internet.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 119
- Stateful filtering: Enabled

## Outbound NNTP Filter Exception for BorderManager 3.7 News Proxy

The following example shows the filter exception that a fresh BorderManager 3.7 installation.

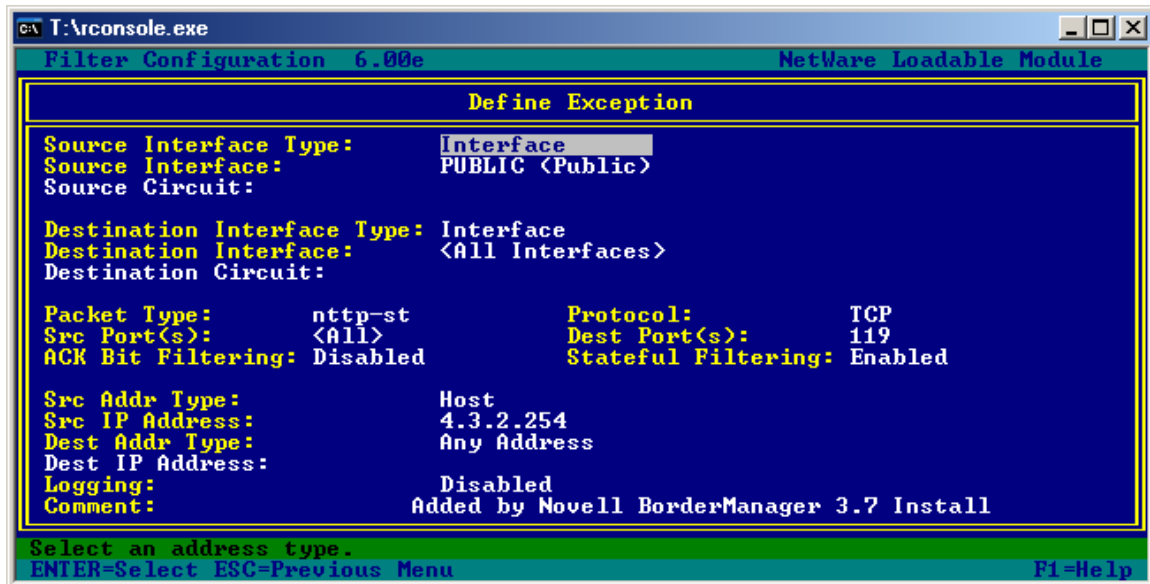


Figure 5-26 - Filter Exception for NNTP from BorderManager 3.7 News Proxy

Note the misspelled packet type (nntp instead of NNTP), which is a holdover from a misspelling in the BorderManager 3.6 BUILTINS.CFG file.

- Source interface: Public
- Destination Interface: All Interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 119
- Stateful filtering: Enabled
- Source IP Address: <BorderManager Public IP address>

## NTP/SNTP

You may have internal hosts that wish to use NTP (Network Time Protocol) or SNTP (Simple Network Time Protocol) to set a clock to an Internet-based time reference server. For example, a UNIX host or NetWare 5 server might use SNTP. A PC using the D4Time program also would use SNTP. In these cases, set up a stateful filter exception to allow port 123 through BorderManager. (It is also easy to set up a Generic UDP Proxy for NTP/SNTP).

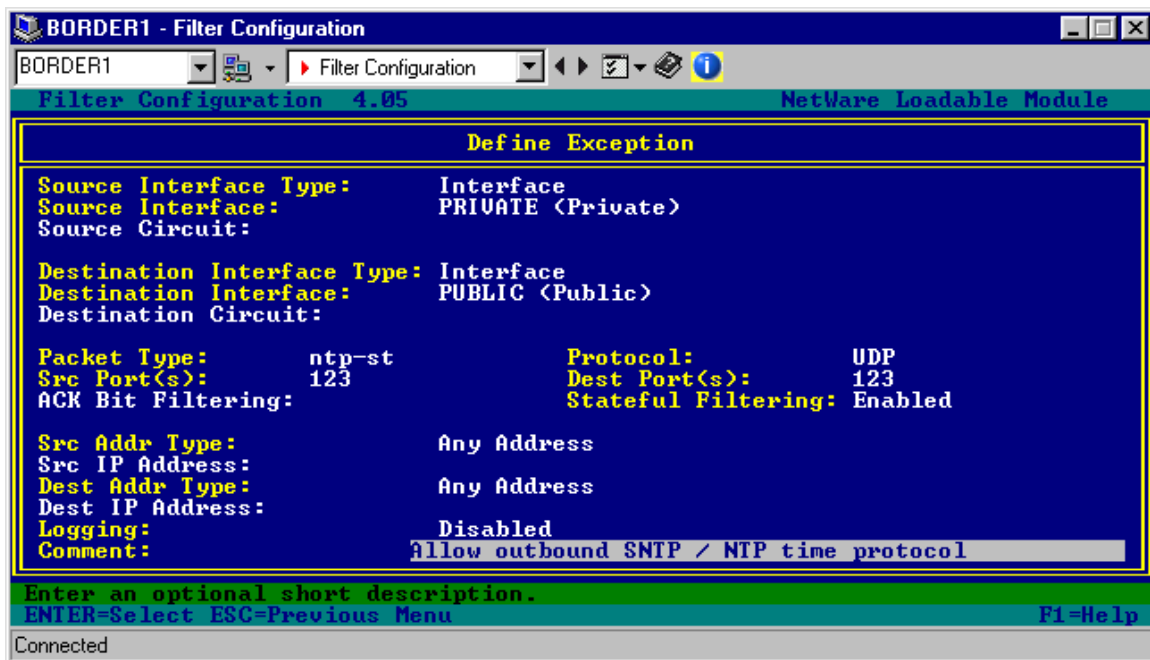


Figure 5-27 - Filter Exception for Outbound NTP, Source Port=123

The filter exception shown in Figure 5-27 allows NTP/SNTP via protocol UDP, with the source port set to 123.

**Note that some NTP clients use port 123 for both source and destination ports, but others might use UDP high ports (1024-65535) for the source ports.** Novell's TIMESYNC.NLM has used both, depending on the version. Using <All> for source ports here would be more flexible, at a slightly increased security risk.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source ports: 123 (or 1025-65535, or All)
- Destination port: 123
- Stateful filtering: Enabled

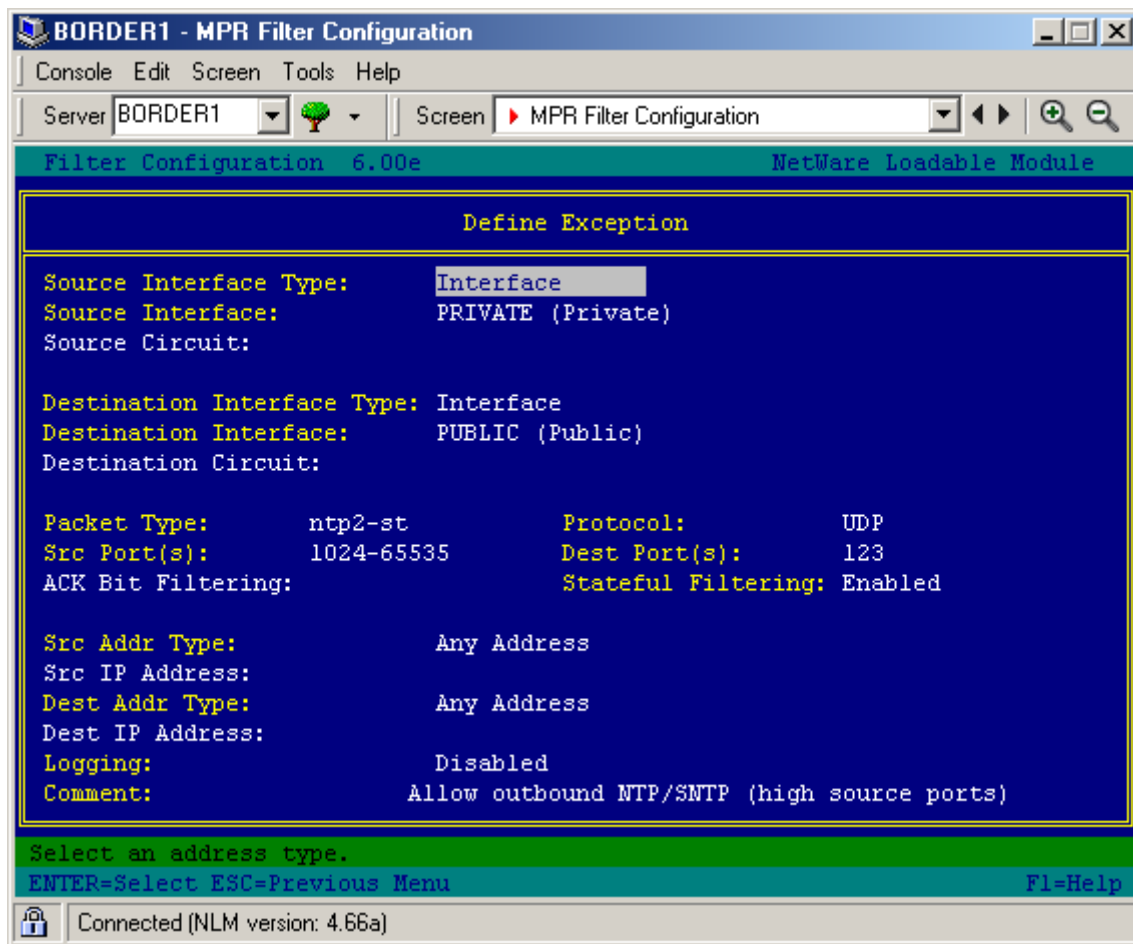


Figure 5-28 - Filter Exception for Outbound NTP, Source Ports=1024-65535

The filter exception shown in Figure 5-28 allows NTP/SNTP via protocol UDP, with the high ports specified for the source ports. This filter exception will allow a NetWare 6.0 server, patched to NW6Sp1 or later, to send outbound NTP requests from behind a BorderManager server.

Note that some NTP clients use port 123 for both source and destination ports, but others might use UDP high ports (1024-65535) for the source ports. Novell's TIMESYNC.NLM has used both, depending on the version. Using <All> for source ports here would be more flexible, at a slightly increased security risk.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source ports: 123 (or 1025-65535, or All)
- Destination port: 123
- Stateful filtering: Enabled

The filter exceptions shown in Figure 5-27 and Figure 5-29 will **not** work if BorderManager is the NTP time server making the request. If the BorderManager server is the single reference or reference time server, it will try to send requests from the public IP address. You would need to change the exception above to call out a source interface of Public, and add a source IP address of your public IP address to modify the filter exception for best results.

---

**Note** More information on using NTP in your LAN can be found in the Novell AppNote "Using Network Time Protocol (NTP) with NetWare 5", July 1999 <http://developer.novell.com/research/appnotes/1999/a9907.htm>

---

## pcANYWHERE

The pcANYWHERE (versions 8, 9 and 10, at least) program uses one of two different UDP ports to locate a pcANYWHERE host, then a particular TCP port to exchange data. Three stateful filter exceptions are needed to allow outbound connectivity for pcANYWHERE.

- UDP destination port 22, source ports 1024-65535 is used to locate another pcANYWHERE host, and may be the only port used to locate an **older version** of pcANYWHERE.
- UDP destination port 5632, source ports 1024-65535 is also used to locate another pcANYWHERE host.
- TCP destination port 5631, source ports 1024-65535 is used to exchange data between pcANYWHERE hosts once the two hosts have located each other using UDP.

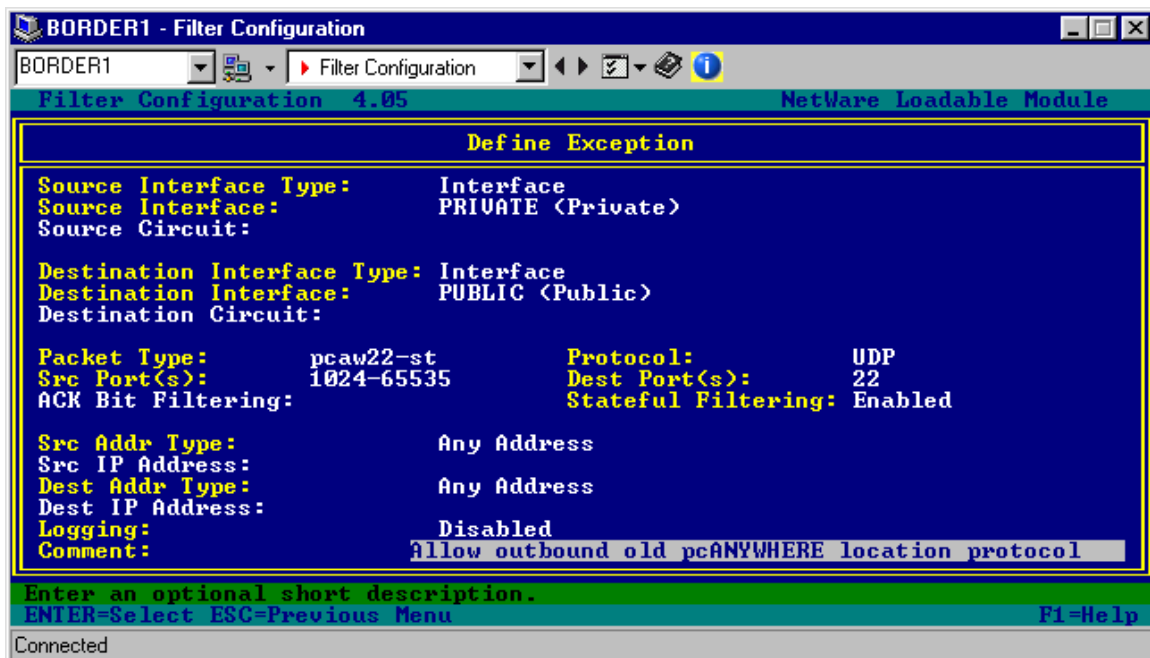


Figure 5-29 - Filter Exception for Outbound pcANYWHERE Location Protocol (Old)

The filter exception shown in Figure 5-29 allows the old (obsolete) pcANYWHERE location protocol.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source ports: 1024-65535
- Destination port: 22
- Stateful filtering: Enabled



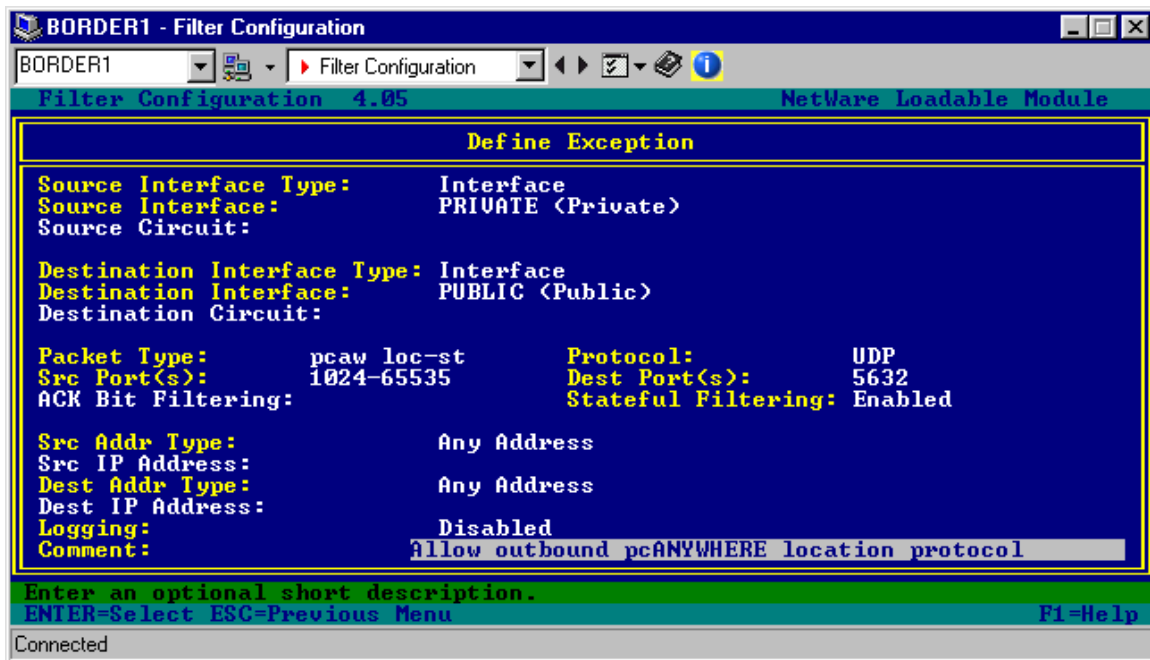


Figure 5-30 - Filter Exception for Outbound pcANYWHERE Location Protocol

The filter exception shown in Figure 5-30 allows the newer pcANYWHERE location protocol.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source ports: 1024-65535
- Destination port: 5632
- Stateful filtering: Enabled

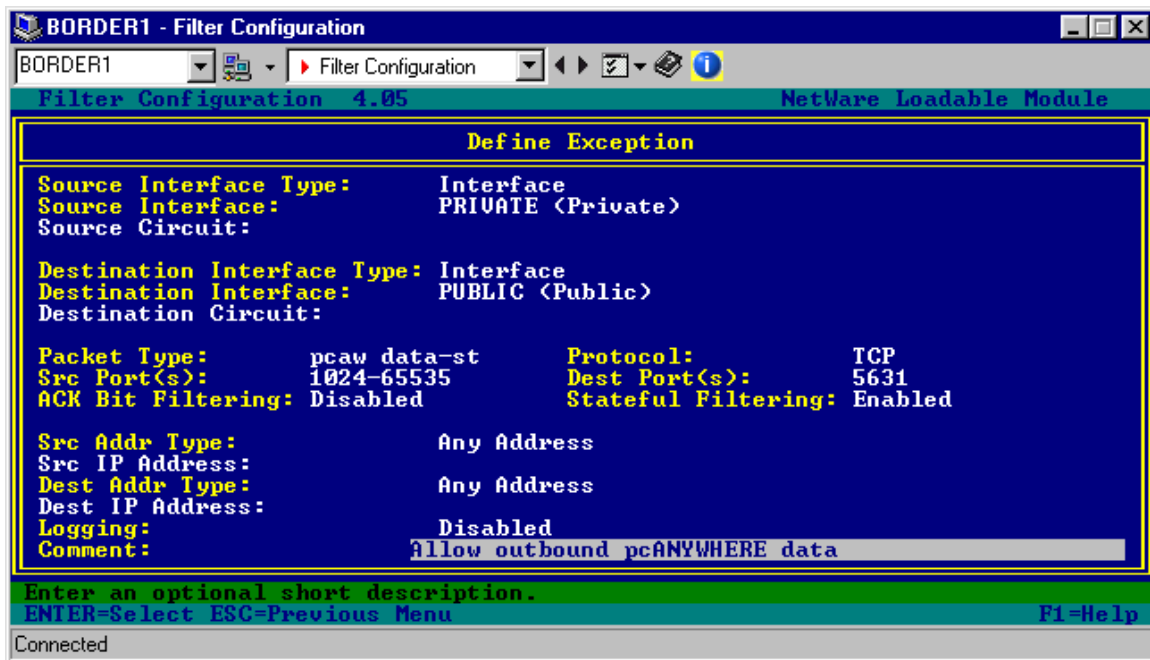


Figure 5-31 - Filter Exception for Outbound pcANYWHERE Data

The filter exception shown in Figure 5-31 allows pcANYWHERE data connections from an internal PC to a pcANYWHERE host on the Internet.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 5631
- Stateful filtering: Enabled

## PING (ICMP)

It is often desirable to be able to PING remote hosts to test basic connectivity. However, the BorderManager default filters will block ICMP packets, and PING tests, which rely on ICMP, will fail. This example shows how to set up a stateful filter exception to allow PING testing outbound, while still restricting PING packets from coming back in.

---

**CAUTION** *ICMP is much more than just PING, and it is important from a security standpoint not to just allow all ICMP to your network! The stateful filter exception shown is secure, but it will not allow your server to be pinged from the public side or allow you to ping from the server console itself.*

---

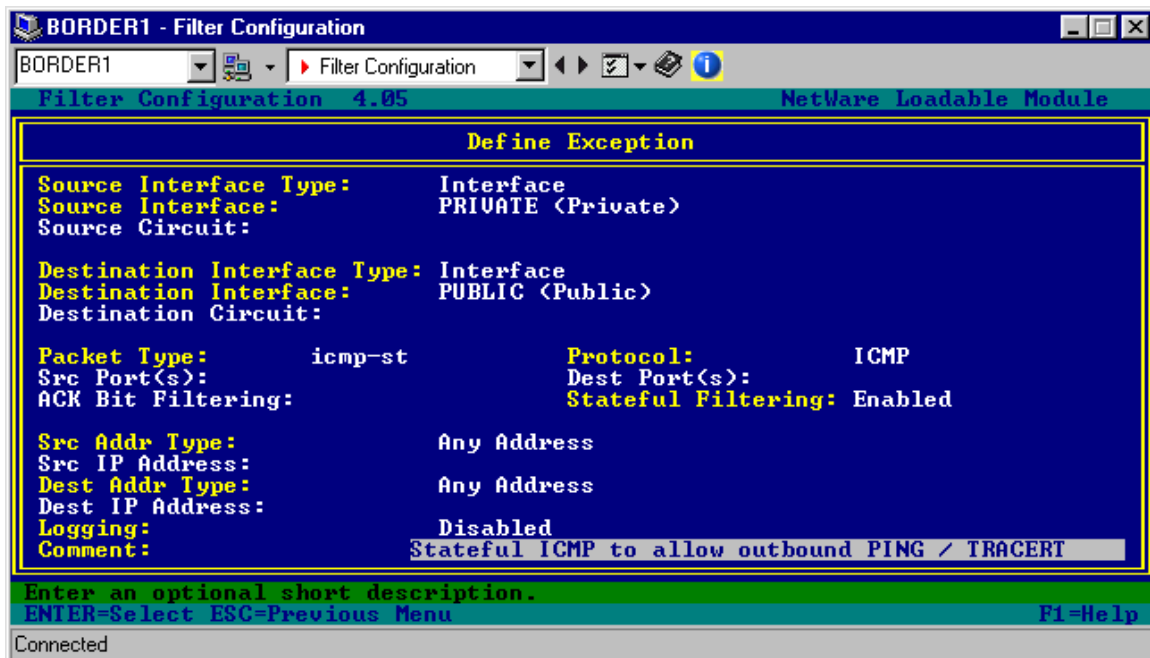


Figure 5-32 - Filter Exception for Outbound ICMP (PING & TRACERT)

The filter exception shown in Figure 5-32 allows protocol ICMP from an internal host to any IP address. It will not allow the BorderManager server itself to ping because it requires the ICMP packets to come across the private interface.

- Source interface: Private
- Destination Interface: Public
- Protocol: ICMP
- Stateful filtering: Enabled

# POP3

## Outbound POP3 Filter Exception for Internal Hosts

If you want to allow any host on your network to simply check their email at an ISP's POP3 server, set up the following stateful filter exception to allow outbound TCP destination port 110.

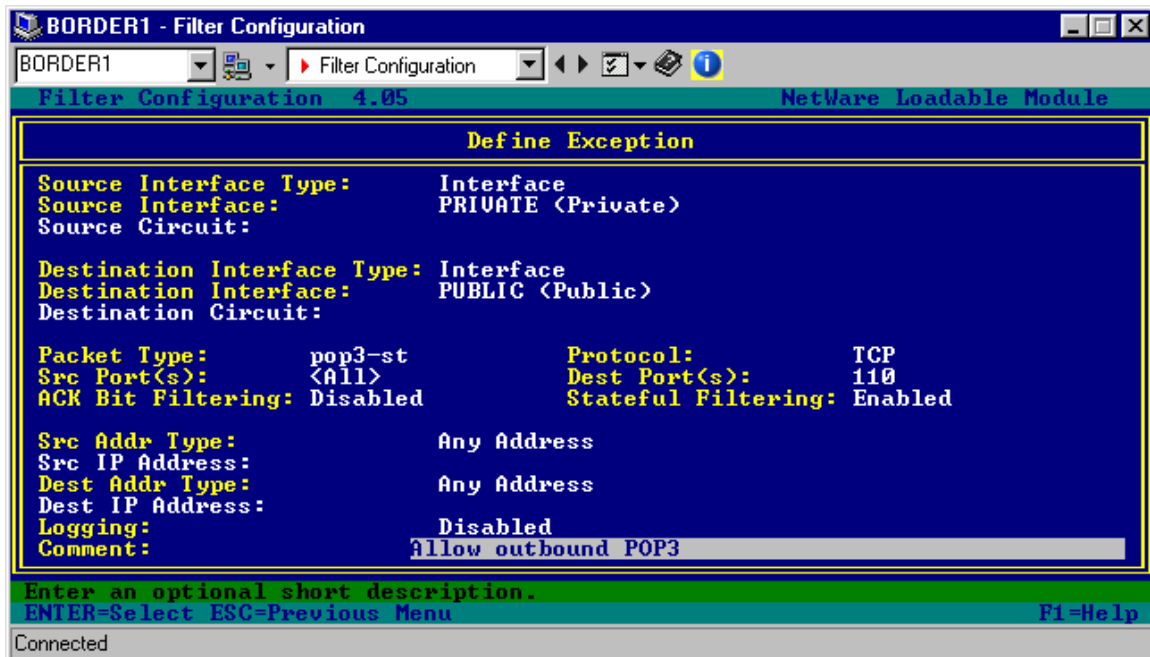


Figure 5-33 - Filter Exception for Outbound POP3

The filter exception shown in Figure 5-33 allows outbound POP3 requests.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 110
- Stateful filtering: Enabled

The alternative to this exception, for BorderManager 3.x, is to use the Mail Proxy.

---

**CAUTION**      *The built-in filter exception for POP3-ST in BorderManager 3.5 is NOT stateful. Either create a new exception, called POP3a-ST or similar, or follow the instructions on page 315 to fix the definition.*

---

## Outbound POP3 Filter Exception for BorderManager 3.7 Mail Proxy

The following example shows the filter exception that a fresh BorderManager 3.7 installation.

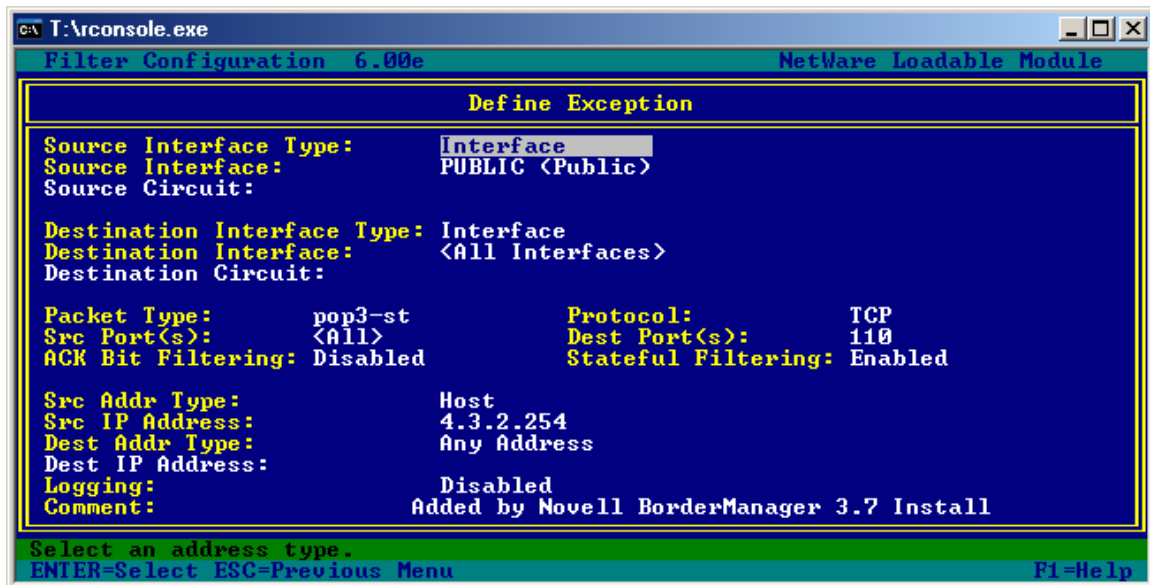


Figure 5-34 - Filter Exception for POP3 from BorderManager 3.7 Mail Proxy

The example shown in Figure 5-34 allows the Mail Proxy to make POP3 requests to a POP3 mail server on the Internet.

- Source interface: Public
- Destination Interface: All Interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 110
- Stateful filtering: Enabled
- Source IP Address: <BorderManager 3.7 Public IP address>

## RDATE

NetWare 3.1x, 4.x and 5.x servers can use a (free) program from <http://www.murkworks.com/> called RDATE.NLM to set their clocks to a time server on the Internet. RDATE uses port 37, so you might want to set up a stateful filter exception to allow UDP port 37 through BorderManager. (It is also easy to set up a Generic UDP proxy for RDATE).

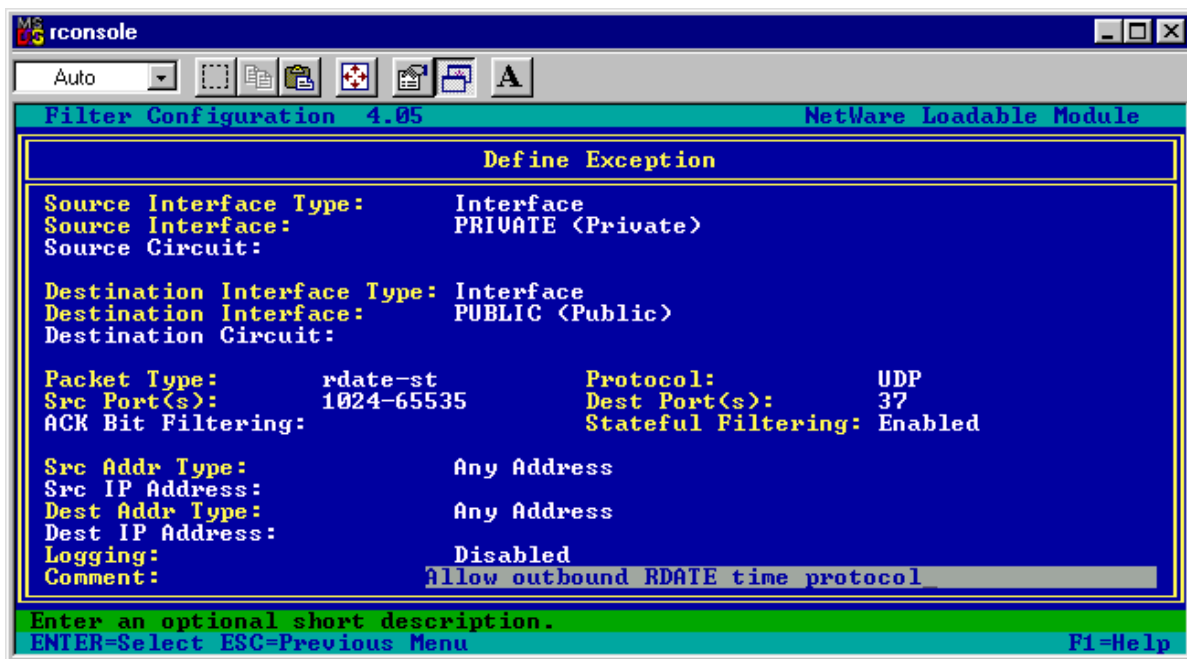


Figure 5-35 - Filter Exception for Outbound RDATE Time Protocol

The filter exception shown in Figure 5-35 allows outbound RDATE requests via protocol UDP, from an internal host.

- Source interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source ports: 1024-65535
- Destination port: 37
- Stateful filtering: Enabled

If the BorderManager server itself is running RDATE, you will need to change the source interface to Public, and make the source IP address equal to your public IP address.

Should you wish to use RDATE on your time reference server, you can use these settings, but read the RDATE documentation so that you thoroughly understand the ramifications of the /M option.

The IP address shown in the example below belongs to a time server in Boulder, Colorado. You may find a list of NTP time servers on the Internet, and some of those may support RDATE using either UDP or TCP protocol.

```
LOAD RDATE /U /V 2 /P 240 /M 9999999 171.64.7.77
```

- /U=UDP
- /V 2 = Allow up to 2 second drift
- / P 240 = Check time every 240 minutes
- / M <large number> = Maximum number of seconds time can be off and RDATE will change it.
- 171.64.7.77 = IP Address of a time server at Stanford University

# RealAudio

## Outbound RealAudio Filter Exception for Internal Hosts

RealAudio streams may come in two formats – PNA (RealAudio) and RTSP. RealPlayer should have two different options to proxy each of these protocols. An exception to allow outbound RTSP proxy is shown in this book on page 180. An exception to allow outbound PNA / RealAudio traffic is shown here.

In BorderManager 3.x, it is possible to use the built in RealAudio Proxy by configuring the RealPlayer settings properly, but if you need to allow RealPlayer G2 traffic using only packet filter exceptions, you can get it working in by opening up TCP port 7070 in both directions. If you have BorderManager 3.x, use a stateful filter to allow TCP port 7070 outbound. If you have BorderManager 2.1, you need to configure two exceptions to allow TCP port 7070 inbound and outbound.

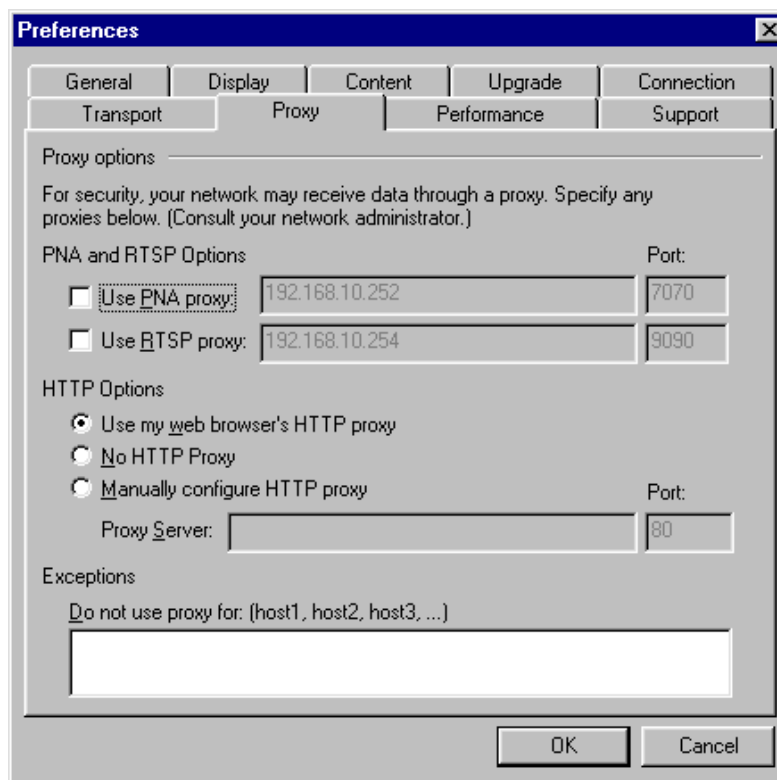


Figure 5-36 - RealPlayer G2 Settings to Bypass PNA & RTSP Proxy

Do not configure RealPlayer to use a PNA Proxy if you wish to bypass the BorderManager 3.x RealAudio proxy.



RealPlayer G2 first uses HTTP to locate a RealAudio site. You must therefore have both DNS and HTTP allowed in some manner for RealPlayer G2 to connect to a site. Once the site has been found, TCP port 7070 carries the data. In the example shown, RealPlayer G2 is configured to use the same proxy settings as the default browser (which means Internet Explorer), which should be port 8080 and the BorderManager private IP address. (BorderManager in this case was set up with HTTP Proxy enabled). DNS in this case was already allowed by a stateful filter exception.

This filter exception was tested using RealPlayer version G2.

At the workstation, first configure RealPlayer G2 under Options, Preferences, Proxy to use your (Internet Explorer) browser's proxy settings, or manually configure the BorderManager private IP address and proxy port number in use (if any). If you are not using the BorderManager HTTP proxy, you must have filter exceptions allowing HTTP port 80 through or RealPlayer G2 will not work.

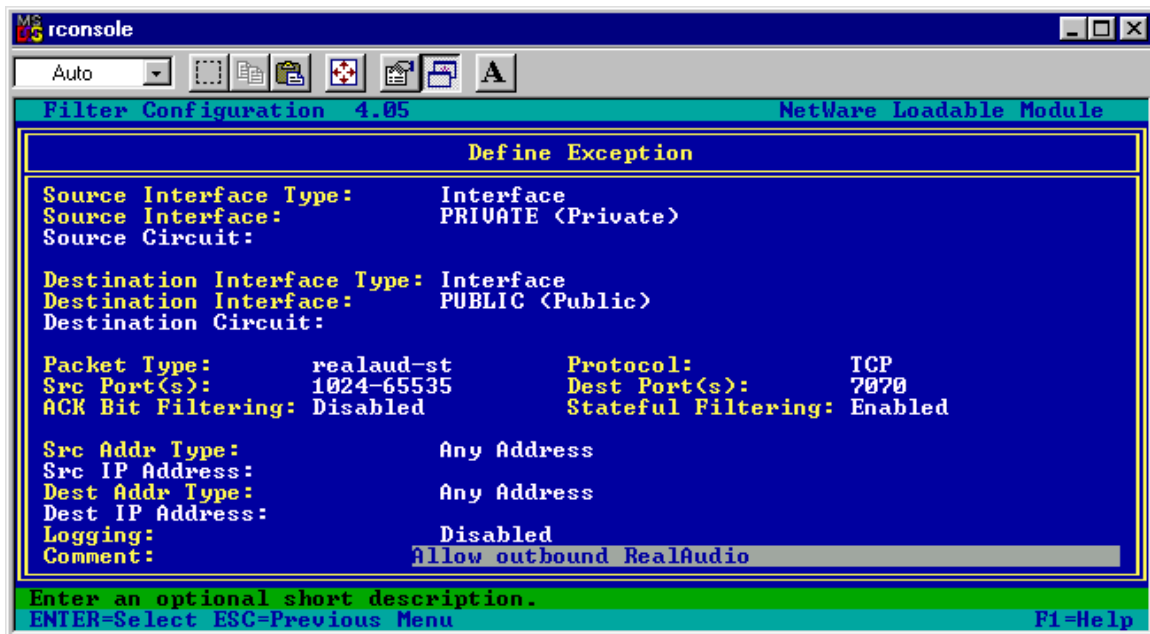


Figure 5-37 - Filter Exception for Outbound RealAudio (PNA)

The stateful filter exception shown in Figure 5-37 allows internal hosts not configured for a RealAudio proxy to access RealAudio (PNA) sources.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 7070
- Stateful filtering: Enabled

## Outbound RealAudio Filter Exception for BorderManager 3.7 RealAudio/RTSP Proxy

The following example shows the filter exception that a fresh BorderManager 3.7 installation provides.

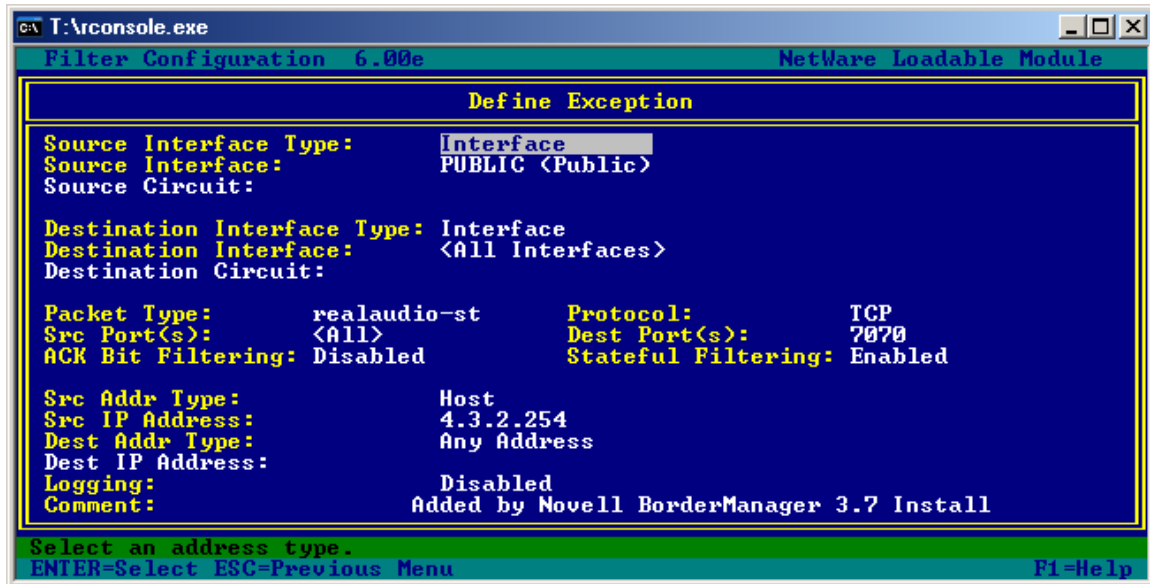


Figure 5-38 - Filter Exception for RealAudio from BorderManager 3.7 RealAudio/RTSP Proxy

The example shown in Figure 5-38 allows the RealAudio/RTSP Proxy to make RealAudio (PNA) requests to a RealAudio on the Internet.

- Source interface: Public
- Destination Interface: All Interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 7070
- Stateful filtering: Enabled
- Source IP Address: <BorderManager 3.7 Public IP address>

# RTSP (Real Time Streaming Protocol)

## Outbound RTSP Filter Exception for Internal Hosts

This exception is useful if you have problems with the RTSP Proxy in BorderManager 3.5 or 3.6, or are using BorderManager 3.0, which doesn't have an RTSP Proxy.

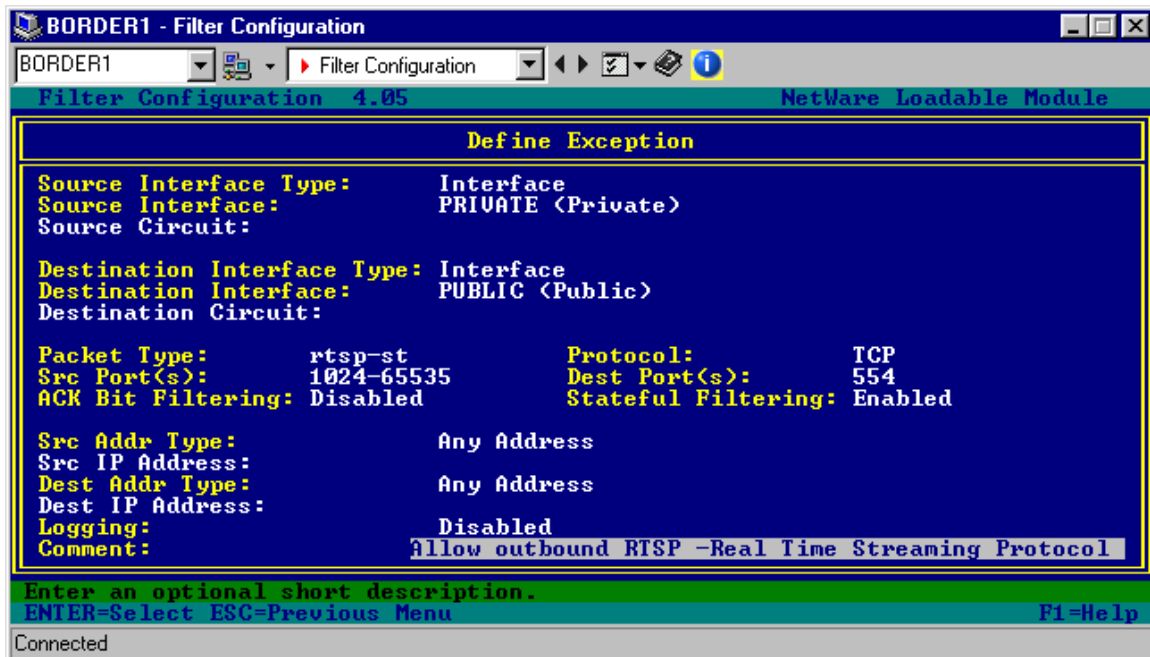


Figure 5-39 - Filter Exception for Outbound RTSP

The filter exception shown in Figure 5-39 should be used when RealPlayer is not configured to use an RTSP Proxy.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 554
- Stateful filtering: Enabled

**Note** A number of versions of PROXY.NLM in BorderManager 3.5 and 3.6 have had problems with RTSP Proxy. The problems should be solved using PROXY.NLM version 022 or later.

Apple's Quicktime program can also be configured to use RTSP protocol, and would use the same filter exception.

## Outbound RTSP Filter Exception for BorderManager 3.7 RealAudio/RTSP Proxy

The following example shows the filter exception that a fresh BorderManager 3.7 installation provides.

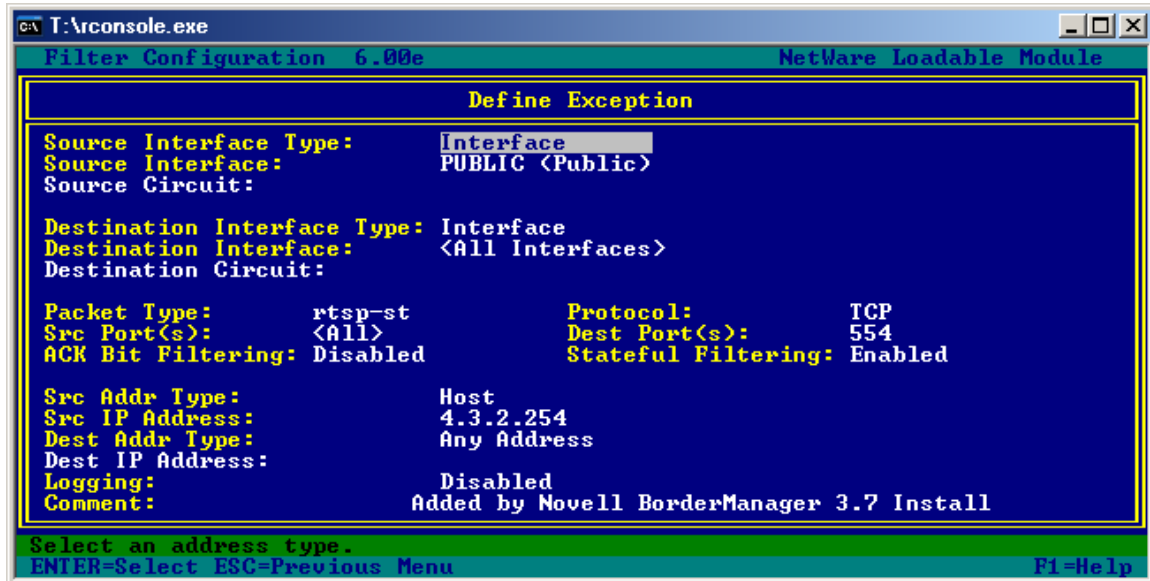


Figure 5-40 - Filter Exception for RTSP from BorderManager RealAudio/RTSP Proxy

The example shown in Figure 5-40 allows the RealAudio/RTSP Proxy to make RTSP requests to a RTSP server on the Internet.

- Source interface: Public
- Destination Interface: All Interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 554
- Stateful filtering: Enabled
- Source IP Address: <BorderManager 3.7 Public IP address>

---

**Note** While RTSP uses TCP destination port 554 between clients, in this case RTSP Proxy and RTSP server, the PC client software (e.g. RealAudio) connecting to the RTSP Proxy needs to specify TCP port 9090 for RTSP transport.

---

# SMTP

## Outbound SMTP Filter Exception for Internal Hosts

Since the BorderManager 3.x mail proxy has had a history of various problems and limitations, it can be useful to set up an SMTP filter exception. The exception shown will simply allow outbound SMTP, so that any host can send mail to an ISP's mail server by allowing port 25 traffic. Note that many ISP's may not allow SMTP relaying off their mail servers unless the SMTP source address originates within the ISP's network.

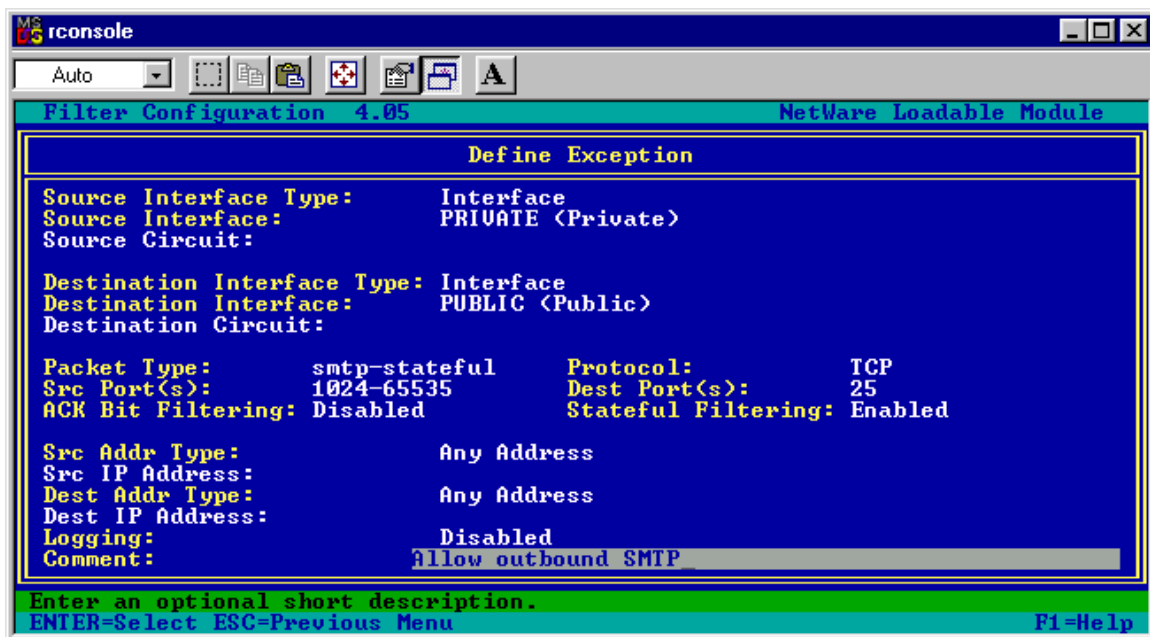


Figure 5-41 - Filter Exception for Outbound SMTP

The filter exception shown in Figure 5-41 allows internal hosts to send email to any external SMTP host that will accept it from your IP address. (Spam relay controls generally applied usually means that you can send SMTP only to an SMTP server at your ISP).

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 25
- Stateful filtering: Enabled

## Outbound SMTP Filter Exception for BorderManager 3.7 Mail Proxy

The following example shows the filter exception that a fresh BorderManager 3.7 installation provides.

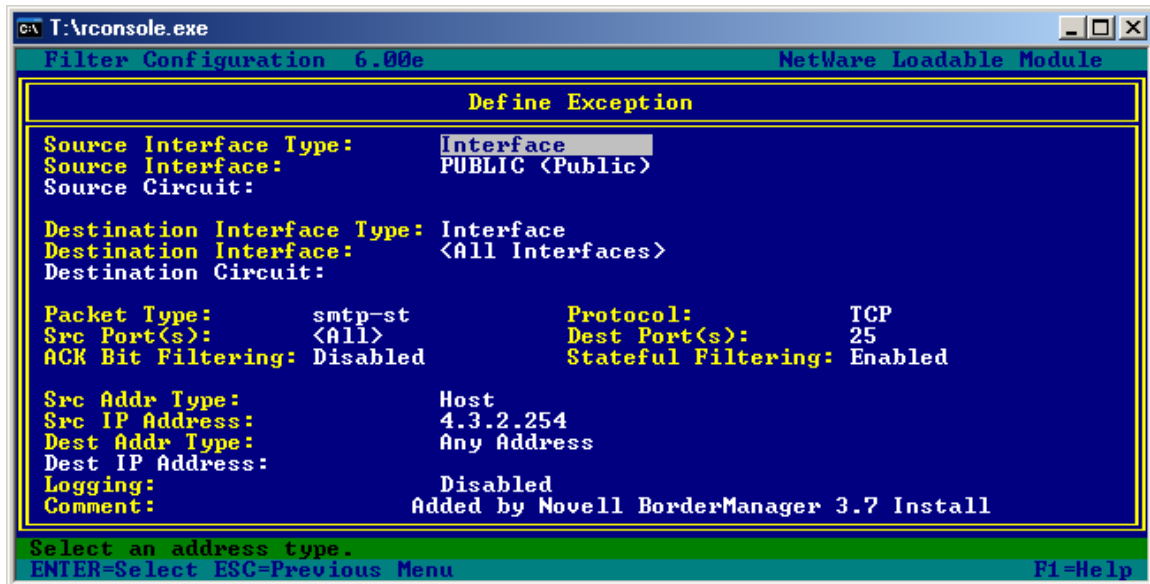


Figure 5-42 - Filter Exception for SMTP from BorderManager 3.7 Mail Proxy

The example shown in Figure 5-42 allows the Mail Proxy to make outbound SMTP requests to a SMTP server on the Internet. It does not allow inbound traffic to the Mail Proxy, so additional exceptions (shown later in this book) are needed for inbound SMTP so that the Mail Proxy can receive email.

- Source interface: Public
- Destination Interface: All Interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 25
- Stateful filtering: Enabled
- Source IP Address: <BorderManager 3.7 Public IP address>

## SSL (HTTPS)

### Outbound SSL Filter Exception for Internal Hosts

Even though you may be using the HTTP proxy to allow outbound web browsing, you may wish to allow SSL traffic to bypass the HTTP proxy. If so, you might also want to set up a stateful filter to allow port 443 out through the BorderManager server.

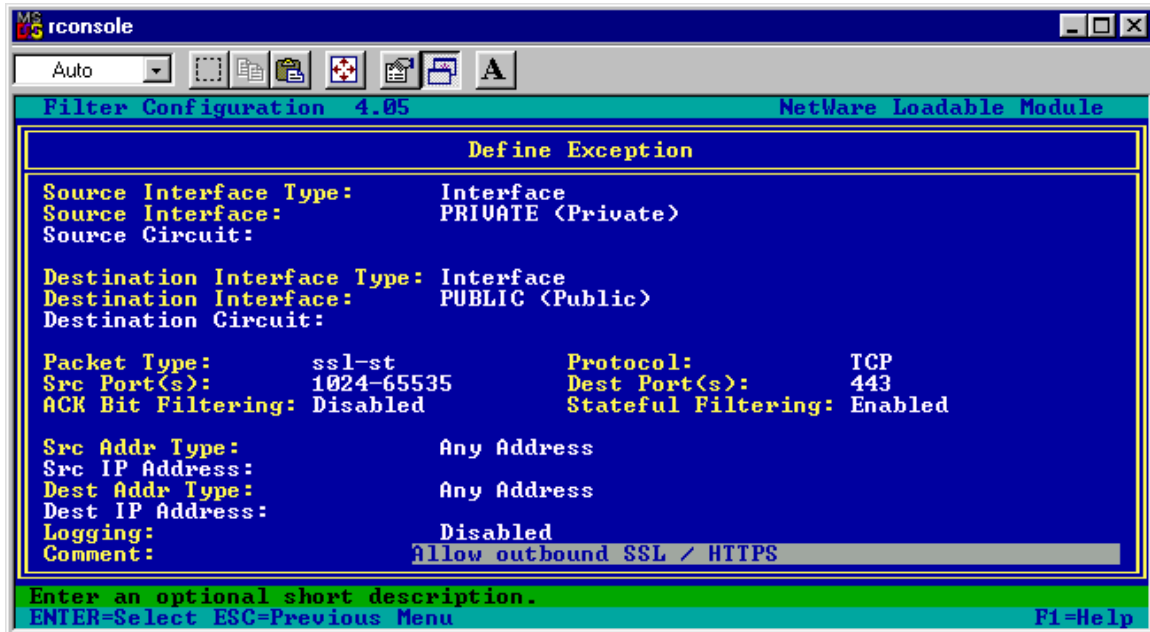


Figure 5-43 - Filter Exception for Outbound SSL / HTTPS

The filter exception shown in Figure 5-43 allows internal hosts to make HTTPS/SSL connections.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 443
- Stateful filtering: Enabled



## Outbound SSL Filter Exception for BorderManager 3.7 HTTP Proxy

The following example shows the filter exception that a fresh BorderManager 3.7 installation provides.

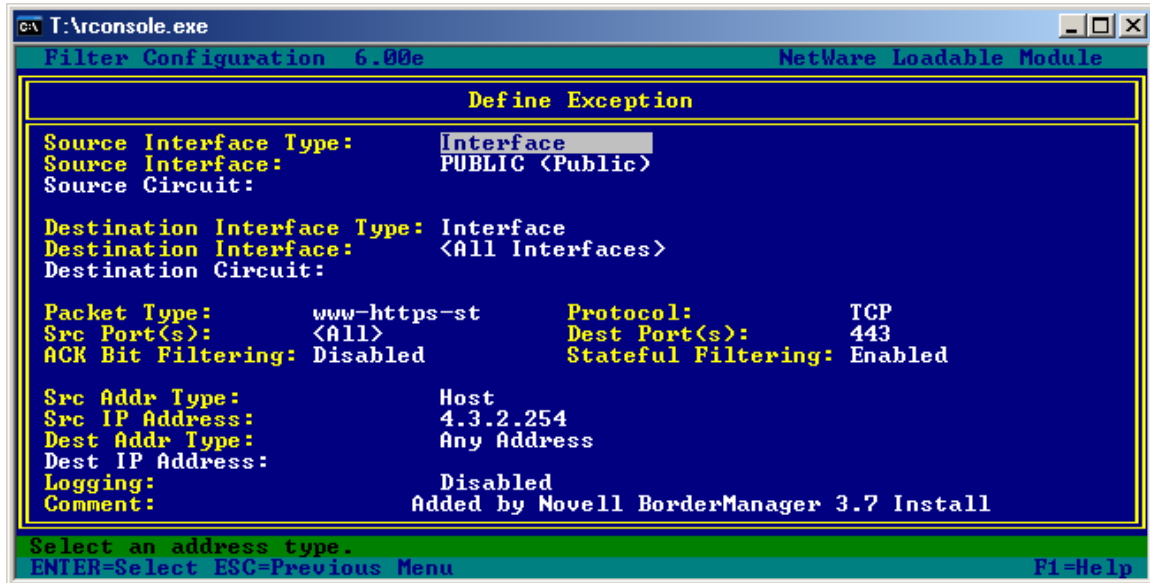


Figure 5-44 - Filter Exception for SSL (HTTPS) from BorderManager 3.7 HTTP Proxy

The example shown in Figure 5-44 allows the HTTP Proxy to make SSL (HTTPS) requests to web servers on the Internet.

- Source interface: Public
- Destination Interface: All Interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 443
- Stateful filtering: Enabled
- Source IP Address: <BorderManager 3.7 Public IP address>

# TELNET

## Outbound TELNET Filter Exception for Internal Hosts

This example will allow any user in your LAN to establish a TELNET session to an external host.

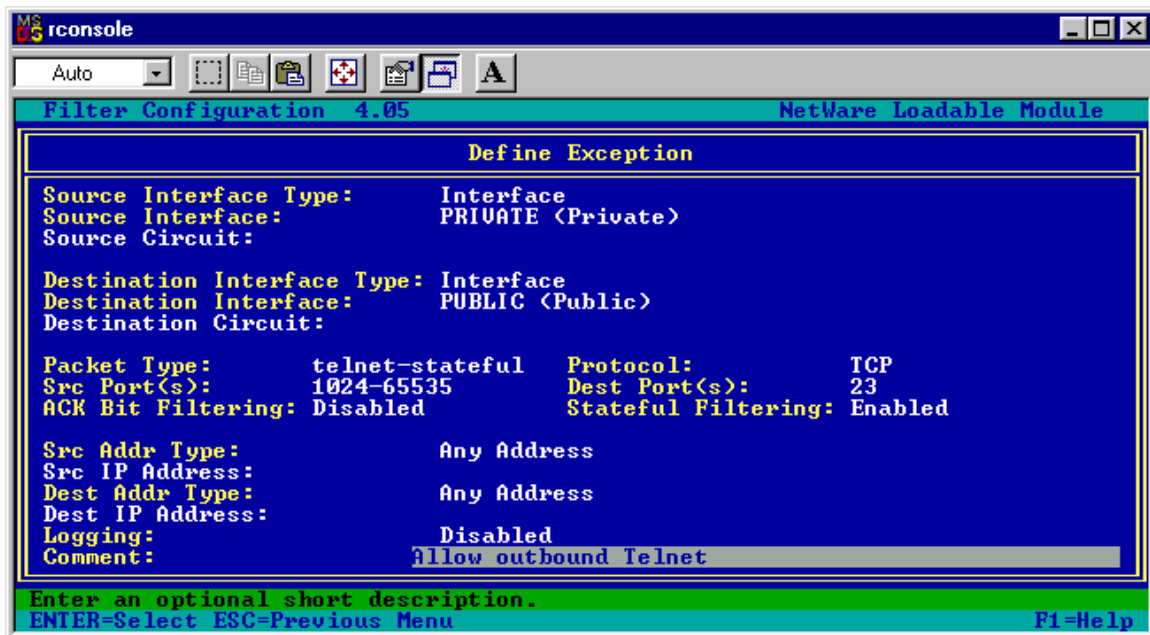


Figure 5-45 - Filter Exception for Outbound TELNET

The filter exception shown in Figure 5-45 allows internal hosts to make outbound TELNET connections on the standard TELNET port number.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 23
- Stateful filtering: Enabled

---

**Note** This filter exception is not strictly necessary for BorderManager 3.5 or 3.6, which provides a Transparent TELNET proxy, but the Transparent Telnet proxy there has had some history of causing problems with the server, such as ABENDS.

---

## Outbound TELNET Filter Exception for BorderManager 3.7 Transparent Telnet Proxy

The following example shows the filter exception that a fresh BorderManager 3.7 installation provides.

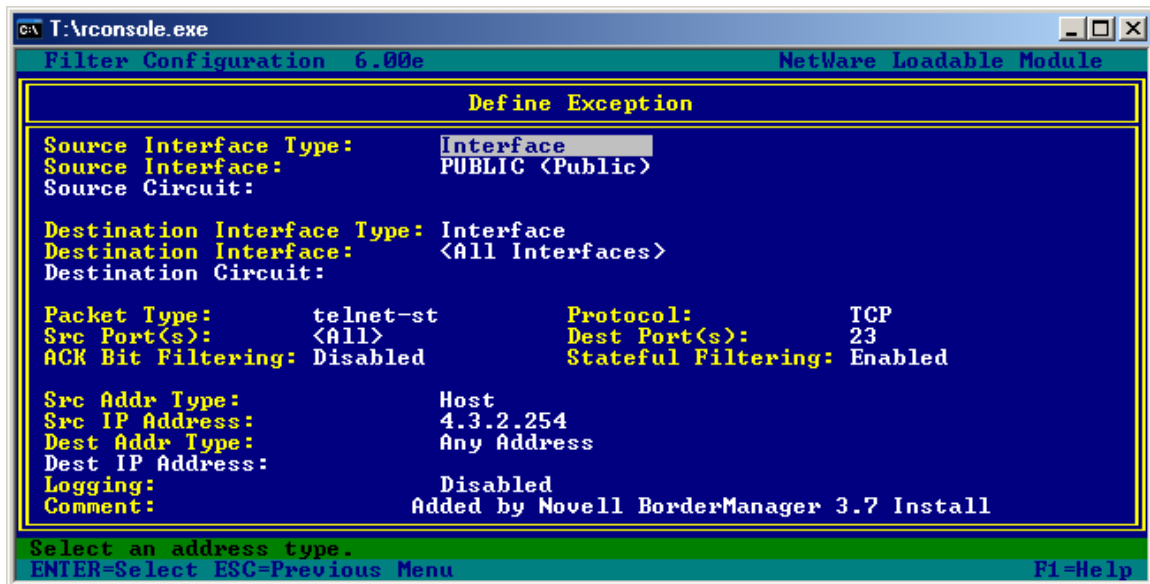


Figure 5-46 - Filter Exception for TELNET from BorderManager 3.7 Transparent Telnet Proxy

The example shown in Figure 5-46 allows the Transparent Telnet Proxy to make TELNET connections to a host on the Internet.

- Source interface: Public
- Destination Interface: All Interfaces
- Protocol: TCP
- Source ports: All
- Destination port: 23
- Stateful filtering: Enabled
- Source IP Address: <BorderManager 3.7 Public IP address>

## Terminal Server

In case you need to access a Microsoft Terminal Server outside your network, use the following filter exception. Should you need to make an internal Terminal Server available to the Internet via Static NAT, see the example later in this book on page 240.

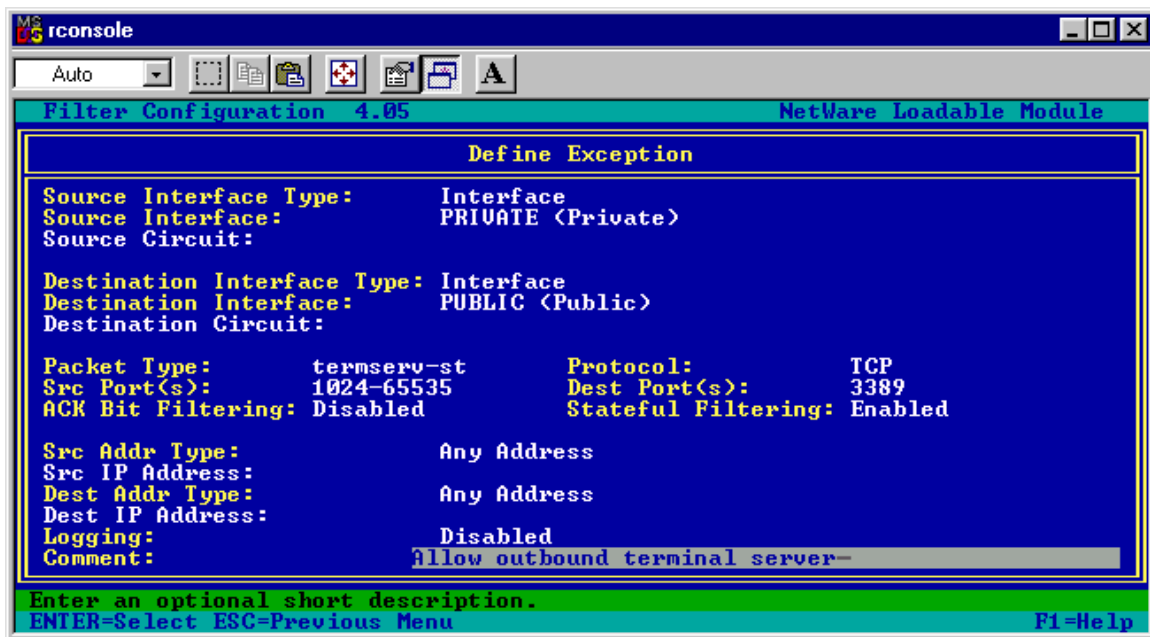


Figure 5-47 - Filter Exception for Outbound Microsoft Terminal Server

The filter exception shown in Figure 5-47 allows an internal host to access a Microsoft Terminal Server on the Internet.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 3389
- Stateful filtering: Enabled

## VNC Viewer

VNC is a free, open-source remote control program that can run on Windows. You can use it as an alternative to a program like pcANYWHERE, though it does not have the features or speed of pcANYWHERE. See the following URL to download VNC.

<http://www.uk.research.att.com/vnc/download.html>

VNC allows multiple sessions to be run at the same time, up to 10, at the time of this writing. Each session requires a different port number, starting at 5900 and going up to 5909. The example shown opens the entire range for the maximum number of simultaneous sessions.

An example for inbound usage through static NAT is shown later.

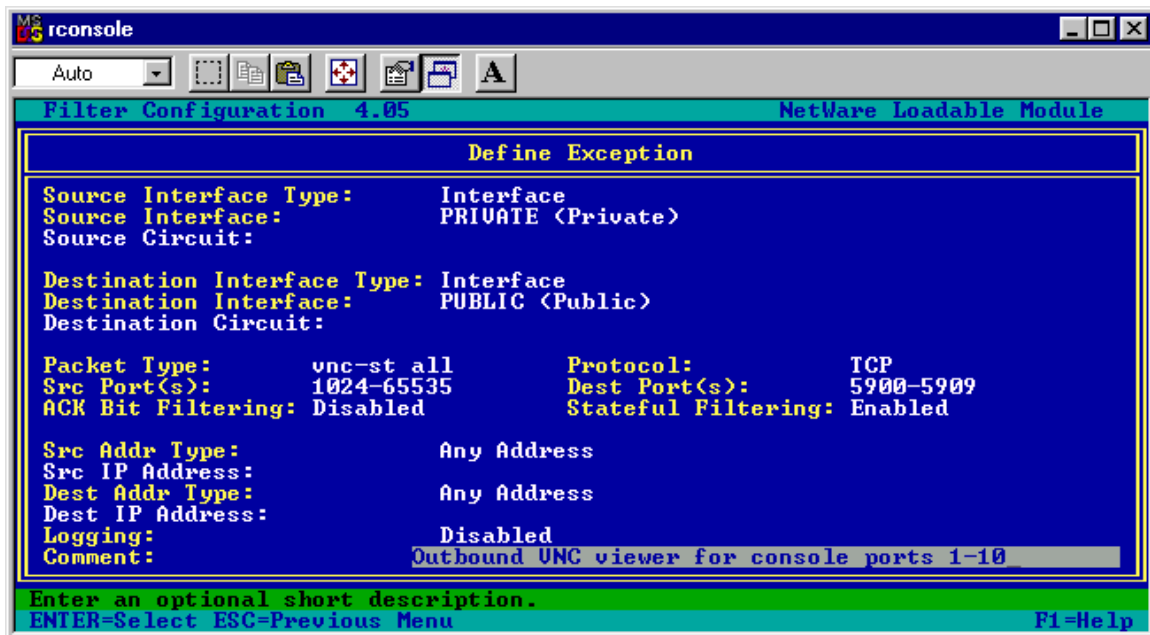


Figure 5-48 - Filter Exception for Outbound VNC Viewer for 10 Console Sessions

The filter exception shown in Figure 5-48 allows an internal host to use the VNC Viewer program to access a VNC server on the Internet.

- Source interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination ports: 5900-5909
- Stateful filtering: Enabled

## VNC Browser Interface

VNC can also be accessed via a browser on ports 5800 through 5809. Similar to the VNC Viewer, up to 10 sessions can be opened, with session 1 using port 5800, session 2 using 5801, etc.

**In addition to the ports shown in this example**, TCP destination ports 5900-5909 must also be opened, as in the previous example for VNC Viewer, and TCP destination port 80 (HTTP) will be used. If the browser is using HTTP Proxy, you do not need to open TCP destination port 80 through BorderManager.

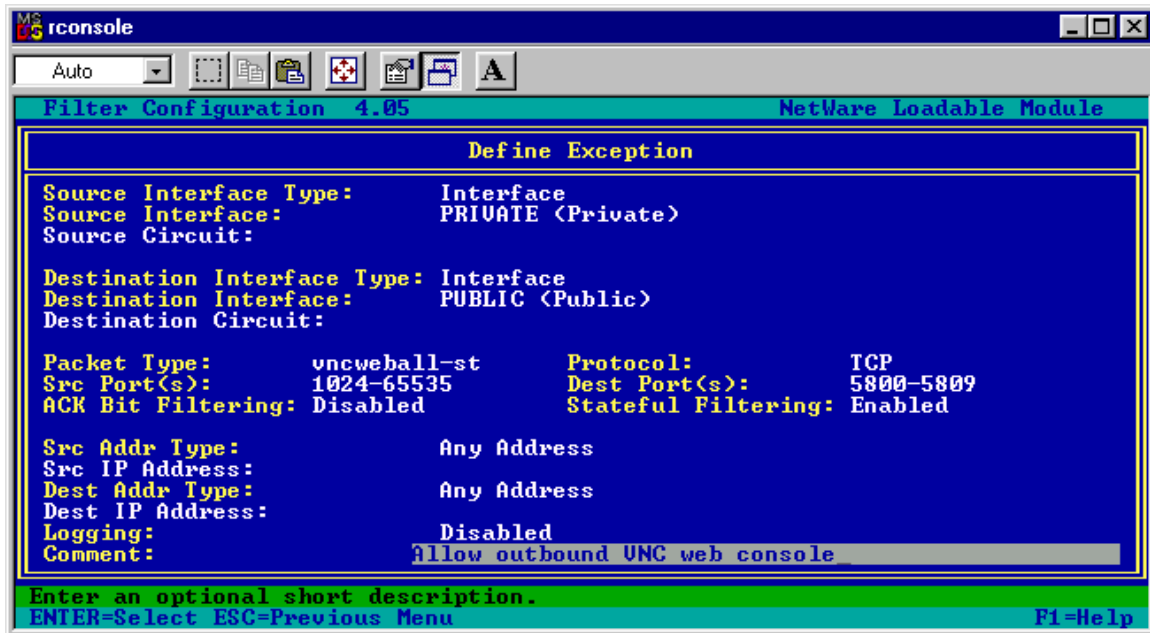


Figure 5-49 - Filter Exception for Outbound VNC through a Web Browser for 10 Console Sessions

The filter exception shown in Figure 5-49 allows an internal host to use a web browser to connect to a VNC host on the Internet, as long as the browser is also able to make an HTTP connection to the host, and as long as the VNC Viewer ports are opened.

- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination ports: 5800-5809
- Stateful filtering: Enabled

# Chapter 6 - Example Inbound Filter Exceptions

---

This chapter describes certain filter exceptions designed to allow traffic to services listening on the BorderManager server public interface, including Reverse HTTP, Reverse FTP and Generic Proxies, as well as services like DHCP that may be running on the server.

The following chapter covers inbound exceptions for Static NAT.

The main difference between filter exceptions in this chapter and filter exceptions for static NAT is that the source and destination IP addresses for Static NAT call out internal IP addresses, not addresses bound on the BorderManager server itself.

---

**Note** Many of the filter exception examples shown here are specifically for BorderManager 3.7, which does not allow all inbound high ports for TCP and UDP as previous versions did. However, if you have customized your BorderManager 3.0, 3.5 or 3.6 server by removing the default exceptions as shown in the Advanced chapter, you will find these examples useful.

---

## DHCP to a PC on the Public Subnet

It can be useful to set up filter exceptions to allow DHCP clients to receive addresses on the public network. **Why?** I find it useful to configure a single DHCP address to be delivered so that I can easily move a laptop PC from the private side to the public side for testing. This is much quicker than manually resetting the IP address. All I have to do is release the old address, plug the laptop into a hub on the public side of the BorderManager server, and renew the address. I get a new IP address, run my tests (which might be checking reverse proxy or static NAT access through new filter exceptions).

Three exceptions are required – one for BOOTPC (BootP Client) and two for BOOTPS (BootP Server).

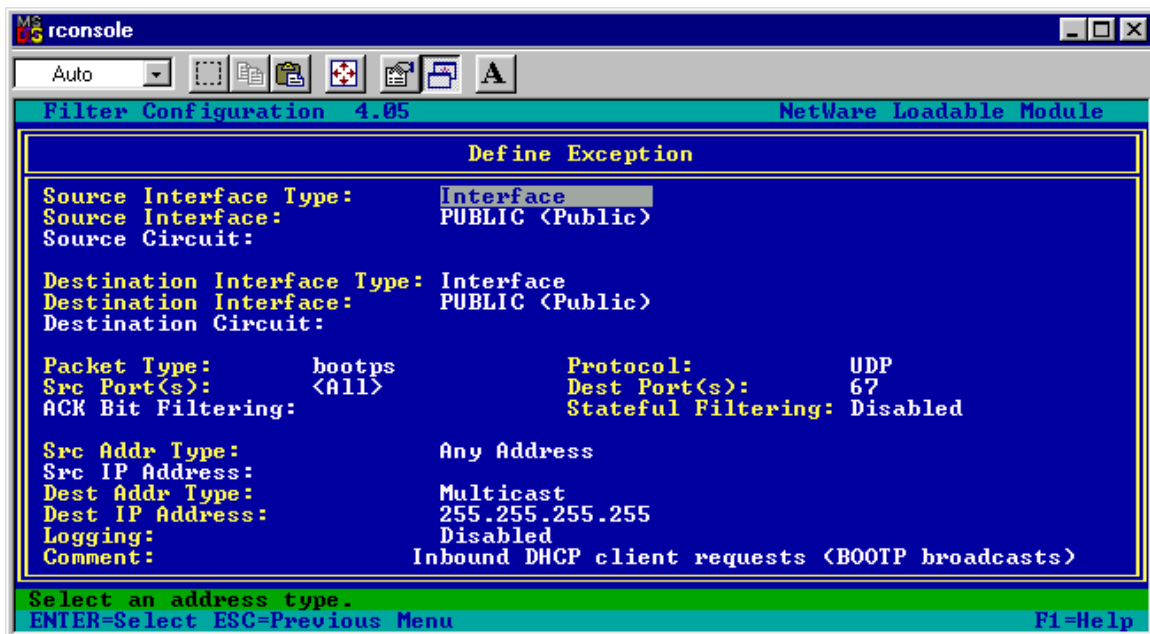


Figure 6-1 - Filter Exception for Initial DHCP Client Request to Broadcast Address on Public Interface

The filter exception shown in Figure 6-1 allows the DHCP requests in to the BorderManager server on the public interface.

- Source interface: Public
- Destination Interface: Public
- Protocol: UDP
- Source ports: <all>
- Destination port: 67
- Destination IP Address: Multicast, 255.255.255.255



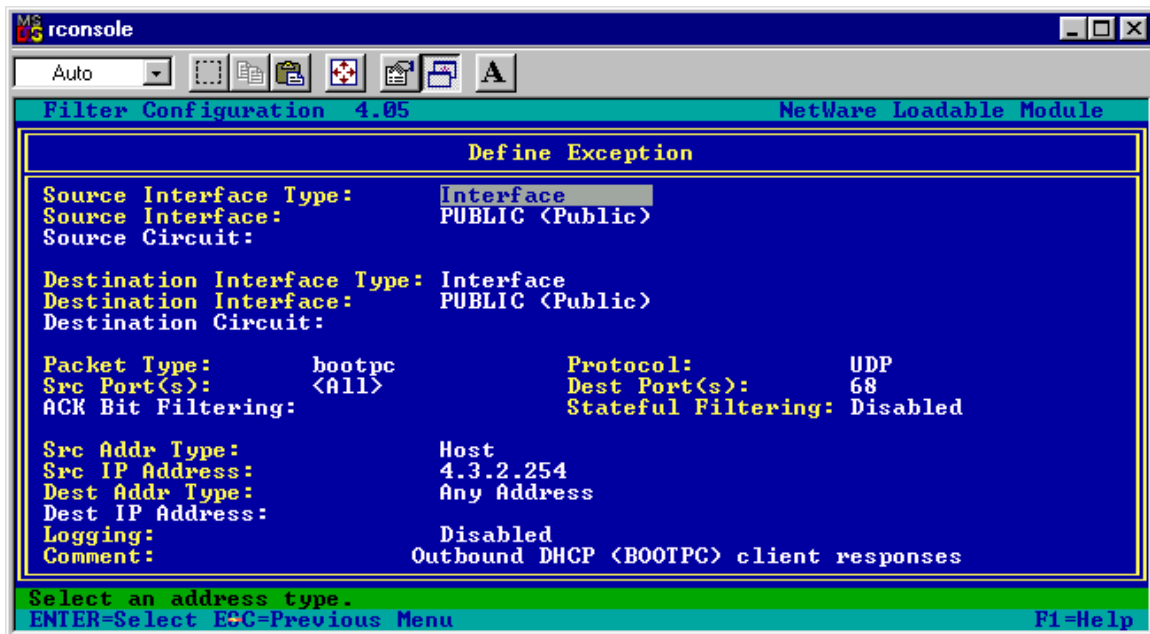


Figure 6-2 - Filter Exception for DHCP Client Responses from Public IP Address

The filter exception shown in Figure 6-2 allows the BorderManager server to respond to DHCP requests

- Source interface: Public
- Destination Interface: Public
- Protocol: UDP
- Source ports: <All>
- Destination port: 68
- Source IP Address: <your public IP address>

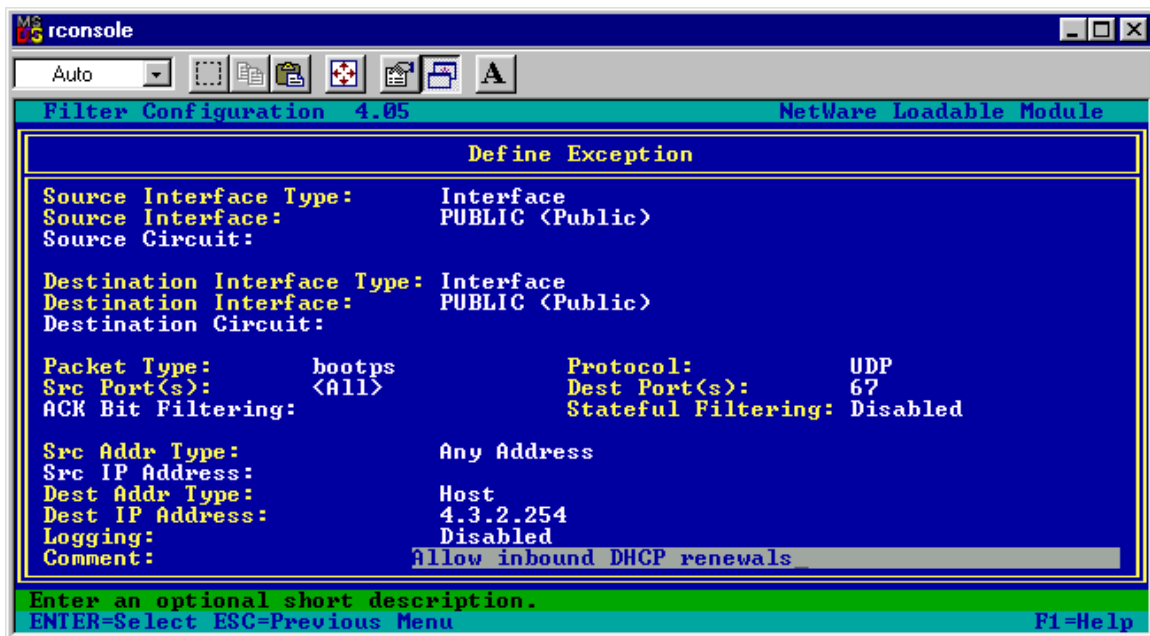


Figure 6-3 - Filter Exception for Inbound DHCP Renewal Requests

The third filter exception, shown in Figure 6-3, allows BOOTPS requests to the public IP address of the BorderManager server. This exception is necessary if you wish to allow DHCP renewal requests from the client.

- Source interface: Public
- Destination Interface: Public
- Protocol: UDP
- Source ports: <all>
- Destination port: 67
- Destination IP Address: 4.3.2.254

## DHCP to the BorderManager Server

This example is for a BorderManager server receiving its public IP address from a DHCP server, for a cable modem connection primarily.

The initial request for a DHCP address is made using a UDP broadcast, and the reply will use a broadcast address, since there is no initial IP address. Once a DHCP lease has been obtained, there will be periodic lease renewals, using UDP unicast addressing. Because of the requirement for using broadcasts, the filter exceptions required cannot limit DHCP traffic to a specific IP address. In fact, because of the nature of DHCP, the filter exceptions shown here will allow inbound DHCP requests to the public interface as well as outbound DHCP requests from the public interface.

DHCP requests are sent out using BOOTPS (UDP destination port 67) and received (by a DHCP client) using BOOTPC. Therefore, the outgoing traffic must include an exception to allow BOOTPS, if you are not using the default filter exceptions. (The default filter exceptions for BorderManager allow ALL outbound IP.) Inbound replies should be limited to broadcast and unicast BOOTPC packets, which use UDP destination port 68. This is different from the previous example, where inbound BOOTPS must be allowed.

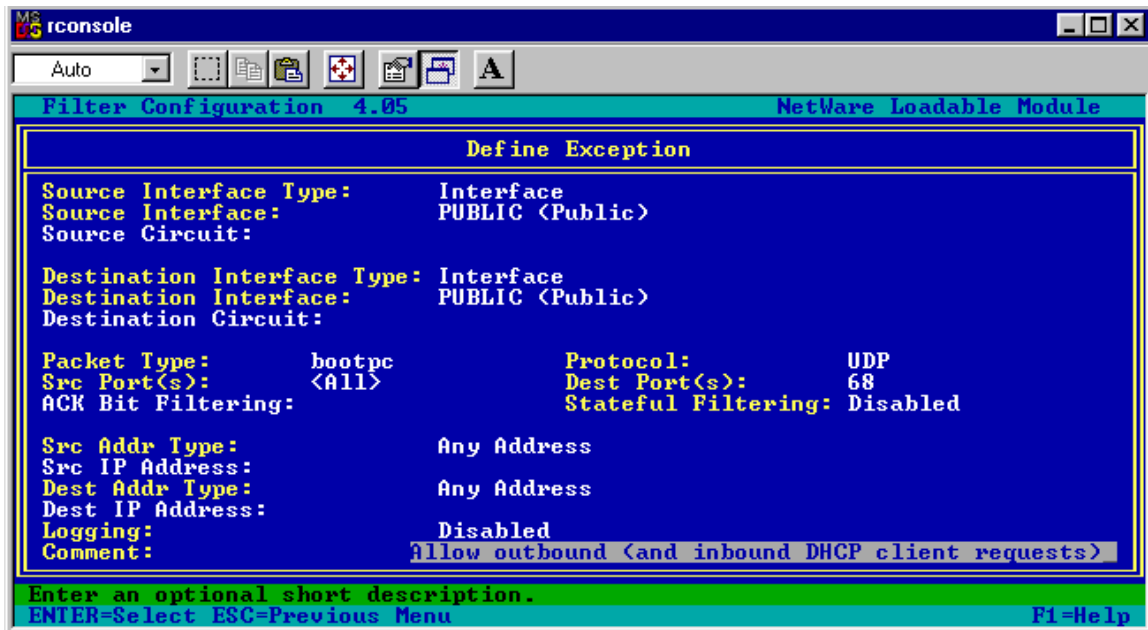


Figure 6-4 - Filter Exception for Public Interface to get DHCP Address

The filter exception shown in Figure 6-4 above shows BOOTPC allowed on the public interface. This filter exception allows the server to send and received broadcast DHCP address requests, and send and receive DHCP lease renewals.

- Source interface: Public
- Destination Interface: Public
- Protocol: UDP
- Source ports: <all>
- Destination port: 68

## Novell Remote Manager (NRM) on Generic TCP Proxy (on Secondary IP Address)

Novell Remote Manager (NRM) is very nice utility included with NetWare 5.1 and later servers that allows a great deal of management and troubleshooting to be done through a web interface. NRM used to be called portal, because it is launched with PORTAL.NLM. Because of the way it works when a user logs in, changing from one port to another and making a new connection, it doesn't work via static NAT. It will work fine through a generic TCP proxy configured for port 8008 and port 8009 (the default ports, which can be changed).

This example allows the inbound traffic for both standard NRM ports. It is possible that you could configure several different generic TCP proxies for different internal NRM servers on a single public IP address, as long as each NRM has been configured to listen on different port numbers. Different port numbers would require another set of custom filter exceptions.

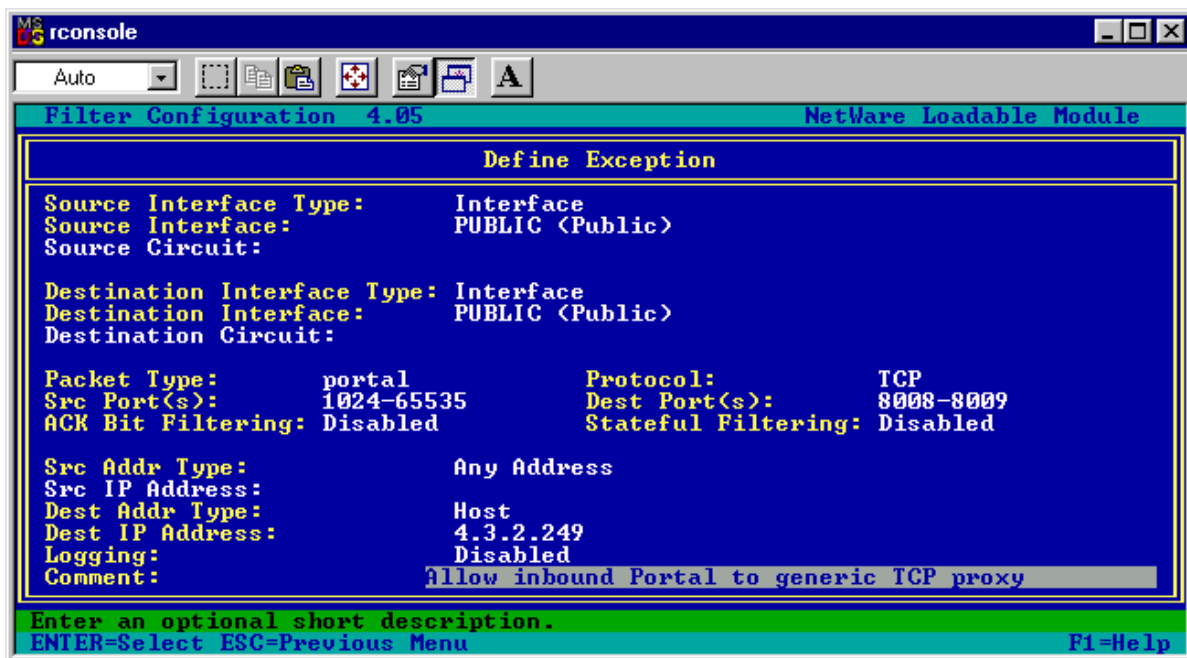


Figure 6-5 - Filter Exception for Inbound NRM to Generic TCP Proxy on Secondary IP Address

The filter exception show in Figure 6-5 allows a web browser on the Internet to send inbound traffic to access NRM via a Generic TCP Proxy listing on the specified public IP address.

- Source interface: Public
- Destination Interface: Public

- Protocol: TCP
- Source ports: 1024-65535
- Destination ports: 8008-8009
- Destination IP Address: <your generic TCP proxy public IP address>

The following exception allows the outbound return traffic from the Generic TCP Proxy for Portal Web Manager traffic.

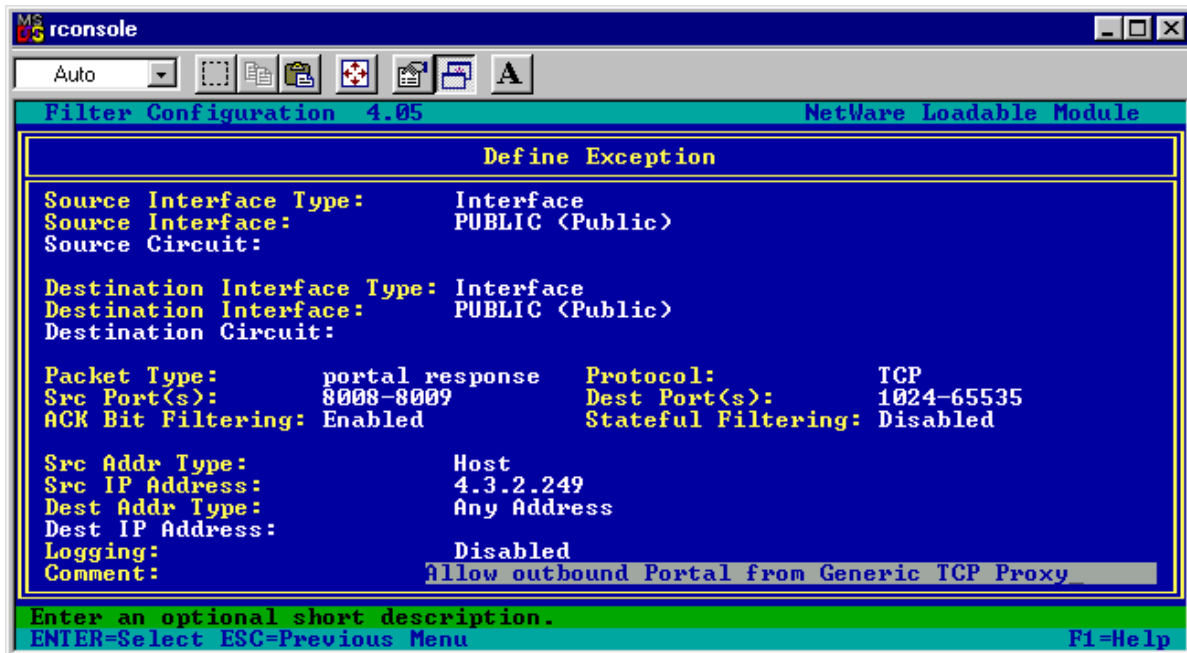


Figure 6-6 - Filter Exception for Portal Responses from Generic TCP Proxy on Secondary Public IP Address

The filter exception shown in Figure 6-6 allows a Generic TCP Proxy on IP address 4.3.2.249 to respond to inbound requests.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source ports: 8008-8009
- Destination ports: 1024-65535
- ACK Bit Filtering: enabled
- Source IP Address: <your generic TCP proxy public IP address>

## Reverse HTTP Proxy (on Secondary IP Address)

Reverse proxy acceleration of an internal web server to the outside is generally preferred. (Examples for accessing web servers through Static NAT are shown in the next chapter). The default filter exceptions allow HTTP and SSL to the main public IP address of the server for reverse HTTP Proxy. However, reverse proxy is often done using a secondary IP address, and the BorderManager default filters will block not only the requests to the secondary IP address, but also the responses from the secondary IP address (from the Proxy server). Therefore, set up the following two filter exceptions to allow HTTP traffic to and from the Reverse Proxy.

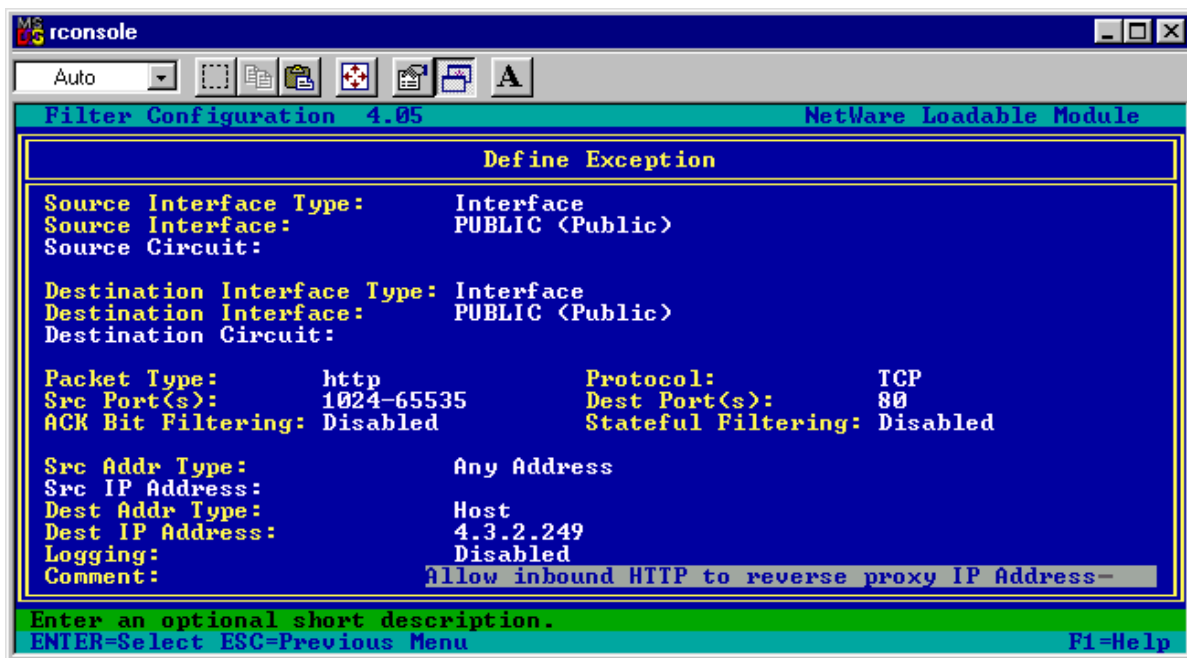


Figure 6-7 - Filter Exception for HTTP to Reverse HTTP Proxy on Secondary Public IP Address

The filter exception shown in Figure 6-7 will allow inbound HTTP requests to a reverse proxy on the specified destination IP address..

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 80
- Destination IP Address: <your reverse HTTP proxy public IP address>

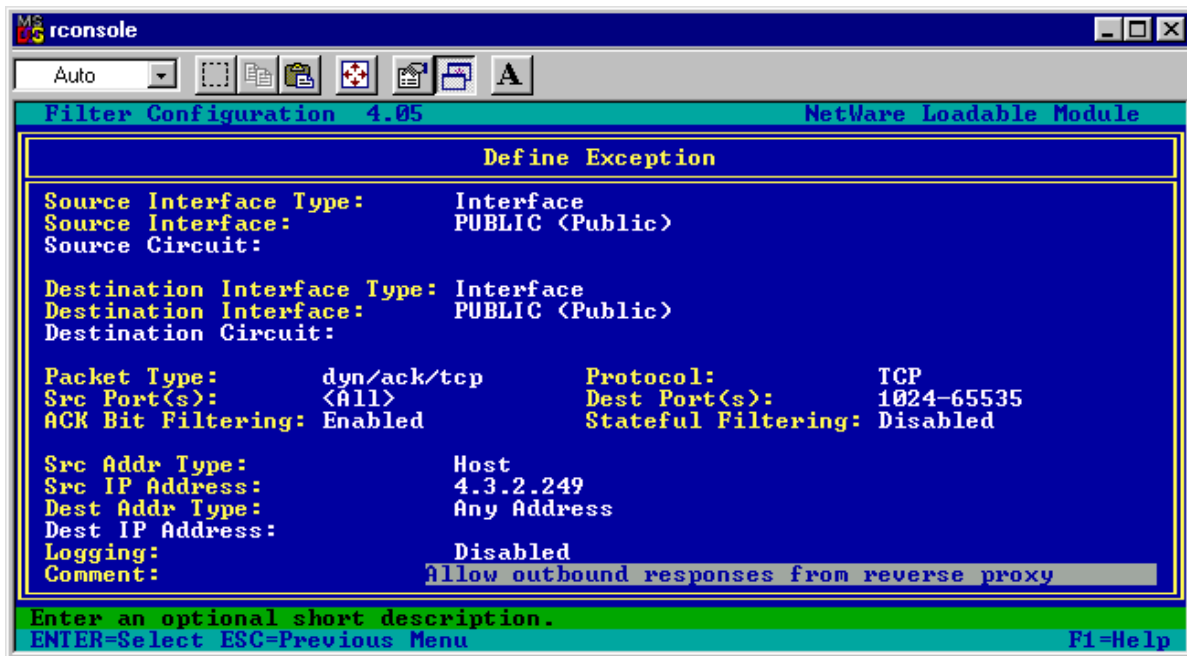


Figure 6-8 - Filter Exception for Reverse HTTP Proxy Responses from Reverse HTTP Proxy on Secondary Public IP Address

The filter exception shown in Figure 6-8 allows the reverse HTTP proxy to respond to inbound requests.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source port: <All> (see text below)
- Destination ports: 1024-65535
- ACK Bit Filtering: enabled
- Source IP Address: <your reverse HTTP proxy public IP address>

This exception shown in Figure 6-8 bears some explanation. Why was the source port left at Any? Why was the default Dynamic/TCP definition not used, with the source IP address called out? Why is the ACK bit set? First, **if** the web server is configured such that **ONLY** standard HTTP port 80 is used, a custom definition HTTP Response could be created, as above, except specifying a source port 80. However, some web servers have content that require HTTPS / SSL (port 443) to log in or receive a certificate and encrypt data. Other content might redirect the browser to non-standard HTTP ports (ports other 80 or 443). The exception shown should allow the web server to communicate in those situations, while still disallowing inbound connections to be made on the high ports (because the ACK bit is set). The requirement that the ACK bit be set ensures that the high ports are only used when the web server initiates the TCP connection.



## SSL to Reverse HTTP Proxy (on Secondary IP Address)

If your internal web server being reverse accelerated requires SSL (HTTPS), you also need to allow SSL port 443 traffic to the secondary IP address of the reverse proxy.

This filter exception also allows SSL Proxy Authentication to a reverse proxy should that option be enabled.

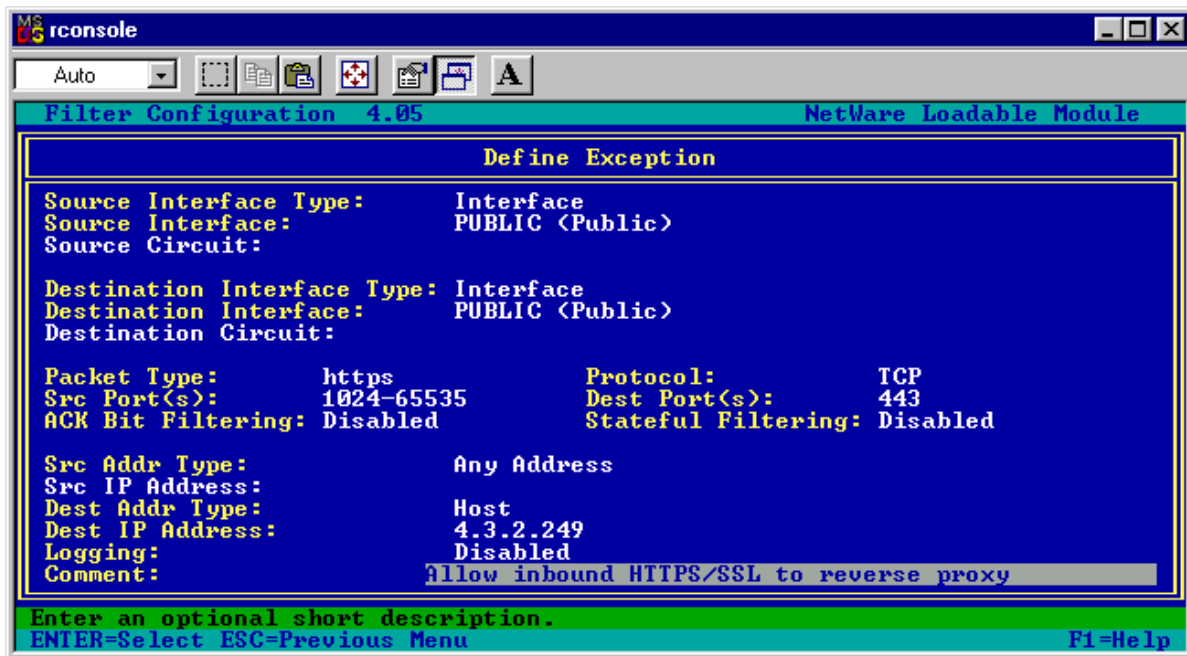


Figure 6-9 - Filter Exception for Inbound HTTPS/SSL to Reverse HTTP Proxy on Secondary Public IP Address

The filter exception shown in Figure 6-9 allows SSL (HTTPS) to the reverse proxy by allowing protocol TCP, any source port, and a destination port equal to 443 to a destination IP address set to the secondary IP address configured for reverse proxy acceleration. A custom exception has been defined that specifies the source ports for improved security.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 443
- Destination IP Address: <your reverse HTTP proxy public IP address>

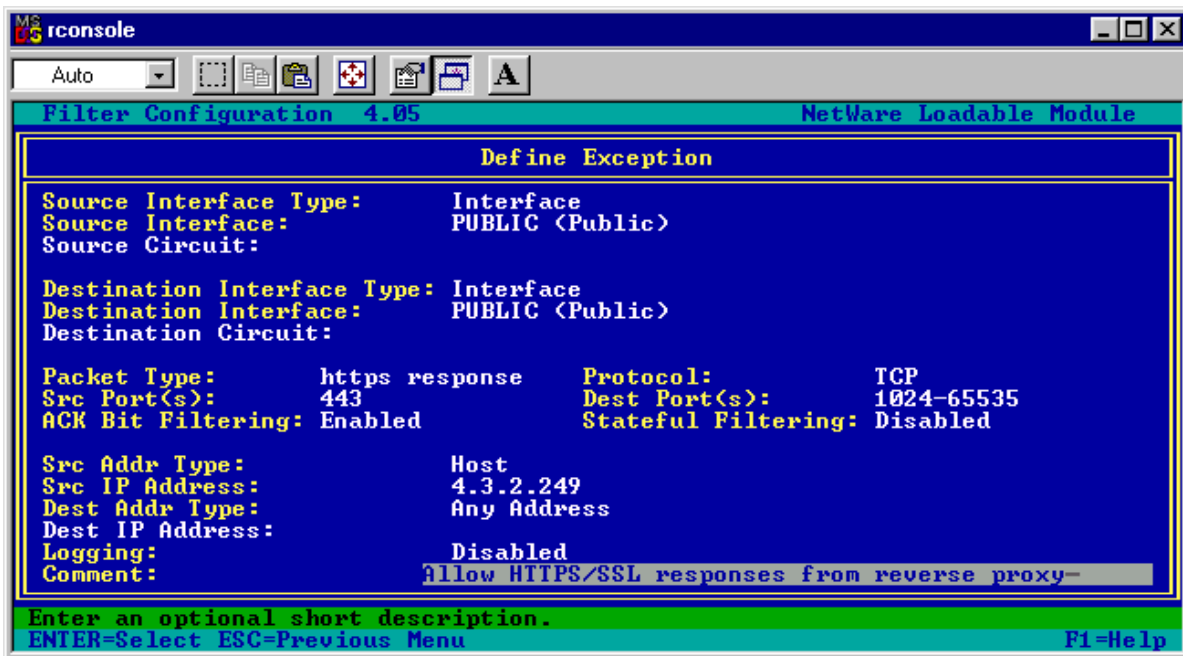


Figure 6-10 - Filter Exception for Outbound HTTPS / SSL Responses from Reverse HTTP Proxy on Secondary Public IP Address

The filter exception shown in Figure 6-10 allows outbound HTTPS / SSL responses from a reverse HTTP proxy on the specified source public IP address.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source port: 443
- Destination ports: 1024-65535
- ACK Bit Filtering: enabled
- Source IP Address: <your reverse HTTP proxy public IP address>

## FTP to Reverse FTP Proxy

FTP to the Reverse FTP Proxy is similar to FTP for a Static NAT connection. Inbound connections should be done with an inbound non-stateful exception (for ports 20 and 21) and one or two outbound non-stateful exceptions to allow the response packets. The IP addresses called out in these exceptions should be the public IP address of the Reverse FTP Proxy, whether that is the primary public IP address or a secondary.

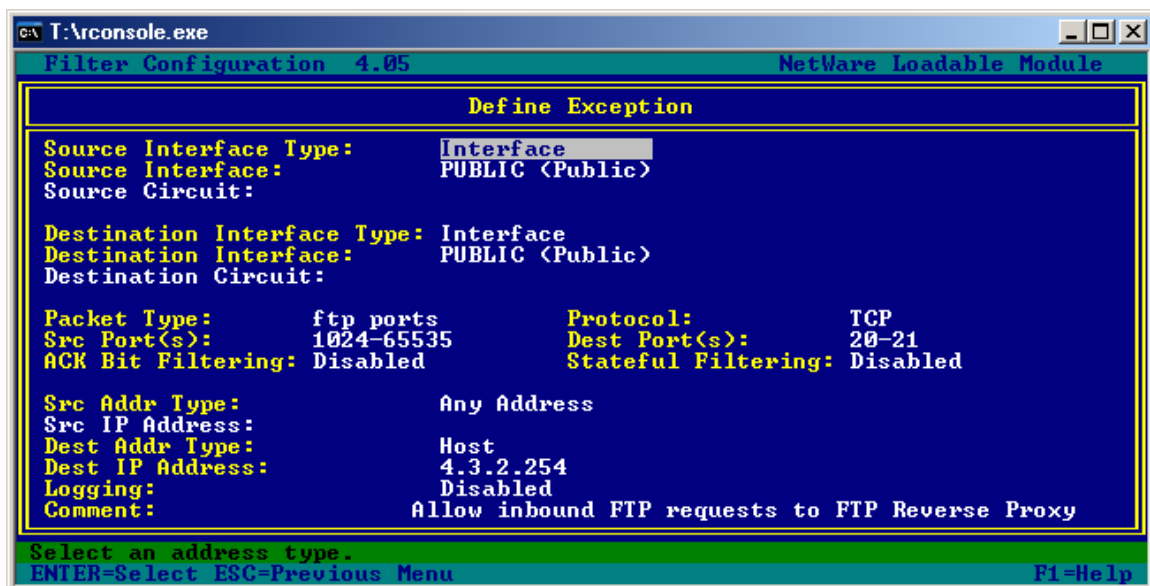


Figure 6-11 - Filter Exception for Inbound FTP Control and Data Ports to Reverse FTP Proxy

The filter exception shown in Figure 6-11 is all that was needed for CuteFTP, and command line FTP to make inbound connections and transfer data.

This custom filter exception uses a source and destination interface of the BorderManager public interface, any source IP address, and a destination IP address of the public IP address of the Reverse FTP Proxy.

- Source Interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Ports: 20-21
- Destination IP Address: <BorderManager public IP address assigned for Reverse FTP Proxy>

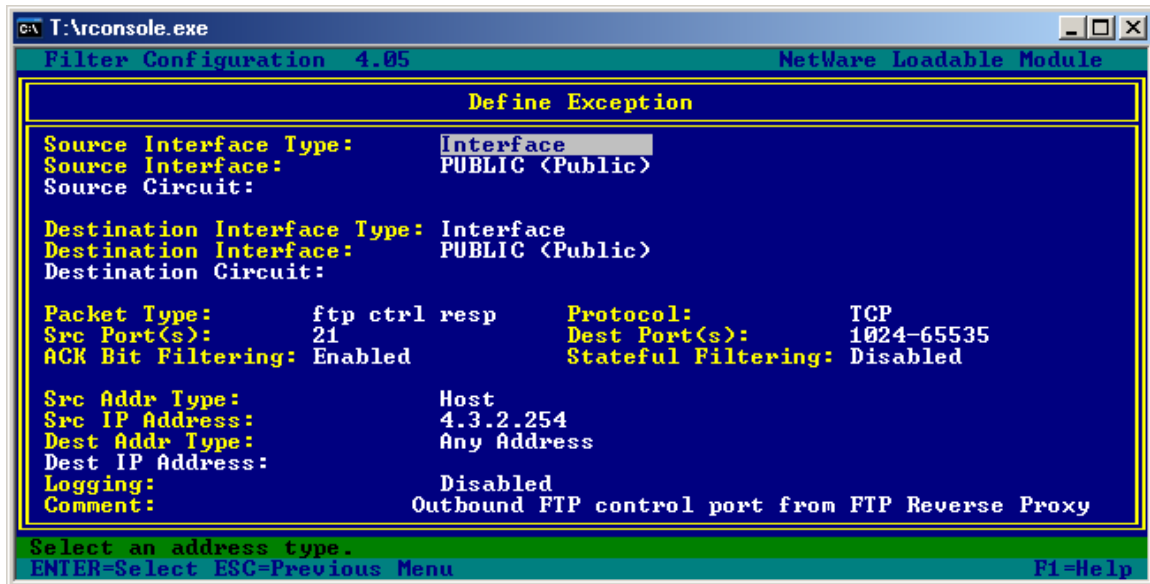


Figure 6-12 - Filter Exception for Outbound FTP Control Port Responses from Reverse FTP Proxy

The filter exception shown in Figure 6-12 allows outbound FTP control port responses from the Reverse FTP Proxy.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 21
- Destination port: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <BorderManager Reverse FTP Proxy public IP address>

This exception would not normally be needed for BorderManager 3.x servers prior to version 3.7, if the default filter exceptions are in place, and the proxy is listening on the primary public IP address. The default filter exceptions in those cases allow all outbound IP traffic from the public IP address (though not from a secondary IP address).

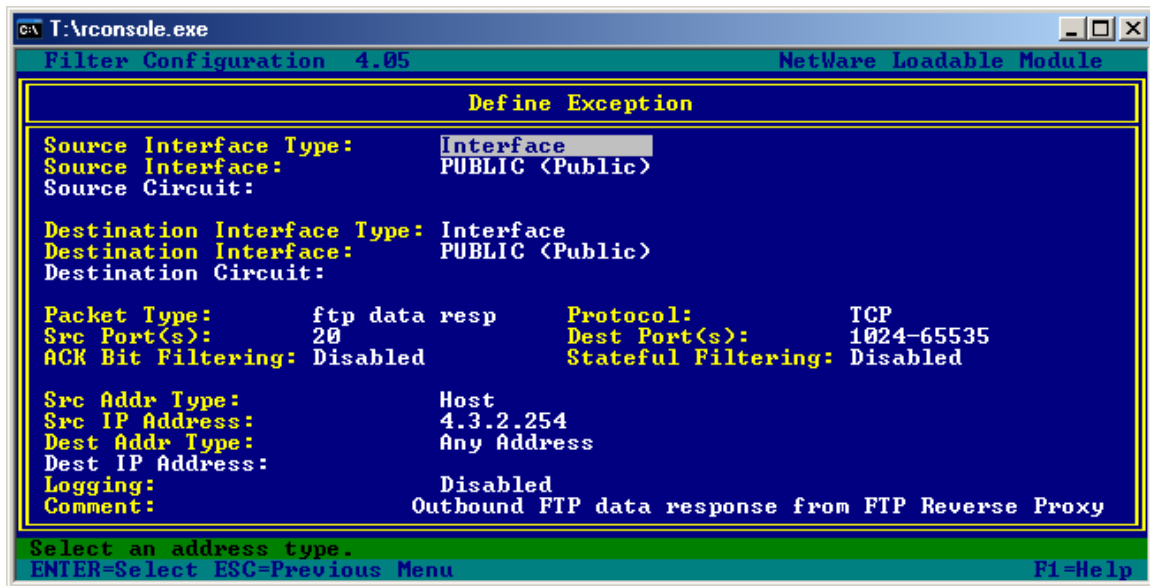


Figure 6-13 - Filter Exception for Outbound FTP Data Port Responses from FTP Reverse Proxy

The filter exception shown in Figure 6-13 allows outbound FTP data responses from the Reverse FTP proxy.

Note that ACK bit filtering has **NOT** been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 20
- Destination port: 1024-65535
- Source IP Address: <BorderManager Reverse FTP Proxy public IP Address>

An alternative to having two filter exceptions for outbound ports 20 and 21 would be to have a single exception for source ports 20-21, but not enable ACK bit filtering on it. If you enabled ACK bit filtering on outbound source port 20, your FTP data connections will fail.

This exception would not normally be needed for BorderManager 3.x servers prior to version 3.7, if the default filter exceptions are in place, and the proxy is listening on the primary public IP address. The default filter exceptions in those cases allow all outbound IP traffic from the public IP address (though not from a secondary IP address).

## SMTP to Mail Proxy or GWIA

The filter exceptions shown in this example allow inbound SMTP mail to be received by the Mail Proxy (or GroupWise GWIA running on the BorderManager server). Inbound connections should be done with an inbound non-stateful exception for port 25 and an outbound non-stateful exception to allow the response packets. The IP addresses called out in these exceptions should be the public IP address of the Mail Proxy, whether that is the primary public IP address or a secondary.

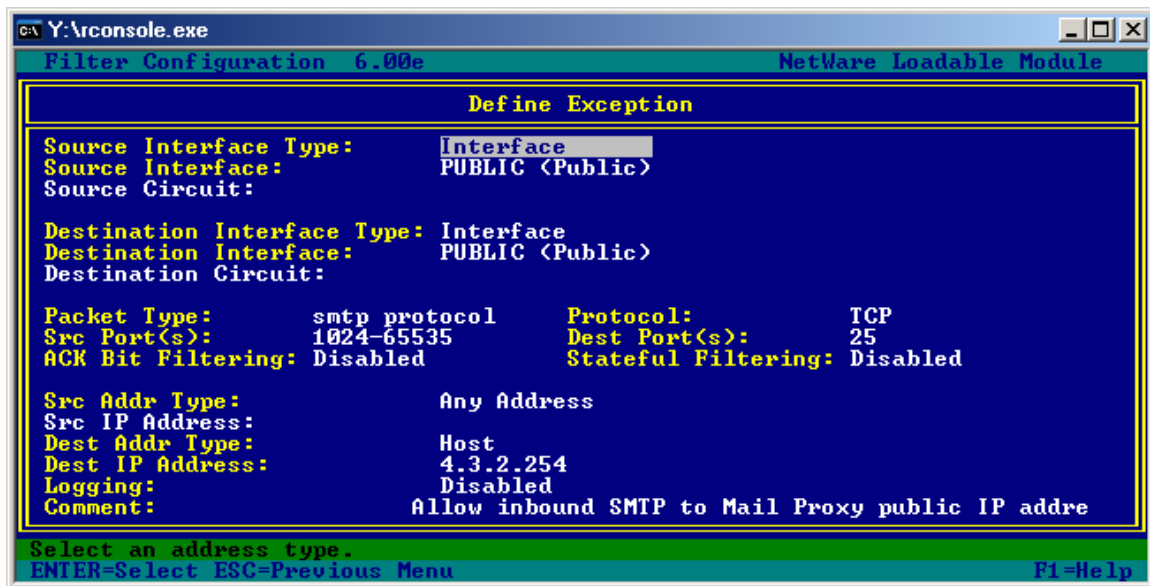


Figure 6-14 - Filter Exception for SMTP to Mail Proxy or GWIA

This custom filter exception uses a source interface and destination interface of the BorderManager public interface, any source IP address, and a destination IP address of the Mail Proxy public IP address

- Source Interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Ports: 25
- Destination IP Address: <BorderManager public IP address assigned for Mail Proxy>

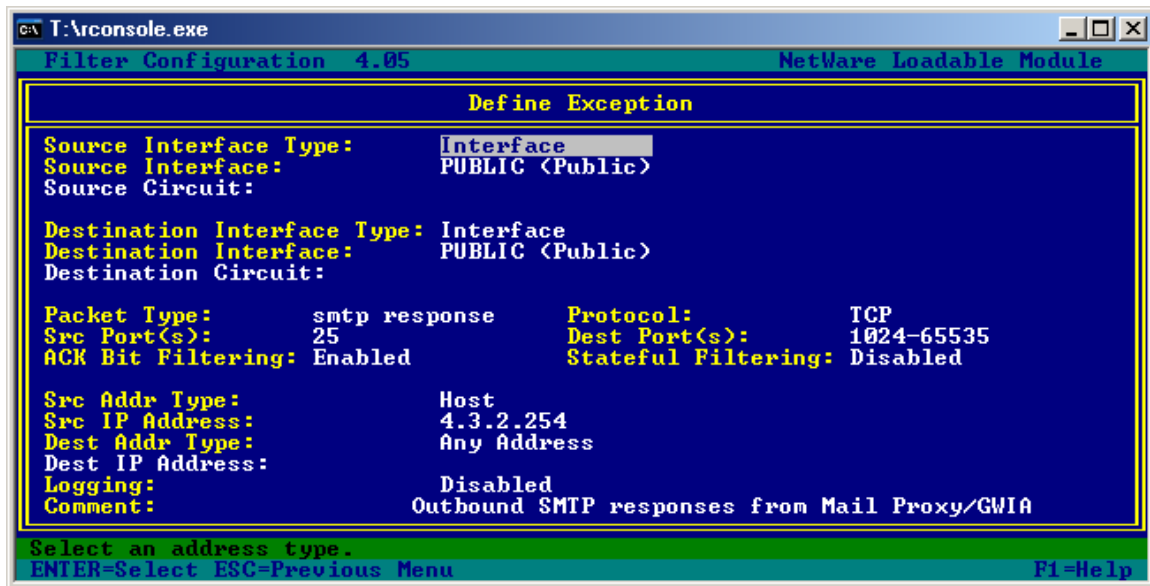


Figure 6-15 - Filter Exception for SMTP Responses from Mail Proxy or GWIA

This filter exception shown in Figure 6-15 uses a source interface and destination interface of the BorderManager public interface, a source IP address of the Mail Proxy public IP address, and any destination IP address.

- Source Interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source Port: 25
- Destination Ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <BorderManager public IP address assigned for Mail Proxy>

This exception would not normally be needed for BorderManager 3.x servers prior to version 3.7, if the default filter exceptions are in place, and the proxy is listening on the primary public IP address. The default filter exceptions in those cases allow all outbound IP traffic from the public IP address (though not from a secondary IP address).

## POP3 to Mail Proxy

The filter exceptions shown in this example allow inbound POP3 requests from a host on the Internet to the Mail Proxy. Inbound connections should be done with an inbound non-stateful exception for port 110 and an outbound non-stateful exception to allow the response packets. The IP addresses called out in these exceptions should be the public IP address of the Mail Proxy, whether that is the primary public IP address or a secondary.

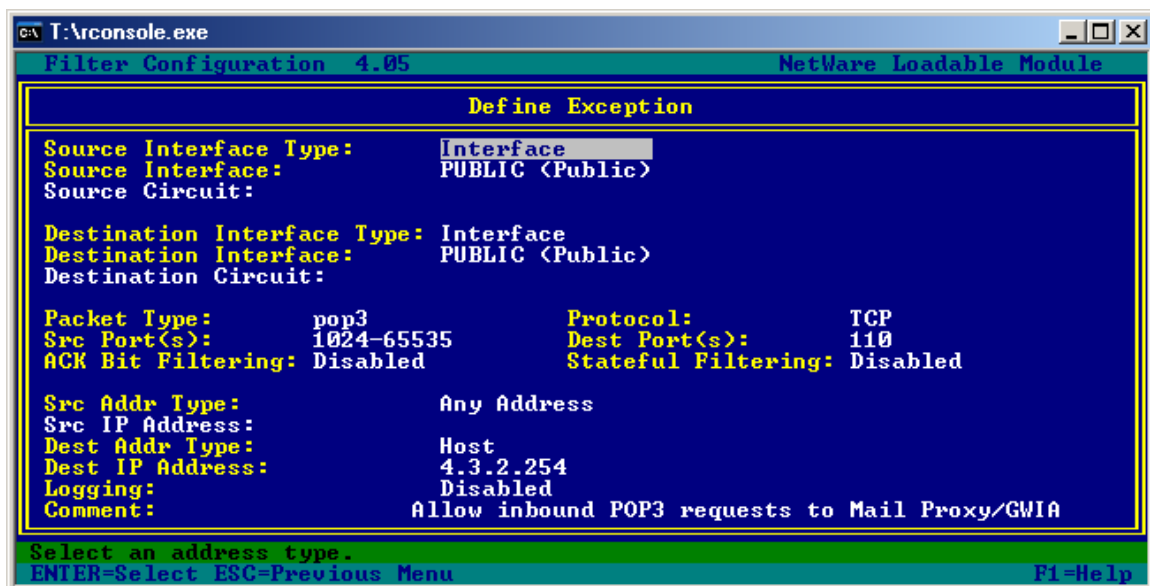


Figure 6-16 - Filter Exception for POP3 to Mail Proxy or GWIA

The filter exception shown in Figure 6-16 uses a source interface and destination interface of the BorderManager public interface, any source IP address, and a destination IP address of the Mail Proxy public IP address

- Source Interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 110
- Destination IP Address: <BorderManager public IP address assigned for Mail Proxy>



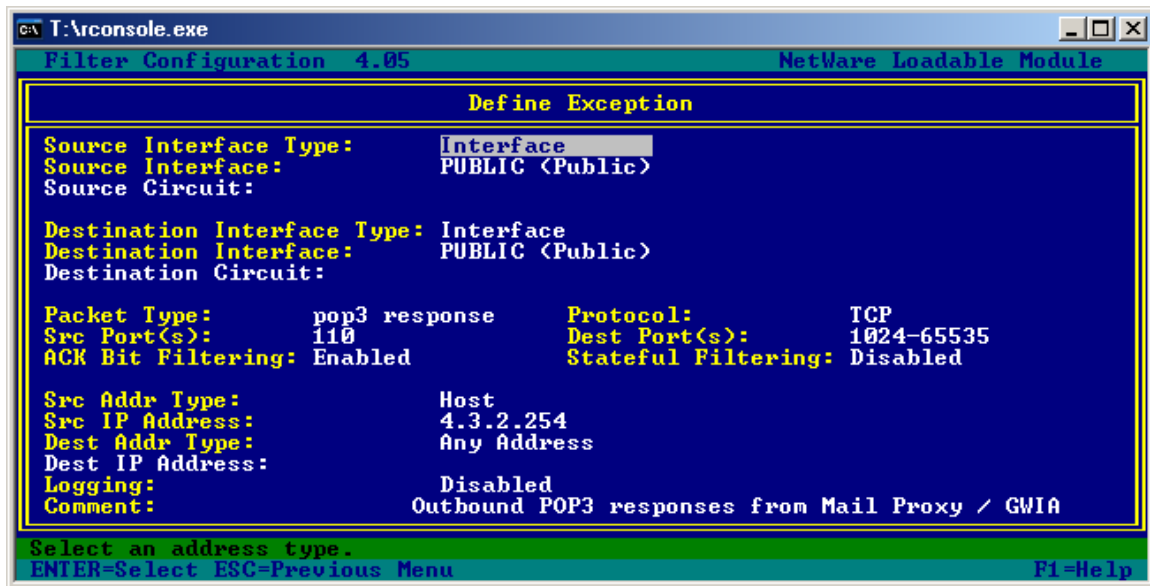


Figure 6-17 - Filter Exception for POP3 Responses from Mail Proxy or GWIA

The filter exception shown in Figure 6-17 uses a source interface and destination interface of the BorderManager public interface, a source IP address of the Mail Proxy public IP address, and any destination IP address.

- Source Interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source Port: 110
- Destination Ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <BorderManager public IP address assigned for Mail Proxy>

This exception would not normally be needed for BorderManager 3.x servers prior to version 3.7, if the default filter exceptions are in place, and the proxy is listening on the primary public IP address. The default filter exceptions in those cases allow all outbound IP traffic from the public IP address (though not from a secondary IP address).

## NNTP to News Proxy

The filter exceptions shown in this example allow inbound NNTP traffic from a Usenet server on the Internet to the News Proxy. This example would only be used if the News Proxy is connecting an internal NNTP server to the Usenet hierarchy on the Internet, not for outbound NNTP requests from internal NNTP clients. Inbound connections should be done with an inbound non-stateful exception for port 110 and an outbound non-stateful exception to allow the response packets. The IP addresses called out in these exceptions should be the public IP address of the News Proxy, whether that is the primary public IP address or a secondary.

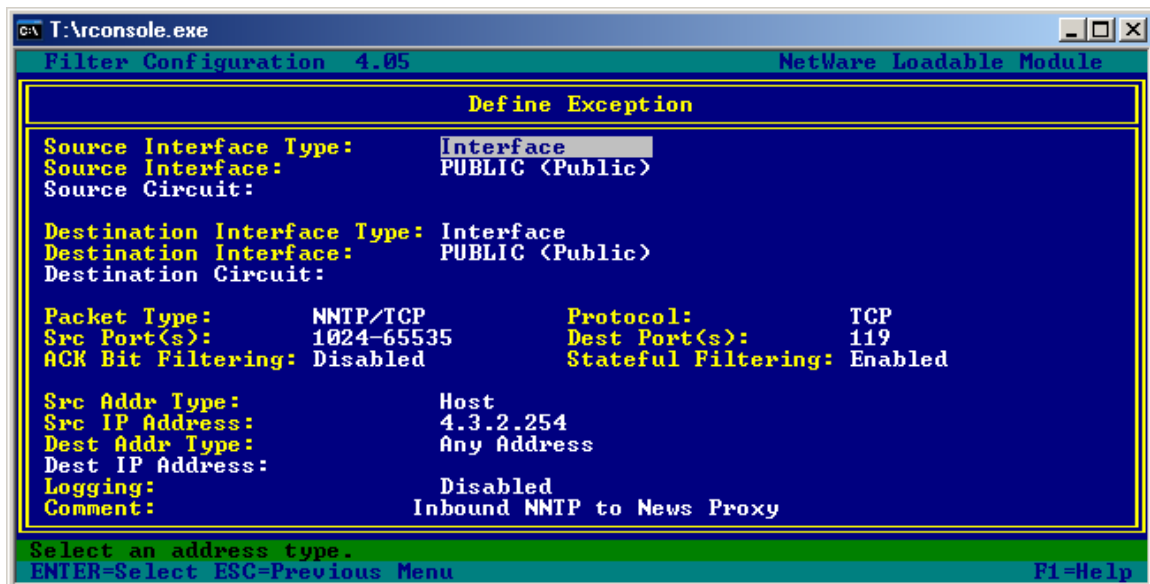


Figure 6-18 - Filter Exception for NNTP to News Proxy

The filter exception shown in Figure 6-18 uses a source interface and destination interface of the BorderManager public interface, any source IP address, and a destination IP address of the Mail Proxy public IP address

- Source Interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 119
- Destination IP Address: <BorderManager public IP address assigned for Mail Proxy>

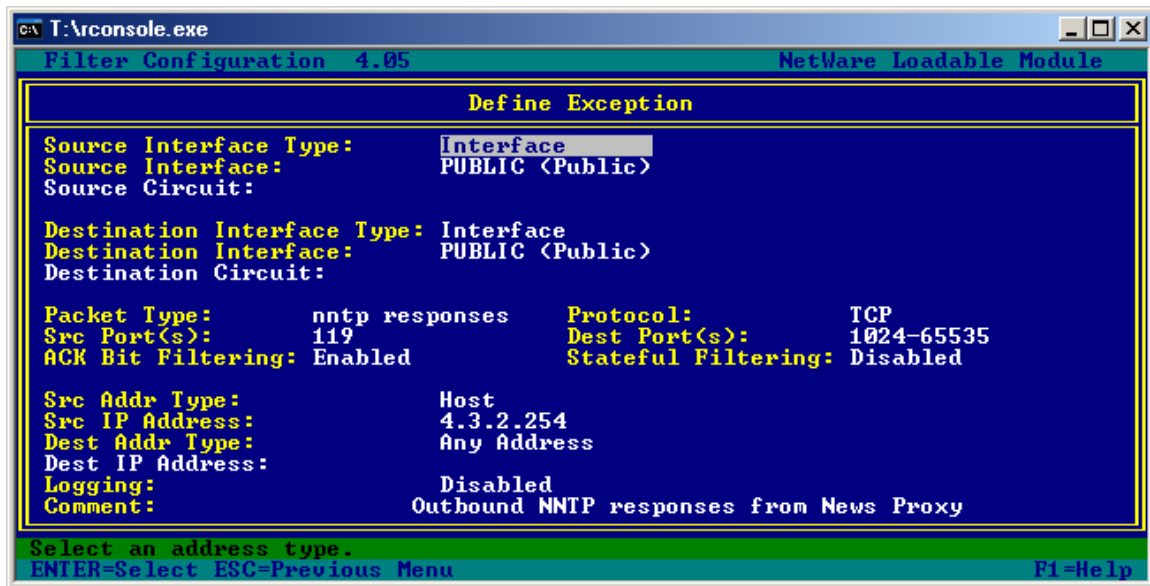


Figure 6-19 - Filter Exception for NNTP Responses from News Proxy

The filter exception shown in Figure 6-19 uses a source interface and destination interface of the BorderManager public interface, a source IP address of the Mail Proxy public IP address, and any destination IP address.

- Source Interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source Port: 119
- Destination Ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <BorderManager public IP address assigned for Mail Proxy>

This exception would not normally be needed for BorderManager 3.x servers prior to version 3.7, if the default filter exceptions are in place, and the proxy is listening on the primary public IP address. The default filter exceptions in those cases allow all outbound IP traffic from the public IP address (though not from a secondary IP address).

## RCONJ to Generic Proxy (on Secondary IP Address)

The point of this example is to allow inbound RCONJ (Java Remote Console) traffic to an internal NetWare server using Generic TCP Proxy.

The default TCP destination port of RCONJ is 2034, which is called out in the command line when loading RCONAG6.

---

**Note** If ZENWorks for Servers has been installed, you might see that RCONJ can be launched in secure mode, using the LOAD RCONAGP ENCRYPT command, which uses port 2037 by default.

---

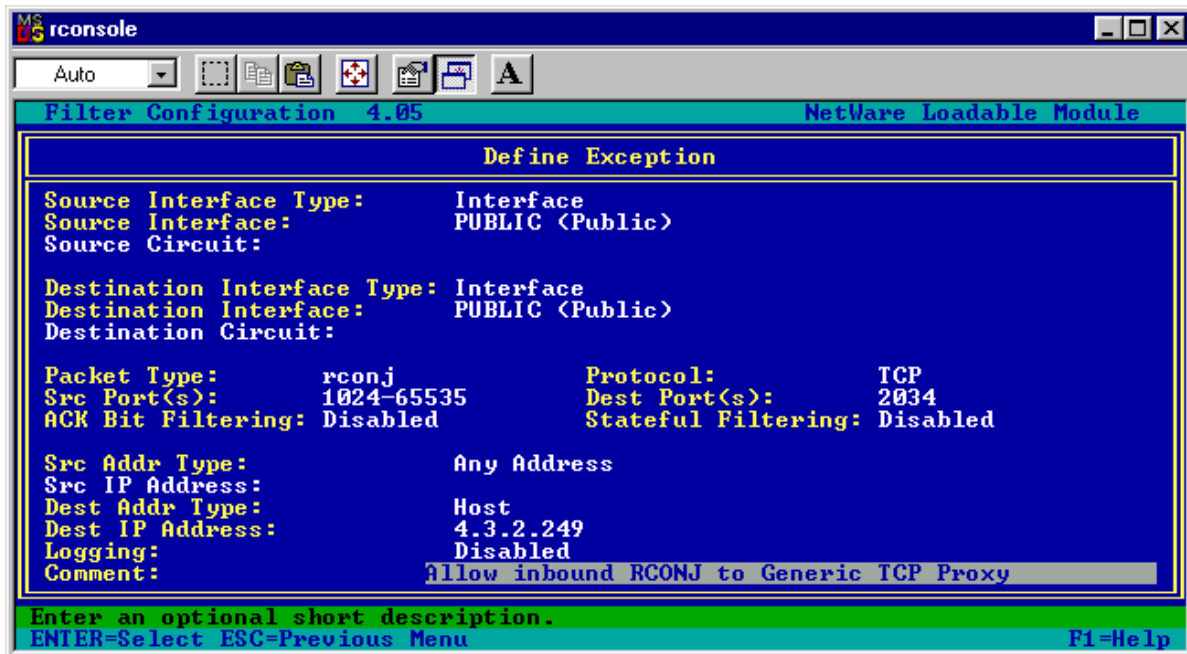


Figure 6-20 - Filter Exception for Inbound RCONJ to Generic TCP Proxy on Secondary Public IP Address

The filter exception shown in Figure 6-20 allows inbound RCONJ traffic to a generic proxy set up for RCONAG6 standard port numbers on the specified destination IP address.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 2034

- Destination IP Address: <your reverse HTTP proxy public IP address>

The following exception allows the outbound RCONJ return traffic.

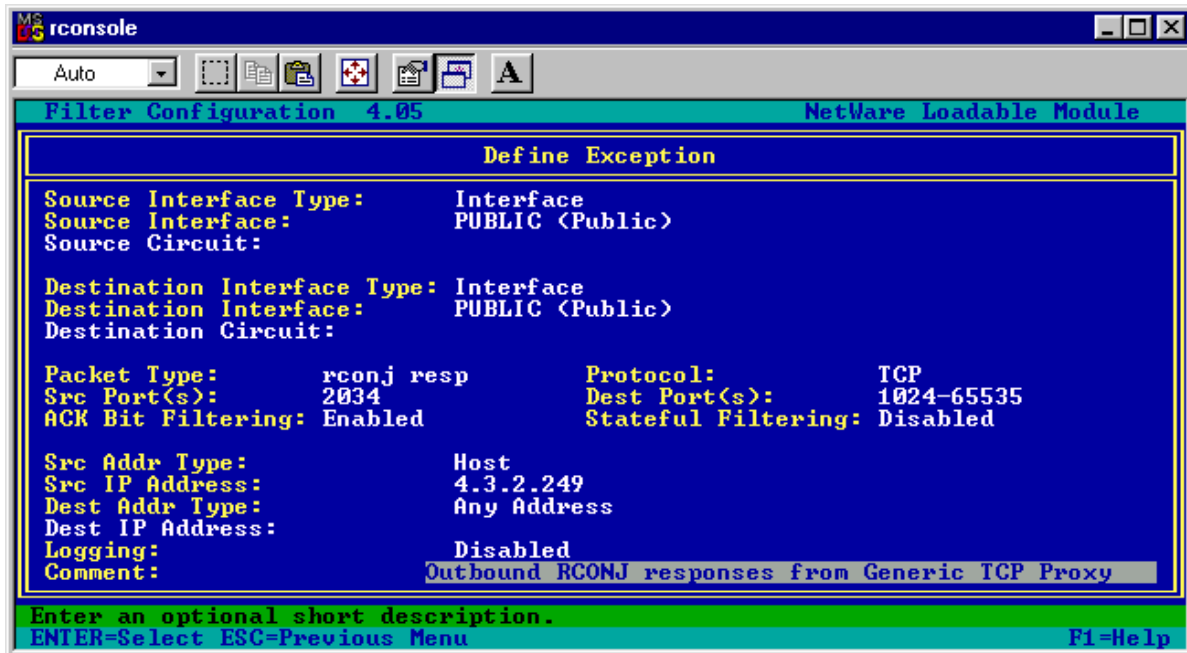


Figure 6-21 - Filter Exception for Outbound Responses from RCONJ on Generic TCP Proxy

The filter exception show in Figure 6-21 allows a Generic TCP Proxy for RCONAG6 on the specified source public IP address to respond to inbound RCONJ requests.

Note that the ACK bit has been set.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source port: 2034
- Destination ports: 1024-65535
- ACK Bit Filtering: enabled
- Source IP Address: <your Generic TCP proxy public IP address>

This exception could be modified to allow RCONJ connections to a BorderManager 3.7 server by changing the source/destination IP addresses to the primary public IP address of the BorderManager server.

## BorderManager 3.7 VPN

BorderManager 3.7 has the same VPN requirements as previous versions, and sets up almost the same default VPN filter exceptions. Unfortunately, the default VPN filter exceptions assume that certain other filter exceptions are present, and those other exceptions are not present by default on a BorderManager 3.7 server. Specifically, there is no exception allowing outbound VPN traffic for the responses to inbound connections, nor are there exceptions which allow Site-to-Site VPN connections outbound, nor are there exceptions which allow Site-to-Site connection responses inbound. Finally, there are no exceptions allowing inbound connections from a remote VPN client behind a NAT connection.

Therefore, neither Client-to-Site nor Site-to-Site VPN will work on a standard BorderManager 3.7 server without additional filter exceptions being added. The following examples show the additional exceptions required for Client-to-Site VPN connections.

### BorderManager 3.7 Default VPN Filter Exceptions

First, three default VPN filter exceptions are assumed to be present. These exceptions can be created using VPNCFG.NLM if they are not already present. These exceptions are:

- VPN-authGW – Allows Client-to-Site connections to be initiated. Allows TCP destination port 353 to the public IP address.
- VPN-KeepAlive – Allows Client-to-Site keepalive traffic. Allows UDP port 353 to the public IP address.
- VPN-SKIP – Allows key exchange, when not using VPN over NAT. Allows the SKIP protocol to the public IP address.

In the past, all IP was allowed outbound from the public IP address, and TCP and UDP high ports were allowed in to the public IP address. These exceptions allowed the response traffic to the above three default exceptions, and allowed outbound Site-to-Site VPN connections. Since these exceptions are not present in BorderManager 3.7 servers by default, you must add exceptions to take their place.

## Client-to-Site Filter Exceptions

### Client-to-Site VPN for Clients Not Behind NAT

The following exceptions are necessary for BorderManager 3.7 servers, if the default 3.7 filter exceptions are in place. Normally, none of these exceptions is required for prior versions of BorderManager with their default exceptions in place.

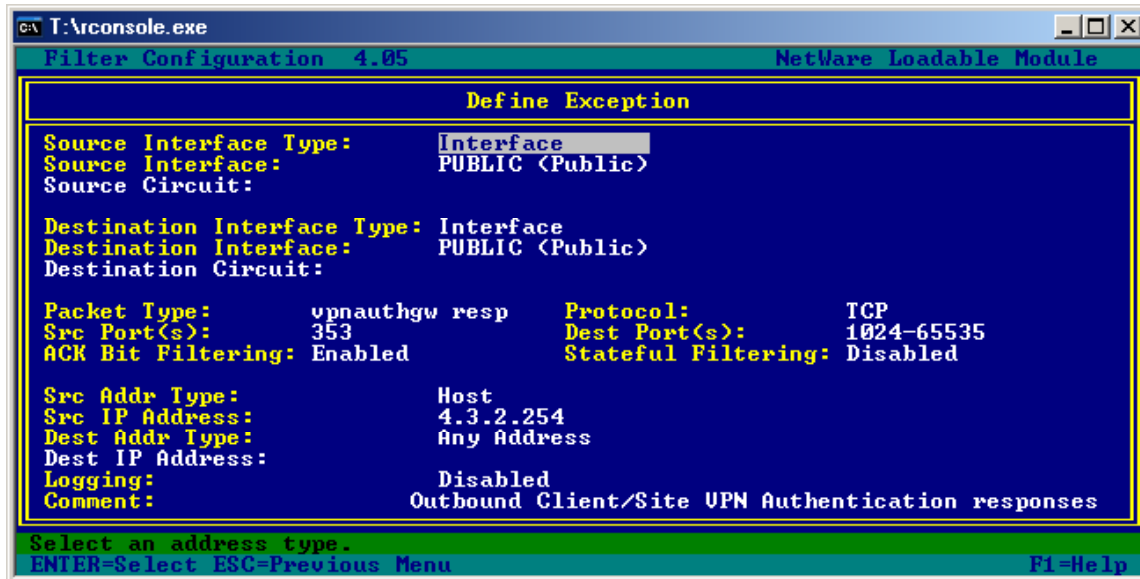


Figure 6-22 - Filter Exception for Client-to-Site VPN Authentication Responses

The filter exception shown in Figure 6-22 allows the VPN server to send authentication replies to a VPN client.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source port: 353
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your BorderManager public IP address>

## Client-to-Site VPN for Clients Behind NAT

---

**Note** This example applies to BorderManager 3.6 and 3.7 servers only, since only 3.6 and later allows the remote VPN client to be behind a NAT connection to the Internet. The BorderManager 3.6 default filter exceptions will allow Client-to-Site VPN over NAT connection. Therefore, these exceptions will only be required in a customized BorderManager 3.6 configuration, or a fresh BorderManager 3.7 installation.

---

This example is for VPN clients on the Internet coming into a BorderManager server. It is not for VPN clients on your LAN going out through NAT to another BorderManager server. That example is shown in the chapter giving outbound filter exceptions.

BorderManager versions prior to 3.7 have default filter exceptions that allow all inbound high ports for both TCP and UDP to the public IP address. (The Dynamic/TCP and Dynamic/UDP exceptions are responsible). Because of these exceptions, a Client-to-Site VPN where the client is behind a NAT connection to the Internet is able to make a VPN connection on a BorderManager 3.6 server. (Client-to-Site VPN over NAT is not possible with versions of BorderManager prior to 3.6). Client-to-Site VPN over NAT uses UDP destination port 2010, in addition to the ports specified in the default VPN filter exceptions.

For BorderManager 3.7, you may need to create the following exceptions to allow Client-to-Site VPN over NAT.

---

**Note** These exceptions will also allow slave servers to communicate in a Site-to-Site VPN.

---



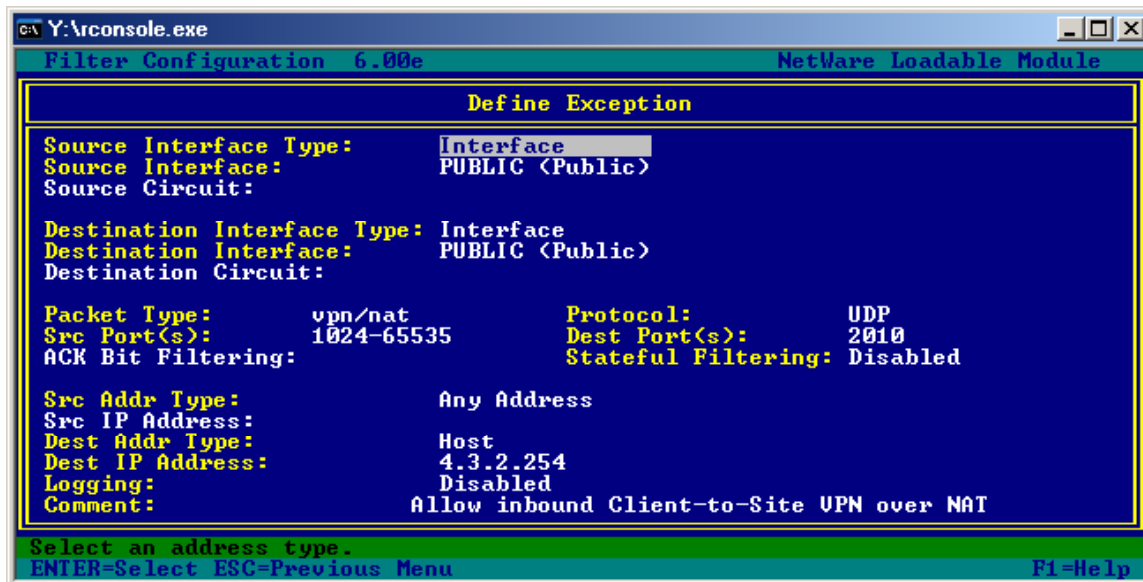


Figure 6-23 - Filter Exception for Inbound Client-to-Site VPN over NAT to BorderManager Server

The filter exception shown in Figure 6-23 allows a VPN client behind a NAT connection on the Internet to make a VPN connection to a BorderManager 3.6 or 3.7 server.

- Source interface: Public
- Destination Interface: Public
- Protocol: UDP
- Source ports: 1024-65535
- Destination port: 2010
- Destination IP Address: <your BorderManager public IP address>

The following exception is also required in order to allow return traffic to the remote VPN client.

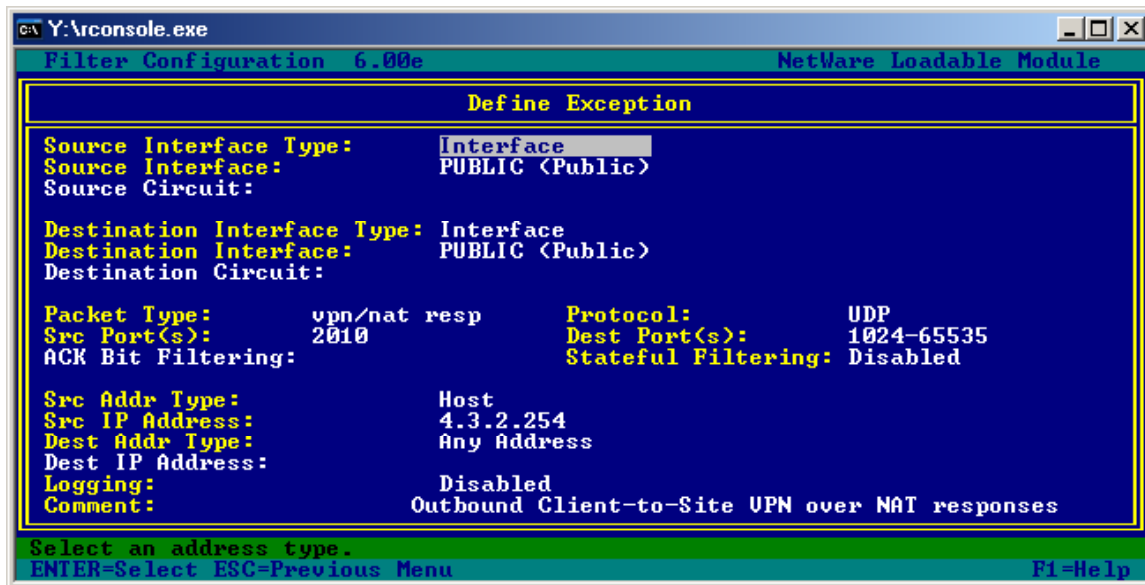


Figure 6-24 - Filter Exception for Outbound Client-to-Site VPN over NAT Responses from BorderManager Server

The filter exception shown in Figure 6-24 allows the VPN server to reply to a VPN client behind a NAT connection to the Internet.

- Source interface: Public
- Destination Interface: Public
- Protocol: UDP
- Source port: 2010
- Destination ports: 1024-65535
- Source IP Address: <your BorderManager public IP address>

## Site-to-Site VPN Exceptions

The default exceptions created in a fresh (not upgraded) BorderManager 3.7 installation do not include some critical exceptions necessary for the VPN to function.

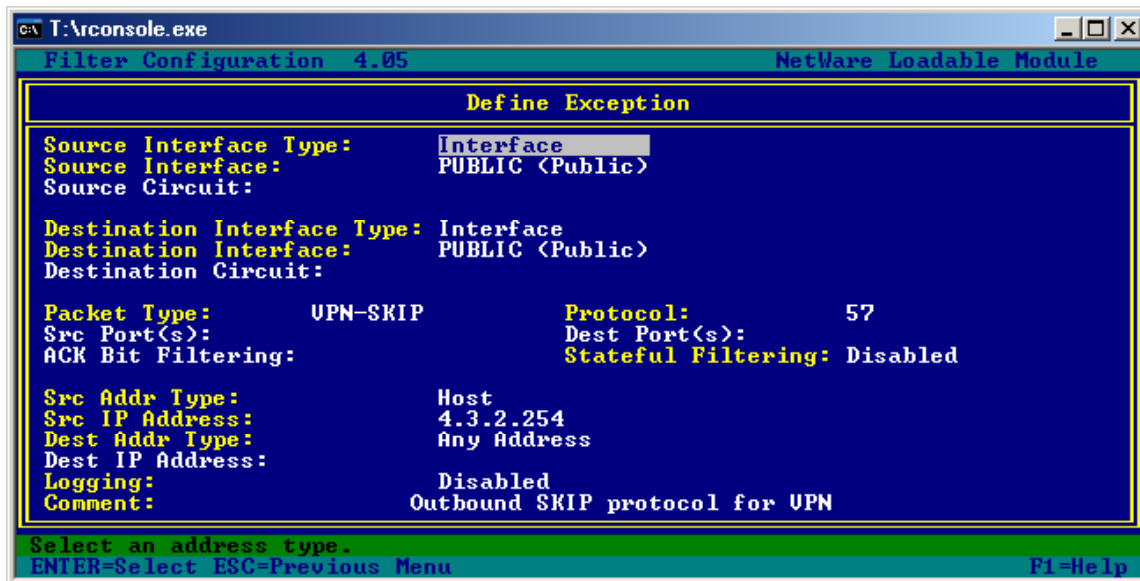


Figure 6-25 - Filter Exception for Outbound SKIP Protocol

The filter exception shown in Figure 6-25 allows the VPN server to send SKIP packets to a master or slave VPN server.

- Source interface: Public
- Destination Interface: Public
- Protocol: 57
- Source IP Address: <your BorderManager public IP address>

One symptom of a slave server not able to send SKIP packets out to the master server is the VPTUNNEL interface not showing up in INETCFG.

---

**Note** The default exceptions created by a BorderManager 3.7 installation for some reason do not include an exception created by default with BRDCFG or VPNCFG in prior revisions of BorderManager. This exception is intended to allow VPN Site-to-Site communications. Even if it was added by BorderManager 3.7's VPNCFG, the response traffic would still have to be manually added.

---

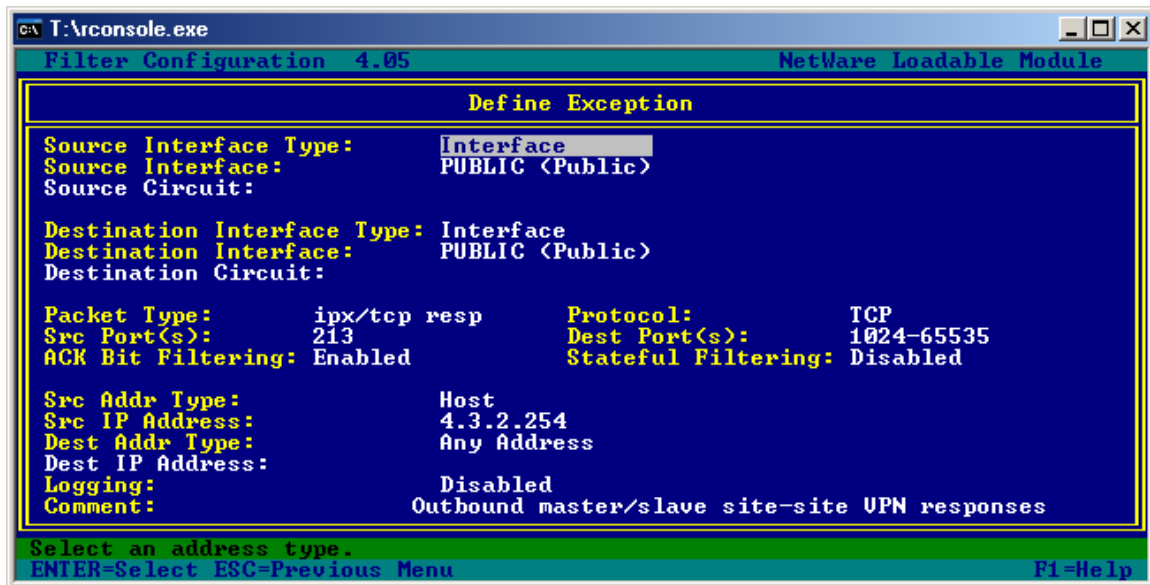


Figure 6-26 - Filter Exception for Inbound Site-to-Site VPN Communications

The filter exception shown in Figure 6-26 allows the master or slave VPN server to send requests to another Site-to-Site VPN server.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 213
- Destination IP Address: <your BorderManager public IP address>

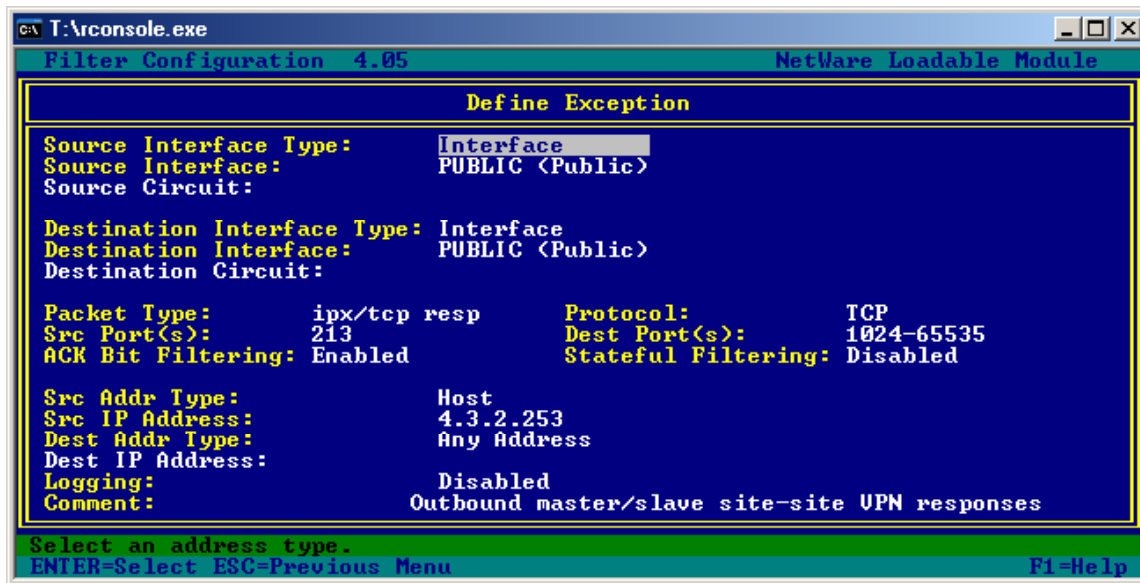


Figure 6-27 - Filter Exception for Site-to-Site VPN Responses

The filter exception shown in Figure 6-27 allows the master or slave VPN server to reply to another Site-to-Site VPN server.

- Source interface: Public
- Destination Interface: Public
- Protocol: TCP
- Source port: 213
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your BorderManager public IP address>

---

**Note** These next two exceptions may be required for slave servers to communicate to each other in a Site-to-Site VPN. If you have configured the UDP 2010 filter exceptions for Client-to-Site VPN clients behind a NAT connection, you do not need to add these exceptions.

---

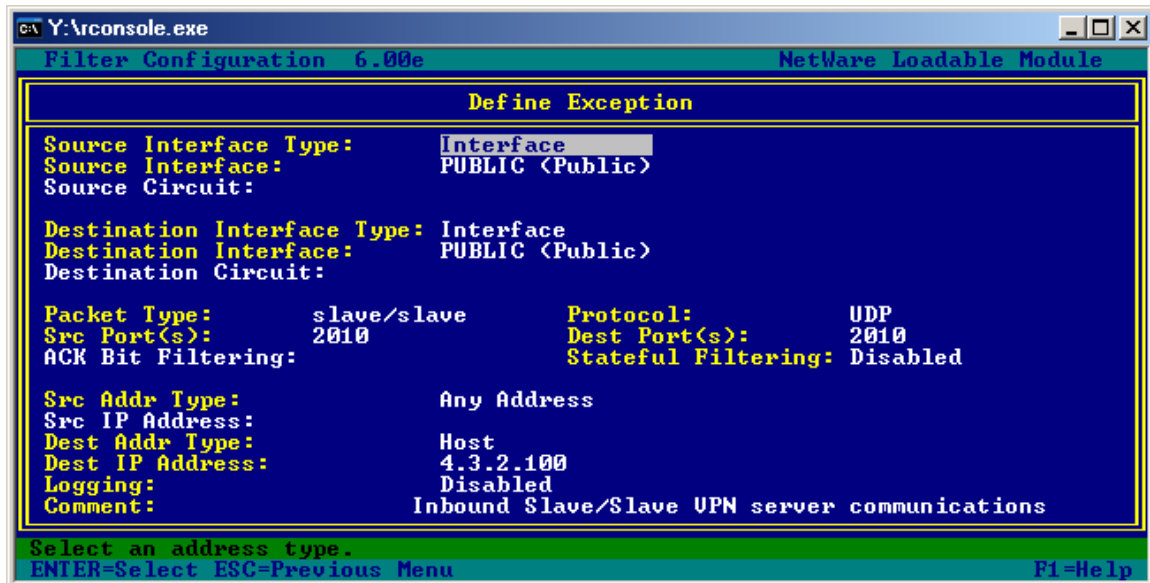


Figure 6-28 - Filter Exception for Slave/Slave VPN Server Site-to-Site Communications Inbound

The filter exception shown in Figure 6-28 allows one slave VPN server to communicate to another Site-to-Site VPN slave server.

- Source interface: Public
- Destination Interface: Public
- Protocol: UDP
- Source port: 2010
- Destination ports: 2010
- Source IP Address: <your BorderManager public IP address>

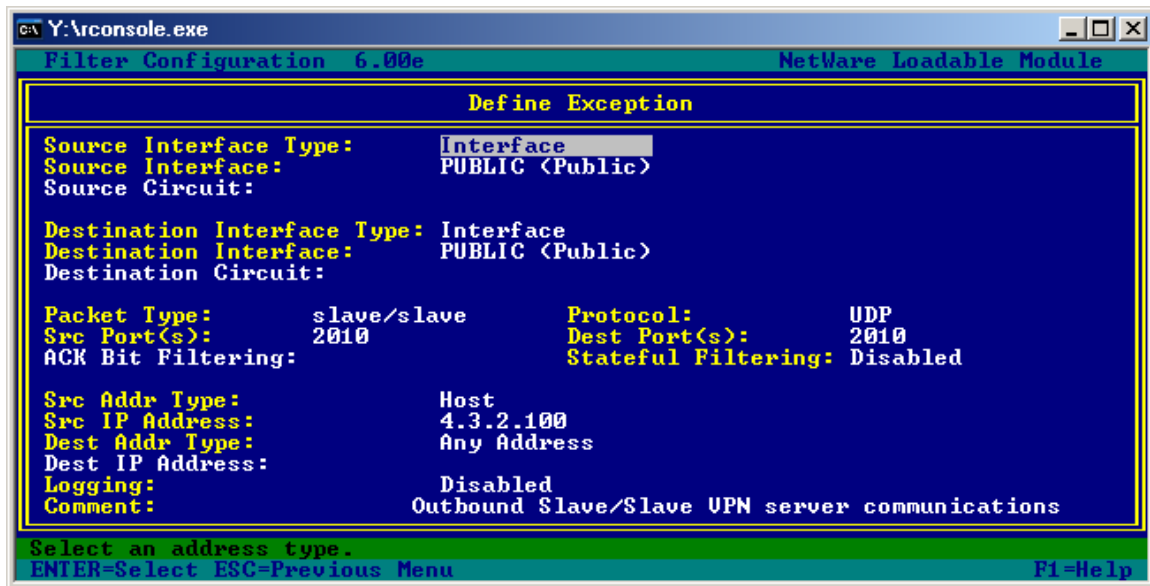


Figure 6-29 - Filter Exception for Slave/Slave VPN Server Site-to-Site Communications Outbound

The filter exception shown in Figure 6-29 allows one slave VPN server to communicate to another Site-to-Site VPN slave server.

- Source interface: Public
- Destination Interface: Public
- Protocol: UDP
- Source port: 2010
- Destination ports: 2010
- Source IP Address: <your BorderManager public IP address>

# Chapter 7 - Example Inbound Filter Exceptions Using Static NAT

---

Static NAT is always done using secondary IP addresses, and the BorderManager default filters should block all traffic to a secondary IP address on the public side. It is therefore necessary to set up two filter exceptions for each static NAT address pair, unless stateful filters are used. Since stateful filters have additional overhead, and you normally aren't worried about hacking into the static NAT traffic from inside your LAN, I recommend using non-stateful filter exceptions with static NAT.

In addition, there is a (rare) security exploit that can be used to bring additional ports in through a stateful filter exception once a session has been established. It can be safer to configure non-stateful exceptions for inbound traffic from the Internet.

---

**Note** You can generally use BorderManager 3.x generic TCP and UDP proxies as an alternative to Static NAT. In this case, the only difference in the filter exceptions would be to change the source/destination IP addresses from the internal IP address of the host to the public IP address of the proxy. You would also need to specify access rules, and the BorderManager PROXY.NLM would have to be running. Some types of traffic (POP3, NNTP, SMTP, etc.) cannot be done with BorderManager *Generic* proxies if a dedicated proxy is provided. (E.g. Mail Proxy must be used for SMTP and POP3).

---



## Citrix WinFrame

Citrix WinFrame hosts can be accessed by two different client types, each requiring their own particular destination port number. The examples shown will allow inbound connections from both a stand-alone Citrix ICA (Independent Computing Architecture) client and a browser-based snap-in client.

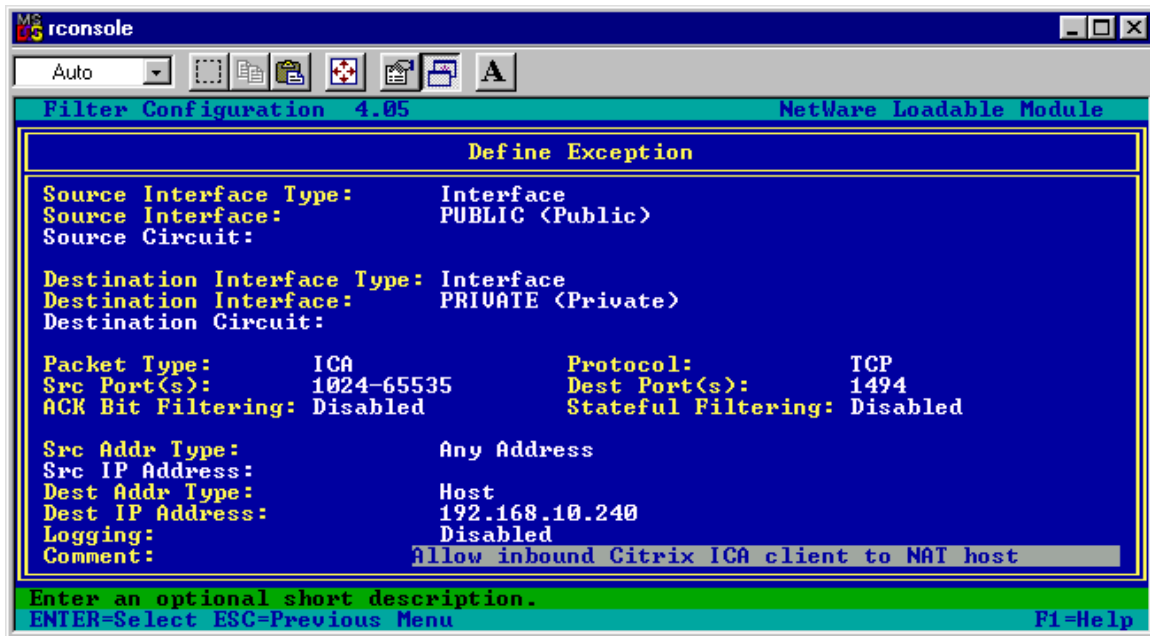


Figure 7-1 - Filter Exception for Inbound Citrix ICA Client

The filter exception shown in Figure 7-1 allows inbound traffic from the Citrix ICA client to an internal Citrix WinFrame host through static NAT.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 1494
- Destination IP Address: <your Citrix server internal address>

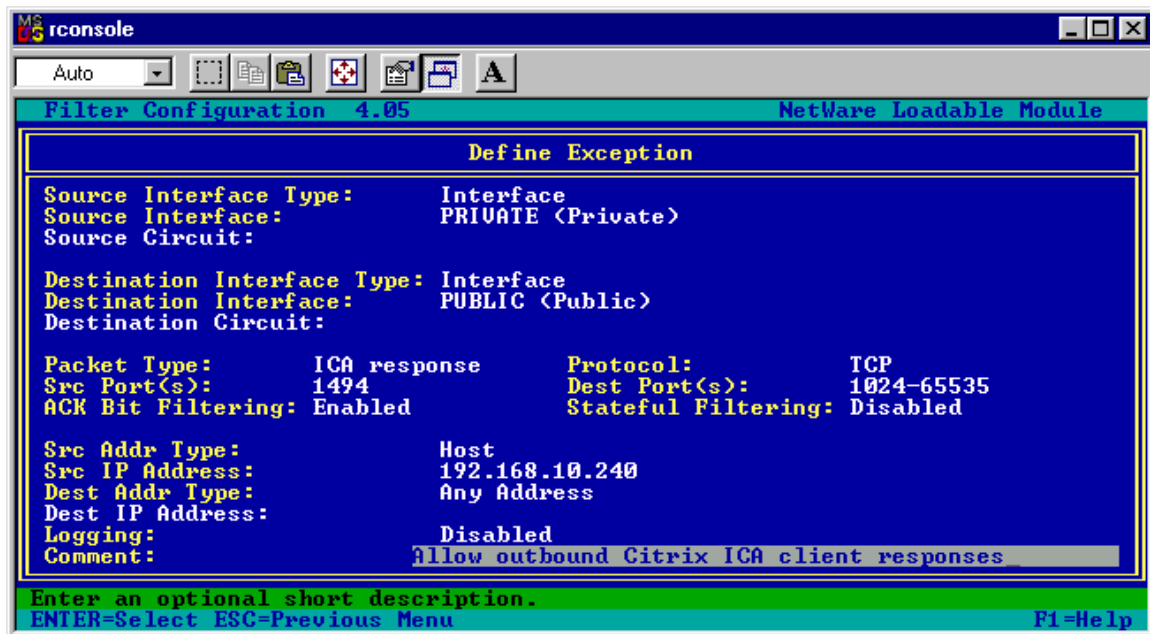


Figure 7-2 - Filter Exception for Outbound Citrix ICA Client Responses

The filter exception shown in Figure 7-2 allows outbound (return) responses from the internal Citrix WinFrame host to an external Citrix ICA client.

Note that the ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 1494
- Destination ports: 1024-65535
- Source IP Address: <your Citrix server internal address>

---

**Note** Citrix needs the altaddr /set x.x.x.x command to be used, plus a correct default route specified, in order to be accessible over static NAT. See your Citrix documentation on the use of the altaddr command.

---

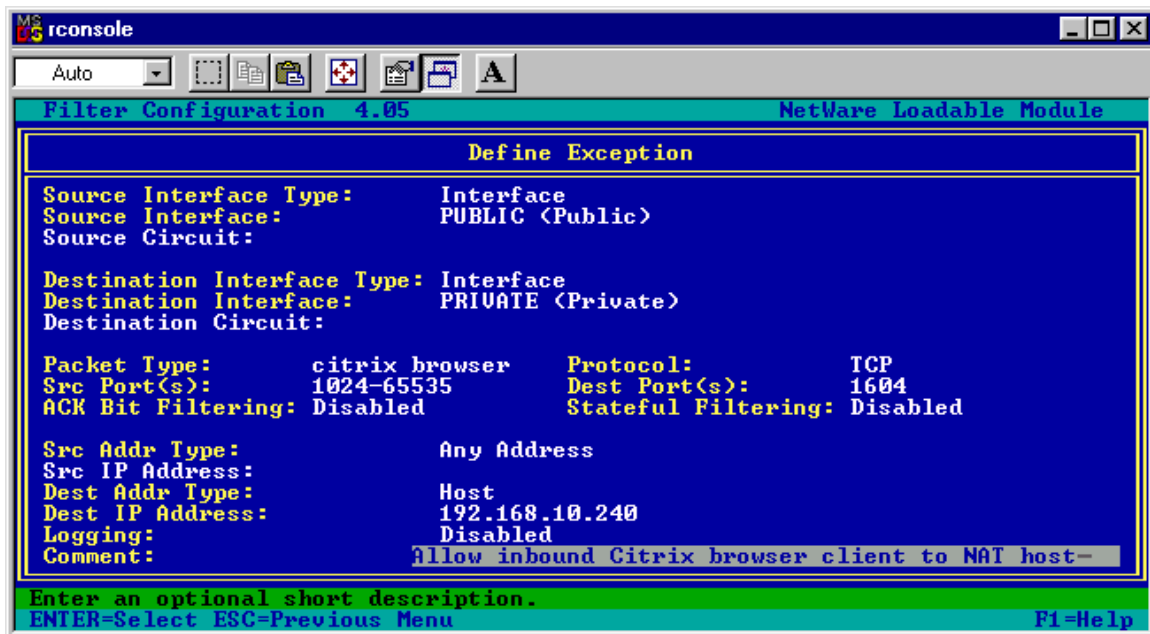


Figure 7-3 - Filter Exception for Inbound Citrix Browser-based Client

The filter exception shown in Figure 7-3 allows inbound traffic from the Citrix browser-based client to an internal Citrix WinFrame / MetaFrame host through static NAT.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 1604
- Destination IP Address: <your Citrix server internal address>

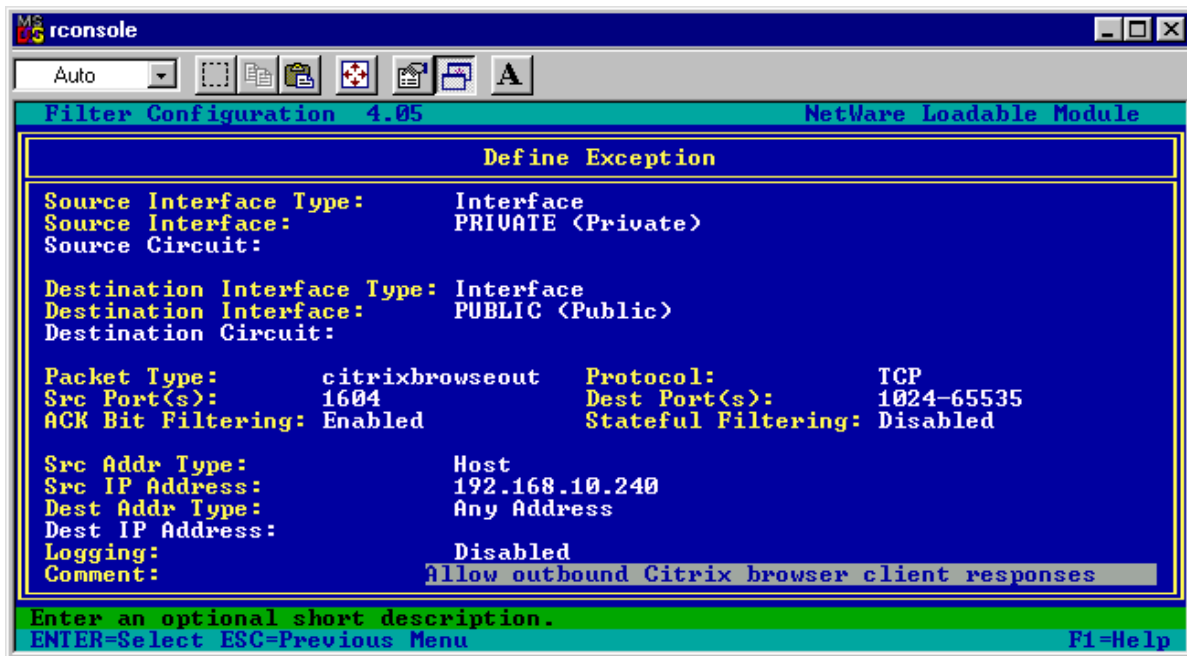


Figure 7-4 - Filter Exception for Outbound Citrix Browser-based Client Responses

The filter exception shown in Figure 7-4 allows outbound (return) traffic from an internal Citrix WinFrame / MetaFrame host to an external Citrix browser-based client.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 1604
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your Citrix server internal address>

# FTP

FTP through a Static NAT connection should be done with an inbound non-stateful exception (for ports 20 and 21) and one or two outbound non-stateful exceptions to allow the response packets.

The testing was done using command prompt FTP in Windows 2000, and CuteFTP 3.0 to a NetWare Novonyx FTP server.

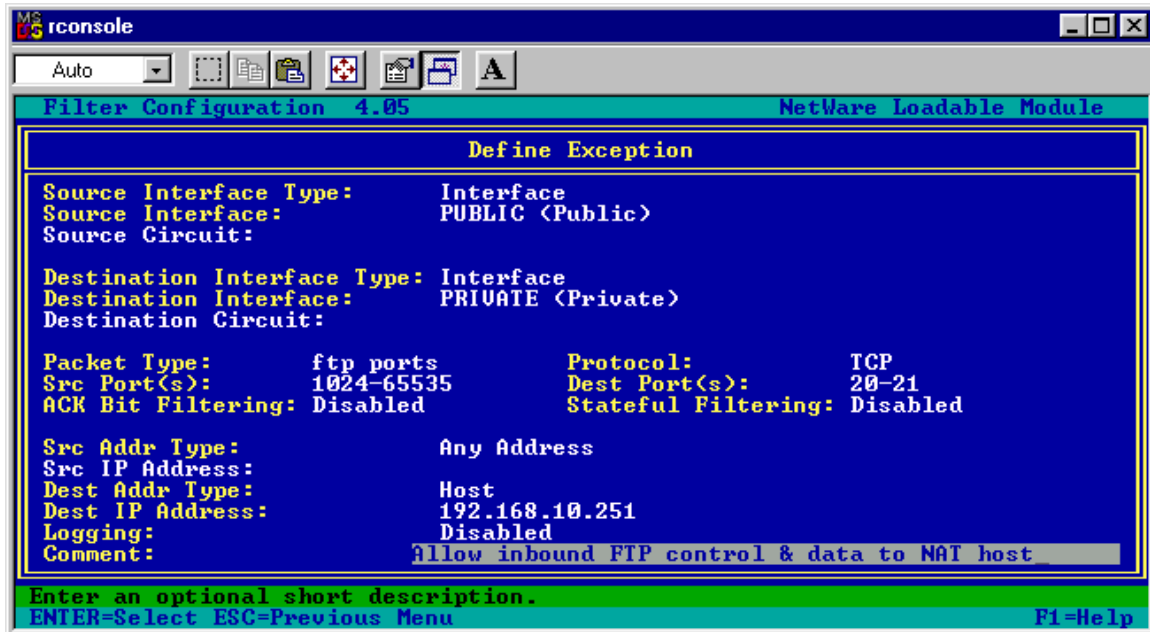


Figure 7-5 - Filter Exception for Inbound FTP Control and Data Ports

The filter exception shown in Figure 7-5 is all that was needed for CuteFTP, and command line FTP to make inbound connections and transfer data.

This custom filter exception uses a source interface of the BorderManager public interface and a destination interface of the any interface, any source IP address, and a destination IP address of the internal static NAT FTP server.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Ports: 20-21
- Destination IP Address: <your FTP server internal address>

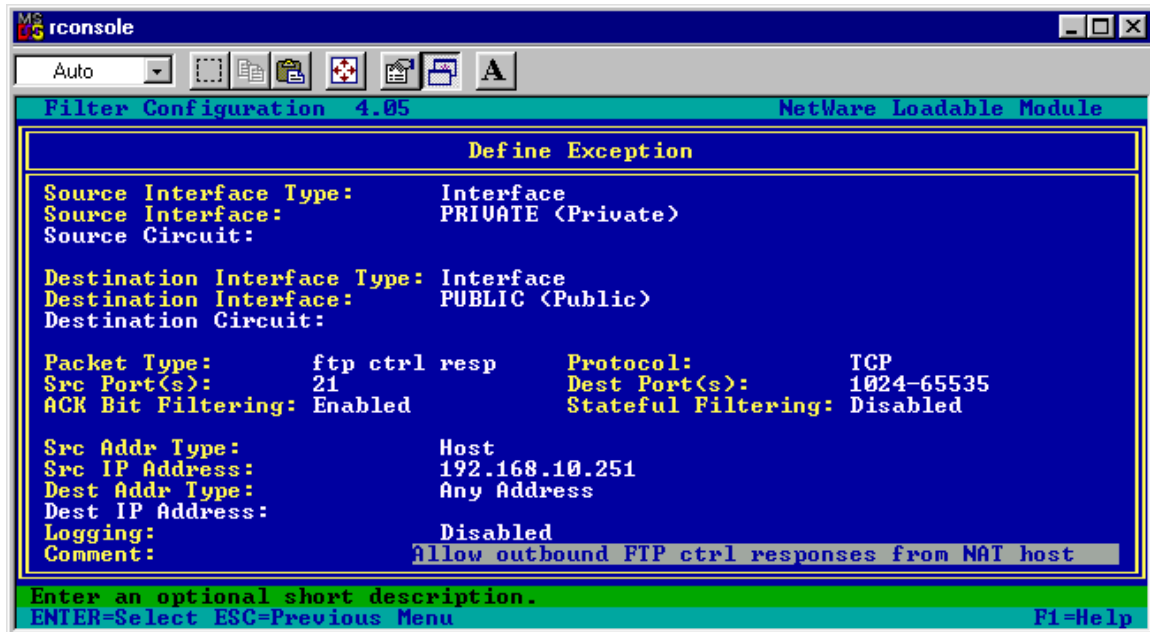


Figure 7-6 - Filter Exception for Outbound FTP Control Port Responses

The filter exception shown in Figure 7-6 allows the FTP control port responses back from an internal FTP server via a Static NAT connection.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 21
- Destination port: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your FTP server internal address>

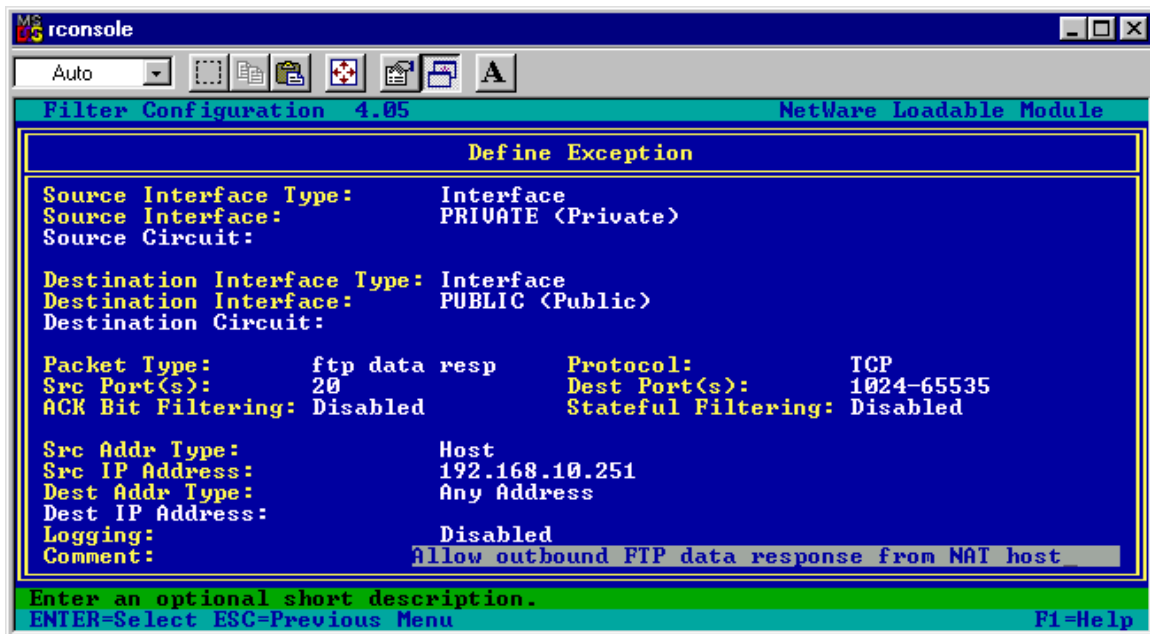


Figure 7-7 - Filter Exception for Outbound FTP Data Port Responses

The filter exception shown in Figure 7-7 allows outbound FTP data responses from an internal FTP server.

Note that ACK bit filtering has **NOT** been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 20
- Destination port: 1024-65535
- Source IP Address: <your FTP server internal address>

An alternative to having two filter exceptions for outbound ports 20 and 21 would be to have a single exception for source ports 20-21, but not enable ACK bit filtering on it. If you enabled ACK bit filtering on outbound source port 20, your FTP data connections will fail.

## GroupWise Remote Client

Should you desire to make a GroupWise client/server connection using the GroupWise Remote Client, instead of using WebAccess, or POP3, you can set up Static NAT between a secondary IP address on the BorderManager public interface and an internal GroupWise POA server. Next, allow TCP destination port 1677 in, and the responses back out with the following two filter exceptions.

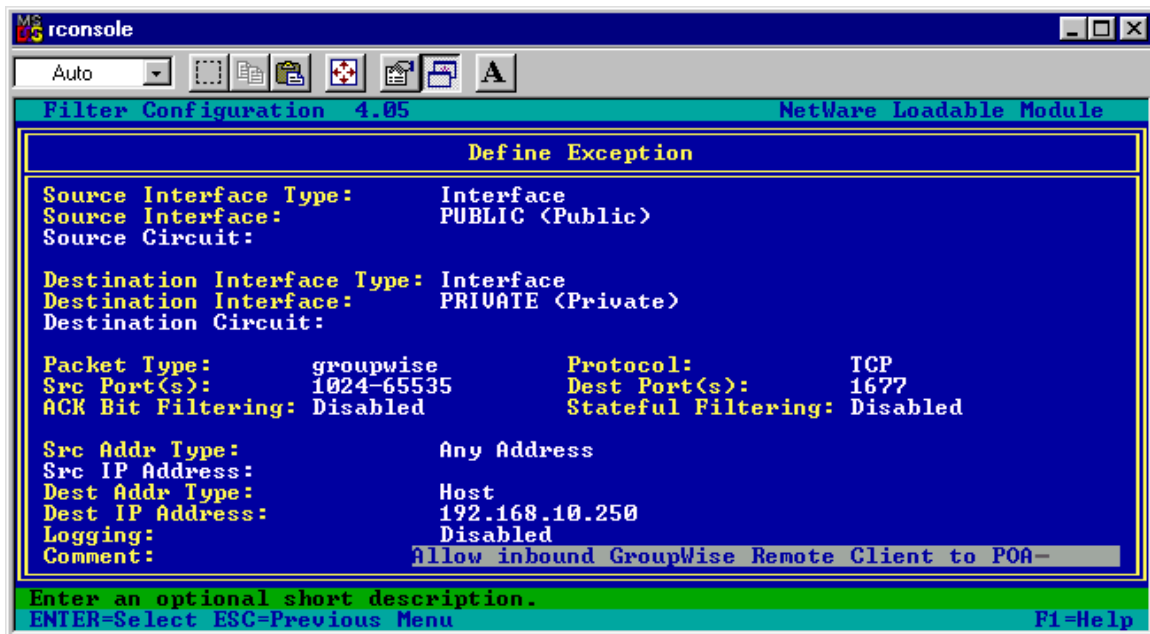


Figure 7-8 - Filter Exception for Inbound GroupWise Remote Client

The filter exception shown in Figure 7-8 allows inbound GroupWise Remote client traffic through static NAT to the host specified at the destination IP address (192.168.10.250).

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 1677
- Destination IP Address: <your GroupWise POA internal address>



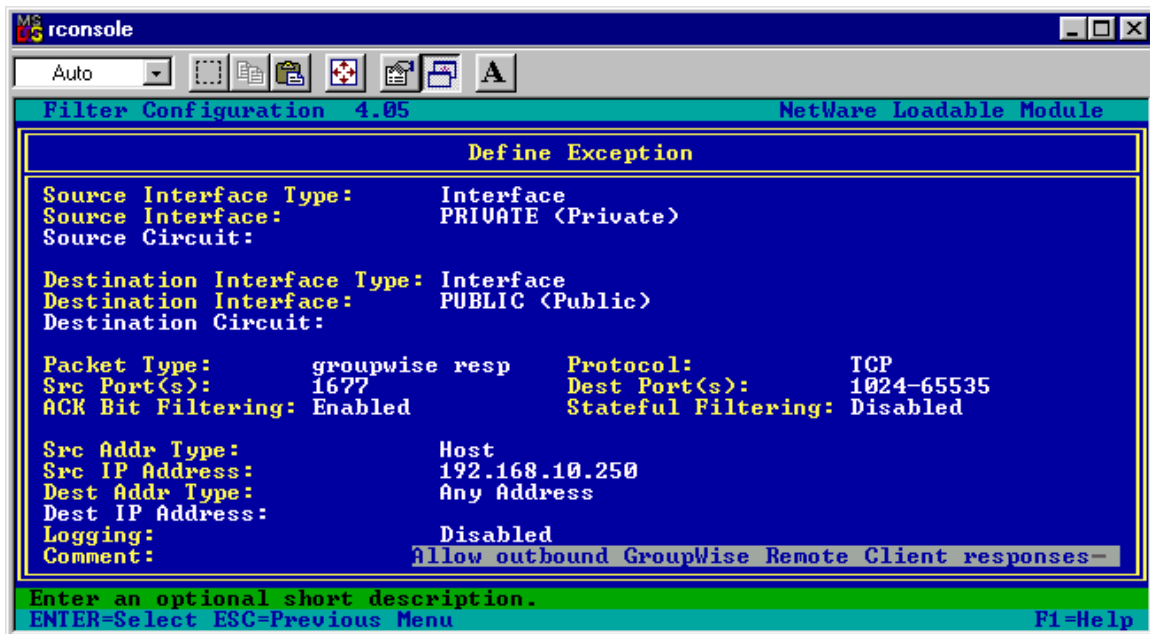


Figure 7-9 - Filter Exception for Outbound GroupWise Remote Client Responses

The filter exception shown in Figure 7-9 allows outbound GroupWise Remote Client responses from an internal host at the specified source IP address to respond to inbound requests.

Note that the ACK bit has been set.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 1677
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your GroupWise POA internal address>

## GroupWise Web Access Spell Check

WebAccess, whether provided via Static NAT or Reverse HTTP Proxy, uses a different port number for the spell check function. Note that if you use reverse HTTP Proxy for WebAccess, you must use a Generic TCP Proxy for the spell check function.

This example is for GroupWise 5.5 Enhancement Pack WebAccess, which uses the Collexion spell check application. Collexion defaults to listening on TCP destination port 9010. As usual for Static NAT, two exceptions are needed, one for inbound traffic and one for outbound responses.

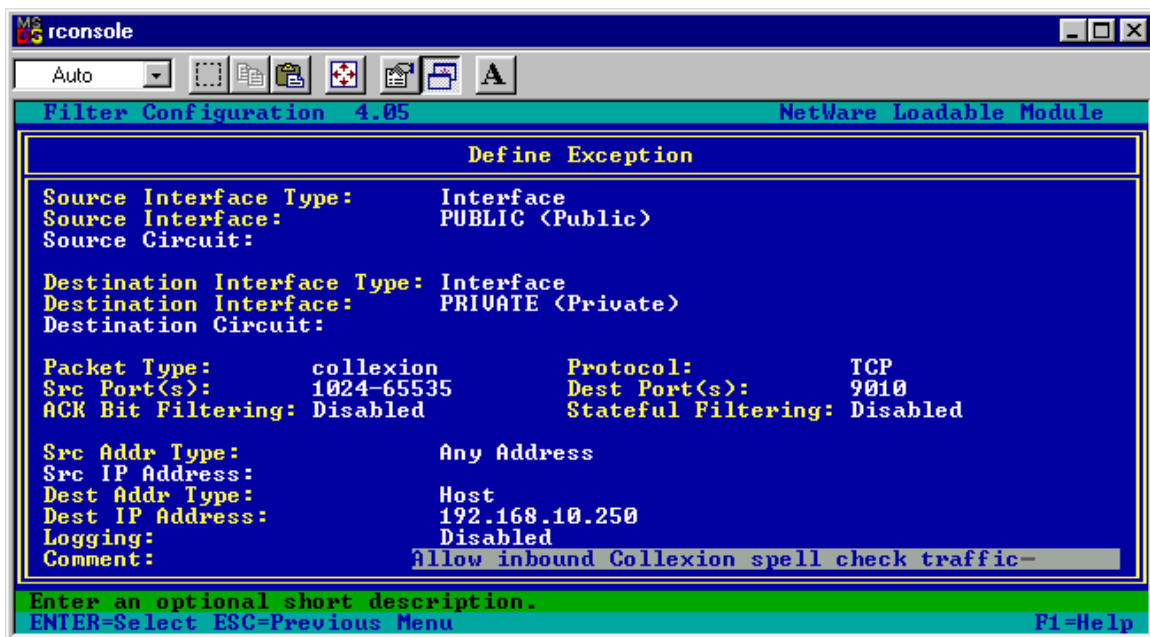


Figure 7-10 - Filter Exception for Inbound Collexion Spell Check Requests

The filter exception shown in Figure 7-10 allows inbound spell check traffic through static NAT on the standard port number used by Collexion to a spell check agent at the specified internal destination IP address.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Ports: 9010
- Destination IP Address: <your Collexion server internal address>

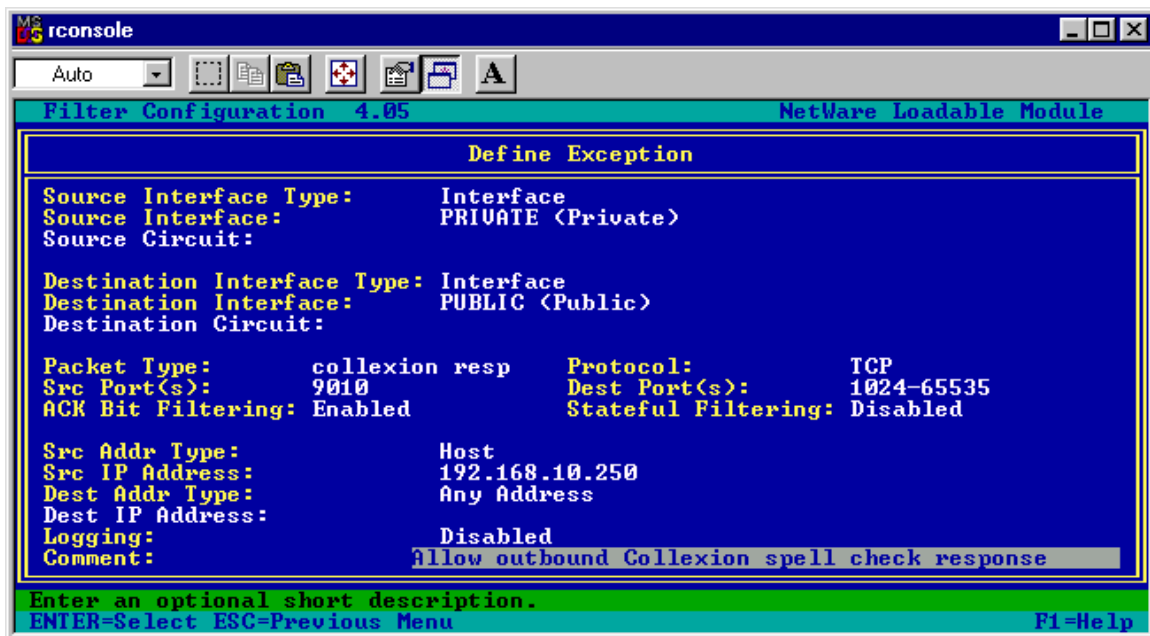


Figure 7-11 - Filter Exception for Outbound Collexion Spell Check Responses

The filter exception shown in Figure 7-11 allows outbound spell check responses from the Collexion spell check application running on an internal mail server at the specified source IP address.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 9010
- Destination port: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your Collexion (WebAccess) server internal address>

# IMAP

IMAP is a mail access protocol. The following pair of filter exceptions allows a user on the Internet to access an internal mail server using IMAP protocol.

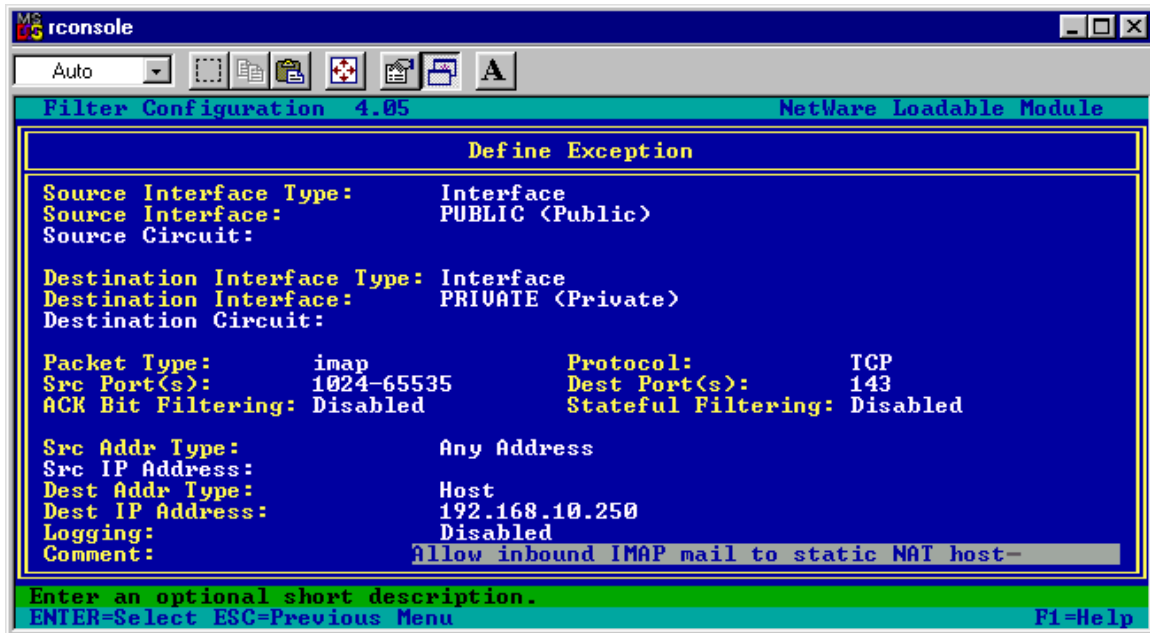


Figure 7-12 - Filter Exception for Inbound IMAP

The filter exception shown in Figure 7-12 allows inbound IMAP traffic through static NAT to an internal mail server at the specified destination IP address.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 143
- Destination IP Address: <your IMAP server internal address>

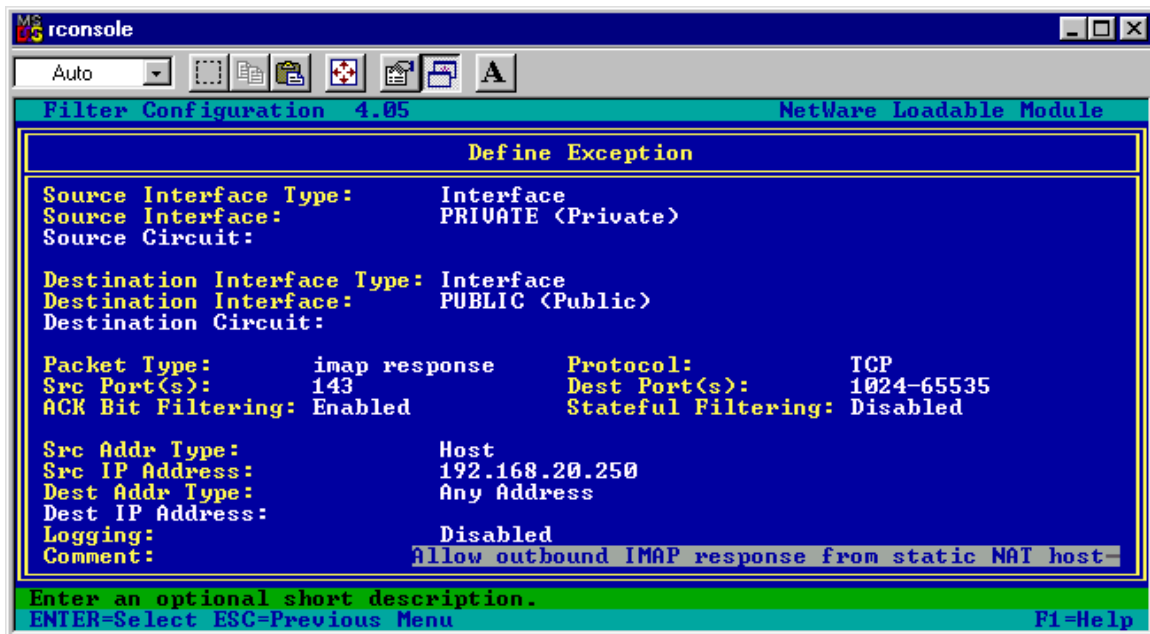


Figure 7-13 - Filter Exception for Outbound IMAP Responses

The filter exception shown in Figure 7-13 allows outbound IMAP responses from a mail server at the specified source IP address.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 143
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your IMAP server internal address>

## Lotus Notes Clients

This filter exception can be used to allow Lotus Notes clients on the Internet to communicate with a Lotus Notes server through a static NAT connection.

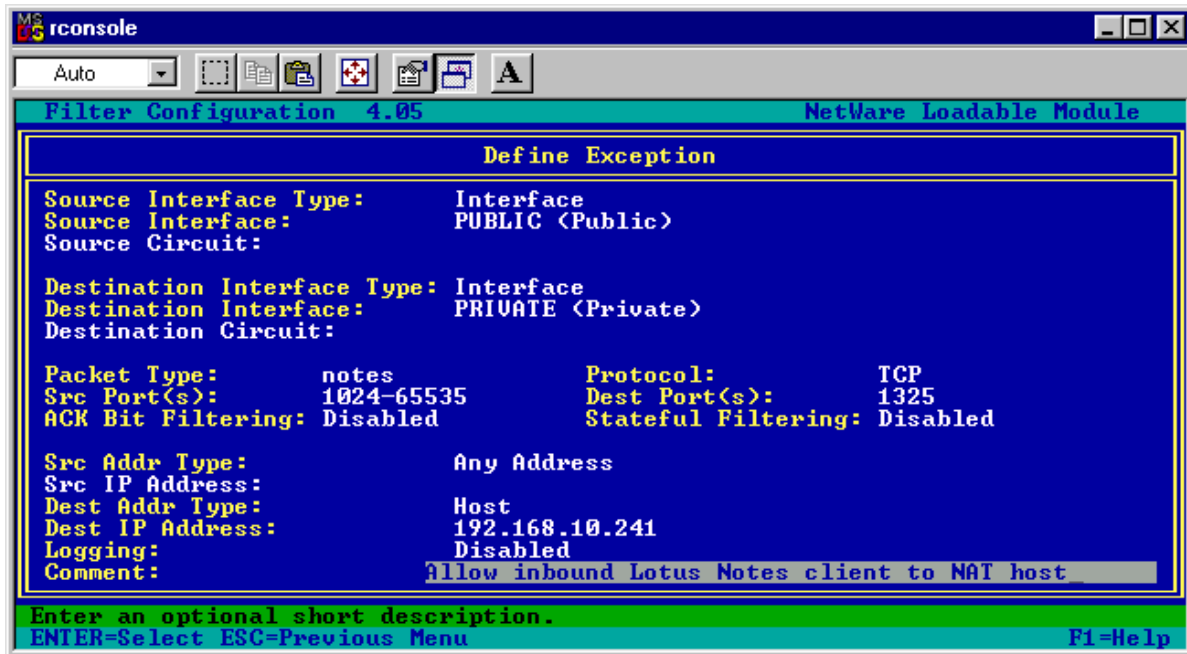


Figure 7-14 - Filter Exception for Inbound Lotus Notes Client

The filter exception in Figure 7-14 allows inbound Lotus Notes client traffic through static NAT to a Notes server at the specified internal IP address.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 1325
- Destination IP Address: <your Notes server internal address>

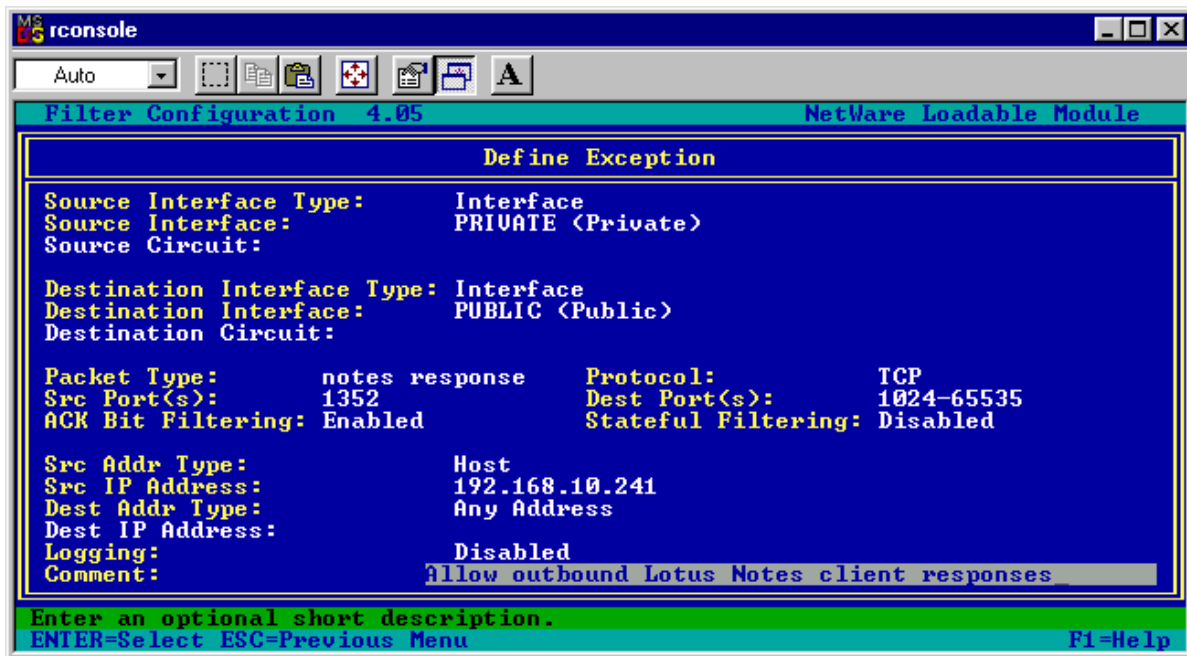


Figure 7-15 - Filter Exception for Outbound Lotus Notes Client Responses

The filter exception shown in Figure 7-15 allows an internal Lotus Notes server at the specified source IP address to respond to inbound Notes Client traffic.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 1352
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your Notes server internal address>

## Microsoft Terminal Server

This pair of exceptions allows you to connect to a Microsoft Windows 2000 Terminal Server via Static NAT.

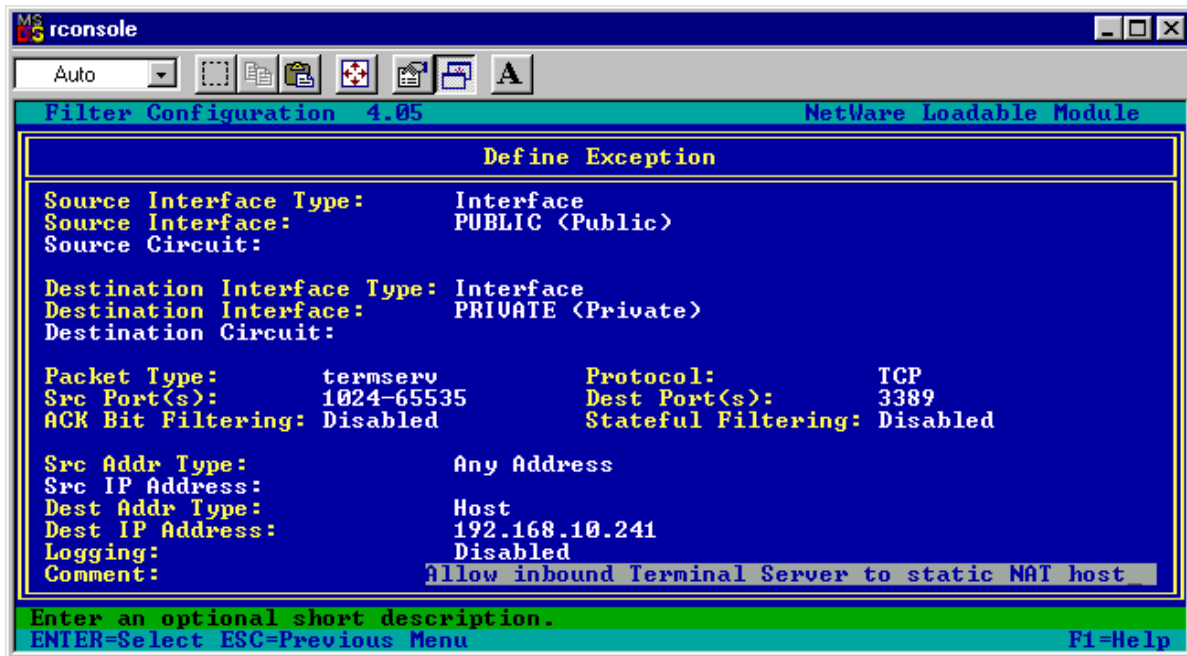


Figure 7-16 - Filter Exception for Inbound Microsoft Terminal Server

The filter exception shown in Figure 7-16 allows inbound Microsoft Terminal Server client requests through Static NAT to an internal Terminal Server at the specified destination IP address.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Ports: 3389
- Destination IP Address: <your Terminal Server internal address>



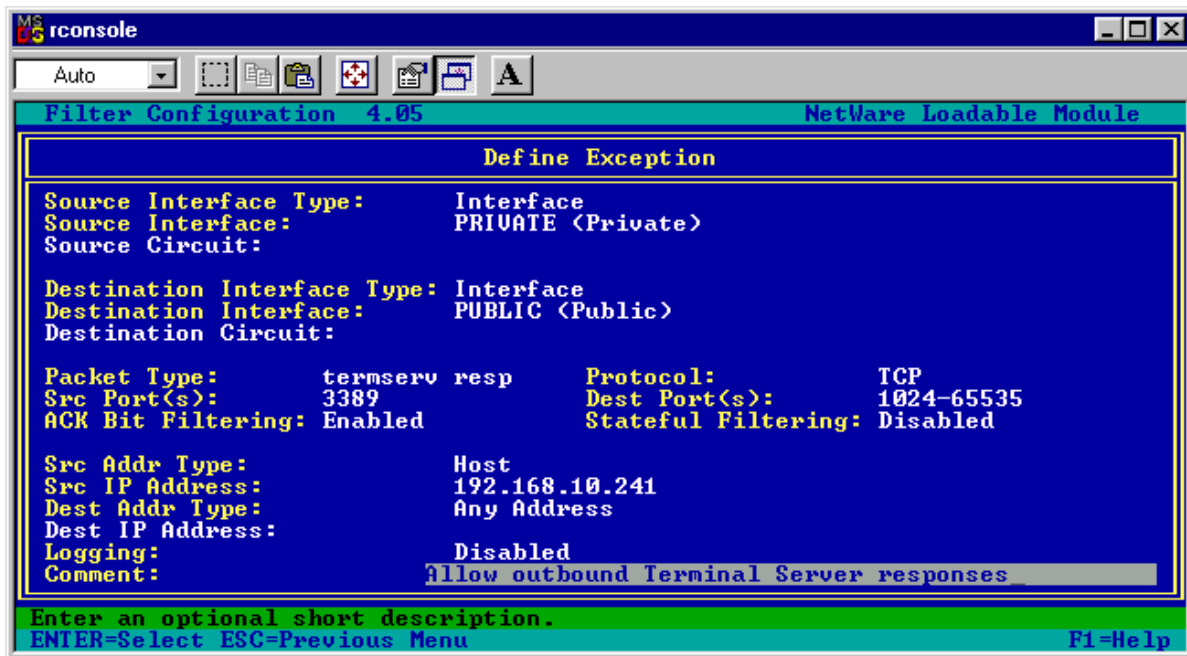


Figure 7-17 - Filter Exception for Outbound Terminal Server Responses

The filter exception shown in Figure 7-17 allows an internal Microsoft Terminal Server at the specified source IP address to respond to inbound client requests.

Note that ACK bit filtering is enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 3389
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your Terminal Server internal address>

## pcANYWHERE

This example covers pcANYWHERE, versions 8.0 through 10, running from a Windows98 PC to a Windows NT 4.0 server running pcANYWHERE version 8.0. (The same exceptions may work with all other versions of pcANYWHERE.)

It appears that pcANYWHERE version 8.0 tries to locate a TCP/IP-based host using UDP port 5632. If it doesn't get an immediate response, it will also try UDP port 22. A UDP response is sent out to UDP port 5632 (or on UDP port 22 if UDP port 22 was used instead of port 5632).

Once a pcANYWHERE host is located, a response is received on UDP port 5632 by the originating host, and a connection is then established using TCP port 5631.

This means that one method of configuring filter exceptions (which would work with BorderManager 2.1 also) is to set up four different filter exceptions:

1. Allow UDP source ports 1024-65535 and destination port 5632 with the static NAT internal IP address as the destination IP address.
2. Allow UDP source port 5632 and destination ports 1024-65535 with the static NAT internal IP address as the source IP address.
3. Allow TCP source ports 1024-65535 and destination port 5631 with the static NAT internal IP address as the destination IP address.
4. Allow TCP destination ports 1024-65535 and source port 5631 with the static NAT internal IP address as the source IP address.

An alternative would be to set up a stateful filter for UDP port 5632 and another for TCP port 5631 and apply it in the appropriate direction (from Public interface to Private interface).

Another alternative is to allow UDP port 22 instead of 5632 in filter exceptions 1 and 2 above.

## Locating Internal pcANYWHERE Host with UDP port 5632

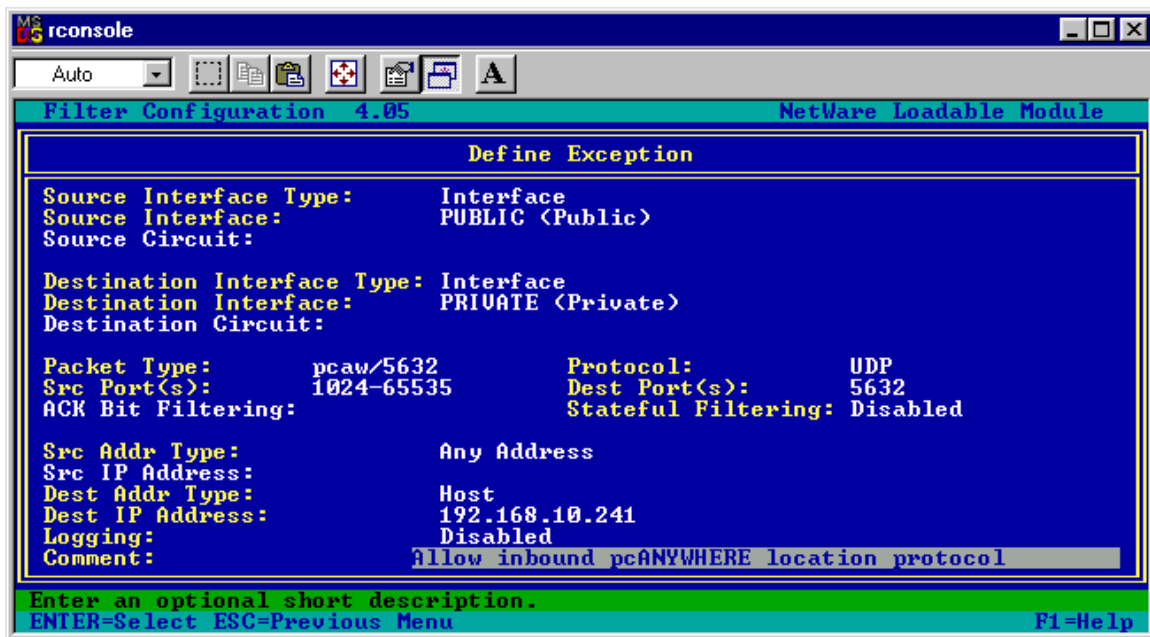


Figure 7-18 - Filter Exception for Inbound pcANYWHERE Location Protocol

The filter exception shown in Figure 7-18 allows pcANYWHERE inbound 'location' traffic, to an internal pcANYWHERE host.

- Source Interface: Public
- Destination Interface: Private
- Protocol: UDP
- Source Ports: 1024-65535
- Destination Port: 5632
- Destination IP Address: <your pcANYWHERE host internal address>

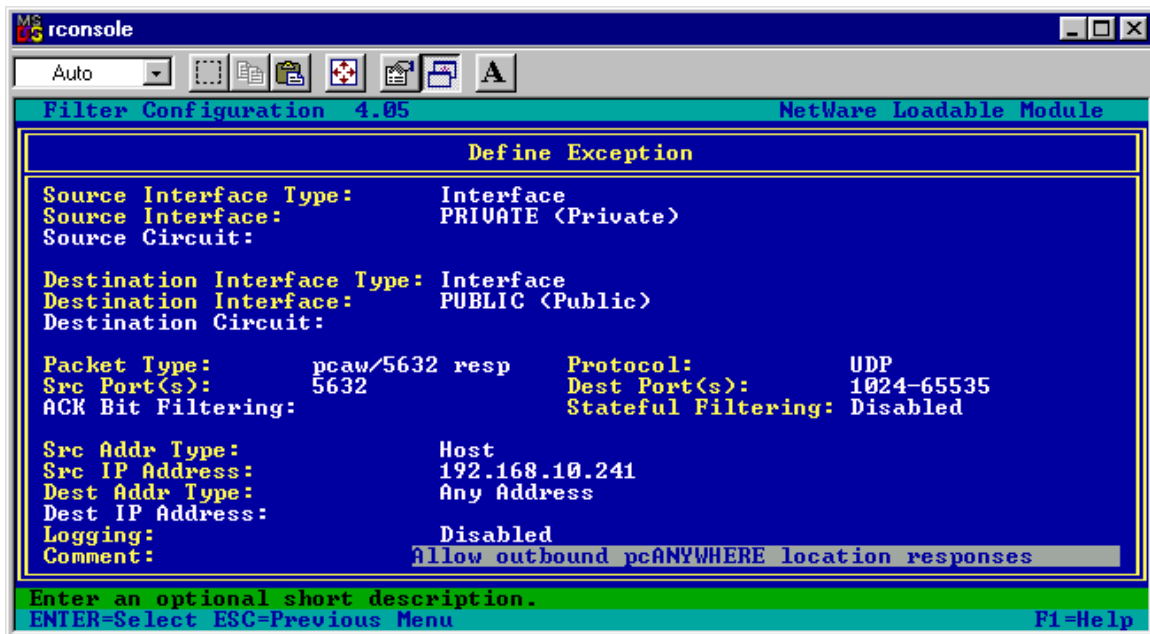


Figure 7-19 - Filter Exception for Outbound pcANYWHERE Location Responses

The filter exception shown in Figure 7-19 allows pcANYWHERE outbound 'location' traffic from an internal pcANYWHERE host.

- Source Interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source port: 5632
- Destination ports: 1024-65535
- Source IP Address: <your pcANYWHERE host internal IP address>

## Data Transfer Between pcANYWHERE Hosts using TCP port 5631

The previous example showed how to set up UDP filter exceptions to allow an internal pcANYWHERE host to be found from the Internet. Once the host is located, a TCP connection using port 5631 must be established to actually perform the remote control functions.

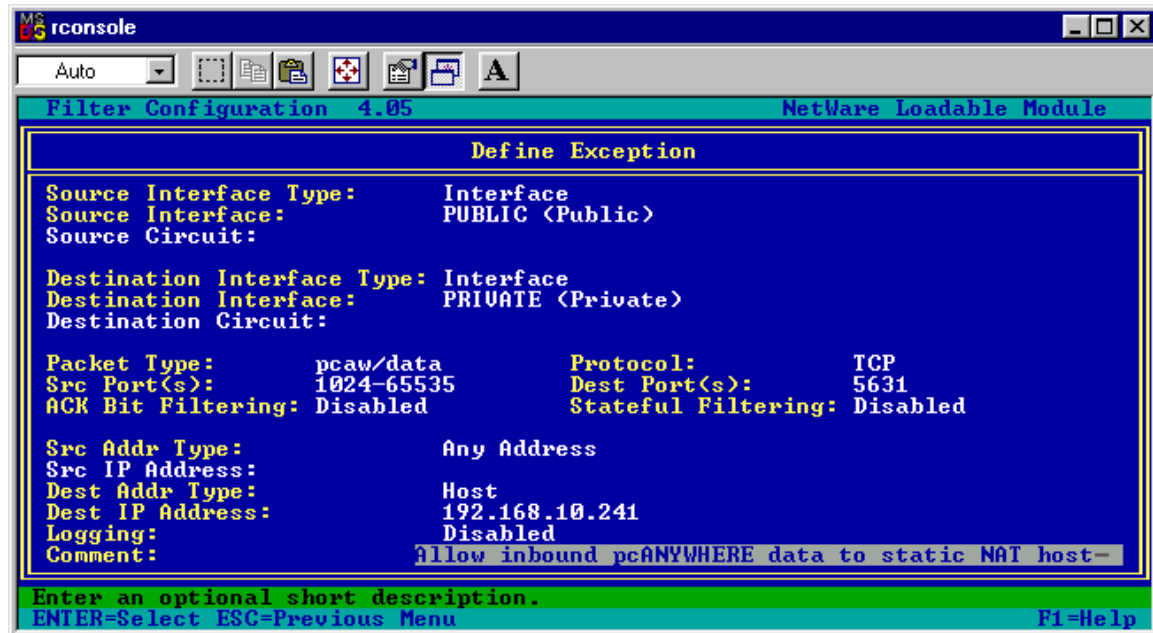


Figure 7-20 - Filter Exception for Inbound pcANYWHERE Data

The filter exception shown in Figure 7-20 allows inbound pcANYWHERE data to the internal pcANYWHERE host through static NAT.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 5631
- Destination IP Address: <your pcANYWHERE host internal IP address>

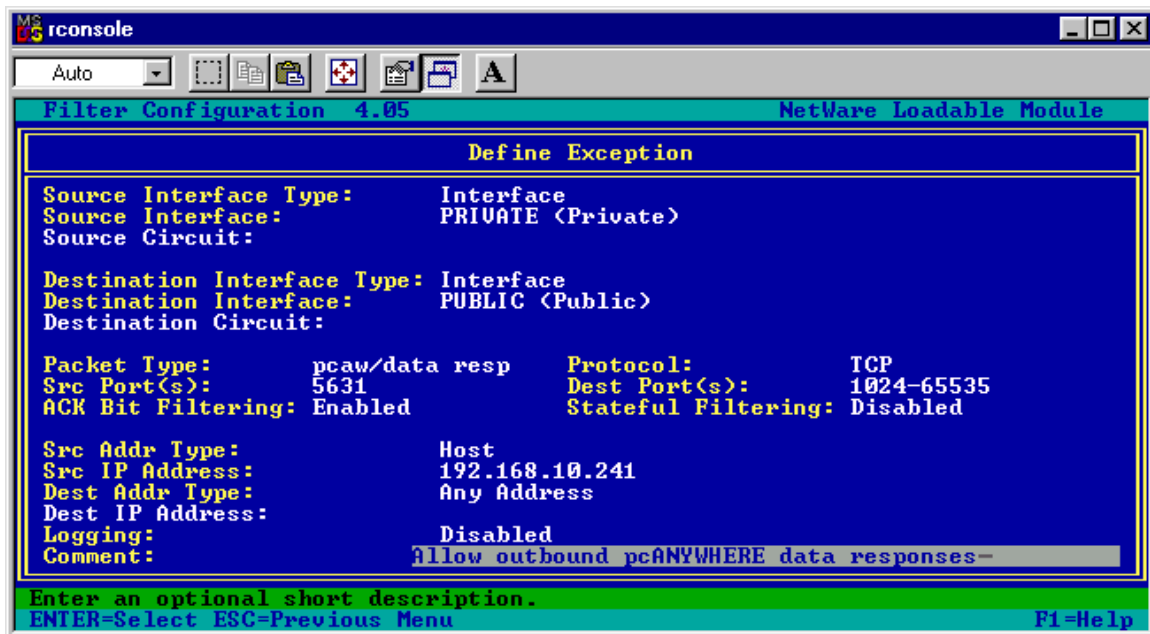


Figure 7-21 - Filter Exception for Outbound pcANYWHERE Data Responses

The filter exception shown in Figure 7-21 allows outbound data from the internal pcANYWHERE host using protocol TCP, source port 5631, destination ports 1024-65535, and a source IP address equal to the static NAT internal IP address of the internal pcANYWHERE host.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 5631
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your pcANYWHERE host internal IP address>

## Alternative - Locating Internal pcANYWHERE Host with UDP port 22

If for some reason you cannot or do not wish to allow UDP port 5632 in and out of your network, you can follow these examples for using UDP port 22 instead (or in addition to).

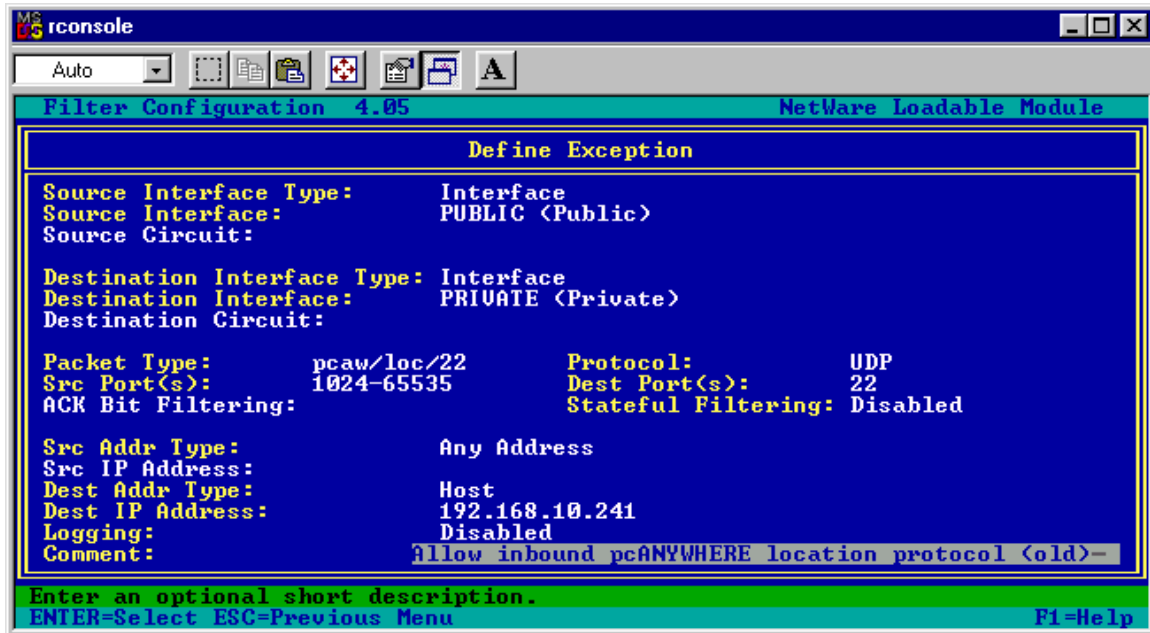


Figure 7-22 - Filter Exception for Inbound Older pcANYWHERE Location Protocol

The filter exception shown in Figure 7-22 shows an alternative to allowing UDP port 5632. It allows inbound 'location' traffic using the obsolete pcANYWHERE UDP port 22 to an internal pcANYWHERE host through static NAT.

- Source Interface: Public
- Destination Interface: Private
- Protocol: UDP
- Source Ports: 1024-65535
- Destination Port: 22
- Destination IP Address: <your pcANYWHERE host internal IP address>

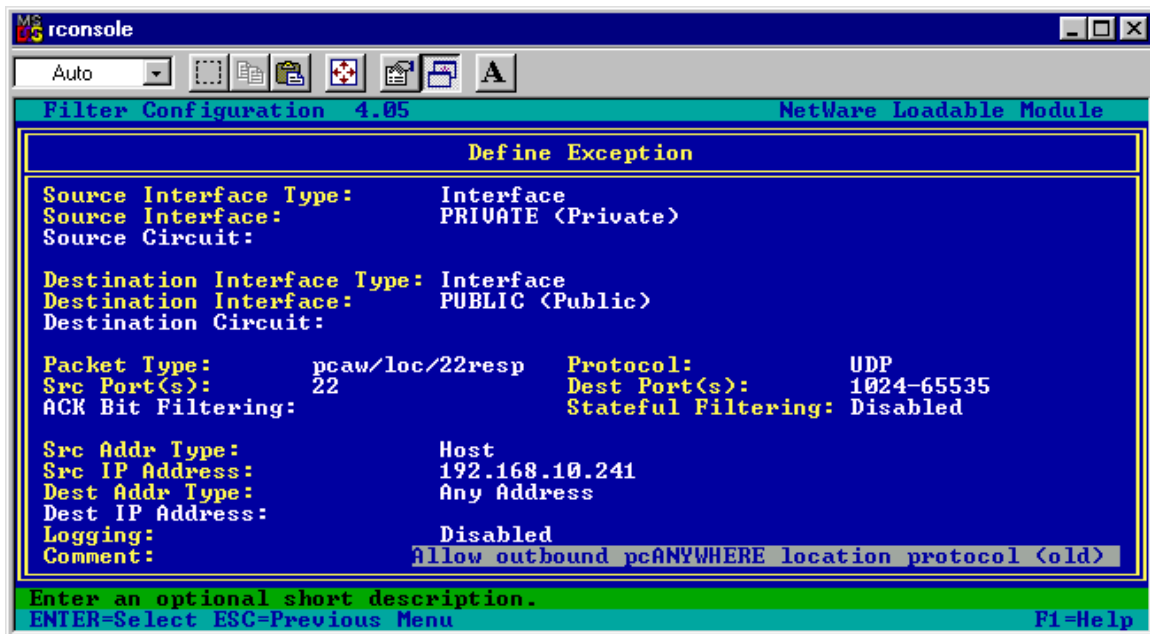


Figure 7-23 - Filter Exception for Outbound Older pcANYWHERE Location Protocol Responses

The filter exception shown in Figure 7-23 shows the second half of the UDP port 5632 alternative. It allows outbound traffic using the obsolete pcANYWHERE location protocol UDP port 22 from an internal pcANYWHERE host.

- Source Interface: Private
- Destination Interface: Public
- Protocol: UDP
- Source port: 22
- Destination ports: 1024-65535
- Source IP Address: <your pcANYWHERE host internal address>



## POP3

The following example shows how to allow POP3 mail traffic to be requested by a host on the Internet to an internal mail server using static NAT.

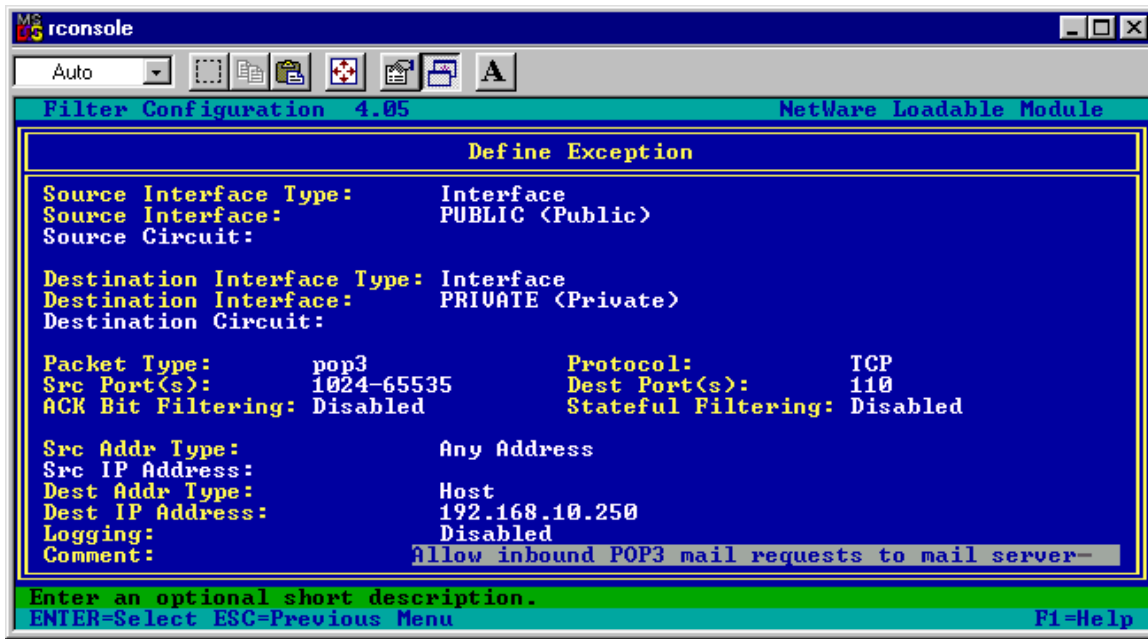


Figure 7-24 - Filter Exception for Inbound POP3 Requests to Internal Mail Server

The filter exception shown in Figure 7-24 allows inbound POP3 mail requests to an internal host at IP address 192.168.10.250.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 110
- Destination IP Address: <your POP3 mail server internal IP address>

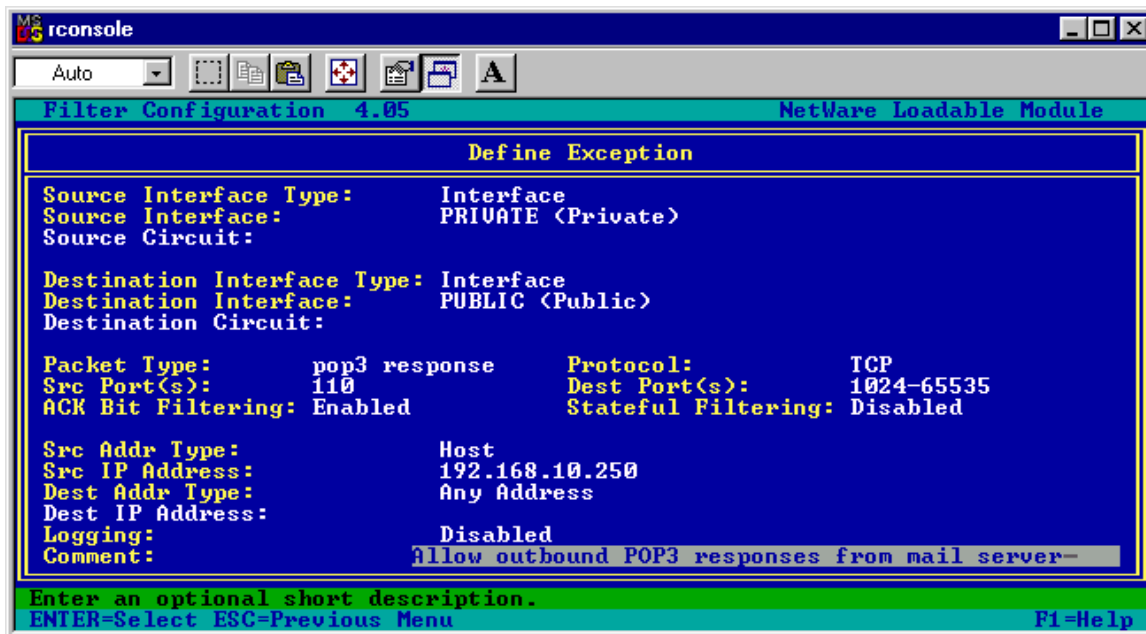


Figure 7-25 - Filter Exception for Outbound POP3 Responses from Internal Mail Server

The filter exception shown in Figure 7-25 allows an internal mail server at IP address 192.168.10.250 to send POP3 replies.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 110
- Destination ports: 1024-65535
- Source IP Address: <your POP3 mail server internal IP address>

## SMTP

The following examples show how to allow SMTP mail traffic to and from an internal SMTP mail server using static NAT. It is often a good idea to further restrict this static NAT traffic to only allow communications between the internal host and the ISP's mail server. (If the ISP has multiple mail servers, set up filter exceptions for each of their mail server IP addresses). Restricting SMTP traffic to only the ISP's mail servers will help prevent someone from using your mail server as a mail relay host (for spamming purposes).

Your SMTP mail server might also need to make DNS queries, and depending on how you have DNS services set up on your network, you may also need to add outbound DNS filter exceptions (one outbound, plus one return traffic exception) for the internal SMTP server IP address.

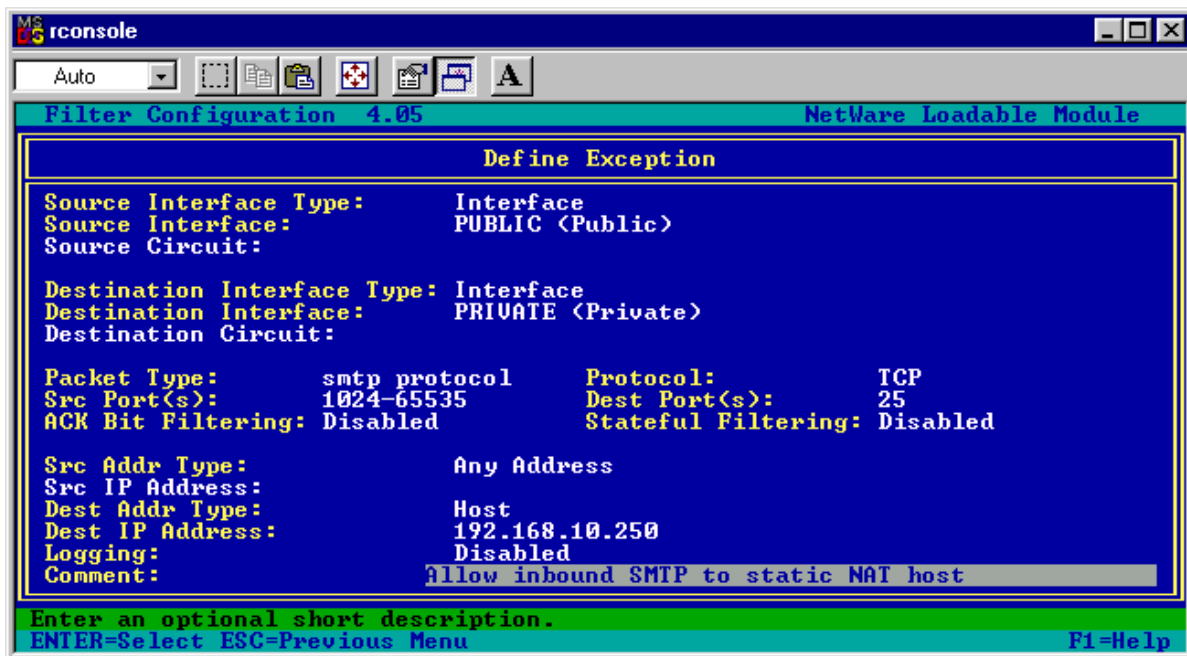


Figure 7-26 - Filter Exception for Inbound SMTP

The filter exception shown in Figure 7-26 allows anyone to send SMTP port 25 mail to the internal SMTP mail server at 192.168.10.250. This filter exception allows protocol TCP with any source port and a destination port of 25 to a destination IP address set to the static NAT internal IP address used by an SMTP mail server.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP

- Source Ports: 1024-65535
- Destination Port: 25
- Destination IP Address: <your SMTP mail server internal address>

---

**Note** Here is where you might want to add your ISP's mail server IP address as a Source IP address.

---

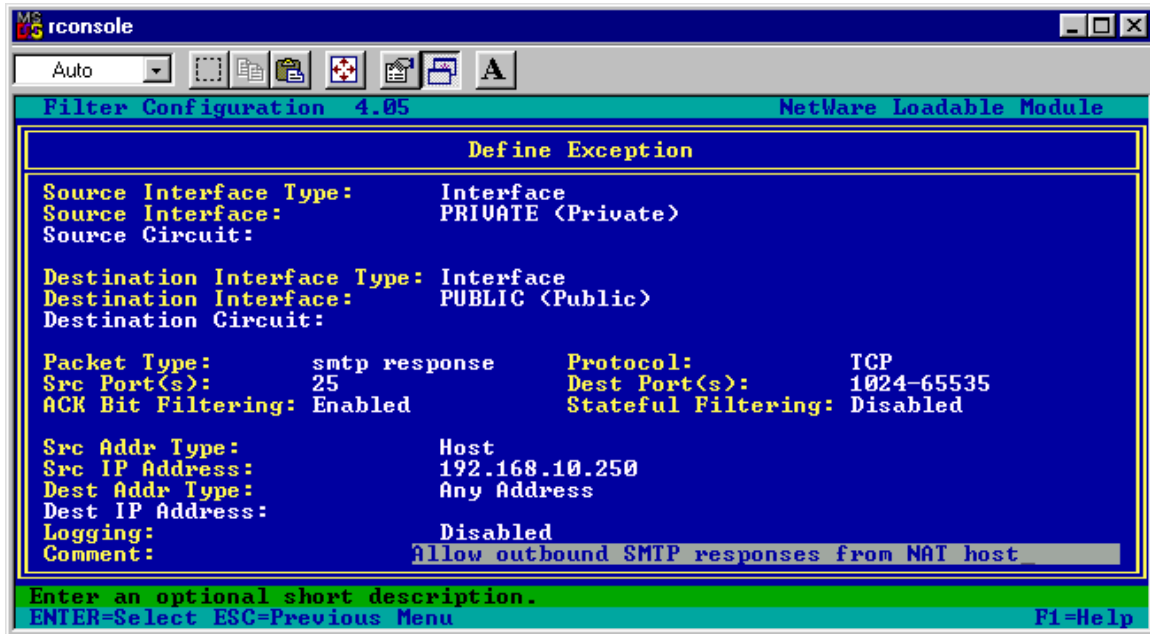


Figure 7-27 - Filter Exception for Outbound SMTP Responses

The filter exception shown in Figure 7-27 allows the SMTP mail host to respond to SMTP requests coming in. This filter exception allows protocol TCP with source port 25 and a destination port range of 1024-65535 from a source IP address equal to the static NAT internal IP address of an SMTP mail server. Set the destination IP address equal to the SMTP server of your ISP if you want to allow communications only to your ISP's mail server(s).

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 25
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your SMTP server internal address>

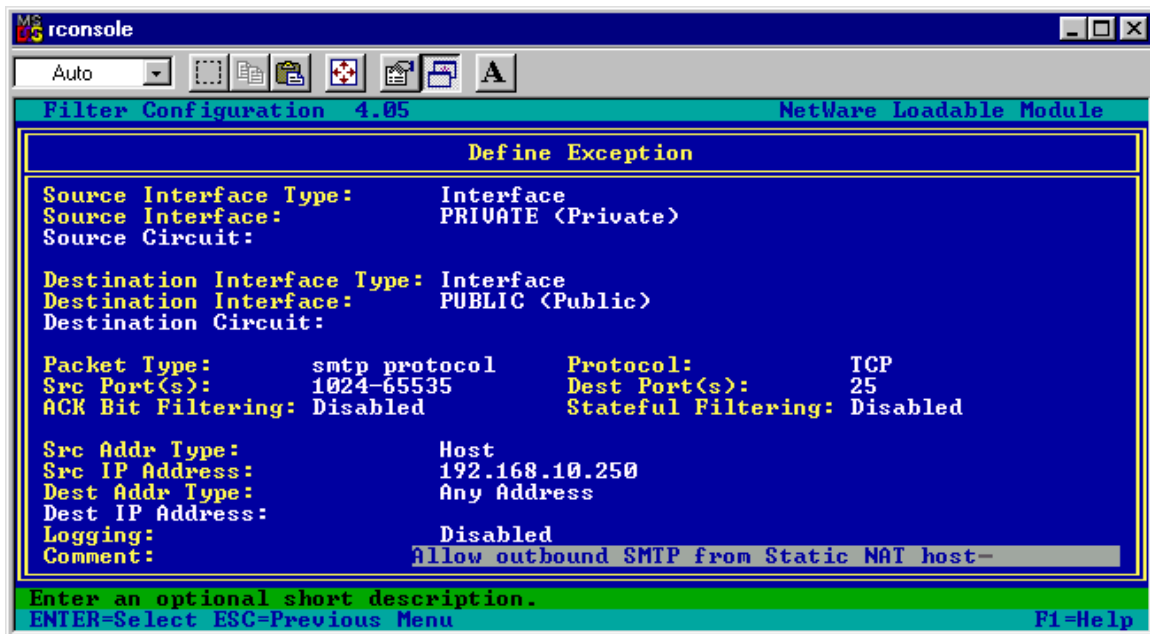


Figure 7-28 - Filter Exception for Outbound SMTP

The filter exception shown in Figure 7-28 allows the internal SMTP mail server to **send** SMTP mail. Please observe that the filter is also applied to the internal IP address and not the public IP address called out in the static NAT table. The filter exception allows protocol TCP with any source port and a destination port of 25 from an IP address set to the static NAT internal IP address of an SMTP server.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 25
- Destination IP Address: <your SMTP server internal address>

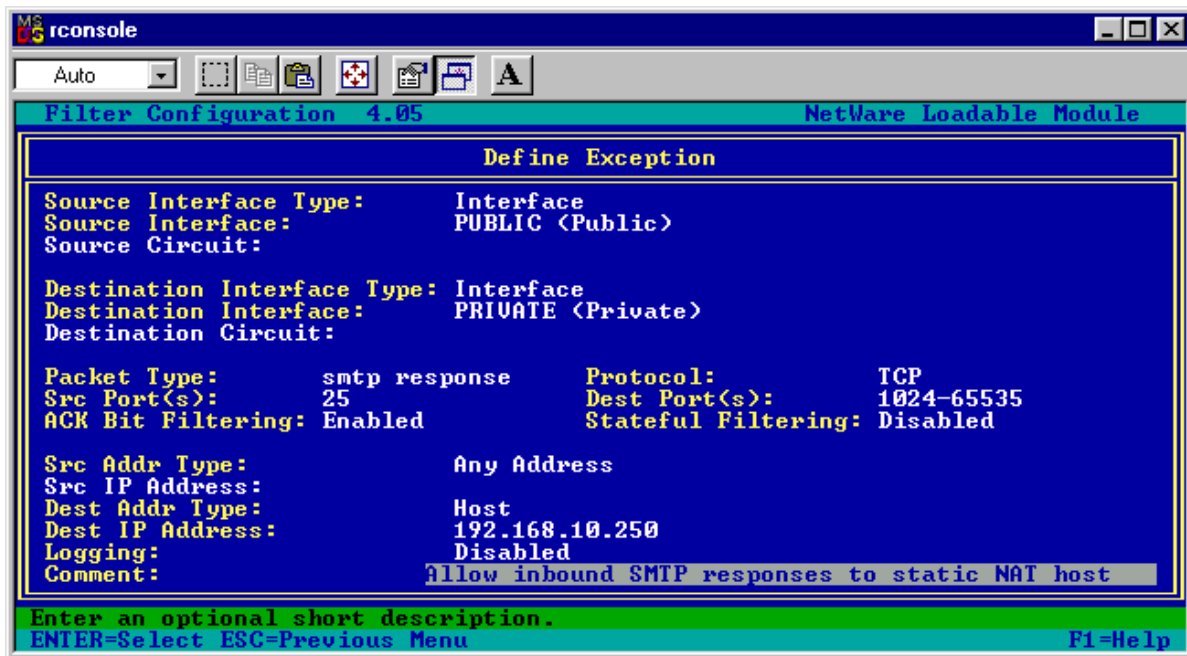


Figure 7-29 - Filter Exception for Inbound SMTP Responses

The filter exception shown in Figure 7-29 allows the internal SMTP mail host to receive responses to SMTP requests coming going out. This filter exception allows protocol TCP with source port 25 and a destination port range of 1024-65535 from any source IP address, and to a destination source IP address equal to the static NAT internal IP address of an SMTP mail server. Set the source IP address equal to the SMTP server of your ISP if you want to allow communications only to your ISP's mail server(s).

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source port: 25
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Destination IP Address: <your SMTP server internal address>

---

**Note** If you are using GWIA for your SMTP mail server, you need to put a ROUTE.CFG file in the DOMAIN\WPGATE\GWIA directory. Check the Novell Knowledgebase for details on this.

---

## VNC

VNC is a free, open source, remote control program that can be run on a variety of platforms. See <http://www.uk.research.att.com/vnc>.

This example shows how to allow VNC to an internal host through Static NAT. Up to 10 VNC console sessions at once are allowed.

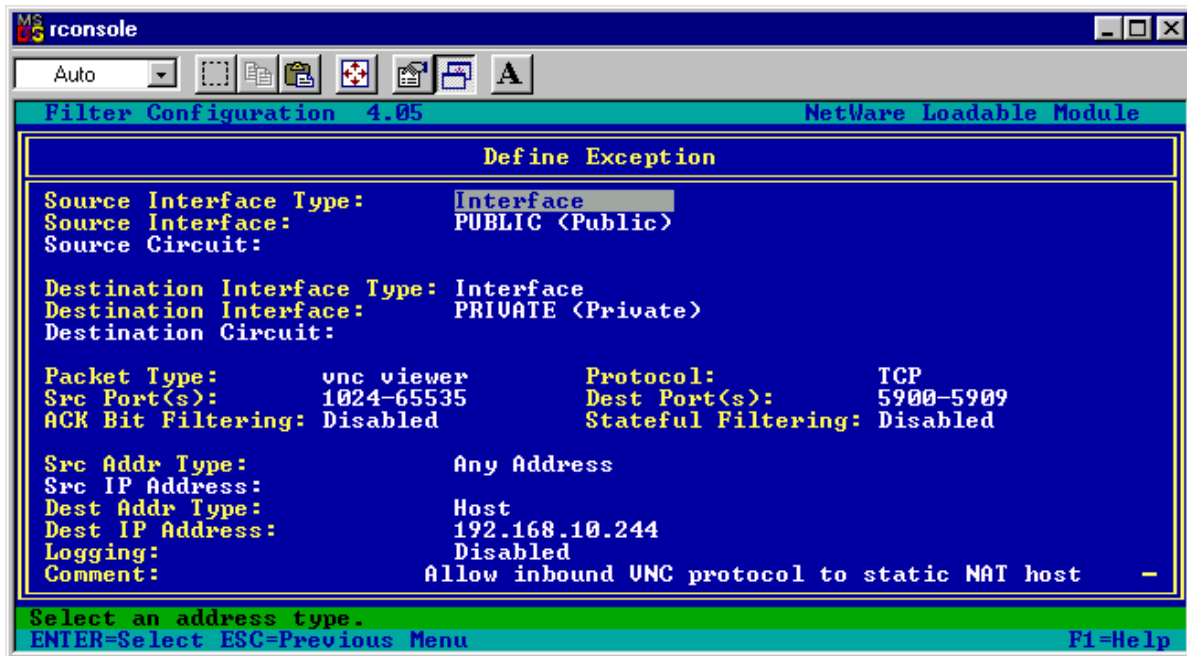


Figure 7-30 - Filter Exception for Inbound VNC Console Connections 1-10

The filter exception shown in Figure 7-30 allows inbound VNC Viewer traffic through static NAT to an internal host at the specified destination IP address.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Ports: 5900-5909
- Destination IP Address: <your VNC server internal address>

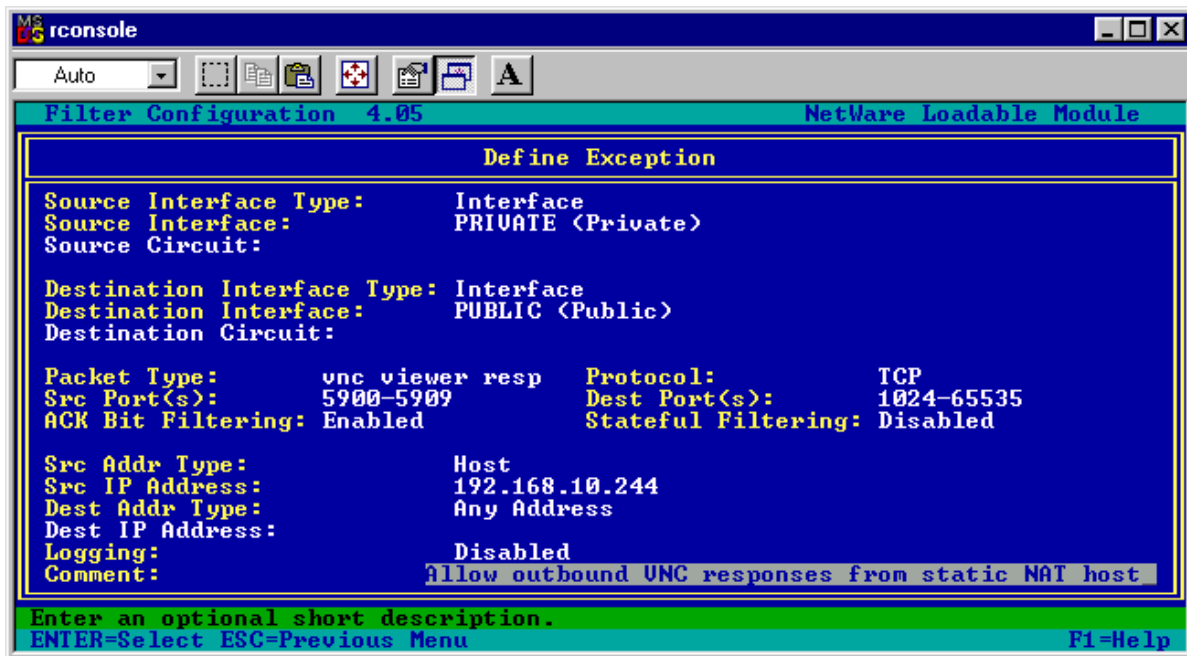


Figure 7-31 - Filter Exception for Outbound VNC Responses

The filter exception shown in Figure 7-31 allows an internal VNC server at the specified source IP address to respond to inbound VNC Viewer requests.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source ports: 5900-5909
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your VNC server internal address>

---

**Note** Should you wish to make your internal VNC host accessible via web browser, you will also need to allow TCP destination ports 5800-5809, and TCP destination port 80 in, and the appropriate responses out. (TCP destination port 80 could be allowed via filter exceptions or reverse proxy).

---



## Web Servers

If you cannot use Reverse Proxy to make an internal web server available to the Internet, the following example will make a web server accessible via static NAT. An additional pair of exceptions for HTTPS / SSL (TCP destination port 443) might also be required.

One reason to use static NAT instead of reverse proxy is that software virtual web servers (multiple web servers sharing the same IP address) are not supported with Reverse Proxy.

### HTTP to Internal Web Server

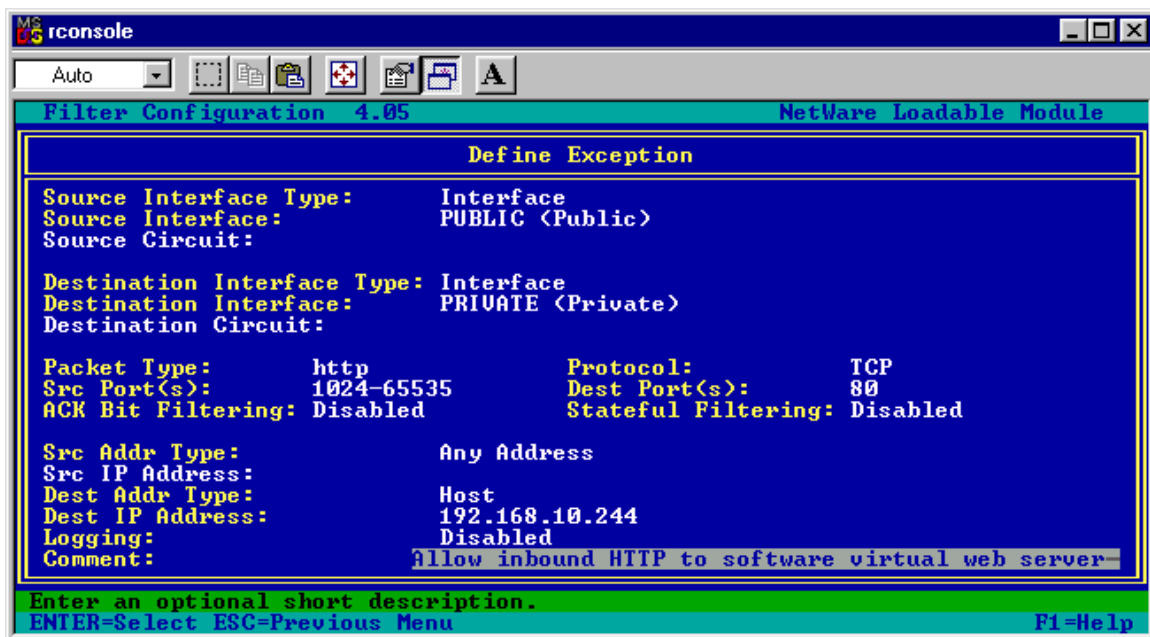


Figure 7-32 - Filter Exceptions for Inbound HTTP to Web Server

The filter exception shown in Figure 7-32 allows inbound web traffic on the standard HTTP port number through static NAT to an internal web server at the specified destination IP address.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 80
- Destination IP Address: <your web server internal address>

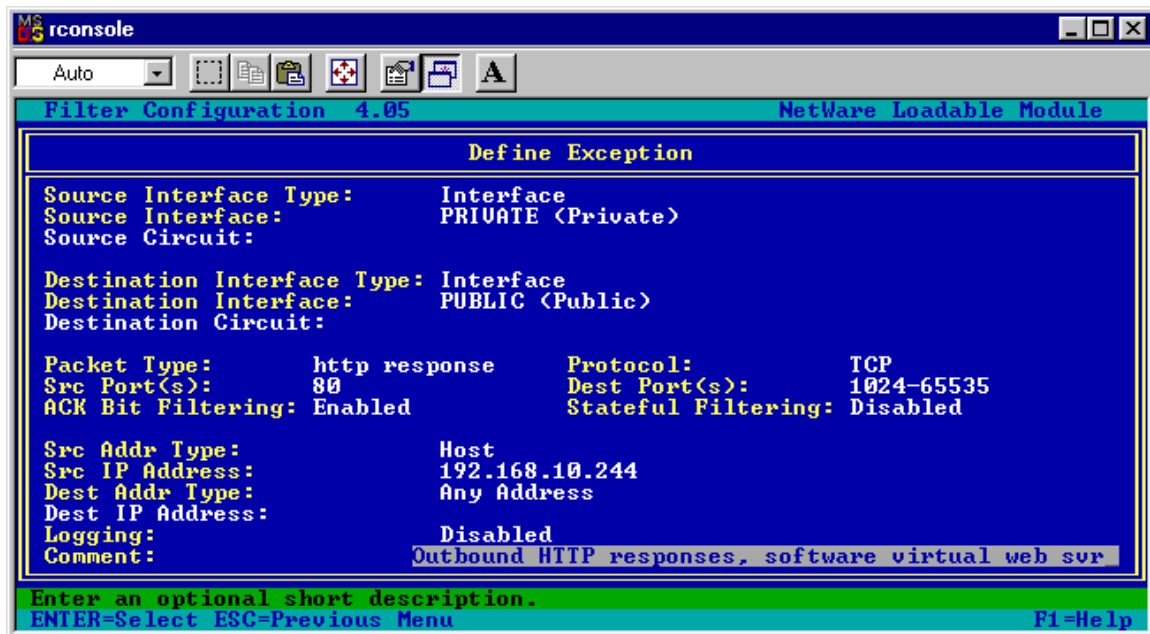


Figure 7-33 - Filter Exception for Outbound HTTP Responses

The filter exception shown in Figure 7-33 allows an internal web server at the specified source IP address to respond to HTTP request on the standard port.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 80
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your web server internal address>

## HTTPS /SSL to Internal Web Server

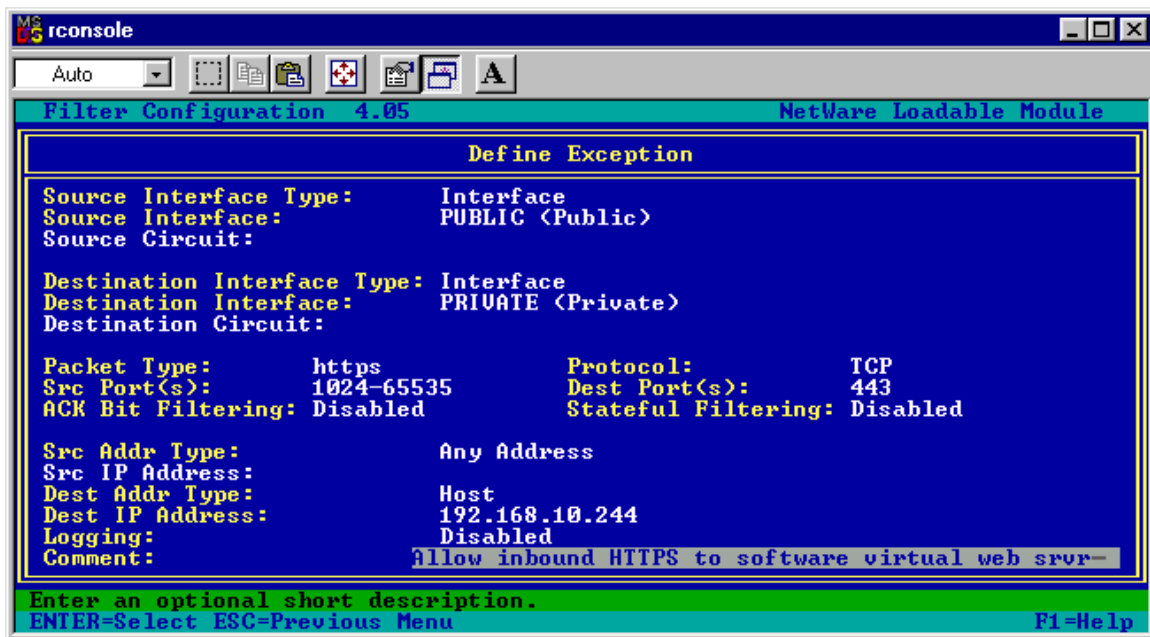


Figure 7-34 - Filter Exception for Inbound HTTPS / SSL

The filter exception shown in Figure 7-34 allows inbound HTTPS / SSL traffic through static NAT to an internal host at the specified destination IP address.

- Source Interface: Public
- Destination Interface: Private
- Protocol: TCP
- Source ports: 1024-65535
- Destination port: 443
- Destination IP Address: <your web server internal address>

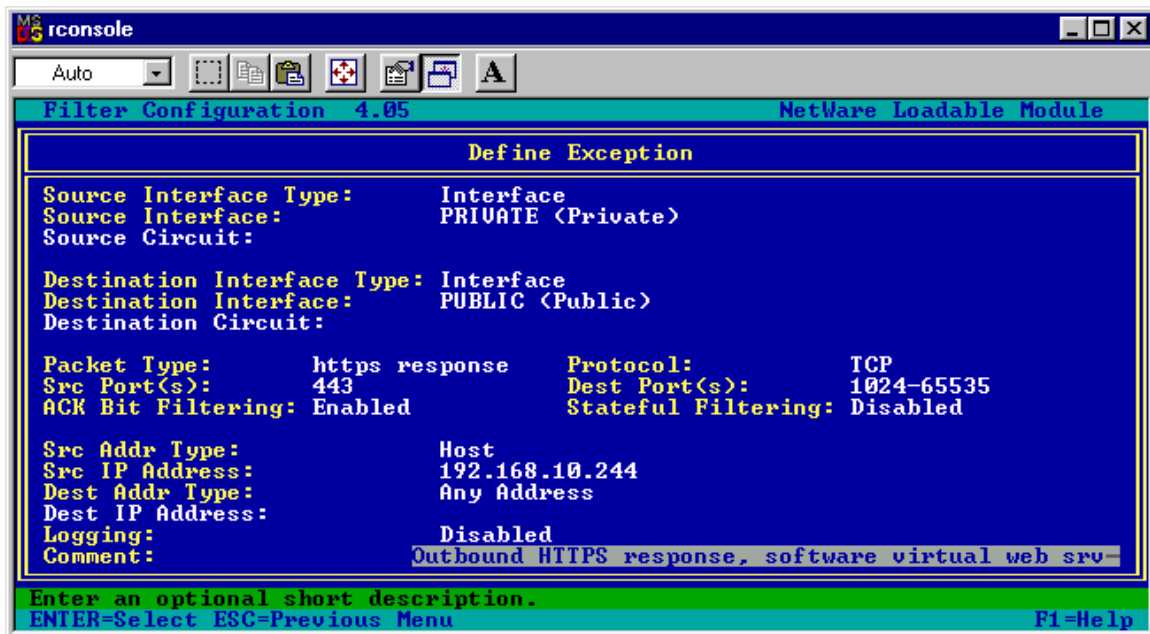


Figure 7-35 - Filter Exception for Outbound HTTPS Responses

The filter exception shown in Figure 7-35 allows HTTPS / SSL responses from an internal host at the specified source IP address to inbound requests.

Note that ACK bit filtering has been enabled.

- Source Interface: Private
- Destination Interface: Public
- Protocol: TCP
- Source port: 443
- Destination ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <your web server internal address>

# Chapter 8 - BorderManager 2.1 – Stateful Filters Alternative

---

Many of the examples of filter exceptions assume the use of stateful filters. Since BorderManager 2.1 does not include stateful filtering capability, you must configure additional exceptions to get things working. BorderManager 2.1 does not include stateful filter capability, ACK filter capability or the special FTP-PORT, FTP-PASV, and FTP-PORT-PASV filters. In general, you will end up needing to set up a Dynamic TCP or Dynamic UDP filter exception to allow the high ports into your network so that responses to outbound queries are not filtered as they return.

For example, you wish to allow outbound DNS requests from any internal host to an external DNS server hosted at your ISP. So you first set up a simple DNS filter exception allowing outbound destination port 53 over UDP. With dynamic NAT enabled, your internal hosts can send their DNS requests out. However, they never get a response.

The reason is that the host is sending out requests to destination UDP port 53, but the request is sent out with a randomly-chosen high source port, let's say port 35145. Your host expects to see a reply to that destination port when a return packet comes back. (Outbound, source port 35145 & **destination** port 53, Inbound reply traffic, **source** port 53 & destination port 35145).

The originating host PC will pick a 'high' port number at random, and that port number can be anywhere between port 1024 and port 65535.

You therefore must set up a Dynamic UDP filter exception on your Public IP address to allow **all** packets between port 1024 and port 65535 into your network. You could set up one filter exception allowing Any source port, or restrict the source port to port 53 only. Restricting the source port to specific port numbers will enhance security, but it will also require you to set up a new Dynamic UDP (or

Dynamic TCP) exception for each outbound port number you wish to allow.

The best security using packet filter exceptions in BorderManager 2.1 for inbound traffic will be to set up individual filter exceptions for each type of return traffic. Specify the source port (usually the same as the destination port of the outgoing traffic) as well as a range of destination ports from 1024-65535. In addition, when the traffic is intended only to go to and from known hosts, add a source IP address to the filter. An example would be to allow SMTP return traffic only from TCP source port 25, destination ports 1024-65535, and source IP address equal to your ISP's mail server IP address. This will require a separate filter exception for each mail server used at the ISP. This short example also addresses only return traffic in an SMTP conversation - not inbound SMTP mail itself. Inbound SMTP mail would require another set of filter exceptions for each of the ISP's mail servers using TCP destination port 25, and requiring the source ports to be in the range of 1024-65535. One begins to see that BorderManager 2.1 filter exceptions get to be complex in the absence of stateful filters as several related filter exceptions are needed to allow traffic out and only the desired return traffic back in. Keep good notes!

## Generic Exception for TCP Return Traffic

If you do not want to set up *individual* filter exceptions to allow return TCP traffic for each application through BorderManager, you can set up a single filter exception that will allow almost all of your outgoing TCP traffic to receive a response.

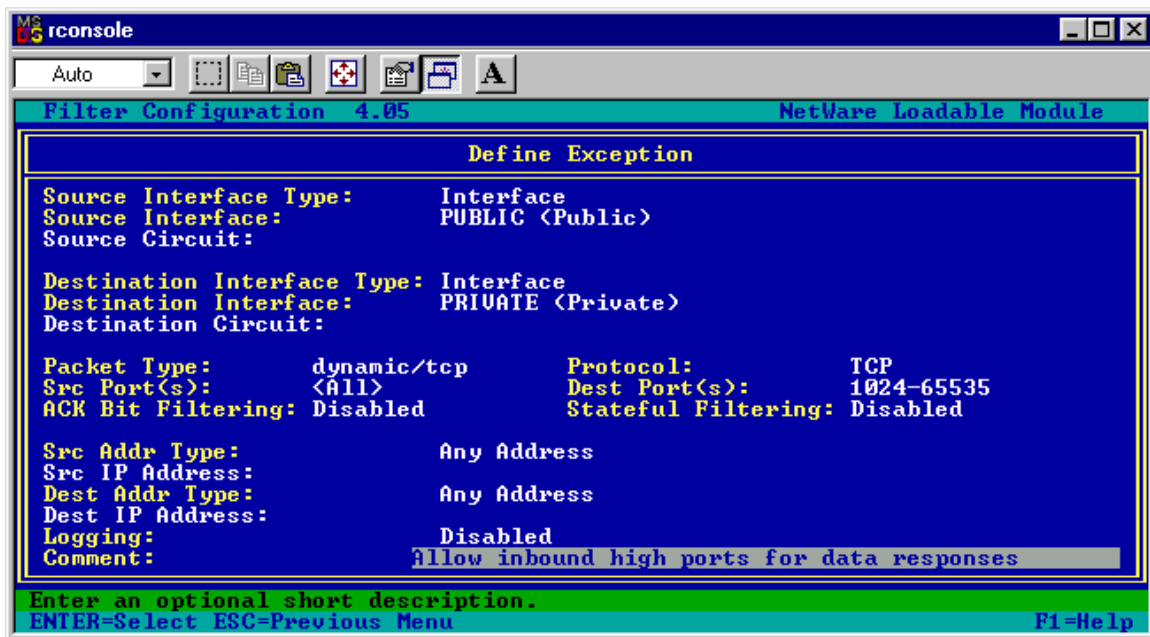


Figure 8-1 - Generic TCP Filter Exception to Allow All Return Traffic

Apply the built-in Dynamic/TCP filter definition to allow all TCP high ports. This filter exception allows destination TCP ports 1024-65535, all source ports, with a source interface = the public interface and a destination interface = the private interface.

## Generic Exception for UDP Return Traffic

If you do not want to set up *individual* filter exceptions to allow return UDP traffic for each application through BorderManager, you can set up a single filter exception that will allow almost all of your outgoing UDP traffic to receive a response.

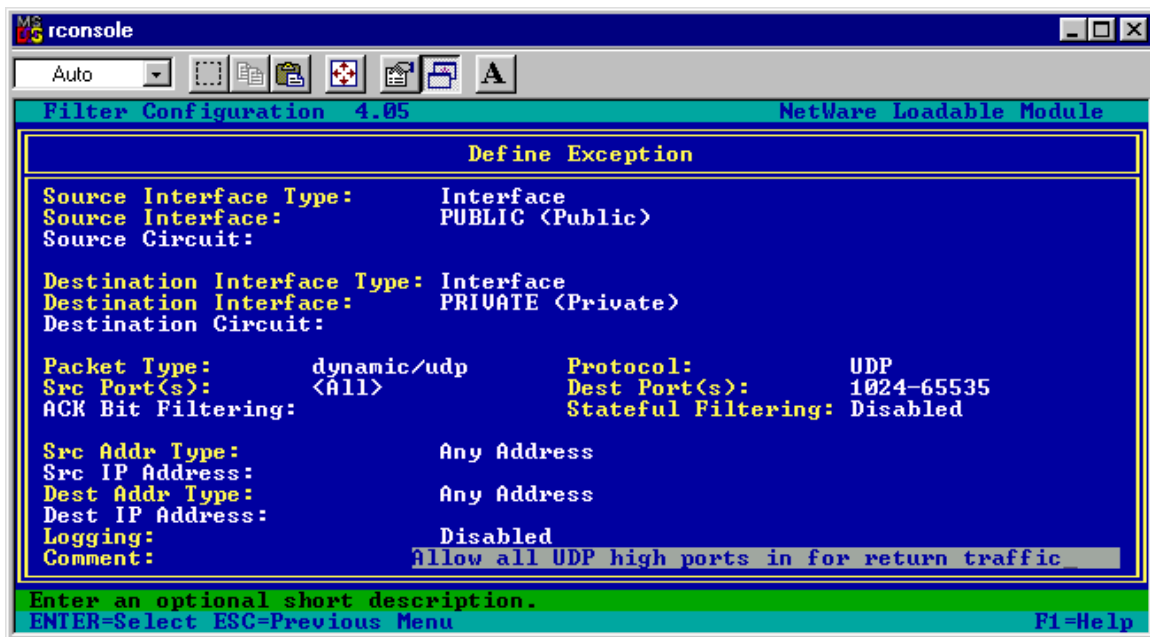


Figure 8-2 - Generic UDP Filter Exception to Allow All Return Traffic

Apply the built-in Dynamic/UDP filter definition to allow all UDP high ports. This filter exception allows destination UDP ports 1024-65535, all source ports, with a source interface = the public interface and a destination interface = the private interface.



# Chapter 9 - Advanced Topics

---

This chapter is not really for beginners – it assumes the reader has understood the previous sections in detail. The sections here are for people who want the most control and security over their Internet connection, and really know what their applications are doing.

## Basic Improvement - Enhance the Security of the Default Exceptions

This section applies mainly to BorderManager versions prior to 3.7.

As discussed earlier in the book, in the section on ACK bit filtering, the default Dynamic/TCP filter exception configured with BRDCFG does not enable ACK bit filtering. This is because the default exceptions were held over from BorderManager 2.1, when ACK bit filtering was not available.

You can **significantly enhance** the security of your BorderManager server by either converting the existing Dynamic/TCP exception to use ACK bit filtering, or by replacing it with your own custom exception.

With the change suggested in this section, you can prevent relaying from the internet to either SOCKS Gateway or Transparent Proxy, prevent certain denial of service attacks, and secure the high TCP ports for various NetWare 5.x and 6.x applications.

---

**CAUTION**     *If you change or replace the default Dynamic/TCP exception to enable ACK bit filtering, you will have to add custom filter exceptions for any inbound TCP high port connections to generic proxies, and services listening on the server's public IP address (such as RCONAG6 if you want to allow that).*

---

Why should this relatively simple change make a difference? Because there are services which run on NetWare that are listening on the public IP address(es). For instance, the CSATPXY.NLM, used for logging purposes, listens on TCP destination port 2000. Before a patch was implemented, there was a way to attack that port and cause a BorderManager server to ABEND. The SOCKS proxy may be listening on the public IP address at port 1080. Portal could be listening on ports 8008 and 8009.

The default Dynamic/TCP exception allowed connections to these ports to be made from the Internet, because it allowed all inbound traffic to the public IP address. Enabling ACK bit filtering prevents inbound connections, but allows inbound responses to connections made by the proxies.

Here is a very short description of how TCP connections are established, so that you can see why ACK bit filtering is important and useful. The way that TCP connections are set up is by what can be referred to as a SYN – ACK – ACK process. Three packets must be exchanged to establish the TCP connection. The first packet enables a SYN bit, but does not have the ACK (acknowledge) bit set. The host that receives such a packet (and wants to set up a connection on the requested port) returns a response with the ACK bit set, and a SYN bit as well. The original host (requesting the connection), sees the ACK bit (and other related fields), and sends back a third packet with the ACK bit set to acknowledge the connection. In this way, each host has acknowledged the other host, and exchanged necessary information (in other fields not described here) so that further communication can take place. All further communications between these hosts will have the ACK bit set.

Only the first packet sent did NOT have an ACK bit set. Therefore, if we wish to prevent inbound connections, we filter for the presence of the ACK bit. (We allow connections in the other direction to flow out without the ACK bit set, so that we can establish a connection). As long as the connection was initiated by our host, we allow the return responses, since they will have the ACK bit set.

Note that the examples in this book generally use ACK bit filtering for all response packets in the static NAT examples. In those cases, we are actually applying ACK bit filtering in the reverse direction, allowing inbound connections only. This allows us to prevent our own internal hosts from making undesired outbound connections, which could be a security risk.

I show two ways to change the exceptions. The first method simply replaces the existing default Dynamic/TCP filter exception. The second method actually changes the definition for the default Dynamic/TCP filter exception. I recommend using the first method, especially for BorderManager 3.7.

## Creating a Custom Dyn/ACK/TCP Filter Exception

In this example, which is the simplest of the two shown, you first create the following custom filter exception, and then you delete the default Dynamic/TCP exception.

You should make a backup copy of the SYS:\ETC\FILTERS.CFG file before making these changes. Should something go wrong, and you want to put the old set of exceptions back in place, UNLOAD IPFLT, copy the files back in, then REINITIALIZE SYSTEM. BorderManager 3.7 users would also need to delete all the objects inside the NBMRuleContainer container, and remigrate the filters with a FILTSRV MIGRATE procedure.

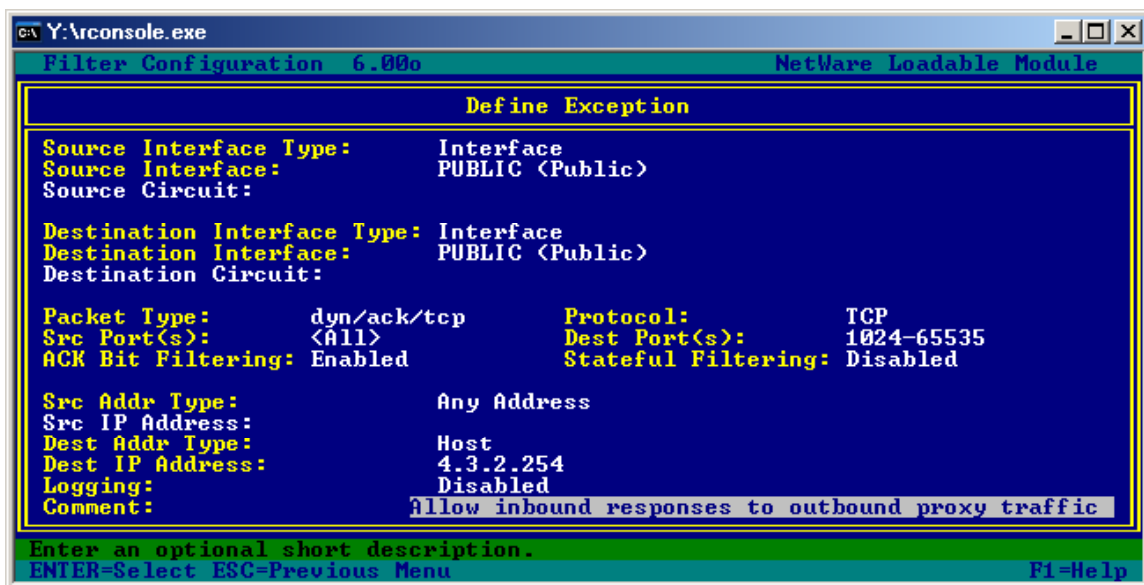


Figure 9-1 - Custom Dyn/ACK/TCP Filter Exception

The example shown is almost the same as the default Dynamic/TCP exception, EXCEPT that ACK bit filtering has been enabled. I have also configured both source and destination interfaces to use the public interface. The destination IP address should be the BorderManager server's primary public IP address.

Remember that if you run BRDCFG again in the future, it will create another default Dynamic/TCP exception which would override the one shown in Figure 9-1.

## Customizing the Default Dynamic/TCP Default Filter Exception

---

**CAUTION** *I do not recommend using this method for BorderManager 3.7 due to the complexity of making it work with filter definitions already in place in NDS. If you want to try using this method on BorderManager 3.7, I recommend you plan on deleting the objects in the NBMRuleContainer container and performing a FILTSRV MIGRATE procedure. It will be essential to make a backup copy of SYS:\ETC\FILTERS.CFG before you start.*

---

If you want to customize the default dynamic/TCP filter exception, you will need to edit the SYS:\ETC\BUILTINS file.

You should make backup copies of the SYS:\ETC\BUILTINS.CFG and SYS:\ETC\FILTERS.CFG file before making these changes. Should something go wrong, and you want to put the old set of exceptions back in place, UNLOAD IPFLT, copy the files back in, then REINITIALIZE SYSTEM. BorderManager 3.7 users would also need to delete all the objects inside the NBMRuleContainer container, and remigrate the filters with a FILTSRV MIGRATE procedure.

---

**Note** Also, see the section in the Odds & Ends chapter called Fixing the BorderManager 3.5 POP3-ST Definition.

---

**Step 1:** The SYS:\ETC\BUILTINS.CFG file is a text file flagged read-only, so you must first flag it as normal in order to edit it. Change to the SYS:\ETC directory and execute

**FLAG SYS:\ETC\BUILTINS.CFG N**

That should change the status to normal.

**Step 2:** Next, use a text editor (Notepad will do) to change the one line in the file for Dynamic/TCP from:

```
PROTOCOL-SERVICE IP, dynamic/tcp, pid=TCP port=1024-65535
srcport=<All>. Dvnamic Destination Ports Over TCP
```

to (adding the text ackfilt=1 as shown below):

```
PROTOCOL-SERVICE IP, dynamic/tcp, pid=TCP port=1024-65535
srcport=<All> ackfilt=1, Dynamic Destination Ports Over
```

---

**Note** Do not add a comma before the 'ackfilt=1'.

---

**Step 3:** Unload IPFLT.NLM and then Reinitialize System.

**Step 4:** Go into FILTCFG.NLM, select the old default exception, and, if necessary, **select the Dynamic/TCP definition again**, and save the exception. (If you have put in a customized dyn/ack/tcp exception as shown in an earlier section of this book, the name of the definition may change when you save the filter exception). You may need to check other filter exceptions as well, if you had called out the Dynamic/TCP definition in some custom exception you made earlier.

If you are using BorderManager 3.7, you may need to remigrate your packet filters and exceptions into NDS with a FILTSRV MIGRATE process. First, you will need to delete all of the existing objects in the NBMRuleContainer container with NWADMN32 or ConsoleOne.

*Be sure to do some testing after making this kind of change, especially for any inbound traffic.*

---

**CAUTION** If you modify the built-in definition for the Dynamic/TCP exception, and reapply the default filters with BRDCFG, you will probably have an incomplete definition where the Dynamic/TCP exception should be. Be sure to go into FILTCFG, and review the default exceptions and be sure you have a Dynamic/TCP exception. If you do NOT modify the built-in Dynamic/TCP exception and use BRDCFG, you will end up with the original Dynamic/TCP exception, and you will need to go into FILTCFG and replace it with a custom DYN/ACK./TCP exception. **Therefore, no matter what you do here, you need to review your exceptions after running BRDCFG again.** If you are truly paranoid about this becoming a problem, delete or rename BRDCFG.NLM so it cannot be run.

---

## More Security - A DMZ Scenario

This section describes how you might set up a DMZ with a single BorderManager server using three network cards. (Another scenario would be to have a firewall, a DMZ, another firewall, and then the internal LAN). You may want to employ a DMZ in addition to enhancing the default filter exceptions as explained earlier, or along with using completely customized exceptions as explained later.

The basic concept in a 3-NIC scenario is that you have a network segment attached to one network interface in a BorderManager server which contains one or more hosts to be accessed from the Internet, (the DMZ segment). The main internal network segment sits behind another network interface. The main network segment should not have any hosts that can be accessed from the Internet. Extremely limited traffic should be allowed between the DMZ and main segment, so that if a host in the DMZ is compromised, it does not have direct access to the main segment. Traffic to and from the DMZ segment is controlled to both the Internet and main segment.

The way to accomplish this goal is to essentially treat the DMZ network interface card as a second public interface. Block all traffic to and from that interface with a pair of filters, and then open up specific ports using proxies and/or static NAT with filter exceptions. All traffic between the main internal segment and the DMZ is allowed only with stateful exceptions for the type of traffic desired.

This scenario does not lend itself well to putting servers in a DMZ which are in the same NDS tree as the main internal segment, though it can be done if you are willing to compromise security somewhat by allowing one or more DMZ hosts to make certain inbound connections to the rest of your LAN.

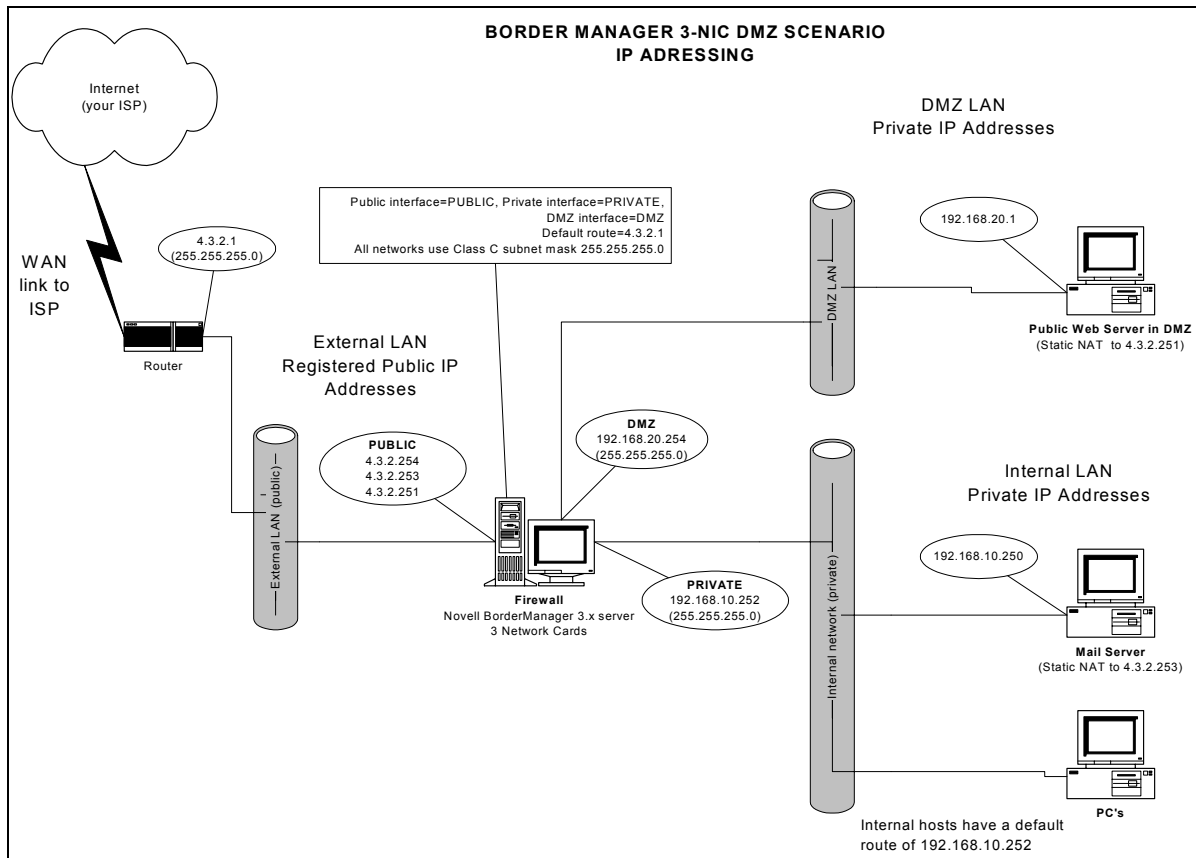


Figure 9-2 - DMZ with Three Network Cards, IP Addressing Diagram

In the example in this section, I have configured a server with three network cards, called PUBLIC, PRIVATE and DMZ. Figure 9-2 shows the IP addressing used for the example.

## Step 1 – Set Filters on the DMZ NIC

**Note** This example was written for BorderManager 3.6, and used the BRDCFG.NLM version for 3.6. For BorderManager 3.7, you should set the filters and exceptions manually.

I initially ran the BRDCFG program against the IP address assigned to PUBLIC. That produced a set of default filters that blocked all traffic to and from PUBLIC, while setting up the usual filter exceptions as well.

Next, I applied BRDCFG again, selecting the IP address assigned to DMZ. This had the effect of adding another set of filters (and exceptions) for the DMZ interface.

Finally, I **deleted all the default filter exceptions for the DMZ address** that were added by BRDCFG. I was therefore left with:

- filters blocking all traffic to and from the PUBLIC interface
- filters blocking all traffic to and from the DMZ interface
- default exceptions for selected traffic to and from the PUBLIC IP address.

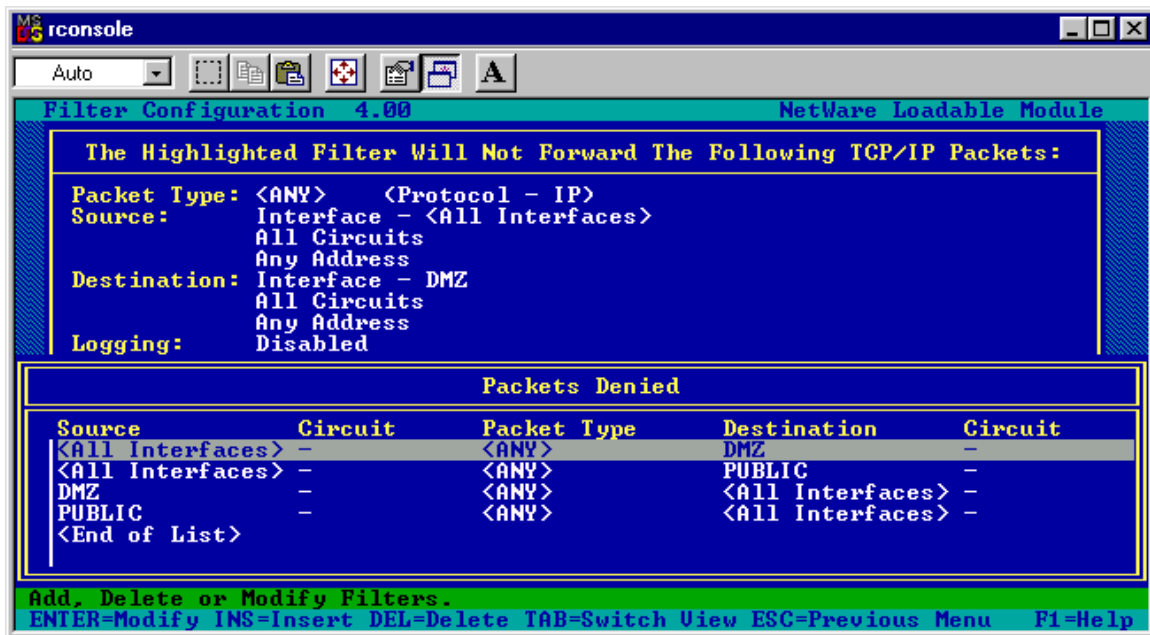


Figure 9-3 - Filters Applied for PUBLIC and DMZ Interfaces

In the example shown in Figure 9-3, you can see the filters (not exceptions) which block traffic for both PUBLIC and DMZ interfaces.



## Step 2 – Open Filter Exceptions for Inbound Traffic from the Internet to the DMZ

In this example, I have set up exceptions for a stand-alone web server residing in the DMZ segment at IP address 192.168.20.1. Access to the web server is being allowed through static NAT. Therefore, HTTP traffic is being allowed in, from the PUBLIC interface, through the DMZ interface, to 192.168.20.1. Another filter exception allows outbound HTTP responses.

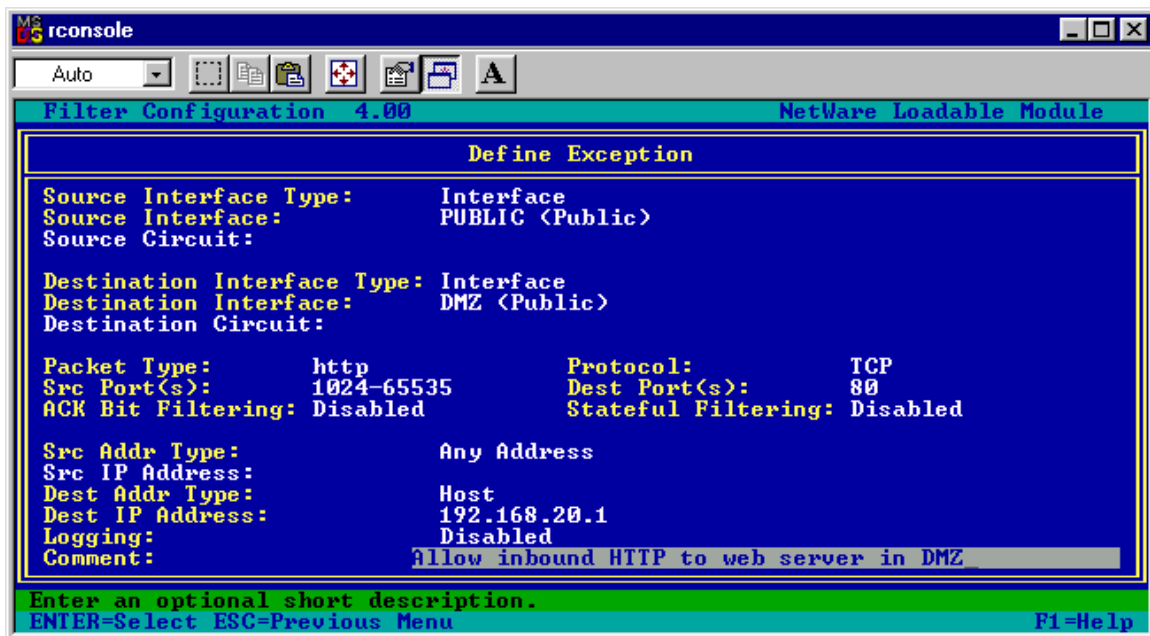


Figure 9-4 - Filter Exception to Allow Inbound HTTP to DMZ Web Server from the Internet

The filter exception shown in Figure 9-4 allows inbound HTTP via Static NAT across the PUBLIC interface to the DMZ web server at IP address 192.168.20.1.

- Source Interface: PUBLIC
- Destination Interface: DMZ
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Port: 80
- Source IP Address: Any
- Destination IP Address: <web server DMZ IP address>

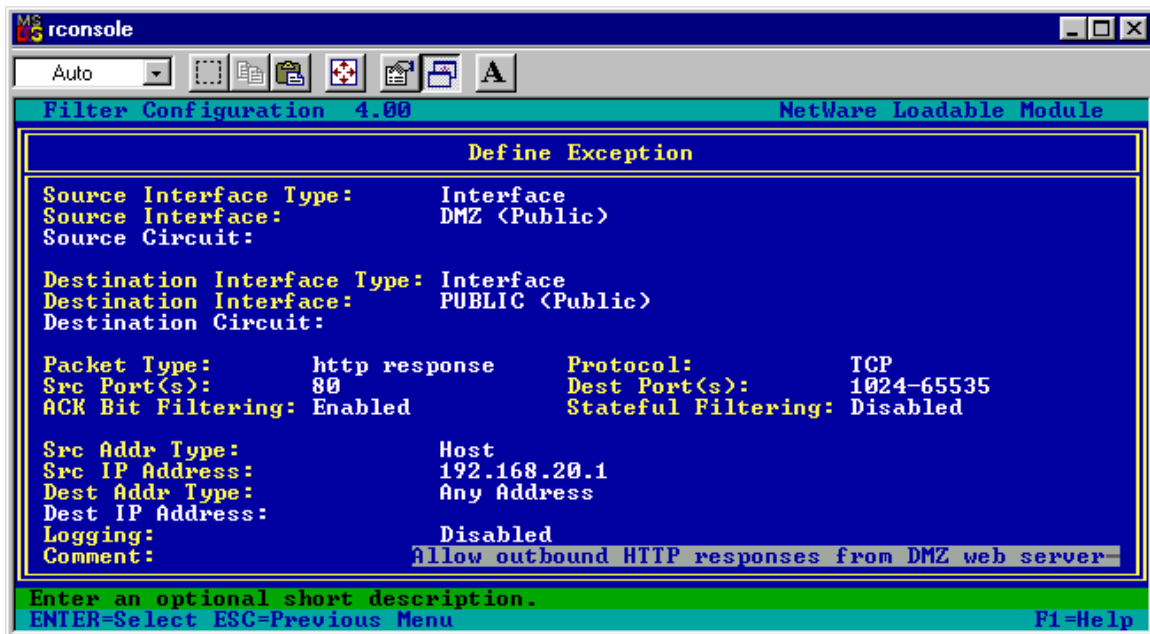


Figure 9-5 - Filter Exception to Allow Outbound HTTP Responses from DMZ Web Server to the Internet

The filter exception shown in Figure 9-5 allows outbound HTTP responses from a DMZ web server at IP address 192.168.20.1 to hosts on the Internet. ACK bit filtering has been enabled.

- Source Interface: DMZ
- Destination Interface: PUBLIC
- Protocol: TCP
- Source Port: 80
- Destination Ports: 1024-65535
- ACK Bit Filtering: Enabled
- Source IP Address: <web server DMZ IP address>
- Destination IP Address: Any

### Step 3 – Open Filter Exceptions for Outbound Traffic from the Internal LAN to the DMZ

Two stateful filter exceptions are set up to allow HTTP and FTP traffic from the internal LAN to the DMZ. The purpose of the HTTP exception is to allow the web server to be seen from the internal LAN. The purpose of the FTP exception is to allow FTP to be used to push web site updates to the web server.

No other exceptions are set up, which means that the web server cannot initiate any contact with the internal LAN. The web server also cannot initiate any contact to the Internet.

---

**Note** All of these exceptions should call out the PRIVATE interface as a source interface, to ensure that traffic is allowed only from the internal network, and not from the Internet.

---

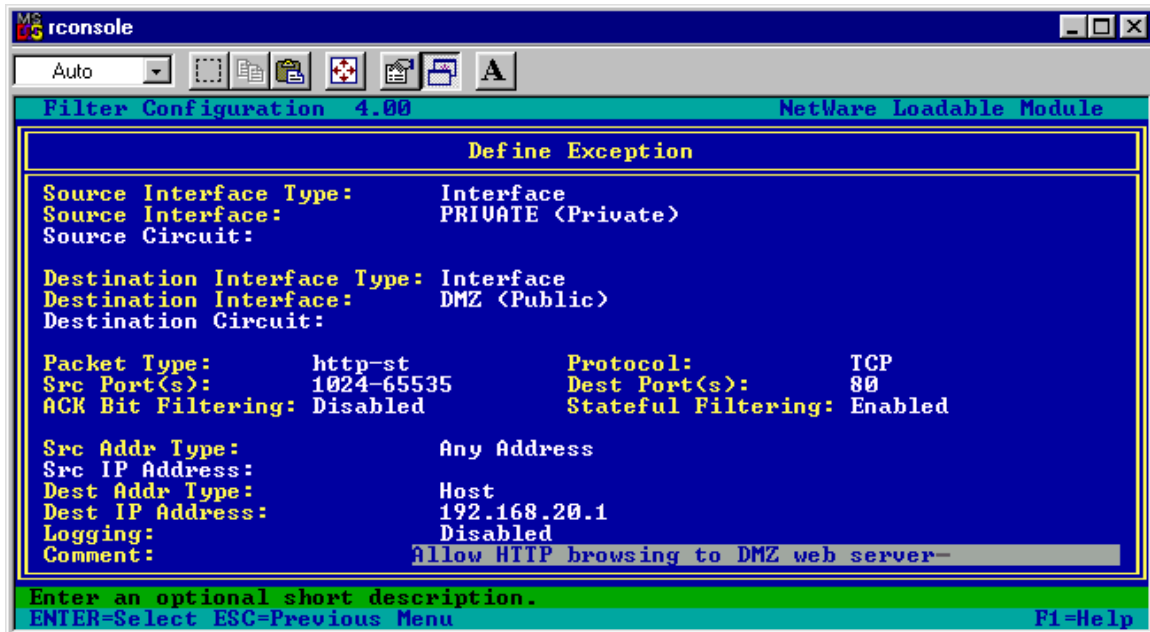


Figure 9-6 - Filter Exception to Allow HTTP to DMZ Web Server from Internal LAN

The filter exception shown in Figure 9-6 allows a PC on the internal LAN to browse to the web server in the DMZ at IP address 192.168.20.1.

- Source Interface: PRIVATE
- Destination Interface: DMZ
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Ports: 80
- Stateful Filtering: Enabled
- Source IP Address: Any
- Destination IP Address: <web server DMZ IP address>

---

**Note** With the filter exceptions shown in this example, even the HTTP Proxy could not be used to browse to the web server, because the default filter exceptions set up by BRDCFG for the DMZ were deleted. In order to use the HTTP Proxy to browse to the web server, change the filter exception above to call out a source interface of 192.168.20.254 (the BorderManager server DMZ IP Address).

---

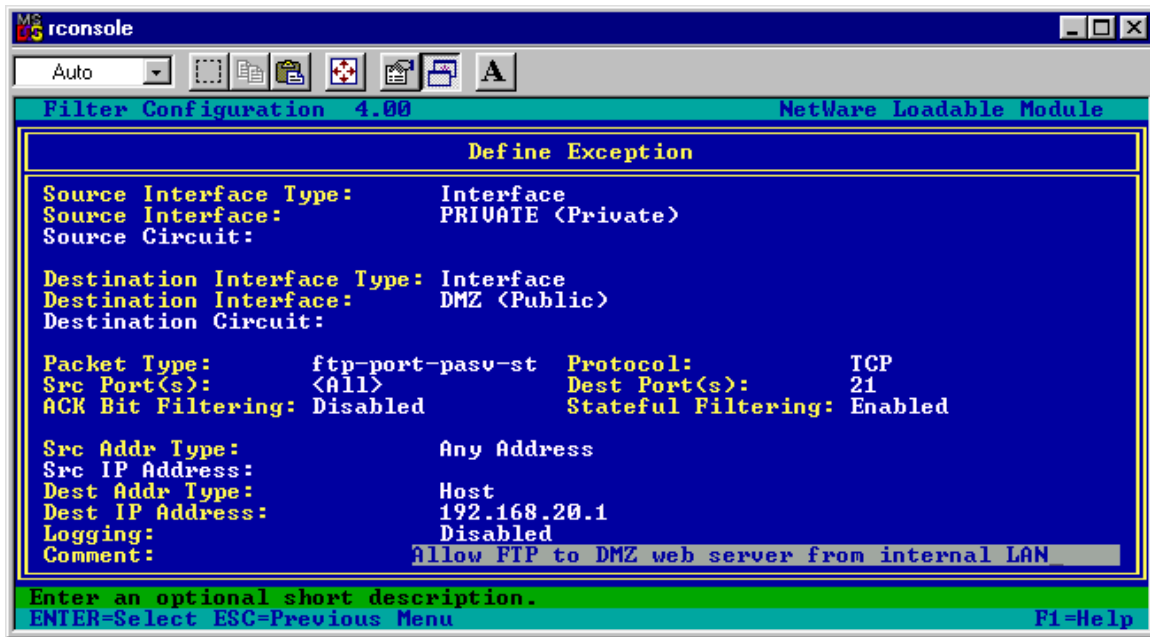


Figure 9-7 - Filter Exception to Allow FTP to DMZ Web Server from Internal LAN

The filter exception shown in Figure 9-7 allows a PC on the internal LAN to make FTP connections to the web server in the DMZ at IP address 192.168.20.1. The purpose is to use FTP to push updates to the web server.

- Source Interface: PRIVATE
- Destination Interface: DMZ
- Protocol: TCP
- Source Ports: 1024-65535
- Destination Ports: 21
- Stateful Filtering: Enabled
- Source IP Address: Any
- Destination IP Address: <web server DMZ IP address>

---

**Note** The built-in ftp-port-pasv-st definition was used for this example in order to save a bit of space in the book. An alternative is to use the examples shown earlier in this book for inbound FTP through Static NAT.

---

## Most Security - Completely Customized Filter Exceptions

If you are the type of person who wants complete control of every packet going across your BorderManager server, this section is for you. You are the type of person who understands every application that wants to access the Internet, and are willing to open one, (or more), custom exceptions for each application to allow it. You have no problem deleting ALL the custom exceptions and dealing with the consequences. You probably have packet sniffing software and know how to use it. (If you don't, go visit <http://www.ethereal.com> for some free software). You are comfortable with using the SET TCP IP DEBUG=1 command and the SET FILTER DEBUG=ON command. You understand how to block certain hosts with dummy static routes. You realize that you will have to set up custom exceptions for each proxy that you enable, since none of them are going to work if you delete the default exceptions. Your users are under your thumb and do not get Internet access that you choose not to allow. Finally,... you may have a fresh installation of BorderManager 3.7, and did not select any proxies during the installation, so you have no filter exceptions at all to start with.

You can also expect to have to do some work for each and every new application that is to be allowed in or out of the network AND for any non-standard web sites that redirect the browser to a port number other than port 80 or 443.

As a start, delete ALL of the default exceptions added by BRDCFG. You now will not be able to send or receive any data.

If you wish to bypass the proxies, using static or dynamic NAT, you can use the filter exceptions as shown earlier. If you wish to use the proxies, you must set up individual filter exceptions as follows:

## Allow Outbound HTTP for the HTTP Proxy Only

You will almost certainly wish to make use of the HTTP Proxy for browsing. The first step is to create a stateful filter exception to allow standard HTTP outbound, from the public IP address of the BorderManager server only.

- Source interface: Public
- Destination interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination ports: 80
- Stateful: Enabled
- Source IP Address: <BorderManager public IP address>
- Destination IP Address: Any

The HTTP Proxy can now send requests to web servers on port 80, and receive responses.

## Allow Outbound HTTPS / SSL for the HTTP Proxy Only

The first example allowed browsing, but not browsing to secure web sites, which generally use HTTPS / SSL. SSL uses TCP destination port 443, but unless you allow that type of traffic out, the default filters will block it.

Create a stateful filter exception to allow standard HTTPS / SSL outbound, from the public IP address of the BorderManager server only.

- Source interface: Public
- Destination interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination ports: 443
- Stateful: Enabled
- Source IP Address: <BorderManager public IP address>
- Destination IP Address: Any

The HTTP Proxy can now send requests to web servers on port 443, and receive responses.

---

**Note** Strictly speaking, you could also now set up a Generic TCP Proxy to use port 443 as well.

---

## Allow Non-Standard Ports Outbound for the HTTP Proxy Only

The two examples allowed browsing on both the standard HTTP and HTTPS, which will allow the HTTP Proxy to access the vast majority of web sites. However, many web sites on the Internet redirect browsers to other port numbers. For every one of these cases, you may have to find out what port numbers are being used, and decide if you want to allow those port numbers by setting up additional stateful filter exceptions.

This discussion is particularly relevant to BorderManager 3.7 installations, where only HTTP and SSL stateful exceptions are created by the install routine.

One common port number in use is 8080. Another is 8008. There are many more. If you want to allow the HTTP Proxy to access a web site using 8080, use the following parameters in a custom filter exception:

- Source interface: Public
- Destination interface: Public
- Protocol: TCP
- Source ports: 1024-65535
- Destination ports: 8080
- Stateful: Enabled
- Source IP Address: <BorderManager public IP address>
- Destination IP Address: Any

The HTTP Proxy can now send requests to web servers on port 8080, and receive responses.

In a similar manner, you may wish to allow port numbers for the other proxies:

<u>Destination Port Number</u>	<u>Proxy</u>
• 25, 110 (TCP)	Mail Proxy
• 119 (TCP)	News Proxy
• 53 (UDP)	DNS Proxy
• 7070, 554 (TCP)	RealAudio & RTSP Proxy
• 20, 21 (TCP)	FTP Proxy
• various TCP	Generic TCP Proxy
• various UDP	Generic UDP Proxy

## Blocking Chat Programs

In a previous chapter, exceptions were shown which allow various Chat programs to communicate over specific (default) port numbers. However, some of the programs contain port-scanning features designed to find other port numbers and use those instead. The problem then arises that if you want to block these programs, and you have opened almost any standard port (such as DNS), you cannot prevent the programs from establishing outbound communications.

Besides the port number scanning capability, most Chat programs have a feature to work through an HTTP Proxy. While you may be able to set up Deny Access Rules to block the sites involved, the technique shown here will work whether or not the HTTP Proxy is configured in the Chat programs.

There is another way to block these programs, though it does not involve filtering. Instead, you set up a dummy static route for one or more critical servers involved.

The Chat programs rely on one or more central servers to communicate. By discovering the IP addresses of those servers, you can redirect all outbound packets to an incorrect next hop with a static route, causing the initial communications to fail. One static route may be necessary for each communication server involved, though in some cases you may be able to redirect an entire subnet of servers with a single static route.

An NSLOOKUP program, such as Cyberkit, is useful to discover all of the IP addresses assigned for the servers involved. You can generally find NSLOOKUP programs (programs with DNS query capabilities) at <http://www.tucows.com/>.

As of this writing, here are some of the IP addresses involved for popular Chat programs

---

**CAUTION**      *You will have to constantly check these programs to see if new server addresses have been put into use.*

---



## Blocking AOL Instant Messenger (as of 02/20/2002)

AOL Instant Messenger relies on logging into a server at **oscar.login.aol.com**. All IP addresses assigned to this URL must be discovered and redirected. add a Deny URL Access Rule for HTTP Proxy to this URL.

AOL's login servers are on these subnets/addresses:

- host 64.12.161.153 (as of Feb. 20, 2002)
- host 64.12.161.185 (as of Feb. 20, 2002) - You might try redirecting the whole subnet 64.12.161.0 (255.255.255.0)

As in any example like this, the technique relies on knowing the IP address of some centralized login servers, and those addresses might change at any time. You will need to periodically check the addresses. One way to check is to try using the latest version of the AOL Instant Messaging software and see if it works.

## Blocking MSN Messenger (as of 11/29/2001)

MSN Messenger relies on logging into a server at **gateway.messenger.hotmail.com**. Add a Deny URL Access Rule for HTTP Proxy to this URL.

At the time of this writing, servers seemed to be changing, but the last time I checked were in the range 64.4.13.170 through 64.4.13.190. It will probably be necessary to constantly check for new server addresses should a redirection technique become necessary. Redirecting subnet 64.4.13.160 (255.255.255.224) will prevent traffic from reaching all addresses from 64.4.13.161 through 64.4.13.191. (Changing that subnet to 64.4.13.128 and the subnet mask to 255.255.255.128 would expand the blocking to 64.4.13.129 through 64.4.13.255). However, I was able to block all access with the Deny URL rule above, and the default filter exceptions.

## Blocking ICQ (as of 2/20/2002)

ICQ relies on logging into a server **login.icq.com** and **http.proxy.icq.com**. (Was **icq.mirabilis.com** and **login.icq.com** previously.)

Start by adding a Deny URL access rule for the URL's above, and be sure the default filters are in place. This may be adequate to block access to ICQ.

Redirect the following hosts or subnets:

- host 64.12.162.57 (as of Feb. 20, 2002) - You might try redirecting the whole subnet 64.12.162.0 (255.255.255.0)
- host 205.188.179.233 (as of Feb. 20, 2002)

Again, you must constantly try to find out what IP addresses may be in use for the login servers if redirection should become necessary.

## Blocking Yahoo Messenger (as of 11/29/2001)

Add a Deny URL Access Rule to block **msg.edit.yahoo.com/\*** for the HTTP Proxy. This, and the default filters, should prevent access to Yahoo Messenger.

However, a reader reports the following addresses in use on Nov. 29, 2001, should you want to try the redirection technique.

csXX.msg.yahoo.com Series

- 216.136.175.143-145
- 216.136.225.83-48
- 216.136.225.12

csXX.msg.sc5.yahoo.com Series

- 216.136.226.209-210
- 216.136.227.166-167

## Adding Dummy Static Routes

While it is easy to add a static route, you should realize that only certain IP addresses would work when entering the next hop IP address. If you should enter an incorrect address, the static route will be ignored, and the Chat program will NOT be redirected. The address you choose should be an internal address to be sure this technique will work.

- You must use an IP address in a directly-connected subnet. This means that if you have an internal interface with IP address 192.168.10.254 bound to it, you should use a next hop IP address in the 192.168.10.x range.
- You cannot use the loopback address 127.0.0.1
- You can use an IP address that is not actually assigned to a host. If you have no internal host at 192.168.10.2, you can use that as a next hop, as long as the 192.168.10.0 subnet is directly connected.
- Do not choose the address of your Internet router. (If you do, it will simply send the packets on to the Chat server!)

---

**Note** The information on blocking chat programs may become dated quickly as companies like Microsoft update their networks and messaging software. This example shows how to redirect traffic with a dummy static route, but do not expect that the IP addressing involved here will still be valid when you read this. You may very well find that other login server names or IP addresses are being used, and you may have to use NSLOOKUP tools or sniffer traces to find that information. Check my web site at <http://nscsysop.hypermart.net> or the Novell public forums for the latest information available on actual IP addressing being used by these chat programs.

---

## Entering a static route in NetWare

LOAD INETCFG, go to **Protocols**, **TCP/IP**, and go into **LAN Static Routing Table**. Make entries for **Network** with the network number, or **HOST**, with the Host IP Address, using a next hop of an IP address that is within a network directly attached to the BorderManager server.

Here is an example for redirecting the MSN Messenger login server login.gateway.hotmail.com at IP address **64.3.13.170**

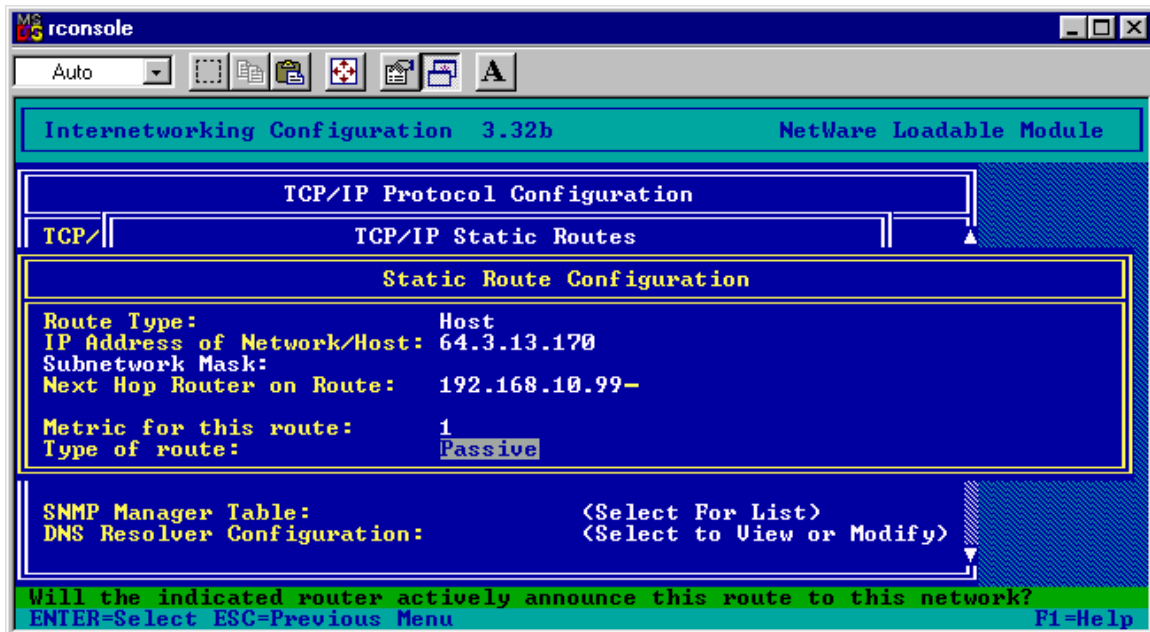


Figure 9-8 - Dummy Static Route to Redirect MSN Messenger

The static route shown in Figure 9-8 redirects requests to the MSN Messenger login server at IP address 64.3.12.170 to a next hop of 192.168.10.99. There is no host at IP address 192.168.10.99, but the 192.168.10.0 subnet is bound to the BorderManager server. (The BorderManager server has an IP address within the 192.168.10.0 subnet).

# Chapter 10 - Troubleshooting

---

There isn't a lot that goes wrong with packet filtering, assuming you have determined that you indeed have a packet filtering issue, but here are a few troubleshooting steps you can take.

## Is It A Filtering Problem?

First, you must determine if you even have a packet filter problem. Normally this means you are filtering some traffic that you want to allow. There are two methods to determine if filtering is stopping your traffic.

- Use **SET TCP IP DEBUG=1** at the server console and look for the word **FILTERING** following a line showing your packets of interest. If you are working with a very busy server, you may find it necessary to use **CONLOG.NLM** to capture the traffic to a text file (which will be saved as **SYS:\ETC\CONSOLE.LOG**) and search for the packets of interest there.
- If **TCP IP DEBUG=1** simply displays too much traffic on the server console to deal with, try **SET FILTER DEBUG=ON**, followed by one of the following (depending on if you are trying to debug a TCP, UDP or ICMP filter exception):
  - **SET TCP DISCARD FILTER DEBUG=1**
  - **SET UDP DISCARD FILTER DEBUG=1**
  - **SET ICMP DISCARD FILTER DEBUG=1**
- **UNLOAD IPFLT.NLM** at the server console. This will immediately disable all IP packet filtering, and if your application now starts working, you can bet something was being filtered. It will then be necessary to determine the port numbers being filtered and determine if you can safely add filter exceptions to allow the traffic. (Not all applications lend themselves to working through a firewall, and getting those applications to function may result in opening up your network so much that the firewall is ineffective).

---

**CAUTION** *UNLOAD IPFLT and disabling your IP filters removes firewall functionality! If you need a highly secure network at all times, you should be doing this sort of testing in a lab environment!*

---

The other kind of filtering issue involves allowing unwanted traffic that you intended to be filtering. There are two things to check if this situation is occurring.

- Be sure that you have enabled the default filter exceptions with BRDCFG.NLM and not manually removed them using FILTCFG.NLM. You can run BRDCFG.NLM again to add back default filters and exceptions.
- Check the filter exceptions in FILTCFG.NLM to ensure that you have not added an exception that inadvertently allows the traffic of interest. Be especially alert for a filter exception that allows Any to Any.
- Check to be sure that IPFLT.NLM is loaded.
- Check to see that your internal hosts and routers have a default gateway setting.

## Stateful Filter Exceptions Aren't Working

This can be especially puzzling as it appears for no apparent reason, and it is clear (from TCP IP DEBUG) that the very destination port you are trying to allow is clearly being filtered in the outbound direction. This situation occurs when applying a filter to an interface instead of an IP address or 'All interfaces'. Changing the stateful filter exception to All Interfaces works sometimes. There are patches to BorderManager 3.x that fix problems with stateful filters by updating the IPFLT31.NLM file. **Be sure to get the latest NetWare support pack on your server, because the filtering modules have been moved from BorderManager patches to the NetWare patches.**

One reason for the problem may be that the problem is not due to IPFLT31.NLM issues at all, but instead involves the server not correctly identifying the public and private interfaces as they appear in the FILTCFG menu. The problem usually occurs when a network interface card (NIC) has been removed, as in going from three network cards to two. Something 'gets confused' in how the interfaces are identified. Variations of this problem can even occur by deleting an interface definition and recreating it, such as changing a network card out for another.

The cure is likely to be recreating the SYS:\ETC\NETINFO.CFG file. Related to that may be the need to also recreate the SYS:\ETC\TCPIP.CFG file, which holds some IP address settings and the NAT definitions. The most thorough way to solve the problem is to:

1. Rename those files to NETINFO.OLD and TCPIP.OLD,
2. LOAD INETCFG, and then recreate all of the network settings again, followed by,

### 3. REINITIALIZE SYSTEM.

---

**Note** Problems in NETINFO.CFG and/or TCPIP.CFG can cause other strange and unexpected behavior, such as static NAT definitions seeming to disappear. Also, be sure to check your filter definition closely to ensure that the 'stateful' option has been selected - it is easy to overlook setting that parameter!

---

## My Filter Exception Looks OK, But My Traffic Is Still Blocked

- Check to see if the traffic is UDP, and the filter is for TCP, or vice-versa.
- Check to see if the filter exception is based on source port, instead of destination port, or vice-versa. Especially with BorderManager 2.1, which has no stateful filters, you need to set up (at least) two filter exceptions - one to allow a destination port out, and the other to allow the return traffic, often in the port range of 1024-65535.

Filter exceptions can be subtle - look very closely at the results of a TCP IP DEBUG trace to see what is actually happening. You need to pay attention to the line directly above FILTERING to determine protocol (TCP or UDP, usually), source port, destination port, and if the traffic was inbound or outbound.

You also might have changed the interface name of the private interface, and have filter exceptions that are invalid, because the exceptions are based on an interface that doesn't exist. The solution is simple: modify each filter exception to call out the new interface name, or change the interface name to match the filter exceptions.

## My Traffic is Blocked, But TCP/IP DEBUG Doesn't Show Any Discards

If TCP/IP DEBUG=1 shows inbound traffic simply being dropped, but with no indications of DISCARD, then you are probably seeing dynamic NAT implicit filtering at work. Try the command

SET NAT DYNAMIC MODE TO PASS THRU=ON

If that helps, go into INETCFG, Protocols, TCP/IP, and disable NAT Implicit Filtering.

---

**Note** The menu option for NAT Implicit Filtering makes its appearance in one of the later NetWare support packs for NetWare 4.11 and 5.0.

---

## None of My Traffic is Blocked – Filters Are Not Working

There are three typical causes here:

1. IPFLT.NLM and IPFLT31.NLM are not loaded. Be sure filtering is enabled in INETCFG, Protocols, TCP/IP, and reinitialize system. You can also LOAD IPFLT, but be sure filtering is enabled in INETCFG.
2. Your public interface does not match the interface called out in FILTCFG, Configure TCP/IP Filters, Packet Forwarding Filters, Filters. (Or that menu is blank!) This problem often occurs if you change the public NIC to a different model and a new NIC driver is automatically selected for you by NetWare. This problem might occur (for the same reason) with installation of a NetWare support pack that upgrades the public interface driver, or an in-place upgrade to a new version of NetWare.
3. You have a 'rogue' filter exception allowing all traffic. This is very easy to put in place accidentally – you simply start to create a new filter exception, then press Escape, and then press Enter to accept the default (Save Filter ). Presto! You just put a filter exception in place that allows Any IP traffic to or from Any interface! Look for the exception and delete it.



## My Filtering Doesn't Seem to Be Working Like this Book

One reason might be that your filtering action is set up 'backwards'. That is, instead of blocking all packets and then allowing by exception, you have FILTCFG set up to allow all packets and block by exception. Look in FILTCFG, Configure TCP/IP Filtering, Packet Forwarding Filters, and look at the Action menu selection. You should see "Deny Packets in Filter List". If you see "Permit Packets in Filter List" instead, you need to do the following:

1. Change the menu entry to "Deny Packets in Filter List"
2. Delete ALL the old RIP, EGP, OSPF and Packet Forwarding filters and exceptions in every menu in FILTCFG. Also delete all entries for IPX filters.
3. If using BorderManager 3.7, delete all filtering objects inside the NBMRuleContainer. You can use ConsoleOne or NWADMN32 to do this, even though the objects show up as a question mark.
4. Run BRDCFG again, to put all the basic filter exceptions back in place in FILTERS.CFG.
5. If using BorderManager 3.7, repeat the FILTSRV MIGRATE procedure.
6. Reconfigure all packet filter exceptions in FILTCFG.

This is not necessarily the only reason your filtering may not seem to be working properly.

- You may have routing issues that seem to you to be filtering issues.
- You may have additional filtering taking place at a router, especially your Internet router.
- You may have NAT issues.
- You may have BorderManager3.7, but older IPFLT.NLM and IPFLT31.NLM versions.

In general, using TCP IP DEBUG or FILTER DEBUG as shown elsewhere in this book will uncover the problem. You may want to hire a consultant who knows what he/she is doing with BorderManager to help you out.

## BorderManager 3.7 Filtering Issues

Because BorderManager 3.7 stores IP filters and filter exceptions in NDS, there are a hosts of potential problems regarding filtering that are unique to version 3.7. The following paragraphs list potential problems and workarounds.

The reader is also strongly recommended to read Novell TID 10070403, “**Novell BorderManager 3.7 Filter Configuration Frequently Asked Questions**”.

There are two important points to keep in mind when troubleshooting BorderManager 3.7 filtering issues: **Filters are read from NDS**, and **the NDS schema must be extended** in order for the filters to exist in NDS. Remember that FILTCFG still has to read filtering information out of NDS, just as the server uses FILTSRV.NLM to read filters from NDS.

Your server must have a master or read/write replica holding at least the BorderManager server object in order for filtering to function as designed.

Do not forget that you have to migrate filters into NDS to start with, using the FILTSRV MIGRATE process, either when you first install BorderManager 3.7, or after using the BRDCFG.NLM, or if you simply delete the filtering objects from the NBMRULECONTAINER container in NDS.

Finally, I have seen a number of BorderManager servers with incorrect filtering modules, including brand new BorderManager 3.7 installs no previous version of BorderManager or NetWare existed. With BorderManager 3.7 installed, your IPFLT.NLM and IPFLT31.NLM should be versions dated 2002 or later – not year 2000!

## No Filters or Exceptions Show Up in FILTCFG for BorderManager 3.7

There are two main possibilities here:

- The filters are not in NDS to read, or
- There is a problem reading the filters or filtering information. (For example, filter objects may have lost the attribute linking them to the proper server object).

The following paragraphs cover more specific symptoms and causes that lead to the problem of nothing showing up in FILTCFG.

### There is No NBMRuleContainer Object in NDS

If the schema was properly extended when BorderManager 3.7 was installed, a special container object should have been created in the same context as the BorderManager server. It has been my personal experience that the schema extensions rarely take place during a 3.7 install, and that I have to manually extend the schema afterwards.

If you do not have an NDS container named NBMRuleContainer, your schema probably was not extended successfully. First check that all servers in the Root replica ring and the BorderManager server's replica ring are up and synchronizing in NDS with no errors. Time synchronization is critical. If the BorderManager 3.7 installation did not extend the schema for you, you should be able to extend it manually with the following command from the BorderManager 3.7 server:

```
LOAD SCHEXT <.admin ID.organization> <admin password>
```

Use the fully-qualified admin ID for your NDS tree. The schema extension should be added to the NDS tree. Afterwards, Reinitialize System and load FILTCFG again, and you should have filters and exceptions.

The SCHEXT file can be found on the BorderManager 3.7 installation CD if it was not copied to the SYS:SYSTEM directory of the BorderManager 3.7 server.

### Cannot Create the NBMRuleContainer Object in NDS

The BorderManager server and at least the master of the Root replica must be running eDirectory (DS.NLM) version 8.71 or later. (I am not clear on the minimum DS version that works, but I recommend at least version 8.82.)

## **FILTSRV MIGRATE Process Gives –603 Errors**

If you tried to migrate filtering information from the FILTERS.CFG file into NDS with the FILTSRV MIGRATE procedure, and you see –603 errors (will display on the Logger screen for NetWare 6.x servers), the schema has not been extended, and you will need to extend it manually as described earlier.

## **FILTSRV MIGRATE Process Gives –608 Errors**

I have seen this, and while I am not clear on the exact cause for a –608 error, my problem went away when deleted all the objects in the NBMRuleContainer and repeated the FILTSRV MIGRATE procedure.

## **FILTSRV MIGRATE Process Gives –659 Errors**

A –659 error indicates that time was not synchronized on the server, and time synchronization will have to be achieved before the FILTSRV MIGRATE process will complete.

## **FILTSRV MIGRATE Completes With No Errors, FILTCFG Sees Nothing**

Verify that some objects got created in the NBMRuleContainer object in NDS. You can use NWADMN32 or ConsoleOne to look in the container, even though snapins to manage the objects are not available. If objects are not there, be sure NDS is healthy, a replica holding the server object exists on the BorderManager 3.7 server, and repeat the migration process.

If objects ARE in the NBMRuleContainer, you might want to try deleting them (NWADMN32 or ConsoleOne will do), and then repeating the migration process. It is possible that the server name attribute of the filters / exceptions is incorrect, and that the filter objects are not properly assigned to the server. Be careful if you have more than one BorderManager 3.7 server in the same container as the filters / exceptions will all be mixed together in the same NBMRuleContainer. If you delete all the filters and exceptions, you will have to remigrate them back on each 3.7 server.

## **Some Duplicate Filter Exceptions Exist in FILTCFG**

Duplicate exceptions in FILTCFG are one consequence of having filters and filter exceptions stored as NDS objects. It can be a problem deleting the objects from FILTCFG. You may need to use ConsoleOne or iManager to find the duplicate objects in NDS and delete them there. You may need to delete both objects and recreate, or you may be able to delete just one of the duplicates. FILTCFG was originally designed not to allow duplicate entries, before NDS was involved, and it has a problem handling duplicate entries.

## **I Have a Filter Exception with No Definition in FILTCFG**

I believe this is one possible consequence of having duplicate filtering objects, and the same procedure should be tried to fix the issue – delete at least one filtering object in NDS, and recreate if necessary.

## **Filter Changes Not Showing Up At the BorderManager 3.7 Server**

If you are making changes to BorderManager 3.7 filters with iManager, you need to have a read/write interface on the BorderManager 3.7 server.

If you make any changes to BorderManager 3.x parameters, you need a read/write interface on the BorderManager server in order for it to be notified of the change and read it from NDS.

## **FILTSRV Returns a –6001 Error When Editing Filters in FILTCFG**

You will see this error occur on the server console (NetWare 5.x) or Logger screen (NetWare 6.x) if you try to manipulate certain filter exceptions in FILTCFG. Classic examples of this are trying to modify or delete a problem exception that shows up with a blank definition, one that keeps reappearing after you have deleted it.

This error seems to be caused when you try to modify an packet type in FILTCFG – the definition of the packet filter itself, not the filter exception. I think modifying the definition creates a new object in NDS, but FILTCFG sees two identical definitions and does not know how to handle the situation.

Ultimately, you will not be able to correct the problem with FILTCFG. You will have to delete the problem objects in NDS. If you have a good FILTERS.CFG file, your easiest course of action is to delete all the objects in the NBMRuleContainer container, and remigrate the filters and exceptions with a FILTSRV MIGRATE procedure. An alternative would be to use ConsoleOne or DSBROWSE –A, and find the problem objects in NDS and delete them.

If you see that you have made a mistake when you created a custom packet type, you would probably be best to create a new version instead of trying to modify the existing version. Once you have the new packet type in your filter exception, you should be able to go back into the packet type definitions list and delete the incorrect version.

## **FILTCFG Sees Filter Exceptions With Blank Packet Type**

See the previous tip on –6001 errors. If you see a filter exception with a blank definition and a –6001 error appears on the server

console or Logger screen, you definitely have a problem that is going to require you to manipulate objects in NDS.

### Deleted Filter Exceptions Keep Reappearing in FILTCFG

See the previous tip on –6001 errors. The problem is typically seen when using FILTCFG and deleting a filter exception. Everything seems fine until you exit and restart FILTCFG, at which point the deleted exception comes back. You will also see –6001 errors on the server console or Logger screen each time you try to delete the problem exception.

If you do not see a –6001 error, the problem may be related to slow NDS synchronization and NDS communications issues. (The object may not be deleting from all replicas, and re-synching back to the BorderManager server from another server). Be sure the BorderManager server has a read/write or master replica holding its own server object, NBMRuleContainer and license objects.

### Server Console Shows “NWDSPutAttrVal returned with error –5”

If you see a message NWDSPutAttrVal returned with error –5 every few minutes on the server console, you might have an incorrect entry for your server in the server’s SYS:\ETC\HOSTS file. Be sure the HOSTS file has the correct private IP address and server name.

Example:

192 168 10 252 BORDER1
------------------------

### Every Time My Server Starts, I See a FILTSRV Error

If you see the message “Error while parsing file SYS:\ETC\BUILTINS.CFG” and when BorderManager 3.7 boots, don’t worry about it. It is essentially a cosmetic error.

## **More Objects Appear in NBMRuleContainer Than I Have Filters or Exceptions**

If you perform multiple FILTSRV MIGRATE processes without first deleting all objects in the NBMRuleContainer container, you will create multiple NDS objects for each filter and exception. This may not cause a problem, but it cannot be good. You should always delete all objects in NBMRuleContainer before performing the FILTSRV MIGRATE procedure.

## **ACK Bit Filtering Not Working**

Applies to BorderManager 3.7: If you enable the ACK bit in a filter definition (service type) using the iManager interface, it may not have been stored correctly to NDS. Try modifying the filter definition using FILTCFG.

## My 3.7 Server Is Driving Me Nuts with Custom Filter Exceptions! How Can I Make It Like BorderManager 3.6?

If you would prefer to return to the way that previous versions of BorderManager worked, where all outbound IP ports were allowed from the public IP address by default, you can add at least these two filter exceptions. I recommend you use FILTCFG to do so. If you want to put in all the filter exceptions that BorderManager 3.6 and earlier versions did, you can also run BRDCFG again, and then remigrate filters and exceptions with a FILTSRV MIGRATE procedure.

### Allow All Outbound IP from Public IP Address

- Source Interface: Public
- Destination Interface: All
- Packet Type: Any (will show protocol as IP)
- Source IP Address: <your BorderManager public IP address>
- Destination IP Address: Any Address
- Comment: Allow all outbound IP traffic from Proxies

### Allow High Port Return Traffic to Public IP Address

I highly recommend you enable ACK bit filtering here. See the Advanced chapter on this subject!

- Source Interface: Public
- Destination Interface: All
- Packet Type: Dynamic/TCP
- Source IP Address: Any Address
- Destination IP Address: <your BorderManager public IP address>
- Comment: Allow all inbound high ports to the public IP address

---

**CAUTION**      *There are definitely security concerns with this exception, and you need to read the Advanced chapter about this. Using ACK bit filtering here will help immensely.*

---



## Some BorderManager 3.7 Proxies Work, Some Don't

BorderManager 3.7 has completely different default filter exceptions from earlier versions. A quick test to see if you have a filtering issue is to a) not enforce rules, and then b) UNLOAD IPFLT to disable IP filtering. If the proxy works then, you may need an access rule change, or a new filter exception, or both.

## No Option in iManager for NBM Access Management for BorderManager 3.7

The plugins should have been automatically added to the server when installing BorderManager 3.7 on NetWare 6.0. You may also need to assign the NBM Access Management role to your user ID with Role Management in iManager. No Option for NBM Access Management Option in iManager

If the schema extensions exist, and you can see filtering information in FILTCFG.NLM, you may need to install iManager on your NetWare 6.0 server.

If you enter iManager, but do not see an option for managing BorderManager filtering (no NBM Access Management option), you might have one of the following issues:

1. The schema was not extended. See the previous troubleshooting tip on manually extending the schema.
2. The BorderManager plugins, consisting of emFrame JAR and XML files, were not copied to the iManager server. The plugins should have been automatically added to the server when installing BorderManager 3.7 on NetWare 6.0. See the following tip on adding NBM Access Management to a NetWare 6 server.
3. Your user ID may not have been set up to manage that option. This involves Role-Based Management, and delegating responsibility for the NBM Access Management role to your user ID, which is done in iManager. See Novell TID 10070403, "Novell BorderManager 3.7 Filter Configuration Frequently Asked Questions".

## Adding NBM Access Management Option to a Non-BorderManager Server

BorderManager 3.7 provides the option of managing IP filtering information in a browser interface within the iManager management

system. However, iManager is offered by Novell only on NetWare 6 servers, while BorderManager 3.7 might be installed on a NetWare 5.1 server.

Should you have another server running NetWare 6, or you simply do not want to have iManager services running on your BorderManager 3.7 server, you can manually add the BorderManager NBM Access Management option to another iManager server.

First, the schema must have been correctly extended in the NDS tree for BorderManager 3.7 filtering support. You should already have a container called NBMRuleContainer in the same context as the BorderManager 3.7 server. See the earlier tip on getting the NBMRuleContainer to show up by manually extending the schema, if necessary. There is a minimum requirement for NDS versions on at least the BorderManager server and the master of the root replica.

Next, iManager must already be installed and functional on a NetWare 6 server.

Finally, copy all of the BorderManager 3.7 emFrame support files.

If you need to reinstall the plugins, or install the plugins to a different server than the BorderManager 3.7 server, follow this procedure:

1. From the BorderManager 3.7 CD, copy the contents of the FILTEMF directory (should see directories named HELP, IMAGES and WEBINF) into the NetWare 6 server directory called SYS:WEBAPPS\EMFRAME. .
2. Flag all the files you just copied as normal. Otherwise, they will be flagged read-only, and that can cause problems patching the server.

Reboot the NetWare 6 server, and the BorderManager NBM Access Management Option should appear as a new choice in iManager. If it does not, see Novell TID 10070403, "**Novell BorderManager 3.7 Filter Configuration Frequently Asked Questions**".

## The BorderManager 3.7 HTTP Proxy Doesn't Work for Some Web Sites

If you set up BorderManager 3.7 as a fresh installation on a new server, and selected the HTTP Proxy during installation, you will have filter exceptions that allow outbound HTTP and SSL, using standard port numbers.

However, there are many web sites on the Internet that do not use standard port numbers (80 for HTTP and 443 for SSL). These sites can be recognized if you look at the status bar of your browser while trying to contact them as the URL will have a colon followed by some port number. For instance, you might see something like

`http://www.oddwebsite.com:9000/index.html`

Note the :9000 in the URL shown. This means the HTML code is redirecting the browser to use TCP destination port 9000 when trying to go to that URL. If you have a standard BorderManager 3.7 configuration, you do not have an exception allowing outbound TCP port 9000, and you will have to add one if you wish to browse to this web site.

This can be quite a problem, since you never really know what port number a web site wants to use until you try it. There are some 'typical' non-standard port numbers you may see, and so by adding exceptions for these port numbers, you may cover a great deal of the problem web sites:

- 81
- 8008
- 8009
- 8080
- 9000

BorderManager versions prior to 3.7 did not have this problem, because all port numbers were allowed from the public IP address by default. Running the BorderManager 3.6 version of BRDCFG (which shipped with BorderManager 3.7) will put the old default exceptions into FILTERS.CFG, and you can then use the FILTERS MIGRATE process to use those exceptions with BorderManager 3.7.

## Server ABENDS if I Unload IPFLT Manually

This is a problem with certain versions of IPFLT.NLM and IPFT31.NLM. The problem was fixed in the IPFLT1.EXE patch, which contained July 2002 versions of each of these NLM's. The problem was also fixed with the versions of IPFLT.NLM and IPFLT31.NLM contained in the BM36SP2A.EXE or BM37SP1.EXE patches.

The symptom is that if you UNLOAD IPFLT, first IPFLT31.NLM unloads (which is normal), but IPFLT.NLM never unloads. If you then try to manually reload IPFLT31, or unload FILTSRV, or try to manually unload IPFLT again, the server ABENDS. A workaround is to disable filtering in INETCFG, PROTOCOLS, TCPIP (also in IPX), and Reinitialize System.

## Clustering Doesn't Work With Filtering Enabled

By default, IP filtering automatically filters network and broadcast addresses, regardless of the filter exceptions you might configure in FILTCFG. This can cause problems with cluster communications depending on the server addresses involved.

The cause is that the node-node cluster keepalive packets are sent out using the network address (not the network broadcast address), which in the case of a class C network is an octet ending in .0. For example, if you have a server with an address of 192.168.10.254, and subnet mask 255.255.255.0, the network address is 192.168.10.0 and the local broadcast address is 192.168.10.255. In this case, clustering will send packets addressed to 192.168.10.0. If IPFLT31 is loaded on a server, the packets addressed to 192.168.10.0 will be filtered, regardless of any exception you try to put in.

### Easy Fix – SET Statement to Change Filtering Behavior

There is a SET statement to change this default filtering action:

```
SET FILTER SUBNET BROADCAST PACKETS=OFF
```

The SET statement should allow the node-node clustering communication packets to get by the packet filtering process.

## Complex Fix – Change Clustering Address

The workaround here is to dual-home the cluster servers, giving them a non-class C subnet for the cluster addresses, while retaining the existing network address for non-clustering communications. As an example (all addresses shown are intended for the private interface):

Node one – uses primary binding of 192.168.10.193 /255.255.255.0

Node two – uses primary binding of 192.168.10.194 /255.255.255.0

The addresses above will not be useable for clustering if filtering is enabled. Add new bindings, similar to the following:

Node one – add another binding of 192.168.11.193 / 255.255.255.192

Node two – add another binding of 192.168.11.194 / 255.255.255.192

In ConsoleOne, designate the 192.168.11.x addresses as the cluster node addresses. Now the node-node keepalive traffic will be sent on network address 192.168.11.192, and IPFLT31 will not filter it.

## APC PowerChute Software Doesn't Work With Filtering

Applies to APC software designed to allow multiple file servers to connect to a single UPS. I have seen the same issue with IPFLT31 filtering APC software updates on the network address as with the clustering communications described earlier. The same fix or workaround is possible. You can use the SET statement described in the previous tip, or you can configure the APC software to use a non-class C binding in a dual-homed server environment.

## NAT Quit Working

The situation I want to cover here generally occurs if you have renamed the public interface at some point. The symptoms are (generally) as follows:

1. Dynamic NAT isn't working even though you have it correctly set up on the public IP binding.
2. Static NAT (for inbound traffic) isn't working either, even though it is correctly set up.

By 'correctly set up', I mean that you look in INETCFG and everything is fine. Yet, if you use SET TCP IP DEBUG=1 to look at your IP traffic, you can see that packets are being forwarded and NOT being NAT'd. (You see packets going out, with the private IP address, and of course no replies come back). Sometimes NAT.NLM will not even be loaded, though it usually is.

The cause is duplicated interface definition in the SYS:\ETC\TCPIP.CFG file. The cure is a simple edit of that file. Here is an example.

## BAD TCPIP.CFG FILE EXAMPLE

```
AutonomousSystem 0
Protocol rip on {
  Interface {
    Address 192.168.10.254
    Port PRIVATE_EII
    Status on
    Cost 1
    Poison off
    SplitHorizon on
    UpdateTime 30
    GarbageTime 120
    ExpireTime 180
    OriginateDefault off
    Version ripI
    Mode normal
  }
  Interface {
    Address 10.70.0.107
    Port PUBLIC_EII
    Status on
    Cost 1
    Poison off
    SplitHorizon on
    UpdateTime 30
    GarbageTime 120
    ExpireTime 180
    OriginateDefault off
    Version ripI
    Mode normal
  }
  Interface {
    Address 10.70.0.107
    Port PUBLIC_EII
    Status on
    Cost 1
    Poison off
    SplitHorizon on
    UpdateTime 30
    GarbageTime 120
    ExpireTime 180
    OriginateDefault off
    Version ripI
    Mode normal
  }
}
Protocol egg off {
}
Protocol ospf off {
  Interface {
    Address 192.168.10.254
    Port PRIVATE_EII
    Status on
    Cost 1
    AreaId 0.0.0.0
    Priority 1
```

```

        RetransmissionInterval 5
        TransitDelay 1
        HelloInterval 10
        RouterDeadInterval 40
        Nbma {

PollInterval 120

Neighbor {

}

    }
    Interface {
        Address 10.70.0.107
        Port PUBLIC_EII
        Status on
        Cost 1
        AreaId 0.0.0.0
        Priority 1
        RetransmissionInterval 5
        TransitDelay 1
        HelloInterval 10
        RouterDeadInterval 40
        Nbma {

PollInterval 120

Neighbor {

}

    }
    Interface {
        Address 10.70.0.107
        Port PUBLIC_EII
        Status on
        Cost 1
        AreaId 0.0.0.0
        Priority 1
        RetransmissionInterval 5
        TransitDelay 1
        HelloInterval 10
        RouterDeadInterval 40
        Nbma {

PollInterval 120

Neighbor {

}

    }
    Interface {
        Address 192.168.10.254
        Port PRIVATE_EII
        Type lan
        RouterDiscovery no
        SolicitationAddress multicast
        NATStatus Disabled
    }
}

```

```

Interface {
  Address 10.70.0.107
  Port PUBLIC_EII
  Type lan
  RouterDiscovery no
  SolicitationAddress multicast
  NATStatus Dynamic
}
Interface {
  Address 10.70.0.107
  Port PUBLIC_EII
  Type lan
  RouterDiscovery no
  SolicitationAddress multicast
  NATStatus Disabled
}
ForwardIPSourceRouting off
NATFiltering off

```

The problem is right at the end. Notice the last two "Interface" entries - there are TWO ENTRIES for an interface named PUBLIC\_EII with address 10.70.0.107. In this example, there are no static NAT entries, but if there were, they would be easily seen in the first of these last two entries. Note that the first entry has "NATStatus Dynamic" while the second has "NATStatus Disabled".

In this example, outbound traffic was not being dynamically NAT'd. INETCFG would see the setting for the first PUBLIC\_EII interface and change NAT to dynamic or disabled just fine, whatever you put in. However, the SECOND entry would never show up in INETCFG, and when INITSYS.NCF ran (in AUTOEXEC.NCF) or if a Reinitialize System command were used, NAT would be enabled by the first entry, and then immediately disabled again by the second entry.

(There are also other duplicate entries in for the public interface in the routing protocols sections of the file. These don't happen to be causing an issue in this example, but should be deleted in the same way as described below).

## Fixing the Problem

The cure is very simple. First, make a backup copy of the file! Then, use a text editor, like Notepad, to delete the last Interface entry. Here is the last part of the above TCPIP.CFG file as it should be.

```

< first section of this file not shown...>
Interface {
  Address 192.168.10.254
  Port PRIVATE_EII
  Type lan
  RouterDiscovery no
  SolicitationAddress multicast
  NATStatus Disabled
}
Interface {

```



```
Address 10.70.0.107
Port PUBLIC_EII
Type lan
RouterDiscovery no
SolicitationAddress multicast
NATStatus Dynamic
}
ForwardIPSourceRouting off
NATFiltering off
```

## NAT Works, but Intermittently, and Communications are Inconsistent or Strange

There is a problem with the installation script for BorderManager 3.6. Regardless of any choices you make, older versions of some critical files will be installed, resulting in inconsistent or failing communications, particularly in regard to NAT. If you have BorderManager 3.6 installed, and NAT.NLM version 1.03 is running, you need to install or re-install the latest NetWare support pack. Some very odd symptoms have been seen with the mixed-up versions of NLM's involved – don't even both troubleshooting until you reapply the support pack as the symptoms are quite variable.

## Some, or All of My Traffic Is Blocked, Even Proxies

### BorderManager 3.7

BorderManager 3.7 does not enable filter exceptions for proxies as BRDCFG did for versions prior to 3.7. Some exceptions will be put in place during a fresh BorderManager 3.7 installation, if you selected certain proxies at that time. If you did not select any proxies, you will not have any filter exceptions at all, and no traffic will be allowed into or out of the Public interface. You will simply have to add filter exceptions (as shown in this book) to allow the proxies to work.

### Any BorderManager Version

Check to see if you have applied the default filters twice, once to the public interface and again to the private interface. If you have done this, your default filter exceptions may look fine, but you will have two sets of filters (not exceptions) blocking traffic first to the Public interface and again to the Private interface. In this case, you should delete all the filters using FILTCFG.NLM (you might also need to

delete any incorrect filter exceptions), and then run BRDCFG.NLM again.

## The Application Keeps Changing Port Numbers

Some applications simply don't work well through a firewall and expect that connections can be established in both directions using any port in the range of 1024-65535. A typical symptom might be that both source and destination port numbers are not the same with each attempt of launching the application. Your choice: don't allow the application or don't filter your network. Often these programs also don't work with a NAT configuration either.

## Stateful Filters or TCP/IP Communications Work, But Quit Working or Are Inconsistent

You probably need to get an updated version of IPFLT31.NLM, and possibly TCPIP.NLM. Check the Minimum Patch List and Novell Public Forums at <http://support.novell.com/> to find out what the latest patch name is. Another possibility is that your server has run out of resources necessary to support stateful filtering - check the readme file from your patches closely to ensure you have made the proper settings.

## My Port Numbers Are Really Weird!

- When using TCP/IP DEBUG you start seeing port numbers that simply don't come close to any of the examples shown in this book.
- You are seeing port numbers reported that are over (WAY over) 65535. In some cases, you may see port numbers like 5356072435! Here is an example:

```
Tcp IP Debug set to 1
      LOOPBACK:pktid:44733 10.1.1.1->10.1.1.1 ttl:128 (UDP)
UDP:Source Port:109903872Destination Port:109903872
RECEIVE:pktid:44733 10.1.1.1->10.1.1.1 ttl:128 (UDP)
UDP:Source Port:109903872Destination Port:109903872
LOCAL:pktid:44733 10.1.1.1->10.1.1.1 ttl:128 (UDP)
UDP:Source Port:109903872Destination Port:109903872
Discard Incoming: cause(FILTERING), reason(5)
LOOPBACK:pktid:44989 10.1.1.1->10.1.1.1 ttl:128 (UDP)
UDP:Source Port:109903872Destination Port:109903872
RECEIVE:pktid:44989 10.1.1.1->10.1.1.1 ttl:128 (UDP)
UDP:Source Port:109903872Destination Port:109903872
LOCAL:pktid:44989 10.1.1.1->10.1.1.1 ttl:128 (UDP)
UDP:Source Port:109903872Destination Port:109903872
Discard Incoming: cause(FILTERING), reason(5)
LOOPBACK:pktid:45245 10.1.1.1->10.1.1.1 ttl:128 (UDP)
UDP:Source Port:109903872Destination Port:109903872
RECEIVE:pktid:45245 10.1.1.1->10.1.1.1 ttl:128 (UDP)
```

You have a version of TCPIP.NLM that is incorrectly reporting the port numbers. This problem was especially prevalent with NetWare 4.11 and some of the early service packs, and is still present in a number of different versions of TCPIP.NLM as of this writing. You may be able to get an updated version of TCPIP.NLM from <http://support.novell.com/> that does not have the problem. It is best to check in the Novell Public Forums as well as the Minimum Patch List to find out what the latest TCPIP.NLM patch is called.

---

**Note** As of this writing, I have become resigned to accepting the problem, since I have seen it with so many versions of TCPIP.NLM. In practice, it causes few issues, because I can infer the outgoing port numbers from the response traffic I see coming back. In addition, the problem does not affect static or dynamic NAT traffic.

---

## FTP-PORT-PASV-ST Stateful Filter Doesn't Work in BorderManager 3.5

There was a bug with the version of the IPFLT31.NLM modules shipped with BorderManager 3.5. I have set up BorderManager 3.5 servers where the filter worked, and others where it did not. The bug was first fixed in the BM35F2A.EXE patch, and the BM35SP1.EXE patch. For several service packs, the updated IPFLT31.NLM modules have been contained in the NetWare support packs instead of in BorderManager patches. Check the Minimum Patch List and Novell Support Connection Public Forums at <http://support.novell.com/> to find out what the latest available BorderManager 3.5 patch is called. Otherwise, set up filter exceptions (as you would have to do with BorderManager 2.1). Allow outbound TCP ports 20-21, source ports 1024-65535. Add another filter exception to allow inbound TCP destination ports 1024-65535, source ports 20-21.

## POP3-ST Stateful Filter Doesn't Work in BorderManager 3.5

There is a bug with the version of the POP3-ST filter definition shipped with BorderManager 3.5. The Stateful parameter was not enabled in the built-in POP3-ST filter definition. You can simply create your own POP3(new)-ST filter definition that enables the stateful parameter, or try manually editing the SYS:\ETC\BUILTINS.CFG file. (See the explanation in the Odds & Ends chapter, page 315 on this).

## All IP Traffic Quits Working After Some Time

You probably need to get an updated version of TCPIP.NLM. Check the Minimum Patch List and Novell Public Forums at <http://support.novell.com/> to find out what the latest patch name is.

You may need to increase the Maximum Physical Receive Packet size setting to allow for larger packets - in some cases up to 4224 bytes! If you have settings below 1600 or so, you may find odd behavior like small files transferring but not larger ones. This setting is particularly required for network cards using Intel chipsets on NetWare 5 servers after applying Service Pack 3.

Also, with versions of TCPIP.NLM 5.31a or similar, try SET TCP IP MAXIMUM SMALL ECBS=65534. (This setting should NOT be needed on TCPIP.NLM versions later than 5.31 – 5.32y is fine without it, as is 5.52).

In general, check these items:

- Get latest TCPIP patch (in the latest NetWare support packs)
- Use SET TCP IP MAXIMUM SMALL ECBS=65534 (only necessary for **older** TCPIP.NLM version 5.31a)
- Get the latest PROXY.NLM patch
- Get the latest IPFLT31.NLM patch (in the latest NetWare support packs)
- Get the latest NAT.NLM patch (in the latest NetWare support packs)

## My Application Works For Me, But Not For My Friend Outside The Firewall

This is a typical problem with certain chat-type software, and could be with any application that tries to establish a direct host-to-host connection between two PC's with one of them behind a firewall. Outbound connections may work, but not the reverse. This type of application is likely to require a static NAT configuration for each internal host. Check the IP traffic using SET TCP IP DEBUG=1 to see if inbound connections are trying to be made on random high ports.

Some programs are simply not suited to working through firewalls as you would have to effectively open all ports up in order to get the program to work.

Some programs will work best using a SOCKS gateway. If the application has a SOCKS option, try using that before going to great lengths setting up filter exceptions.

## I Can't Filter Traffic That Brings Up My Dial-Up Connection!

Well... you're right. Unfortunately, you can't, due to the way that Novell works with dial-up and filtering. First, a dial-up link is opened, and THEN filtering is activated because the link is opened before the packet even gets to the router. About the only way around this problem is to get a dial-up router upstream from the one connected to your BorderManager server, and activate some filtering on that router. This problem is inherent to how the Novell TCPIP stack is written and would require a fundamental change to the coding. So in short - unless something major has changed with the TCPIP stack, you are wasting your time trying to keep those dial-up WAN links from coming up by using filtering techniques.

One reason for a dial-up link coming up periodically on a NetWare 5.x server is multicast traffic advertising the public IP address for NCP services. This is a common problem with BorderManager servers since NCP tries to use the first IP address bound. (First IP address showing in the interface sections in the SYS:\ETC\TCPIP.CFG file, actually). It is always a good idea on a BorderManager server to SET NCP INCLUDE IP ADDRESS = <your private IP address> so that NCP only uses the private IP address. This will cure a variety of problems, including stopping attempted multicast advertising on the public interface.

---

**Note** The NCP Include parameter was introduced with one of the later service packs for NetWare 5.x.

---

## Can't PING the Private IP Address of the BorderManager Server over Client-Site VPN

This is NOT a filtering issue. This is an issue related to NAT, which should be fixed with NAT version 6.00d or later.

---

**CAUTION** *If you should see NAT version 1.03 in use, you need to reapply the latest NetWare support pack, because you have probably experienced the install bug with BorderManager 3.6 back-revving certain critical files.*

---

However, if you do not have that version of NAT, or still find the problem with that version or later, there is an old workaround. On the public IP binding, be sure you have at least static NAT enabled. (Static and Dynamic is also OK, if you are already using dynamic NAT but not static). Add a static NAT mapping of the private IP address to itself, ignore the error message that comes up, and reinitialize system. You should then be able to ping the private IP address over Client-to-Site VPN.

## No VPTUNNEL Interface in INETCFG for Site-to-Site VPN Slave Server

The VPTUNNEL interface will not appear in INETCFG on a slave server until the slave has communicated with the master server to get configuration data.

In order to communicate with the master server, the packets must not be filtered, including the SKIP protocol packets. BorderManager 3.7 servers do not include default exceptions that allow all the required traffic. See the VPN filter exception examples in the inbound filter exception examples chapter. One symptom of a filtering problem (which might also be at the ISP) is repeated messages in the CSAUDIT log stating that SKIP construction failed.

Another reason that SKIP might appear to be failing includes the two servers being more than one hour off from each other in universal (UTC) time. Check the time on each server.

Finally, once filtering and time issues have been fixed, you may want to do two more things to speed up the process of getting the VPN configuration to initialize:

1. Reinitialize System (on master and slave servers)
2. Synchronize All (or Synchronize Selected) in the Site-Site VPN option in NWADMN32, BorderManager Setup menu. Synchronizing the VPN servers also pushes out the protected network settings to each server, resulting in each server getting static routes to the other VPN sites.

# Chapter 11 - Odds & Ends

---

This section is intended to relay some information I have received from various sources, but for which I do not have examples. In some cases, I have not personally tried out the suggested filter exceptions.

## Other Useful Port Numbers

I have heard users of the Novell Public Forums report that they have been able to get the following services to work through BorderManager with the destination port numbers shown. I have not confirmed this information personally.

A fairly comprehensive list of port numbers can be found at <http://www.ec11.dial.pipex.com/port-num0.shtml>.

### LDAP

Try setting up a stateful filter for TCP destination port 389, and source port Any (or source ports 1024-65535). If LDAP is using SSL encryption, the port number defaults to TCP destination port 636.

### SQL

SQL traffic normally uses TCP destination port 1433

### SSH

SSH (Secure Shell) traffic normally uses TCP destination port 22.

### SOCKS

SOCKS normally uses TCP destination port 1080.

### NetWare NCP Over IP

Novell NetWare NCP uses TCP destination port 524. TCP port 524 is used by NDS and client communications when IP is being used instead of IPX.

### NDPS

See Novell TID 10056598. The broker uses TCP ports 3014, 3016, 3017, 3018 and 3019. The NDPS manager uses TCP port 3396.



Printer agents use TCP destination port 3396, and UDP destination port 3396.

## **SNMP**

SNMP uses UDP destination port 161. NDPS also uses SNMP queries to check printer status, so if you use NDPS exceptions, you probably will need SNMP exceptions.

## **SCMD**

Use TCP destination port 2302 and UDP destination port 2645.

## **SLP**

The SLP listening port is UDP destination port 427, but SLP by its nature works on multicast addresses, so keep that in mind when developing filter exceptions.

## **IPP**

IPP printing uses TCP destination port 631.

## **WEBMIN**

WEBMIN is a very useful (and free) program from [www.webmin.com](http://www.webmin.com) used to manage UNIX / LINUX hosts in a browser, and it normally uses TCP destination port 10000.

## Renaming Your Interfaces to Public and Private

---

**CAUTION** *This procedure does not apply to BorderManager 3.7 without doing additional work! The FILTERS.CFG file is not used once the server has been configured and filters migrated to NDS. The filters that are stored in NDS must be manually modified, using iManager, FILTCFG, or a process involving exporting and importing the filtering information. If you use the technique described here, you probably will need to delete all objects in the server's NBMRuleContainer container, and remigrate the filters using the FILTSRV MIGRATE procedure.*

---

You will have noticed that all of the filtering examples in this book call out interface names PUBLIC and PRIVATE. My guess is that by now, you wish YOUR BorderManager server names were also PUBLIC and PRIVATE! This section tells you how to (easily) convert the board names, without deleting and recreating the board definitions.

---

**CAUTION** *This procedure can cause major problems if you make a mistake. Be absolutely sure you have backed up the files involved before proceeding!*

---

There are only four files you have to modify to change the names. The first step is to make a backup copy of the files, in case you make a mistake and need to go back to the original settings. Make a backup copy of the following files in a directory on the server. (If the files are on the server, you can copy them back into place with TOOLBOX.NLM should you lose communications and not be able to log into the server. Toolbox is available in two versions at <http://support.novell.com/misc/patlst.htm> in the etbox7.exe and tbox7.exe files.) Another good place to back up the files is on a floppy disk.

- SYS:\ETC\NETINFO.CFG
- SYS:\ETC\NETINFO.CHK
- SYS:\ETC\FILTERS.CFG
- SYS:\ETC\TCPIP.CFG

Once you have a good backup copy, you can proceed with the following steps.

**In this example, I assume you have a public interface with a board name currently set at 3C90X\_1, and a private interface**

**with a board name of 3C90X-2. Your desire is to end up with interfaces named PUBLIC and PRIVATE.**

1. With a text editor such as WordPad (one which has a search and replace capability, unlike Notepad), edit the NETINFO.CHK file, and change whatever value is in there to zero.
2. Edit the NETINFO.CFG file. Search and replace all instances of 3C90X\_1 with PUBLIC. Next search and replace all instances of 3C90X\_2 with PRIVATE. Save the file.
3. Edit the TCPIP.CFG file. Search and replace all instances of 3C90X\_1 with PUBLIC. Next search and replace all instances of 3C90X\_2 with PRIVATE. Save the file.
4. Edit the FILTERS.CFG file. Search and replace all instances of 3C90X\_1 with PUBLIC. Next search and replace all instances of 3C90X\_2 with PRIVATE. Save the file.
5. If you were saving the files on your PC, copy them to the SYS:ETC directory on the server.
6. Make sure INETCFG is unloaded on the server.
7. UNLOAD IPFLT on the server.
8. REINITIALIZE SYSTEM on the server.
9. LOAD INETCFG, and confirm that your board names are now PUBLIC and PRIVATE, that you have the correct name for the public and private IP addresses, and that any NAT definitions are on the correct binding.
10. (Applies to BorderManager 3.7 servers only). You have modified your FILTERS.CFG file with new interface names, but the old filters and exceptions are stored in NDS. You must now delete the old exceptions in NDS, and migrate the new changes into NDS using the FILTSRV MIGRATE process. You can use iManager, ConsoleOne or NWADMN32 to delete the old filtering objects in the server's NBMRuleContainer. Then perform a FILTSRV MIGRATE procedure, and reinitialize system.
11. LOAD FILTCFG and confirm that all of your filters now call out the PUBLIC interface, and that your filter exceptions call out PUBLIC and PRIVATE as needed.

## Fixing the BorderManager 3.5 POP3-ST Definition

The Novell-supplied filter definition called POP3-ST in BorderManager 3.5 has a little problem - Novell didn't enable the Stateful attribute on this supposedly stateful filter definition! If you

look at the filter definition in FILTCFG.NLM, you will see that the filter is NOT stateful. What's more, since the filter definition is a built-in definition, you are not allowed to edit it, or replace it with another by the same name. Most people simply create a new definition with a slightly different name, like POP3A-ST, and enable the stateful attribute there.

The Novell-supplied filter definitions are in a file in SYS:ETC called BUILTINS.CFG. Editing this file is all that you need to do to fix the filter definition.

1. The SYS:\ETC\BUILTINS.CFG file is a text file flagged read-only, so you must first flag it as normal in order to edit it. Change to the sys:etc directory and execute

```
FLAG SYS:\ETC\BUILTINS.CFG N
```

That should change the status to normal.

2. Next, use a text editor (Notepad will do) to change the one line in the file for POP3 from:

```
PROTOCOL-SERVICE IP, pop3-st, pid=TCP port=110  
srcport=<All>, Stateful POP3 Service
```

to (adding the text stfilt=1):

```
PROTOCOL-SERVICE IP, pop3-st, pid=TCP port=110  
srcport=<All> stfilt=1, Stateful POP3 Service
```

**Do not add a comma** before the 'stfilt=1'.

3. Unload IPFLT.NLM and then Reinitialize System, or reboot the server.

Now you should have a POP3-ST filter definition that is actually stateful!

## Novell's FILT01A.EXE File

Novell has for some time provided a file called FILT01A.EXE that provides some predefined filter exceptions for DNS, HTTP, and FTP for both BorderManager 2.1 and BorderManager 3.x. You would replace your existing FILTERS.CFG file with one of the examples provided in the file. There are also some filter worksheets provided in various formats. The file is available from Novell's support site, but there are better examples in this book.

## Packet Filter Logging

While I don't recommend using packet filter logging, since it takes a lot of server resources, and using the debug options shown in this book is easier, it can be done. See Novell TID 10021711 for some information.

## NetWare for Small Business (SBS)

NetWare for Small Business has come bundled with Novell BorderManager for years. In that past, some versions came without the Client-to-Site VPN licenses, but otherwise, the version of BorderManager is no different than BorderManager purchased as a separate product.

NetWare for Small Business implementations often use all services installed on a single server. While this is certainly possible, it does make for a complicated installation, and the administrator has to be very careful to research patch levels for the various product suites (NetWare, BorderManager, GroupWise, Netshield, FAXWare, etc.) to be sure that they will not interfere with each other.

From a purely packet filtering viewpoint, the administrator of a SBS system should consider that services such as GroupWise running on the BorderManager server itself should be treated in the same way as for proxies. For example, GWIA running on the BorderManager server is the same as Mail Proxy running on the BorderManager server, from a packet filtering perspective. The examples show in this book for inbound traffic to proxies will apply, whereas the inbound static NAT examples will probably not apply.

# Chapter 12 - Other References

---

Check my web site at <http://nscsysop.hypermart.net> for updates!

A fairly comprehensive list of port numbers can be found at <http://www.ec11.dial.pipex.com/port-num0.shtml>. (This URL has changed recently, so you might have to go to the main site URL and browse to the port number link.)

Other sources of information on port numbers can be found at Marcus Williamson's web site at <http://www.connectotel.com/border/>.

It is always a good idea to check a manufacturer's web site for information on setting up a product with a firewall. However, I have found several occasions where the manufacturer's web site or technical support person was simply incorrect on the port numbers being used by an application. You can always see exactly what port numbers are used by making use of the TCP/IP DEBUG or FILTER DEBUG commands shown elsewhere in this book.

Well-known port numbers are described in RFC 1700 and on numerous Internet web sites. Use a web search engine to find a site detailing these port numbers.

Novell describes how to set up a number of filter exceptions in one Technical Information Document (TID) or another. Look these up using the Knowledgebase link at <http://support.novell.com> and entering a search term for the type of traffic you want to allow. Many of the TID's describe BorderManager 2.1 filter exceptions (no stateful filters).

The Novell Knowledgebase also contains many TID's on BorderManager issues or problems.

Proprietary programs may change the port numbers they use between revisions. Check the manufacturer's web site for guidance in setting up the program with a firewall.

# Index

---

---

## 6

6001 error · 293, 294  
603 error · 292  
608 error · 292  
659 error · 292

---

## A

ACK bit filtering · 21, 26, 204, 205, 226, 230, 231, 235, 237, 239, 241, 250, 252, 256, 258, 260, 265, 266, 267, 274, 296  
ackfilt · 268  
AIM · 16, 137  
altaddr · 226  
AOL · 16, 137, 155, 281  
AOL Instant Messenger · 16, 137, 155, 281  
ARP table · 77

---

## B

Blocking Chat Programs · 156, 280  
BorderManager 2.1 · 16, 17, 40, 88, 142, 151, 176, 242, 261, 262, 265, 287, 308, 316, 318  
BorderManager 3.0 · 26, 40, 52, 58, 59, 71, 151, 180, 191  
BorderManager 3.5 · 45, 172, 180, 186, 268, 308, 315  
BorderManager 3.6 · 76, 142, 164, 216, 217, 272, 296, 299, 305, 310  
BorderManager 3.7 · 18, 19, 20, 21, 23, 32, 40, 55, 56, 58, 69, 70, 73, 74, 75, 77, 78, 79, 80, 110, 113, 114, 115, 116, 118, 134, 135, 136, 149, 150, 153, 158, 164, 173, 179, 181, 183, 185, 187, 191, 213, 214, 215, 216, 219, 220, 267, 268, 269, 272, 277, 279, 289, 290, 291, 292, 293, 294, 295, 297, 298, 299, 305, 311, 314, 315  
BRDCFG · 53, 58, 59, 69, 70, 71, 73, 74, 220, 265, 267, 269, 272, 275, 277, 286, 289, 290, 296, 299, 305, 306  
BUILTINS.CFG · 78, 98, 110, 164, 268, 294, 308, 316

---

## C

Chat · 156, 280, 283  
Cisco VPN · 16, 138, 139  
Citrix · 16, 140, 141, 225, 226, 227, 228  
Client-to-Site VPN · 77, 142, 143, 144, 214, 215, 216, 217, 218, 222, 310, 317

CLNTRUST · 145  
Collexion · 234, 235  
CONLOG · 51, 52, 74, 81, 82, 91, 285  
ConsoleOne · 79, 80, 134, 135, 269, 289, 292, 293, 301, 315  
CSAUDIT · 311  
CuteFTP · 203, 229  
Cyberkit · 148, 280

---

## D

D0.NCF · 51  
D1.NCF · 51  
DEBUG · 27, 43, 51, 52, 74, 77, 81, 82, 88, 89, 90, 91, 285, 286, 287, 288, 289, 307, 318  
default filters · 24, 25, 50, 53, 55, 58, 59, 71, 73, 74, 82, 83, 84, 85, 145, 171, 199, 224, 269, 272, 278, 282, 286, 305  
default route · 24, 30, 31, 32, 34, 35, 74, 77, 226  
DHCP · 16, 191, 192, 193, 194, 195, 196  
DMZ · 21, 270, 271, 272, 273, 274, 275, 276  
DNS · 16, 40, 70, 82, 85, 137, 147, 148, 149, 150, 177, 251, 261, 279, 280, 316  
DNS Proxy · 70, 147, 149, 279  
DOMAIN · 254  
Dummy Static Route · 283, 284  
Dyn/ACK/TCP · 267  
dynamic NAT · 27, 36, 37, 40, 42, 44, 45, 48, 53, 82, 83, 138, 139, 142, 261, 277, 288, 307, 310  
Dynamic NAT · 41, 42, 43, 48, 83, 144, 301

---

## E

emFrame · 297, 298

---

## F

FILTCFG · 18, 19, 53, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 73, 74, 75, 76, 80, 82, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 117, 125, 128, 130, 134, 136, 269, 286, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 300, 305, 314, 315, 316  
FilterConfiguration · 113, 115, 122  
FILTERS.CFG · 19, 55, 56, 74, 78, 79, 80, 110, 267, 268, 289, 292, 293, 299, 314, 315, 316  
FILTSRV · 19, 20, 55, 69, 73, 78, 79, 80, 267, 268, 269, 289, 290, 292, 293, 294, 295, 296, 299, 300, 314, 315

FILTERSV MIGRATE · 19, 20, 69, 78, 79, 80, 267, 268, 269, 289, 290, 292, 293, 295, 296, 299, 314, 315  
 FILTERSV MIGRATE Procedure · 20, 80  
 FILTERSV\_BACKUP\_FILTERS · 79  
 firewall · 18, 30, 38, 45, 69, 72, 73, 97, 138, 139, 140, 141, 270, 285, 306, 309, 318  
 Firewall · 309  
 FTP · 16, 24, 45, 70, 90, 120, 151, 152, 153, 191, 203, 204, 205, 229, 230, 231, 261, 274, 276, 279, 308, 316  
 FTP-PORT-PASV-ST · 308

---

## G

Generic TCP Proxy · 197, 198, 212, 213, 234, 278, 279  
 Generic UDP Proxy · 165, 279  
 GroupWise · 12, 16, 154, 206, 232, 233, 234, 317  
 GWIA · 206, 207, 208, 209, 254, 317

---

## H

HTTP · 18, 25, 40, 50, 59, 67, 68, 69, 70, 84, 85, 86, 87, 151, 157, 158, 160, 161, 177, 184, 185, 190, 191, 199, 200, 201, 202, 213, 234, 257, 258, 273, 274, 275, 278, 279, 280, 281, 282, 299, 316  
 HTTP Proxy · 18, 25, 40, 67, 69, 70, 84, 85, 86, 87, 157, 158, 160, 161, 177, 185, 190, 199, 200, 201, 202, 234, 275, 278, 279, 280, 281, 282, 299  
 HTTPS · 59, 68, 184, 185, 200, 201, 202, 257, 259, 260, 278, 279

---

## I

ICMP · 26, 83, 89, 171, 285  
 ICQ · 16, 137, 155, 156, 282  
 iManager · 18, 19, 21, 73, 79, 80, 110, 111, 112, 113, 114, 116, 117, 136, 292, 293, 295, 297, 298, 314, 315  
 IMAP · 16, 159, 236, 237  
 IPFLT · 76, 78, 79, 80, 109, 267, 268, 269, 285, 286, 288, 289, 290, 297, 300, 315, 316  
 IPFLT31.NLM · 73, 76, 80, 88, 286, 288, 289, 290, 300, 306, 308, 309  
 IPP · 313

---

## J

JAR · 297

---

## L

LDAP · 312

Logger screen · 74, 80, 91, 292, 293, 294  
 LOGGER.TXT · 91  
 loopback · 52, 283  
 Lotus Notes · 16, 238, 239

---

## M

Mail Proxy · 70, 172, 173, 183, 206, 207, 208, 209, 210, 211, 224, 279, 317  
 Media Player · 16, 161, 162  
 Microsoft Terminal Server · 16, 188, 240, 241  
 MSN Messenger · 16, 160, 281, 284  
 multicast · 52, 303, 304, 305, 310, 313

---

## N

NAT · 18, 27, 37, 40, 42, 43, 44, 45, 46, 49, 50, 52, 77, 138, 142, 143, 214, 215, 216, 217, 218, 222, 224, 251, 253, 286, 288, 289, 301, 304, 305, 306, 309, 310, 315  
 NBM Access Management · 110, 113, 297, 298  
 NBMRuleContainer · 80, 110, 125, 134, 267, 268, 269, 289, 291, 292, 293, 294, 295, 298, 314, 315  
 NCF Files · 91  
 NCP · 146, 310, 312  
 NDPS · 312, 313  
 NETBIOS · 81  
 NETINFO.CFG · 49, 286, 287, 314, 315  
 NETINFO.CHK · 314, 315  
 NetWare 6 · 19, 52, 73, 74, 91, 110, 111, 166, 292, 293, 297, 298  
 NetWare for Small Business · 317  
 News Proxy · 70, 164, 210, 211, 279  
 NNTP · 13, 16, 17, 163, 164, 210, 211, 224  
 Notes · 238, 239  
 Novell Public Forums · 306, 307, 308, 312  
 Novell Remote Manager · 197  
 NRM · 197  
 NSLOOKUP · 148, 280, 283  
 NTP · 16, 120, 165, 166, 167, 175  
 NWDSPutAttrVal · 294

---

## P

PASV · 261  
 pcANYWHERE · 16, 24, 168, 169, 170, 189, 242, 243, 244, 245, 246, 247, 248  
 PING · 16, 52, 82, 83, 171, 310  
 POA · 232, 233  
 POP3 · 16, 159, 172, 173, 208, 209, 224, 232, 249, 250, 268, 308, 315, 316  
 port 10000 · 138, 313  
 port 1080 · 265, 312  
 port 110 · 172, 208, 210  
 port 119 · 70, 163  
 port 123 · 165, 166



port 143 · 312  
 port 161 · 313  
 port 1677 · 232  
 port 1755 · 161  
 port 1863 · 160  
 port 20 · 142, 143, 152, 205, 212, 216, 231, 265  
 port 2010 · 142, 143, 216  
 port 2037 · 212  
 port 21 · 59, 63, 70, 151  
 port 213 · 59, 63  
 port 22 · 168, 242, 247, 248, 312  
 port 23 · 28, 70, 313  
 port 2302 · 313  
 port 25 · 28, 29, 70, 182, 206, 251, 252, 254, 262  
 port 2645 · 313  
 port 3396 · 312  
 port 353 · 59, 64, 65, 214  
 port 37 · 174  
 port 389 · 312  
 port 427 · 313  
 port 443 · 59, 68, 184, 200, 201, 257, 278  
 port 500 · 138  
 port 5190 · 137, 155, 156  
 port 524 · 145, 312  
 port 53 · 70, 82, 147, 261  
 port 554 · 181  
 port 5631 · 168, 242, 245, 246  
 port 5632 · 168, 242, 243, 247, 248  
 port 631 · 313  
 port 636 · 312  
 port 66 · 117, 119  
 port 67 · 195  
 port 68 · 195  
 port 7070 · 176, 177  
 port 80 · 59, 67, 70, 85, 87, 157, 177, 190, 197, 200, 256, 277, 278, 279  
 port 8008 · 197  
 port 8009 · 197  
 port 8080 · 87, 177, 279  
 port 9090 · 181  
 Portal · 16, 198, 265  
 ports 1024-65535 · 168, 242, 246, 262, 263, 264, 308, 312  
 Private interface · 128, 129, 242, 305  
 protocol 57 · 59  
 Public interface · 75, 242, 305  
 Public IP address · 158, 164, 173, 179, 181, 183, 185, 187, 261

---

## Q

Quicktime · 180

---

## R

RCONAG6 · 212, 213, 265  
 RCONJ · 16, 212, 213  
 RDATE · 16, 174, 175

RealAudio · 16, 70, 176, 177, 178, 179, 181, 279  
 RealPlayer G2 · 176, 177  
 relay · 182, 251, 312  
 Rename · 286  
 Reverse Proxy · 45, 50, 67, 199, 205, 257  
 Role Management · 297  
 routing · 16, 18, 25, 30, 32, 35, 40, 77, 87, 289, 304  
 RTSP · 16, 70, 176, 179, 180, 181, 279

---

## S

SBS · 317  
 SCMD · 313  
 Secondary IP address · 39  
 security · 21, 26, 30, 37, 45, 50, 61, 69, 71, 72, 165, 166, 171, 201, 224, 261, 262, 265, 266, 270, 296  
 Security · 26, 44, 71, 265, 270, 277  
 Service Type · 116, 117, 118, 119, 121, 122, 125, 126, 127  
 SET FILTER DEBUG=ON · 52, 77, 89, 90, 91, 277, 285  
 SET TCP IP DEBUG · 52, 74, 76, 77, 81, 88, 277, 285, 301, 309  
 SET TCP IP MAXIMUM SMALL ECBS=65534 · 308, 309  
 SKIP · 59, 66, 214, 219, 311  
 SLP · 60, 71, 313  
 SMTP · 16, 28, 182, 183, 206, 207, 224, 251, 252, 253, 254, 262  
 SNMP · 313  
 SNTP · 16, 165, 166  
 SOCKS · 265, 309, 312  
 Spell Check · 16, 234, 235  
 SQL · 312  
 SSH · 312  
 SSL · 16, 59, 68, 184, 185, 199, 200, 201, 202, 257, 259, 260, 278, 279, 299, 312  
 stateful filtering · 17, 26, 120, 261, 306  
 Stateful filtering · 137, 138, 139, 140, 141, 142, 143, 144, 145, 147, 148, 149, 150, 152, 153, 154, 156, 157, 158, 159, 160, 162, 163, 164, 165, 166, 168, 169, 170, 171, 172, 173, 174, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190  
 static NAT · 16, 24, 26, 27, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 154, 189, 191, 192, 197, 224, 225, 226, 227, 229, 232, 234, 236, 238, 242, 245, 246, 247, 249, 251, 252, 253, 254, 255, 257, 259, 266, 270, 273, 287, 304, 309, 310, 317  
 Static NAT · 24, 45, 46, 47, 49, 50, 52, 188, 191, 199, 203, 224, 229, 230, 232, 234, 240, 255, 273, 276, 301  
 static route · 30, 277, 280, 283, 284, 311

---

## T

T0.NCF · 91  
 T1.NCF · 91  
 TCPCON · 30, 77

TCPIP.CFG · 49, 146, 286, 287, 302, 304, 310, 314, 315  
Telnet · 186  
TELNET · 16, 28, 186, 187  
Terminal Server · 16, 188, 240, 241  
TOOLBOX.NLM · 91, 314  
Transparent HTTP Proxy · 70, 157, 158  
Transparent Telnet Proxy · 70, 187  
Troubleshooting · 20, 110, 285

---

## *U*

UTC · 311

---

## *V*

VNC · 16, 189, 190, 255, 256

VPN · 13, 16, 20, 25, 27, 49, 53, 58, 59, 63, 64, 65, 66, 69, 70, 77, 138, 142, 143, 144, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 310, 311  
VPTUNNEL · 54, 219, 311

---

## *W*

Web Access · 16, 234  
WEBMIN · 313

---

## *X*

XML · 73, 297