



IxLoad Tcl API Programming Guide



Release 5.00

Part Number 913-0920 Rev. A

March 2010



Copyrights and Trademarks

Copyright © 2010 Ixia. All rights reserved.

This publication may not be copied, in whole or in part, without Ixia's consent.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

Ixia, the Ixia logo, and all Ixia brand names and product names in this document are either trademarks or registered trademarks of Ixia in the United States and/or other countries. All other trademarks belong to their respective owners.

The information herein is furnished for informational use only, is subject to change by Ixia without notice, and should not be construed as a commitment by Ixia. Ixia assumes no responsibility or liability for any errors or inaccuracies contained in this publication.



Contents

Copyrights and Trademarks	2
Technical Support.....	xvii
About this Guide	xix
Conventions.....	xx
Related Documentation	xxii
Introduction	23
Background Reading	24
Using a License Server	24
Network Setup.....	25
Configuring a Network Address on the IxLoad Development Station	26
Testing the Development Station's Routing	26
Configuring a Permanent Route to Ixia Ports	27
Setting Ixia Chassis Base Addresses	29
Backward Compatibility.....	29
Deprecated Commands.....	30
Quick Start.....	31
Windows	31
Using The Sample Tcl Scripts	32
Running the sample scripts	33
Monitoring Status and Retrieving Results	34
Unix/Linux	34
Installing IxOS Tcl	35
Installing IxLoad Tcl	35
Editing the setup_simple.tcl script	35
Running the sample scripts.....	36
Monitoring Status and Retrieving Results.....	37
API Overview	38



Tcl API Structure	39
Mandatory Objects to Complete a Script.....	40
Multi Version Support	41
General Tcl Conventions.....	41
Object Creation.....	41
Subobjects.....	42
Lists of Objects	42
Constants	43
Strings and Numbers	43
TCL API Internal Overview.....	43
Windows Overview	43
Unix Overview	45
Object Structure.....	46
Building an IxLoad Test.....	46
Step 1: Initial Overhead.....	47
Step 2: Define the TrafficFlow	49
Step 3: Define the TrafficColumn	49
Step 4: Define the NetTraffic	50
Step 5: Define ixSubscriberNetTraffic	52
Step 6: Define the NetworkGroup	52
Step 7: Define the NetworkGroup	53
Step 8: Define the NetworkRange.....	56
Step 9: Define the ixTimeline.....	57
Step 10: Prepare to Run the Test	58
Step 11: Start the Test.....	64
Stopping a Test by Pressing Enter	65
Running an IxLoad Tcl Script.....	65
Windows.....	66
Unix / Linux	66
Maximum Numbers of Scripts That Can Be Run.....	67
Modifying Older Scripts	68
API Description.....	68
Network Commands.....	69
DUT Commands.....	74
Traffic Commands.....	74
Test Structure Commands.....	75
Test Operation Commands	76
Debugging	78
Sample Scripts Shipped with IxLoad	79
Example Program.....	83
IxLoad Tcl API Commands.....	106
::IxLoad.....	109
ixChassisChain.....	112
ixCustomPortMap	115
Steps for Custom Traffic Mapping	115

ixPort	119
Submap	122
ixSubmapRange	124
ixIntRange	125
ixRepository	127
statCollectorUtils	129
ixStatCatalogItem	131
ixStatFilter	133
ixStatSpec	135
ixTest	137
ixTestController	141
ixTestControllerMonitor	148
statCollectorUtils	150
ixScriptGen	160
ixTimeline	163
ixSubscriberNetTraffic	170
ixNetTraffic	172
activityList	177
ixTrafficFlow	179
ixTrafficColumn	180
ixNetworkGroup	182
ixDut	184
ixDutConfigVirtual	186
ixDutNetworkRange	189
ixDutProtocolPortRange	192
ixDutConfigVip	194
ixDutConfigSLB	195
ixView	197
ixClientNetwork	199
ixClientTraffic	204
ixClientTrafficNetworkMapping	207
ixNetworkRange	217
ixServerNetwork	221
ixServerTraffic	226
ixServerTrafficNetworkMapping	230
statCollectorUtils	233
Internal Commands	235
ixConfig	236
ixConfigSequenceContainer	238
ixConfigSortedNamedItemList	241
Network Stack API	244
Setting the Global Plug-ins List	245
Creating the Stack	245
List Programming in the Network Stack	246
Creating the Ranges	247
Network- and Session-Specific Data	248
Network Stack Configuration in pre-4.10 Releases	250
Network Stack Configuration in 4.10 and Later Releases	251
Network Stack Object Model	253
Network Stack Object Model - Base Classes beginning with 'A-DH'	253
Network Stack Object Model - Base Classes beginning with 'DN'	253
Network Stack Object Model - Base Classes beginning with 'T'	253
Network Stack Object Model - Base Classes Beginning with 'P'	254

Network Stack Object Model - Base Classes Beginning with 'R'	255
Network Stack Object Model - Base Classes Beginning with 'Se'	256
Network Stack Object Model - Base Classes Beginning with 'St'	256
Network Stack Object Model - Base Classes Beginning with 'St' (cont.).....	257
Physical Layer	258
L1EthernetPlugin.....	259
EthernetELMPlugin.....	264
Physical Layer Example	267
Layer 2 Protocols	270
L2EthernetPlugin.....	271
L2EthernetPortGroupData.....	273
MacSessionData.....	274
MacRangeList.....	276
VlanIdRangeList.....	279
Layer 2 Example.....	284
Emulated Router Plugin	288
EmulatedRouterRangeList.....	290
Emulated Router Example.....	295
IP Plugin.....	300
IpRangePortGroupData	301
IpSessionData.....	302
IpV4V6Plugin.....	304
IP Plugin Example	313
DHCP Client and Server	318
DHCPPlugin.....	319
DHCPServerPlugin.....	363
Authentication Extension Plugins	378
DNPluginBase	379
DNRangeBase	384
WebAuthPlugin	390
Dot1xPlugin.....	408
Nacl3Plugin	429
CertInfo	469
Impair Plugin.....	482
ImpairRangeList.....	484
ImpairProfileList	490
ImpairSessionData.....	506
Impair Plugin Example.....	507
IPSec Plugin	514
IPSecRangeList	516
IPSecPortGroupData	534
IPSecSessionData.....	537
IPSecTunnelSetup	540
LOD_IPSecEapTuple	543
LOD_IPSecManualKeyingOpts	547
LOD_IPSecAssociate	551
IPSecCertificates	553
IPSec Example.....	557
PPPoX Plugin.....	564
PppoxPortGroupData.....	566
PLSessionDataBase	570
PppoxRangeList	574
PppoxAcNameList.....	590
PppoxAcMacList.....	593
PppoX Plugin Example	596
L2TP Plugin	602
L2tpPortGroupData	604

PLSessionDataBase	607
L2tpRangeList	611
L2tpLnsIpAddressList	631
PLAssociate	633
L2tp Plugin Example	634
GTP Plugin	641
GTPSPPlugin	642
GTPGPlugin	680
Global Plugins	692
FilterPlugin	693
GratArpPlugin	697
DnsPlugin	700
TCPPlugin	710
RoutesPlugin.....	721
IxLoadSettingsPlugin	728
Mobile Subscribers Plugins.....	730
MobileSubscribersPlugin.....	731
Radius Plugin.....	754
Mobile Subscribers Example	771
AppReplay	780
Objectives.....	781
Application Replay Peer Agent.....	782
Flow Definition	784
LoopBeginCommand.....	795
LoopEndCommand.....	796
Think	797
availableTosList	798
Advanced Options	800
Global Statistics.....	801
Application Test	808
Objectives.....	808
IxVerify VAAT in IxLoad	809
Initial Overhead.....	809
Adding an IxVerify Activity	810
Statistics	810
Bulk MGCP	814
API Overview.....	816
MGCP Client API	817
Objectives	817
MGCP Server API.....	823
MGCP Server Agent.....	823
Parameters	826
MGCP Client Agent.....	828
Parameters	830
Low Level Parameters	833
DNS Record.....	835
Endpoint Names	837
Media Settings	841
Commands	850
Custom Endpoint Names	854



MGCP Server Agent	856
Parameters	858
Low Level Parameters	860
DNS Updates	861
Endpoint Names	865
Custom Endpoint Names	868
Bulk MGCP Statistics	869
Bulk MGCP Client Statistics	869
Bulk MGCP Server Statistics	874
Bulk SIP	878
Overview	880
Objectives.....	880
SIP Client Commands	881
SIP Client Agent.....	881
General Settings.....	881
Content of Messages.....	882
Rules.....	883
State Machine	883
Media Settings	884
Audio Clips Pool	885
Video Settings	885
Scenarios.....	886
SIP Server Commands	886
SIP Server Agent	886
General Settings.....	887
Content of Messages.....	887
Rules.....	887
State Machine	887
Media Settings	887
Audio Clips Pool	887
Scenarios.....	888
SIP Client Agent.....	889
General Settings.....	897
Content of Messages.....	902
State Machine	907
Media Settings	910
Video Settings	920
Scenarios.....	922
SIP Server Agent.....	932
General Settings.....	939
Content of Messages.....	941
State Machine	942
Media Settings	944
Scenarios.....	945
Using Variables in SIP Fields	948
Bulk SIP Statistics	950
Bulk SIP Client Statistics	950
Bulk SIP Server Statistics.....	963
DDoS	976
API Overview.....	978
Objectives.....	978
DDoS Client Agent.....	978

DDoS Client Agent	979
DDoS Command	981
DDoS Statistics	987
DHCP	988
Overview	990
Objectives	990
DHCP Client Agent.....	994
DHCP Command List	996
Advanced Options	1004
Relay Agent.....	1007
Option.....	1010
Option Set.....	1017
Option Set Manager.....	1020
Option Choices	1021
IP Address	1023
Using Variables in DHCP Fields.....	1025
DHCP Statistics.....	1026
Effect of Options on DHCP Packet Size	1034
DNS	1035
Overview	1036
Objectives	1036
DNS Client Agent.....	1036
DNS Server Agent	1037
DNS Client Agent	1040
DNS Client Query	1043
DNS Client Advanced Options.....	1045
DNS Server Agent.....	1047
DNS Server Zone Management.....	1051
DNS Server Zone Configuration	1054
DNS Server Advanced Options	1055
DNS Server Resource Record	1057
DNS Statistics	1060
DNS Client Statistics	1060
DNS Server Statistics	1063
FTP	1066
Overview	1067
Objectives.....	1067
FTP Client Agent	1067
FTP Client Action.....	1068
FTP Server Agent.....	1068
realFileList.....	1068
FTP Client Agent	1070
FTP Client Action.....	1076
FTP Server Agent.....	1084
FTP Statistics	1087
FTP Client Statistics	1088
FTP Server Statistics	1090
HTTP.....	1092



Overview	1094
Objectives	1094
HTTP Client Agent.....	1094
HTTP Server Agent	1096
HTTP Client Agent	1100
HTTP Client Profile.....	1110
HTTP Client Action.....	1113
HTTP Server Agent.....	1118
ixCookieContent.....	1124
ixCookieObject.....	1127
ixResponseHeader	1130
PageObject.....	1135
CustomPayloadObject	1139
Supported Ciphers	1142
Using Your Own Web Pages In IxLoad.....	1146
Using Sequence Generators in HTTP Client Commands and Server Header Name=Value Fields	1148
Using System Variables.....	1149
Statistics	1151
HTTP Server Statistics	1151
HTTP Client Statistics	1158
TCP Reset Statistics.....	1175
IxLoad Statistics Interpolation.....	1176
IMAP.....	1178
API Overview.....	1179
Objectives	1179
IMAP Client Commands	1179
IMAP Server Commands.....	1180
IMAP Client Agent	1185
IMAP Commands.....	1188
IMAP Client Advanced Options.....	1195
IMAP Server Agent.....	1197
IMAP Server Advanced Options	1202
IMAP Server Config.....	1204
Using Auto-Generated Strings	1218
IMAP Statistics	1219
IMAP Client Statistics	1219
IMAP Server Statistics	1222
IPTV/ Video	1225
Overview	1227
Video	1227
IPTV	1227
Objectives	1228
Video Client API Structure.....	1229
Video Client Commands	1229
Video Server Commands.....	1232
Video Client Agent.....	1236
Commands.....	1243
Advanced.....	1258
Header	1263
Signaling.....	1264
Profiles.....	1268
Channel View	1271
IPTV Options.....	1272

Stats	1274
Video Server Agent	1277
Video Properties	1283
Advanced Options	1294
Video Config	1298
IPTV / Video Statistics	1299
IPTV / Video Client Statistics	1299
Global Stream Statistics.....	1299
Per-Stream Statistics.....	1303
Multicast and VoD Global Statistics	1308
IGMP and MLD Conditional Statistics	1310
IPTV Global Statistics	1311
RTSP Conditional Statistics	1315
Video Quality Statistics	1317
IPTV / Video Server Statistics	1328
LDAP	1332
Overview	1333
Objectives	1333
LDAP Client Commands.....	1333
LDAP Client Agent.....	1337
Command List	1341
Global Options.....	1350
Control.....	1353
Modification	1355
Attribute.....	1357
Attribute Type and Values.....	1359
LDAP Statistics	1360
Peer-to-Peer Application	1366
Overview	1368
Objectives	1368
Peer-to-Peer Application Agent Commands.....	1368
Peer-to-Peer Application Agent	1370
FlowDefinition	1372
Peer-to-peer Global Statistics.....	1378
Packet Monitor	1381
Overview	1383
Packet Monitor Server Agent	1384
PacketMonitor Server Agent	1385
PluginProperties	1387
PacketFilter	1389
ProtocolFilter.....	1395
EthernetFilter.....	1396
IPProtocolFilter	1398
ArpProtocolFilter.....	1400
TCPProtocol Filter.....	1402
UdpProtocol Filter	1404
IcmpProtocol Filter.....	1406
MACAddressRange.....	1408



IPAddressRange	1410
PortRange	1412
POP3	1413
Overview	1415
Objectives	1415
POP3 Client Agent	1415
POP3 Server Agent.....	1416
POP3 Client Agent	1418
Pop3Command	1422
POP3 Server Agent	1426
MailBoxItem.....	1430
Using Auto-Generated Strings	1432
POP3 Statistics	1433
POP3 Client Statistics	1433
POP3 Server Statistics.....	1436
QuickHTTP.....	1438
Overview	1439
Objectives	1439
QuickHTTP Client Agent.....	1439
QuickHTTP Server Agent	1440
QuickHTTP Client Agent.....	1442
QuickHTTP Client HTTP.....	1446
QuickHTTP Client Advanced.....	1449
QuickHTTP Client Command	1451
QuickHTTP Server Agent	1453
QuickHTTP Server HTTP	1456
QuickHTTP Server Advanced	1458
QuickHTTP Statistics.....	1459
HTTP/QuickHTTP Client Statistics	1459
HTTP/QuickHTTP Server Statistics.....	1463
QuickTCP.....	1465
Overview	1466
Objectives	1466
QuickTCP Client Agent.....	1466
QuickTCP Server Agent.....	1467
QuickTCP Client Agent	1468
QuickTCP Command	1471
QuickTCP Client Advanced	1474
QuickTCP Server Agent.....	1476
QuickTCP Server TCP	1478
QuickTCP Statistics	1479
Radius	1480
Overview	1482
Objectives	1482
Radius Client Agent	1486
Radius Command List	1488

Global Config	1494
Specific Secrets	1496
Vendor List	1498
Attribute List	1499
AccessAttribSetList	1502
AcctngAttribSetList	1504
RADIUS Client Statistics	1505
RTSP	1511
Overview	1513
Objectives	1513
RTSP Client Agent	1513
RTSP Server Agent	1515
RTSP Client Agent	1517
RtspCommand	1524
RtspHeaders	1527
RtspsetParamOptionList	1531
RtspgetParamOptionList	1533
RTSP Server Agent	1535
PresentationItem	1540
Content	1542
RTSP Statistics	1545
RTSP Client Statistics	1546
RTSP Server Statistics	1551
SMTP	1553
Overview	1554
Objectives	1554
SMTP Client Agent	1555
SMTP Server Agent	1557
SMTP Client Agent	1557
SmtCommand	1561
MailMessage	1563
SMTP Server Agent	1570
SMTP Statistics	1573
SMTP Client Statistics	1573
SMTP Server Statistics	1575
SSH	1577
API Overview	1578
Objectives	1578
SSH Client Agent	1581
SSH Command List	1583
Option Set	1591
Option Set Manager	1593
Global Config	1596
SSH Client Statistics	1597
Stateless Peer	1600
Stateless Peer Overview	1601
Objectives	1601
Stateless Peer Commands	1601



Stateless Peer Agent	1602
Stateless Peer Advanced Options	1603
Stateless Peer Protocol Flows	1604
Stateless Peer Available TOS List	1610
Telnet	1612
Telnet RFCs	1614
IxLoad Telnet Client and Server	1614
Telnet Server	1614
Telnet Client	1616
Creating a Telnet Test	1617
Configuring a Telnet Server	1618
Configuring a Telnet Client	1620
Telnet Statistics	1628
Telnet Client Statistics	1629
Telnet Server Statistics	1634
TFTP	1638
Overview	1640
Objectives	1640
TFTP Client Agent	1642
TFTP Command List	1644
TFTP Client Advanced	1649
TFTP Server Agent	1651
fileList	1653
advanced	1655
TFTP Client Statistics	1658
TFTP Server Statistics	1660
Trace File Replay	1662
Overview	1664
Objectives	1664
Trace File Replay Client Commands	1664
Trace File Replay Server Commands	1666
Trace File Replay Client Agent	1668
Options	1670
Filter List	1672
Enable Filter	1674
Trace File Replay Server Agent	1675
Trace File Options	1677
Advanced Options	1679
Statistics	1681
Trace File Replay Client Statistics	1682
Trace File Replay Server Statistics	1684
VoIP H.248 Peer	1686
Limitations	1688
VoIP H248 Peer API Commands	1689
VoIP H248 MGC/MGW Peer API Objects	1690
VoIP H248 TermGroup Peer API Objects	1691

VoIP H248 Peer Agent.....	1693
Simulated MGC.....	1699
Simulated MGW	1701
H248 TermGroups.....	1703
MGW Automatic.....	1705
MGC Automatic.....	1708
Profiles	1714
Packages.....	1716
Events.....	1718
Properties.....	1720
Signals.....	1722
Statistics	1724
H248 Settings.....	1726
Codec Settings.....	1729
Data Codecs	1731
Codecs	1733
Other Settings.....	1740
SDP Settings.....	1743
RTP Settings.....	1745
Audio Settings	1747
Execution Settings.....	1751
Scenario Settings.....	1753
VoIP H.323 Peer.....	1755
API Overview.....	1757
Limitations	1758
VoIP H323 Peer API Commands.....	1759
VoIP H323 Peer API Objects	1759
VoIP H323 Peer Agent.....	1761
Codec Settings.....	1771
Codecs	1772
Data Codecs	1778
Other Settings.....	1780
RTP Settings.....	1783
Audio Settings	1785
Video Settings	1789
Alternative Capability Value Set List.....	1795
Capability List.....	1797
Custom Activity Link Settings.....	1799
Execution Settings.....	1801
Simultaneous Capability.....	1803
H323 Settings.....	1804
Simultaneous Capability Value Set List.....	1809
Alternative Capability List.....	1811
Alternative Capability	1813
Dial Plan.....	1814
Terminal Capability Set.....	1817
Simultaneous Capability List	1818
Scenario Settings.....	1820
Custom Parameters.....	1822
VoIP MGCP	1824
Limitations	1826
VoIP MGCP Peer API Commands.....	1827
VoIP MGCP CA/MGW Peer API Objects	1828



VoIP MGCP Endpoint Peer API Objects	1829
MGCP GW Agent	1830
MGCP Settings (GW)	1838
Automatic Settings (GW)	1841
Endpoints.....	1843
MGCP CA Agent	1845
MGCP Settings (CA)	1847
Automatic Settings (CA)	1850
Endpoints.....	1852
Gateways.....	1854
Scenario Settings.....	1856
Execution Settings.....	1857
Custom Activity Link Settings	1859
Simulated Endpoints	1863
Data Codecs	1865
Codecs	1867
SDP Settings.....	1872
RTP Settings.....	1874
Audio Settings	1876
Other Settings.....	1880
VoIP SIP Cloud	1882
Limitations	1883
VoIP SIP Cloud API Commands	1884
API Objects	1884
VoIPSIP Cloud Agent	1885
Settings.....	1887
SIP Server List	1888
VoIP SIP Peer	1890
Limitations	1893
VoIP SIP Peer API Commands	1894
VoIP SIP Peer API Objects	1894

VoIP SIP Peer Agent.....	1897
Codec Settings.....	1916
Data Codecs	1917
Codecs	1919
Other Settings.....	1926
Signaling Settings.....	1929
Edit Contact.....	1934
RTP Settings.....	1936
Audio Settings.....	1938
Video Settings	1942
T.38 Settings	1946
T.30 Settings	1950
Timer Settings	1955
SRTP Settings	1957
Custom Activity Link Settings	1960
Execution Settings.....	1963
Transfer Address	1966
Scenario Settings.....	1968
Dial Plan.....	1969
TLS Settings.....	1972
TLS Cyphers	1975
Custom Parameters.....	1976
Advanced Settings.....	1979
Cloud Servers	1981
Server Rules	1983
Cloud Rules.....	1985
RuleData.....	1987
VoIP Skinny Peer	1991
Limitations	1993
VoIP Skinny Peer API Commands.....	1994
VoIP Skinny API Objects.....	1994
VoIP Skinny Peer Agent	1996
Scenario Settings.....	2012
Execution Settings.....	2013
Dial Plan.....	2015
Skinny Settings.....	2020
Call Managers	2023
Codec Settings.....	2025
Data Codecs	2026
Codecs	2028
RTP Settings.....	2034
Audio Settings.....	2036
Other Settings.....	2040
Custom Activity Link Settings	2043
Custom Parameters.....	2045
Nessus (Vulnerability)	2047
Overview	2048
Objectives.....	2048
Nessus Client Commands.....	2048
Nessus Client Agent.....	2052
Parameter Categories.....	2054
Targets	2080



Attacks Search.....	2086
Attacks Select.....	2087
Attacks List.....	2089
Parameters Data Table.....	2091
Parameters Data List.....	2092
Category List.....	2094
Statistics	2095
IP, TCP, Run State, and Curve Segment Statistics	2096
Per-Interface and TCP Statistics.....	2097
TCP Statistics	2097
Advanced TCP Statistics	2103
Per-Interface Statistics	2104
Run State Statistics.....	2104
Curve Segment Statistics.....	2105
Connection Latency Statistics.....	2105
IxServer Layer 2-3 Statistics	2106
Index.....	2109



Technical Support

You can obtain technical support for any Ixia product by contacting Ixia Technical Support by any of the methods mentioned below.

Ixia sales office contact information (http://www.ixiacom.com/contact_us/map/) is available on the Ixia website.

USA, Canada, and Latin America	Ixia Worldwide Headquarters 26601 W. Agoura Rd. Calabasas, CA 91302 USA	877 367 4942 (toll-free inside USA) +1 818 871 1800 Option 1 (outside USA) support@ixiacom.com Hours: 0600 to 1800 PST
Europe, Middle East, and Africa (EMEA)	Ixia Europe Limited One Globeside, Fieldhouse Lane Marlow, SL7 1HZ United Kingdom	+44 1628 405797 eurosupport@ixiacom.com online support form (http://www.ixiacom.com/support/inquiry/?location=emea): Hours: 0700 to 1730 GMT
Asia Pacific	Asia Pacific Representative Office 210 Middle Road #08-01 IOI Plaza Singapore 188994	+65 6332 0126 support-field-asia-pacific@ixiacom.com online support form: (http://www.ixiacom.com/support/inquiry) Hours: 0900 to 1730 local time
Japan	Ixia KK Aioi Sampo Shinjuku Building, 16th Floor 3-25-3 Yoyogi Shibuya-Ku Tokyo 151-0053 Japan	+81 3 5365 4690 support@ixiacom.com online support form: (http://www.ixiacom.com/support/inquiry)



India	Ixia India No. 508, 6th Main 6th Cross ST Bed, Koramangala 4th Block Bangalore 560 034 India	+91 80 32918500 support-india@ixiacom.com online support form: (http://www.ixiacom.com/support/inquiry/?location=india) Hours: 0900 to 1730 local time
-------	--	---



About this Guide

This section contains information that explains the typographical conventions used in this documentation. This information will aid you in using the documentation most effectively. Also provided is a list of related documentation that you may find useful.

In this section:

Conventions	xx
Related Documentation	xxii

Conventions

The following typographical conventions are used in this documentation:

- Italics are used to indicate the names of software fields and parameters, titles of books or documents, and first references to words, terms, phrases, or concepts that have a special meaning or require special identification or emphasis. For example:
 - In the *userid* field, enter your assigned user identification number.
 - *Norton's Telecom Dictionary* is a helpful reference tool.
 - The term *tolerance level* refers to the standard deviation setting.
 - The variable *n* represents any numerical value.
- Menu names and options appear as bold blue text in online Help, and appear in small capital letters in documents. For example:
 - To save your input, choose the **FILE>SAVE** menu option.
- Bold black type is used to indicate the names of buttons, commands, and files that are part of procedures, as well as to identify field and parameter options. In addition, bold text emphasizes important information in text or in caution, warning, or danger statements. For example:
 - To proceed to the next step, click **OK**.
 - Use the **copy** command to duplicate the field entry.
 - Save and close the **books.xml** file.
 - Always save your test configuration.
- Courier text is used to indicate typed text input. For example:
 - Access the new file name at the command line: `c = newbook.gif`.
 - Enter the setup.ini location: `setupini = Ixia\Code\New`.
- PC keys are indicated in all caps, using the following conventions:
 - Simultaneous keystrokes are shown by joining the key names with a plus sign (+), For example, CTRL+Q.
 - Sequential keystrokes are shown by joining the key names with a comma (,). For example, SHIFT, F7.

Icons are used throughout the documentation to draw attention to caution, warning, and danger statements. Icons also identify notes, important information, and the start of a procedure.



CAUTION: Provides information that is needed to avoid potential harm to equipment or the environment.



WARNING: Provides information needed to prevent certain harm to a person, piece of equipment, or the environment.



DANGER: Provides information to prevent certain severe bodily harm to a person.



Note: Provides information that needs special attention within text.



IMPORTANT: Provides information that is especially important for correct product operation.



Indicates the start of a procedure.



Related Documentation

The following documentation may be helpful in gaining more understanding of IxLoad. The documentation is available from the Help pull-down menu in IxLoad or from the IxLoad CD.

Ixia user documentation is also available in the Support>User Guides area of [ixiacom.com](http://www.ixiacom.com) (<http://www.ixiacom.com>). User registration is required to view this online documentation.

- *Getting Started with Aptixia IxLoad*
- *IxLoad User Guide*



Introduction

The *IxLoad Tcl API* is a set of Tcl commands that enable you to run IxLoad tests from Tcl scripts. The API provides most of the same capabilities available from the GUI.

In this chapter:

Background Reading	24
Network Setup	25
Backward Compatibility	29
Deprecated Commands	30

Background Reading

In order to use the Tcl API, you should also have the following other documents:

The *IxLoad User Guide* should be read and understood before attempting to use the API. In particular, the following two chapters are essential:

- *Introduction* discusses the background to understand Internet protocol testing in general and the manner in which Ixia approaches it specif
- *Creating and Running an IxLoad Test* describes how to create the test infrastructure. Care must be taken to assign IP addresses correctly and to provide required routes.

The *Creating and Running an IxLoad Test* chapter uses the term *Management Station* to refer to the host that runs the IxLoad GUI application. In this guide, that hose is The host that runs a Tcl program using the IxLoad Tcl API is in this same position. We shall refer to this as the *Development Station* in the remainder of this manual.

The *Ixia Tcl Development Guide* describes the general method for developing Tcl scripts for use with Ixia equipment. Only a few of the commands described in that guide are necessary to construct an IxLoad Tcl API-based test, but you should review the entire guide to familiarize yourself with the general structure and functioning of Tcl-based tests.

Using a License Server

If you are using a central license server with IxLoad, make sure to set the name of the server in IxLoad's SETTINGS > PREFERENCES menu choice.

To allow use of a central license server by the TCL API, the system environment variable `IXN_LICENSE_SERVER` must be set on the client PC.

If you are running your Tcl program on a Unix client, the `IXN_LICENSE_SERVER` environment variable must be set on the Windows host running the Tcl Server, and on the Unix client through the user shell initialization script.



To set the License Server environment variable on a Windows host:

1. Right-click on the My Computer icon on the desktop, then select `PROPERTIES`.
2. Click the `Advanced` tab.
3. Click `Environment Variables`.
4. In either the `User variables for...` or `System variables` lists, click `New` to add a new variable.
5. Name the variable `IXN_LICENSE_SERVER`.
6. Set the variable value to the name or IP address of the license server host.
7. Click `OK` to close the window.

Configuring the IxAppOptions.ini file

The IxAppOptions.ini file contains various parameters that determine some aspects of IxLoad's behavior. Normally, you should not have to change any of the parameters in IxAppOptions.ini. However, some test scenarios may require you to edit the IxAppOptions.ini file.

The IxAppOptions.ini file is stored in the following directory on the IxLoad cliPC:

C:\Program Files\Ixia\IxLoad\Client

License Server Parameters

The [GlobalOptions] section contains two parameters that define the license server being used. In order to run IxLoad from Tcl scripts, you must configure these parameters, because there is no way to define a license server from an IxLoad Tcl API script. The license server parameters are:

[GlobalOptions]

license_server_enabled = Specifies whether the license is stored on the test chassis or on an external license server.

False: The license is stored on the same Ixia chassis that is being used by the Tcl script (Default).

True: The license is stored on an external license server. Specify the license server's host name or IP address in the license_server parameter.

license_server = If the license is stored on an external license server, specify its host name or IP address.

local_hardware_manager = This option is added in IxAppOptions file. It can be True or False.

False: Ixload talks to hardware manager on chassis to get chassis related information.

True: Ixload talks to hardware manager on local machine to get chassis related information.

Network Setup

You may need to configure IP addresses or routes for IxLoad Tcl API testing. Review the following sections to see if you need to set or change any addresses:

- To change the IxLoad Tcl API development station's IP address, see [Configuring a Network Address on the IxLoad Development Station](#) (on page 26).
- If the route to your Ixia chassis includes one or more routers, see [Configuring a Permanent Route to Ixia Ports](#) (on page 27).
- If you need to change the internal network used by an Ixia chassis, see [Setting Ixia Chassis Base Addresses](#) (on page 29).



Configuring a Network Address on the IxLoad Development Station

To use the *IxLoad Tcl API*, you must configure your development station with an address on its local network that is routeable to all of the Ixia chassis that you will use for testing.

 To configure routing:

1. Click Windows' Start button and select Settings > Network and Dial-up Connections. Windows displays the connections currently configured on your PC.
2. Right-click Local Area Connection and select Properties. Windows displays the Local Area Connections Properties window.
3. Click Internet Protocol (TCP/IP), then click Properties. Windows displays the LAN connection's TCP/IP properties.
4. Click the Use the following IP address button, then enter addresses in the following fields:
 - *IP address*: Enter an IP address that is routeable to all the Ixia chassis that you will use for IxLoad testing.
 - *Subnet mask*: Enter a subnet mask appropriate to the IP address you entered.
 - *Default gateway*: Enter the IP address of the gateway you will use to access the network that the Ixia chassis are on.
5. If you want to use DNS, enter the DNS servers' IP addresses in the Preferred DNS server and Alternate DNS server fields.
6. Click OK to close the window.

Testing the Development Station's Routing

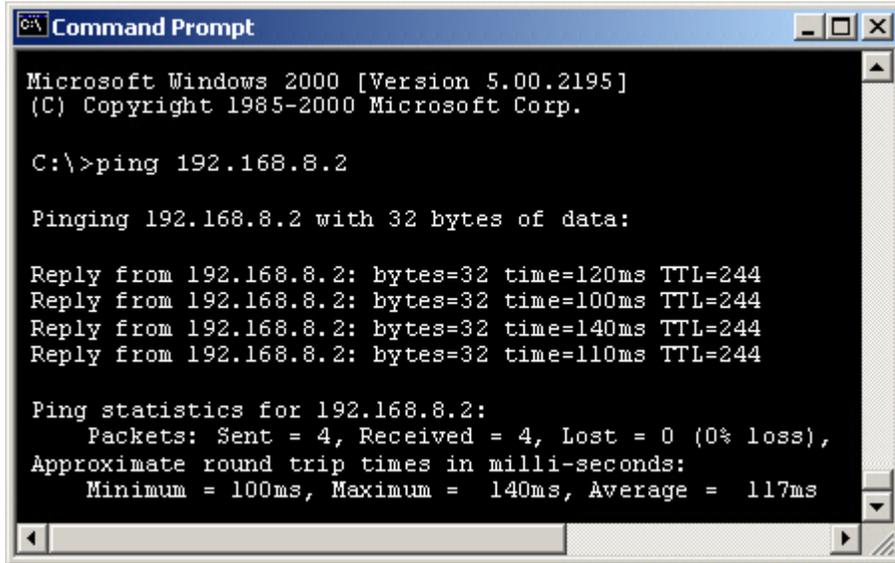
After you have configured the development station's IP address, you should test its routing to ensure it can communicate with the Ixia chassis you will use with IxLoad.

 To test the routing:

1. Click Windows' Start button and select PROGRAMS > ACCESSORIES > COMXPROMPT.

Windows displays a Command Prompt window.

Equation 1: -1.Ping Command



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>ping 192.168.8.2

Pinging 192.168.8.2 with 32 bytes of data:

Reply from 192.168.8.2: bytes=32 time=120ms TTL=244
Reply from 192.168.8.2: bytes=32 time=100ms TTL=244
Reply from 192.168.8.2: bytes=32 time=140ms TTL=244
Reply from 192.168.8.2: bytes=32 time=110ms TTL=244

Ping statistics for 192.168.8.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 100ms, Maximum = 140ms, Average = 117ms
```

2. Use the ping command to test that your development station can communicate with each chassis:
`ping aaa.bbb.ccc.ddd`
(replace `aaa.bbb.ccc.ddd` with the IP address of the Ixia chassis).
3. Repeat the ping command for each chassis. Each chassis should return a reply. If any do not, check their TCP/IP configurations.



Note: You cannot ping Ixia ports (the chassis' internal 10.0.0.0 network) until you have started a test. Refer to [Configuring a Permanent Route to Ixia Ports](#) on page 1-6 on how to set up routing so you can access the addresses assigned to Ixia ports.

Configuring a Permanent Route to Ixia Ports

You must configure a route from the IxLoad development station to the Ixia port management base addresses.

To establish a permanent route on a Windows system, you can either use the IxLoad GUI (see chapter 4—step 3) or the following procedure:

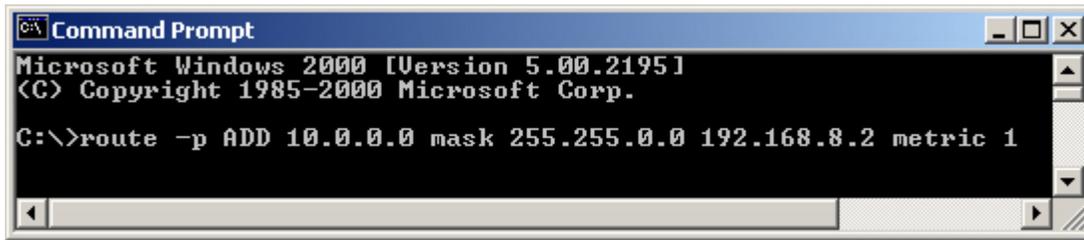


To establish a permanent route:

1. At the IxLoad development station, click Windows' Start button and select `PROGRAMS > ACCESSORIES > COMMAND PROMPT`.

Windows displays a Command Prompt window.

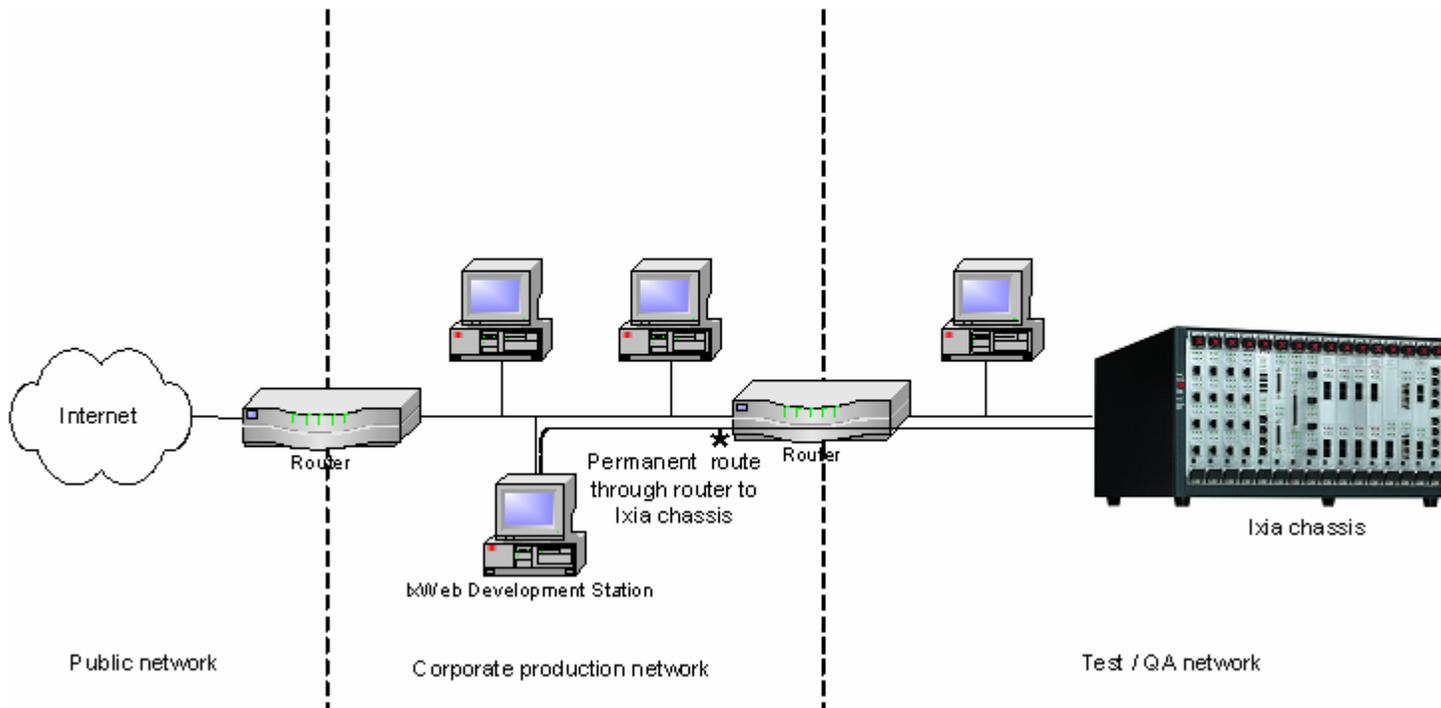
Equation 2: -2.Route Command



2. Use the route command to create a permanent route:

```
route -p ADD 10.0.0.0 mask 255.255.0.0 aaa.bbb.ccc.ddd metric 1
```

- If the Ixia chassis is on the same subnet as the development station, replace `aaa.bbb.ccc.ddd` with the IP address of the Ixia chassis.
- Many IxLoad test environments resemble the one shown in the figure below: IxLoad Tcl API running on a PC connected to a corporate production netan Ixia chassis connected to a test or QA network behind one or more routers, and a DUT network connected only to the Ixia chassis.



If the Ixia chassis is not on the same subnet as the development station, as shown in the figure above, replace `aaa.bbb.ccc.ddd` with the address of the router that will provide a connection to the Ixia chassis. That router, and all other intermediate routers to the chassis, should contain routes for the 10.0.0.0 (or modified) address range. These routes in the last router should refer to the Ixia chassis as a gateway.

Note that in a network shown like the one shown in the figure, the router(s) may be configured to disallow access from the production network, or they may route IxLoad requests intended for the Ixia ports (by default, a 10.0.0.0 network) elsewhere (usually to the Internet) or may drop them altogether.

Ensure that no other addresses assigned in IxLoad fall into this range. This setup may be tested using the ping command as described in Testing the Development Station's Routing (on page 26), but only in the final stages of running a test.

Setting Ixia Chassis Base Addresses

All ports on an Ixia chassis are initially configured so that they may be internally addressed, for IxLoad management purposes, as:

```
10.0.<card>.<port>
```

For example, card 2 port 3 has an internal IP address of 10.0.2.3. These addresses must be routeable from the development stations to the Ixia chassis.

The first two octets of the address (10.0) are called the *base address*. If you are using IxLoad on an existing network, you may want to change the base address to conform to your existing network layout. If two or more chassis are used for IxLoad GUI or IxLoad Tcl API testing, all but one of the chassis base addresses will need to be changed.

Note that the Ixia ports on a chassis will use only a limited range of addresses on their subnet. For example, if the base address is 10.0, and there are sixteen 8-port cards in the chassis, then the range of addresses used will be:

```
10.0.1.1 - 10.0.16.8
```



To change the base address of a chassis, use IxExplorer:

1. Open IxExplorer.
2. Select the Chassis Chain object in the tree and right click and choose **ADD CHASSIS**. Enter the name or IP address of the chassis that will be used.
3. Right-click on the newly created chassis and select **PROPERTIES**.
4. Select the **IXROUTER** tab.
5. You may change the base address in the *IP Network* field. Make sure to only modify the top two octets and do not change the *Mask* field.

Backward Compatibility

IxLoad Tcl provides backward compatibility for:

- Scripts that configure and run tests.
- Scripts that run tests from a repository.
- Scripts that modify repositories, as long as the script was written for and tested with repositories from the same IxLoad release as the script, or an earrelease.

For example, if you write a script for IxLoad 4.0, that script can modify any repository created in IxLoad 4.0 or earlier.

IxLoad Tcl *does not* provide backwards compatibility for scripts that modify repositories that were created or saved from releases after the release that the script was written for and tested on.

For example, if you write a script for IxLoad 4.0, that script *should not* modify a repository created in a release later than IxLoad 4.0.



Deprecated Commands

Beginning with the 4.10 release, the objects listed below are deprecated. If used in a script, they will continue to function, but their functionality has been assumed by objects in the Network Stack API.

- ixArpSettings
- ixDhcp
- ixDhcpPD
- ixDhcpPdClient
- ixDhcpOptions
- ixIpSec
- ixPppoe
- ixNetworkRange
- ixDns
- ixDnsServerItem
- ixDnsSuffixItem
- ixImpairment
- ixTcpParameters
- ixBandwidthLimit
- ixEmulatedRouterIpAddressRange
- ixNetworkRange



2

Quick Start

This section describes how to modify a sample script to run an IxLoad Tcl API test. You can use this section to quickly familiarize yourself with the basic steps required to run a simple IxLoad script. Once you have modified and run a sample script, you can refer to the following sections in this guide to learn about the IxLoad Tcl API in greater detail.

- To run a sample script from Windows, see Windows (on page 31).
- To run a sample script from Unix/Linux, see Unix/Linux (on page 34).

In this chapter:

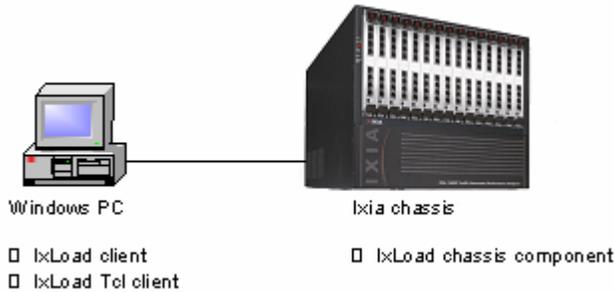
Windows	31
Unix/Linux	34
Editing the setup_simple.tcl script	35
Running the sample scripts	36
Monitoring Status and Retrieving Results	37

Windows

The section describes how to run a sample Tcl script included with IxLoad on Windows. To run IxLoad Tcl scripts, you must install the IxLoad Tcl 8.4 shell, which is an option in the IxLoad Windows client installation package.



Note: The IxOS wish console cannot be used to execute IxLoad Tcl scripts.



Using The Sample Tcl Scripts

The Tcl scripts require the IxLoad Tcl 8.4 wish console to run. The wish console can be launched using either of two commands:

- C:\Program Files\Ixia\Tcl\<version>\bin\wish.exe
- C:\Program Files\Ixia\Tcl\<version>\bin\tclsh.exe

The included sample Tcl scripts can be found in the following subdirectories under C:\Program Files\Ixia\IxLoad\<version>\TclScripts\Samples:

- Samples\Application Features contains scripts that demonstrate various IxLoad features
- Samples\Network contains scripts that create various network configurations
- Samples\Protocols contains scripts that generate different types of protocol traffic
- Samples\Stats contains scripts that demonstrate how to retrieve statistics

One script from the Samples directory, `setup_simple.tcl`, must be modified to work with your network topology. See Editing the `setup_simple.tcl` script (on page 33).

\TclScripts\bin contains two important files:

`IxiaWish.tcl`: This script, when sourced, sets the `auto_path` value so that when you execute a `package req IxLoad` command, the command loads the Ixia wish console. When you install IxLoad, the installer creates a shortcut to the Ixia wish console (with a label including the version) on your desktop. That shortcut launches TCL and sources this file (for the specific installed version that the shortcut corresponds to) so that you can immediately source a sample script, or execute a `package req IxLoad` command.

`SetOldRegistryKeys.tcl`: This script sets the old pre-multi-version registry keys to reference the version of `SetOldRegistryKeys.tcl`. This is a work-around for those who have copied the old (deprecated) `setup_ixload_paths.tcl` logic directly into their scripts. The new `setup_ixload_paths.tcl` script just sources `IxiaWish.tcl`.



Note: The `setup_ixload_paths` code no longer works. The current method, `stanacross` all Ixia applications, is to source the `TclScripts\bin\IxiaWish.tcl` for the application, and follow that with a `package require` command.



Note: Scriptgen scripts no longer attempt to set the `autopath` value. Scriptgen scripts should work directly when sourced from the installed Wish shortcut. If you are using a different way of launching wish, you need to be sure to source `IxiaWprior` to sourcing Scriptgen-created scripts.

Editing the `setup_simple.tcl` script

You must edit the `setup_simple.Tcl` script to include the correct addresses in use on your network.

- On Windows, the file is located at : `..\IxLoad\<version>\TclScripts\Samples`
- On Unix/Linux, the file is located at: `../IxLoadTclApi<version>/Samples/`



To edit the `setup_simple.tcl` script:

1. In an editor, open the `setup_simple.tcl` script.
2. Set the Tcl server address:

```
variable ::IxLoadPrivate::SimpleSettings::remoteServer n.n.n.n
```

Tcl server must run on a Windows not on the chassis. When running a script from Unix, change this value to the IP address of the IxLoad client that the script will run on. When running a script from Windows, this variable must still be set, but its value is not used.

3. Set `chassisName` to the hostname or IP address of the chassis you will use:

```
variable ::IxLoadPrivate::SimpleSettings::chassisName n.n.n.n
```

4. `CARD_ID` and `PORT_ID` are local variables used between the `setup_simple.tcl` script and all Ixia-provided sample Tcl scripts. Set `CARD_ID` and `CARD_PORT` (in the `serverPort` and `clientPort` array) to the card and port you will use:

```
array set ::IxLoadPrivate::SimpleSettings::clientPort {  
CARD_ID    "4"  
PORT_ID    "5"
```

5. Save and close the file.

Running the sample scripts

Follow the instructions below to launch the `ixwish` shell, and call the Tcl script. In the procedure below, replace (replace `<version>` with the correct directory name).



To run a sample script:

1. Start the wish shell by double-clicking the ActiveTcl Wish IxLoad shortcut on the desktop.
2. Change the path to the directory that contains the script that you want to run. Scripts are stored in directories under `<installDir>/IxLoad/<version>/TclScripts/Samples`.
 - `Samples/Application Features` contains scripts that demonstrate various IxLoad features
 - `Samples/Network` contains scripts that create various network configurations
 - `Samples/Protocols` contains scripts that generate different types of protocol traffic
 - `Samples/Stats` contains scripts that demonstrate how to retrieve statistics

For example, to change to the Protocols directory, type:



```
cd Samples/Protocols
```

3. To start the script, use the `source` command to run it.

For example, to run the HTTP.tcl script, type

```
source HTTP.tcl
```

Monitoring Status and Retrieving Results

While a test is running, status messages display in the wish console window.

The results (in CSV format) are placed in the `Results\ subfolder where your Tcl script is located.`

For example: `C:\Program`

`Files\Ixia\IxLoad\`

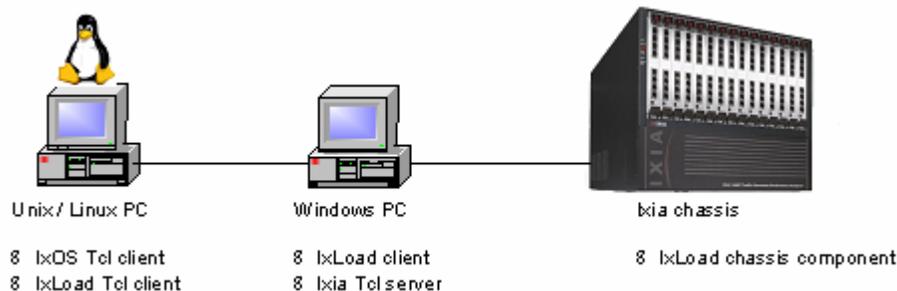
Unix/Linux

On Unix/Linux, you must install the Unix/Linux versions of IxOS Tcl and IxLoad Tcl. There are a number of IxLoad Tcl packages available for Unix/Linux; select the version approfor your distribution:

IxLoad Tcl Client Installation File	Description
IxLoadTclAPI<X.YY>linux.bin	RedHat Linux, up to version 9
IxLoadTclAPI<X.YY>solarisSparc.bin	SUN Solaris version of Unix
IxLoadTclAPI<X.YY>genericLinux.bin	RedHat Linux, after version 9 (RedHat Enterprise 3 and 4)

See the following sections:

Installing IxOS Tcl (on page 35)



Installing IxOS Tcl

This section describes how to install the Unix/Linux version of IxOS Tcl.

 To install Unix/Linux IxOS Tcl:

1. Copy the IxOS<version>genericLinux.bin file to the Linux system.
2. Change the file's attribute to make it executable.
Example: `chmod +x IxOS<version>genericLinux.bin`
3. Execute the IxOS installer file (use the -console option if installing via console or remote shell).
Example: `./IxOS<version>genericLinux.bin`
4. When the installer prompts you, select Tcl version 8.4 (required) and the location.
Example: `/etc/ixosTcl84`
5. Follow the rest of the instructions to complete the installation.

Installing IxLoad Tcl

This section describes how to install the Unix/Linux version of IxLoad Tcl.

 To install Unix/Linux IxLoad Tcl:

1. Copy the IxLoadTclAPI<version>genericLinux.bin file to the Linux system.
2. Change the attribute to make it executable.
Example: `chmod +x IxLoadTclAPI<version>genericLinux.bin`
3. Execute the installer file (use the “-console” option if installing via console or remote shell).
Example: `./IxLoadTclAPI<version>genericLinux.bin`
4. Follow the prompts to complete the installation. The installer should autodetect the IxOS Tcl client installation directory before it installs IxLoad Tcl.

Editing the setup_simple.tcl script

You must edit the setup_simple.Tcl script to include the correct addresses in use on your network.

- On Windows, the file is located at : ..\IxLoad\<version>\TclScripts\Samples
- On Unix/Linux, the file is located at: ../IxLoadTclApi<version>/Samples/

 To edit the setup_simple.tcl script:

1. In an editor, open the setup_simple.tcl script.
2. Set the Tcl server address:
`variable ::IxLoadPrivate::SimpleSettings::remoteServer n.n.n.n`



Tcl server must run on a Windows not on the chassis. When running a script from Unix, change this value to the IP address of the IxLoad client that the script will run on. When running a script from Windows, this variable must still be set, but its value is not used.

3. Set `chassisName` to the hostname or IP address of the chassis you will use:

```
variable ::IxLoadPrivate::SimpleSettings::chassisName n.n.n.n
```

4. `CARD_ID` and `PORT_ID` are local variables used between the `setup_simple.tcl` script and all Ixia-provided sample Tcl scripts. Set `CARD_ID` and `CARD_PORT` (in the `serverPort` and `clientPort` array) to the card and port you will use:

```
array set ::IxLoadPrivate::SimpleSettings::clientPort {  
CARD_ID    "4"  
PORT_ID    "5"
```

5. Save and close the file.

Running the sample scripts

Once the `setup_simple.tcl` script is configured, use the following procedure to launch the Tcl shell and run a sample script.

 To run a sample script:

1. Change to the bin directory where IxOS Tcl is installed.
2. Copy `/bin/ixwish` to `bin/ixTclsh`.
3. Start the Tcl shell:

```
./bin/ixTclsh
```

4. Scripts are stored in directories under `/etc/ixosTcl8.4/IxLoadTclAPI<version>/Samples`.
 - `Samples/Application Features` contains scripts that demonstrate various IxLoad features
 - `Samples/Network` contains scripts that create various network configurations
 - `Samples/Protocols` contains scripts that generate different types of protocol traffic
 - `Samples/Stats` contains scripts that demonstrate how to retrieve statistics

Change your path to the directory that contains the script that you want to run.

For example, to change to the `Protocols` directory, type:

```
cd /etc/ixosTcl8.4/IxLoadTclAPI<version>/Samples/Protocols
```

5. Source the sample script that you want to run. For example, to run the `HTTP.tcl` script, type:

```
source HTTP.tcl
```

Monitoring Status and Retrieving Results

While a test is running, status messages display in the Linux shell.

The log files are stored on the Windows host. The log file name is determined by the `set LogName` command in the script.

- If you specify no path or a partial path, the log file is stored relative to the `\remoteScriptingService` directory on the IxLoad installation path.
- If you specify an absolute path, the log file is stored in that location.

The log file will be prefixed with the specified name, followed by “-x-00” where *x* is a session ID from 1 through 4. The sample scripts all set the log name to be the same as the script name. For example:

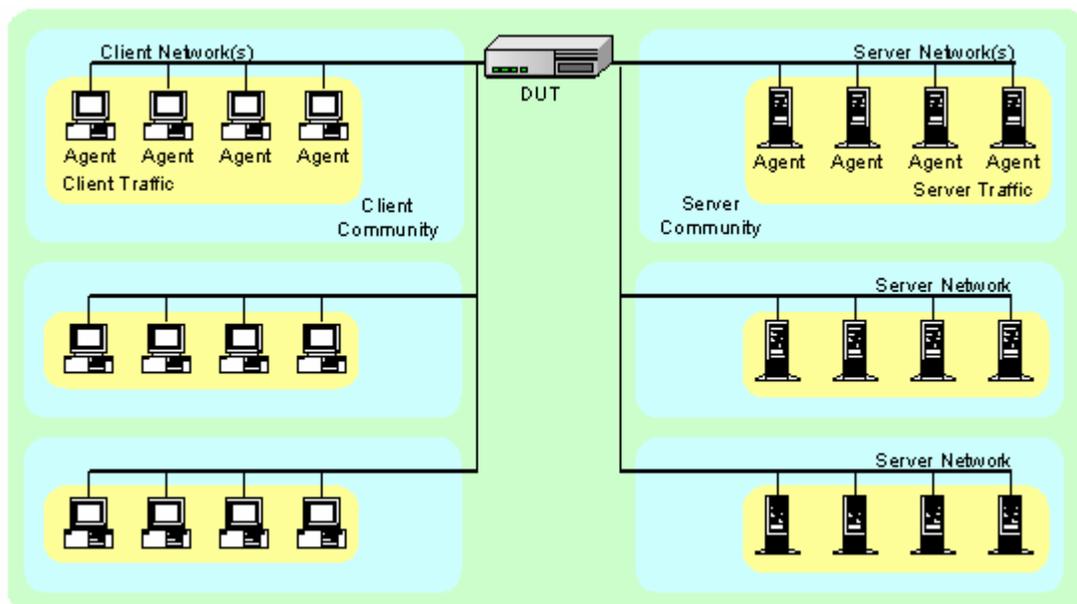
```
C:\Program
Files\Ixia\IxLoad\\TclScripts\remoteScriptingService\RESULTS\
```

3

API Overview

An IxLoad test consists of one or more Client Communities sending traffic through the DUT (Device Under Test) to Server Communities. The structure of both a client and server community is the same: Traffic sent over a network. Traffic is generated or handled by one or more agents.

The conceptual view of an IxLoad test is shown in the figure below.

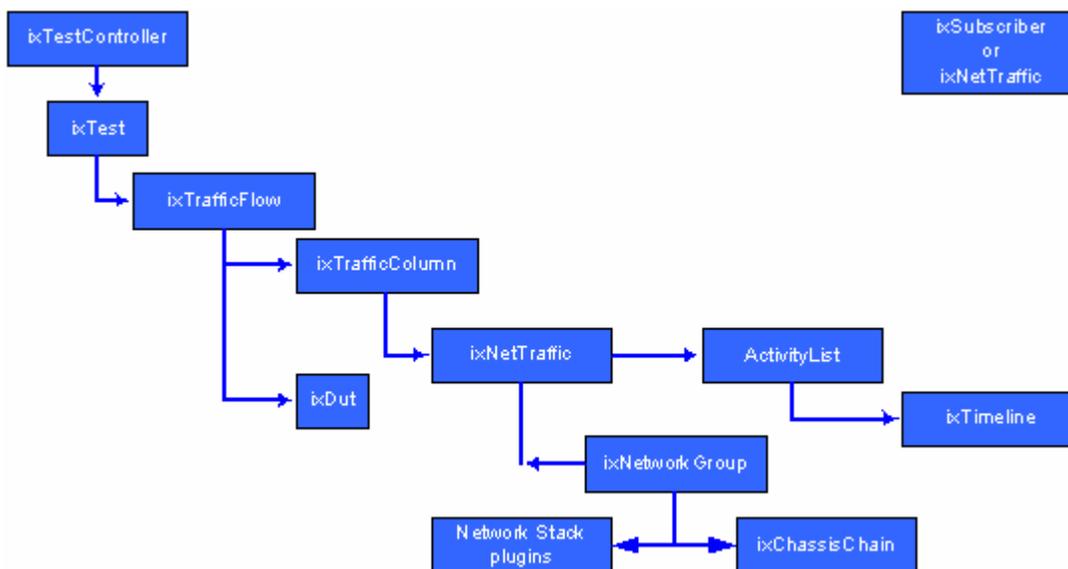


In this chapter:

Tcl API Structure	39
General Tcl Conventions	41
TCL API Internal Overview	43
Building an IxLoad Test	46
Running an IxLoad Tcl Script	65
API Description	68
Sample Scripts Shipped with IxLoad	79
Example Program	83

Tcl API Structure

The Tcl API's structure follows the conceptual model. Its main components are shown in the figure below.



The table below describes the components of the API shown in the figure.

Component	Description
ixTestController	testController controls the running of a test. No test can run successfully without this object. It has no relationship with any other object.
ixTest	The top level object which co-ordinates the client and server communities. It holds separate lists of each type of communities.

Component	Description
ixTrafficFlow	This object lists the test scenario.
ixTrafficColumn	This is a container of ixNetTraffic and ixDut objects.
ixNetTraffic	This object contains client and server activities, and traffic.
ixNetworkGroup	This object describes the global network configuration.
Protocol Agents	IxLoad uses a pluggable structure that allows multiple Internet protocols to be used, such as HTTP and FTP. Ixia supplies a growing set of agents, which are described in separate appendixes of this manual. The agents provide the intelligence to generate/handle protocol specific traffic. Some client agents use Protocol Actions to describe their operation.
Network Stack plugins	Protocol and associated extensions that provide the network that the traffic protocols run over.
Protocol Actions	Some of the Protocol Agents describe their operation in terms of specific actions. These protocol dependent objects detail those operations.
ixChassisChain	This independent object describes the list of chassis that will be used in a test.
ixDut	This object holds the type and type-specific information about a DUT.
ixTimeline	This object configures the time in the test when the activities in the NetTraffics come online, and how long they stay up for. It is also used to configure the test's objectives.
ixSubscriber	The ixSubscriberNetTraffic object is a special type of NetTraffic that simulates the traffic patterns created by residential customers that receive voice, video, and data service (Triple-play) over a single physical connection (usually a cable or DSL connection).
ixImpairment	The ixImpairment object impairs one or more types of traffic from a client and server network.
activityList	Generates traffic for one side of a particular protocol.

Mandatory Objects to Complete a Script

The following mandatory objects are required to complete a script:

- an ixTestController
- an ixRepository or an ixChassisChain (ixRepository includes a chassis chain)
- ixViewOptions
- ixTest
- ixTrafficFlow
- ixTrafficColumn
- ixNetTraffic
- ixTimeline
- activityList

Multi Version Support

You can install and use multiple versions of IxLoad on the client PC and on the chassis. Installing multiple versions allows you to try out the new features in a new release of IxLoad without having to overwrite your existing copy of IxLoad.

Refer Chapter 2: - API Quick Start and Running an IxLoad Tcl Program on page 3-27 for more information.

General Tcl Conventions

IxLoad's Tcl API is somewhat different from other Ixia Tcl APIs that you might have used. Rather than a single set of global commands that are associated with an Ixia port, IxLoad uses the concept of instances of commands—called *objects*. This guide uses the words *command* and *object* to refer to the same thing.

Object Creation

The general paradigm for the creation of IxLoad objects is to make a 'new' copy of a command, saving the result in a Tcl variable:

```
set my_network [::IxLoad new ixClientNetwork $chain\  
-name "my_client_network4" ]
```

The variable `my_network` is an instance of the `ixClientNetwork` object. Each instance occupies its own area of memory. Multiple objects of the same type can be created and added to lists of items.

The `::IxLoad` reference is to a utility routine that allows new objects of any type to be created. The `::` means in the global context and is a safe means of refer `ixLoad` from any program location.

In addition to command-specific subcommands, each IxLoad command supports a set of subcommands described in the following table.

Subcommand	Usage
config	Allow any option of the command to be set.
cget	Read the value of any command option.
getOptions	Get the names of all of a command's options.

The `ixLoad` command provides a convenient means of creating an object and setting options at the same time. One need only append option names and values to the end of the command. See the following example:

```
set my_network [::IxLoad new ixClientNetwork \  
-name "my_client_network4"
```

This is the standard means by which IxLoad objects are created.



Subobjects

Some IxLoad objects can contain other objects, making them *subobjects*. A subobject can use the same subcommands as its parent object (those listed in the table under Object Creation (on page 41)).

For example, for the name option in `ixClientNetwork`, you can reference the subobject's option as follows:

```
$my_network network.config -name lk_hosts
```

Lists of Objects

Most IxLoad commands contain one or more options that are lists of other objects. For example, `networkRangeList` in `ixClientNetwork` is a list of items of type `ixNetworkRange`. Such lists are commonly built up using the `appendItem` subcommand. For example:

```
$my_network portList.appendItem \  
-chassisId 1 \  
-cardId 2 \  
-port 3
```

As in the `::IxLoad new` command, you can set the values of a list member's options while creating the item. All such lists have a number of associated sub-commands, described in the following table.

Subcommand	Usage
clear	Remove all elements from the list.
appendItem	Add an item to the end of the list.
configItem	Configure the options of one item of the list.
deleteItem	Delete an item from the list.
getItem	Return an instance reference to an element of a list. This can be used to directly manipulate that list member.
insertItem	Add an item into the middle of the list.
find	Search for an item in a list. The indexes of all matching list members is returned.
indexCount	Returns a count of the number of items in a list.

In addition, if you know the index of an item in a list, it may be directly manipulated by the common configuration commands listed in the table under Object Creation (on page 41). For example, to configure the first item in a list:

```
$my_netTraffic activityList(0).config -userObjectiveValue 100000
```

Constants

Predefined constants within the IxLoad Tcl API are associated with particular commands and placed in an array corresponding to a command. For example, the `ixCard` object holds a definition for different Ixia card types, one of which is `kCard1000Txs4`. The proper means of referencing this constant is:

```
$::ixCard(kCard1000Txs4)
```

Text strings may frequently be set using provided strings. Refer to the various reference pages to determine availability.

Strings and Numbers

Tcl variables are considered type-less. That is, no special distinction is made between the string of characters "1.0" and the numeric value 1.0. Within the IxLoad Tcl API, however, items that look like numbers (for example, 111) are converted to numbers. In the specification of arguments and values to the IxLoad Tcl APIs, it is best to enclose these items in quotes if they are not to be interpreted as numbered values.

For example, if you want to name an IxLoad 123, you should enclose the name in quotes: "123".

TCL API Internal Overview

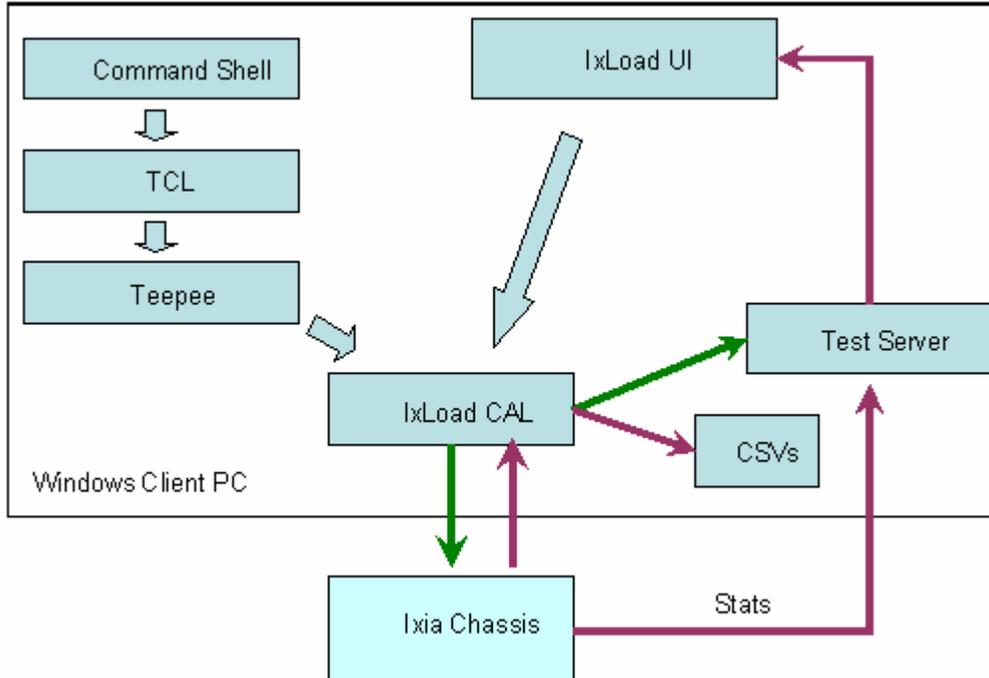
The following sections provide an overview of how the Tcl API functions on Windows or Unix/Linux platforms.

Windows Overview

When running scripts on Windows:

- For IxLoad Unit Limit:
 - User Interface counts as two
 - Scripts count as one

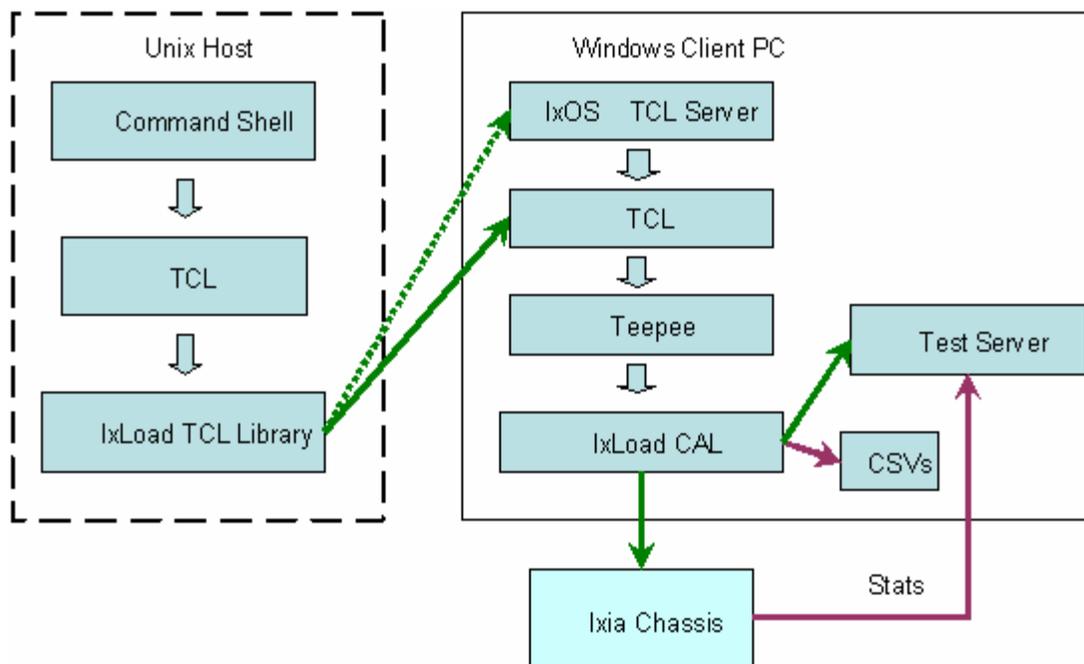
- TCL Server is not required
- The `:IxLoad connect/disconnect` command is ignored
- Log files are stored in the same directory as the script
- Relative files depend on the Client directory path



Unix Overview

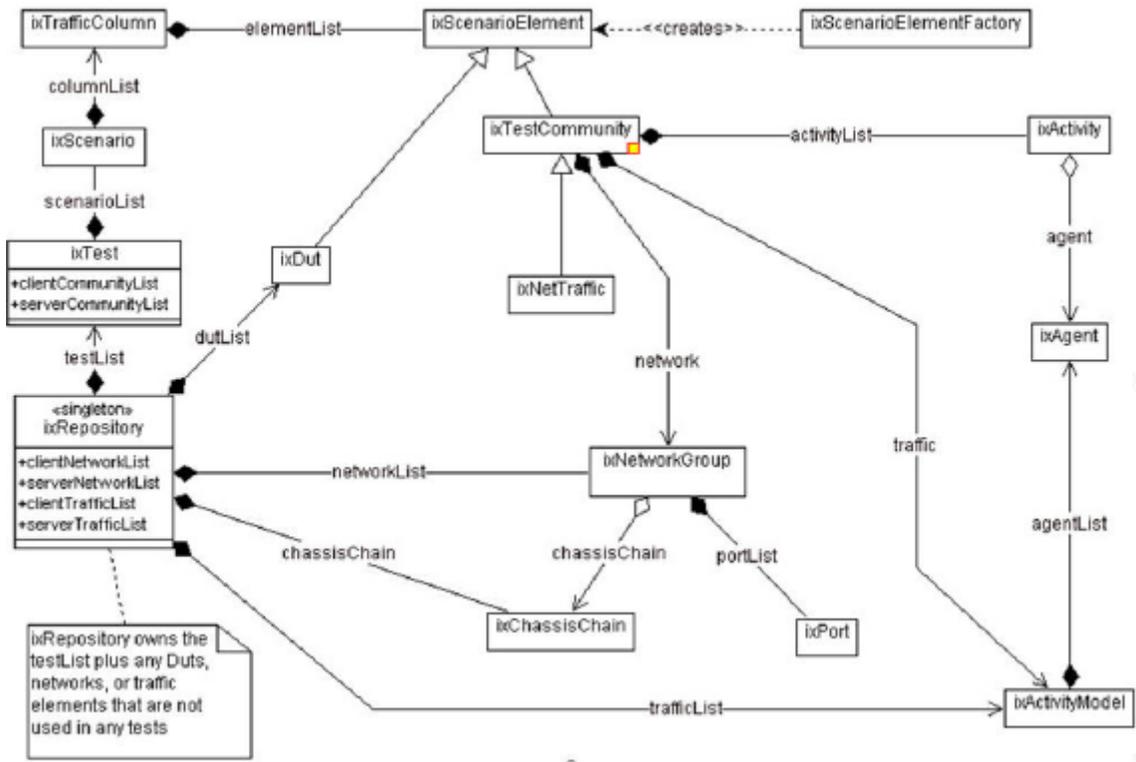
When running scripts on Unix:

- Unix script counts the same as a Windows script
- TCL Server is required on Client PC
- `::IxLoad connect/disconnect` command is required
- Log files go to PC
 - `C:\Program Files\Ixia\IxLoad\Client\tclext\remoteScriptingService` direc
 - Or, internal debug file `c:rssN.log` (*N* is session # (0-4))
- Result files go to PC
 - Path is on Windows system.
 - If path is relative, then path is relative to the `remoteScriptingService` directory



Object Structure

The figure below shows the current TCL object structure.



Building an IxLoad Test

The following set of steps is a suggestion for the manner in which the Tcl API may be used to create a complete test. These steps mirror our suggested use of the IxLoad GUI. This is by no means the only manner in which a test may be developed. The code fragments used in this discussion are from a complete example, included at the end of this chapter.

Step 1: Initial Overhead

This section performs the necessary overhead steps to get a test started. This includes:

- Add the `IxWish` or `tclsh` installer location. This script, when sourced, sets the `auto_path` so that `package req IxLoad` loads the version from which `IxiaW` was sourced. The `IxiaWish.tcl` is sourced from different path depending on a multiversion or non-multiversion release.
- Connect to the remote server. When running an `IxLoad Tcl` program on a non-Windows client it is necessary to use a Windows machine as a remote server. This may either be the Ixia chassis or some other Windows client. When run on a Windows machine, the `localhost` specification should be used. A `catch { ... }` construct is used to ensure that a matching `disconnect` operation is invoked at the end of the script.
- A `package require` for the statistics utilities.
- Load the protocols needed for the test. A separate call to `$ixAppload` should be performed for each protocol required. See the table below for the protocols and corresponding string to be passed.
- Creation of a chassis chain to include a list of test related chassis.
- Creation of the top level `ixTest` object.

Protocol	String to Pass
HTTP	HTTP
FTP	FTP
POP3	POP3
RTSP	RTSP
IMAP	IMAP
LDAP	ldap
MGCP	MGCP
QuickHTTP	QuickHTTP
QuickTCP	QuickTCP
SIP	SIP
Telnet	Telnet
Video	Video
DDOS	DDoS
DHCP	dhcp
RADIUS	radius
SSH	ssh
Capture Replay	capturereplay
Application Test	verify
Vulnerability Attacks	nessus
TFTP	TFTP



- Creation of a chassis chain to include a list of test related chassis.
- Creation of the top level ixTest object.

```
#-----
# Set path to find Tcl API
#-----
set MY_IXLOAD_INSTALL "C:\\Program Files\\Ixia\\IxLoad"
lappend ::auto_path [file join $MY_IXLOAD_INSTALL "client" "tclx" "teepee" "stage"]

#-----
# Uncomment the following if you'll be using the Ixia Standard Tcl API
#-----
#set MY_IXTCLHAL_INSTALL "C:\\Program Files\\Ixia\\TclScripts"
#lappend ::auto_path [file join $MY_IXTCLHAL_INSTALL "lib" "ixTcl1.0"]

#-----
# When running on Unix clients, it's necessary to connect to a remote
# server. For Windows clients, this is unnecessary. In the line below,
# change localhost to the IP address of your remote server
#-----
::IxLoad connect localhost

# This catch is used to ensure that we disconnect from the remote
# server regardless of how we exit
catch {

    #-----
    # package require the stat collection utilities
    #-----
    package require statCollectorUtils
    global ixAppPluginManager
    $ixAppPluginManager load "HTTP"

    #-----
    # Build Chassis Chain
    #-----
    set chassisName birdie
    set chassisChain [::IxLoad new ixChassisChain]
    $chassisChain addChassis $chassisName

    #-----
    # Create the test
    #-----
    set test [::IxLoad new ixTest \
        -name          "my_test" \
        -statsRequired 0 \
        -enableResetPorts 0
    ]
}
```

Step 2: Define the TrafficFlow

In this step, we create the TrafficFlow that will list the test scenario.

This involves:

- Creation of an ixTrafficFlow instance.
- Appending the client, server and DUT object of ixTrafficColumn.

```
#-----  
# Create TrafficFlow  
#-----  
set TrafficFlow1 [::IxLoad new ixTrafficFlow]  
$TrafficFlow1 config \  
-name "TrafficFlow1"  
  
#-----  
# Append client object  
#-----  
$TrafficFlow1 columnList.appendItem -object $Client  
  
set Client [::IxLoad new ixTrafficColumn]  
$Client config \  
-name "Client"
```

Step 3: Define the TrafficColumn

This is a container of ixNetTraffic and ixDut objects.

This involves:

- Creation of an ixTrafficColumn instance
- Defining and configuring client, server and DUT.

```
#-----  
# Create the client instance of ixTrafficColumn  
#-----  
  
set DUT [::IxLoad new ixTrafficColumn]  
$DUT config \  
-name "DUT"
```



Step 4: Define the NetTraffic

This step involves the configuration of client and server activities, configuring traffic, mapping traffic to network.

This involves:

- Creation of an `ixNetTraffic` instance
- Configuring traffic
- Configuring the client or server network
- Appending `activityagent`
- Defining and configuring the activity. For each protocol:
 - Define and append an agent to its `agentList`.
 - Perform protocol dependent settings; for example, add actions to the agent's operation by creating an instance of `ixHttpAction` and defining the options.
 - Declare a timeline for each activity.

```

set HTTP_client_client_network [::IxLoad new ixNetTraffic]

#-----
# Activity newAgent1 of NetTraffic HTTP client@client network
#-----
set Activity_newAgent1 [$HTTP_client_client_network activityList.appendItem \
-protocolAndType          "HTTP Client" ]

#-----
# Defining Activity newAgent1
#-----
$Activity_newAgent1 config \
-enable                    1 \
-name                     "newClientActivity1" \
-enableConstraint         false \
-userObjectiveValue       100 \
-constraintValue          100 \
-userObjectiveType        "simulatedUsers" \
-timeline                  $Timeline1

#-----
# Configuring Activity newAgent1
#-----

$Activity_newAgent1 agent.config \
-vlanPriority              0 \
-enableHttpsProxy         0 \
-enableSsl                 0 \
-cookieRejectProbability  0.0 \
-enableUnidirectionalClose false \
-ipPreference              2 \
-loopValue                 true \
-maxPersistentRequests    1 \
-enableEsm                 0 \
-certificate               "" \
-sequentialSessionReuse   0 \
-tos                       0 \
-maxPipeline               1 \
-maxHeaderLen              1024 \
-maxSessions               3 \
-enableHttpProxy           0 \
-enableTos                  false \
-enable                    1 \
-browserEmulation          1 \
-cookieJarSize             10 \
-privateKey                 "" \
-privateKeyPassword        "" \
-urlStatsCount             10 \
-followHttpRedirects       0 \
-tcpCloseOption            0 \
-enableVlanPriority         false \
-esm                       1460 \
-httpVersion                0 \
-sslVersion                 3 \
-name                       "newClientActivity1" \
-enableCookieSupport        0 \
-enableLargeHeader          false \
-clientCiphers              "DEFAULT" \
-httpProxy                  ":80" \
-keepAlive                  0 \
-httpsProxy                 ":443"

$Activity_newAgent1 agent.actionList.clear

```

```
#-----  
# Add actions to this client agent  
#-----  
set my_ixHttpAction [::IxLoad new ixHttpAction]  
$my_ixHttpAction config \  
-profile -1 \  
-namevalueargs "" \  
-destination "HTTP server_newServerActivity1:80" \  
-abort "None" \  
-command "GET" \  
-arguments "" \  
-pageObject "/4k.html"  
  
$Activity_newAgent1 agent.actionList.appendItem -object $my_ixHttpAction
```

Step 5: Define ixSubscriberNetTraffic

The `ixSubscriberNetTraffic` is a special type of `NetTraffic` that simulates the trafpatterns created by residential customers that receive voice, video, and data service (Triple-play) over a single physical connection (usually a cable or DSL connection).

A Subscriber `NetTraffic` allows you to control the interactions between protocols for each user. This produces a traffic pattern that more accurately reproduces the pattern created by actual triple-play customers.

This involves:

- Creation of an `ixSubscriber NetTraffic` instance
- Configuring an `ixBandwidthLimit` object
- The rest is similar to `ixNetTraffic` described in Step 4: Define the `NetTraffic` (on page 50).

Step 6: Define the NetworkGroup

This step involves the global network configuration.

This involves:

- Creation of an `ixNetworkGroup` client and server network instance
- Clearing the global plugins list

```
set Network1 [::IxLoad new ixNetworkGroup $chassisChain]  
$Network1 config \  
-comment "" \  
-name "Network1" \  
-macMappingMode 1 \  
-linkLayerOptions 0  
  
$Network1 globalPlugins.clear
```

Step 7: Define the NetworkGroup

This step involves the network stack configuration.

This involves:

- Creating the network stack, including any extension protocols, appending the network stack plugins to the global plugin list, and then configuring them.
- Configuring the global settings (Dynamic Control plane)

```
set Filter [::IxLoad new ixNetFilterPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $Filter
```

```
$Filter config \
  -all                false \
  -pppoecontrol       false \
  -isis               false \
  -name               "Filter" \
  -auto               true \
  -udp                "" \
  -tcp                "" \
  -mac                "" \
  -pppoenetwork       false \
  -ip                 "" \
  -icmp               ""
```

```
set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP
```

```
$GratARP config \
  -enabled            true \
  -name               "GratARP"
```

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

```
$TCP config \
  -name               "TCP" \
  -tcp_orphan_retries 0 \
  -tcp_max_tw_buckets 180000 \
  -tcp_wmem_default   4096 \
  -tcp_low_latency    0 \
  -tcp_rmem_min       4096 \
  -tcp_adv_win_scale  2 \
  -tcp_wmem_min       4096 \
  -tcp_port_min       1024 \
  -tcp_stdurg         false \
  -tcp_port_max       65535 \
  -tcp_fin_timeout    60 \
  -tcp_no_metrics_save false \
  -tcp_dsack          true \
  -tcp_mem_high       49152 \
  -tcp_frto           0 \
  -tcp_app_win        31 \
  -ip_no_pmtu_disc    false \
  -tcp_window_scaling false \
  -tcp_max_orphans    8192 \
  -tcp_mem_pressure   32768 \
```



```
-tcp_syn_retries                5

set DNS [::IxLoad new ixNetDnsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $DNS

$DNS config \
  -domain                        "" \
  -name                          "DNS" \
  -timeout                       5

$DNS hostList.clear

$DNS searchList.clear

$DNS nameServerList.clear

set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
  -teardownInterfaceWithUser     false \
  -name                          "Settings" \
  -interfaceBehavior            0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
  -negotiationType              "master" \
  -negotiateMasterSlave        true

$Ethernet_1 config \
  -advertise10Full              true \
  -name                         "Ethernet-1" \
  -autoNegotiate                true \
  -advertise100Half             true \
  -advertise10Half              true \
  -speed                        "k100FD" \
  -advertise1000Full            true \
  -advertise100Full             true \
  -cardElm                      $my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

set MAC_VLAN_2 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_2

$MAC_VLAN_2 config \
  -name                         "MAC/VLAN-2"

$MAC_VLAN_2 childrenList.clear

set IP_3 [::IxLoad new ixNetIPv4V6Plugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_2 childrenList.appendItem -object $IP_3

$IP_3 config \
  -name                         "IP-3"

$IP_3 childrenList.clear

$IP_3 extensionList.clear
```

```

$MAC_VLAN_2 extensionList.clear

$Ethernet_1 extensionList.clear

#####
# Setting the ranges starting with the plugin on top of the stack
#####
$IP_3 rangeList.clear

set IP_R3 [::IxLoad new ixNetIpV4V6Range]
# ixNet objects needs to be added in the list before they are configured!
$IP_3 rangeList.appendItem -object $IP_R3

$IP_R3 config \
    -count 1 \
    -name "IP-R3" \
    -gatewayAddress "0.0.0.0" \
    -enabled true \
    -autoMacGeneration true \
    -mss 1460 \
    -incrementBy "0.0.0.1" \
    -prefix 16 \
    -gatewayIncrement "0.0.0.0" \
    -gatewayIncrementMode "perSubnet" \
    -generateStatistics false \
    -ipAddress "10.10.0.4" \
    -ipType "IPv4"

set MAC_R2 [$IP_R3 getLowerRelatedRange "MacRange"]

$MAC_R2 config \
    -count 1 \
    -name "MAC-R2" \
    -enabled true \
    -mtu 1500 \
    -mac "00:0A:0A:00:04:00" \
    -incrementBy "00:00:00:00:00:01"

set VLAN_R1 [$IP_R3 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
    -incrementStep 1 \
    -uniqueCount 4094 \
    -name "VLAN-R1" \
    -innerIncrement 1 \
    -innerUniqueCount 4094 \
    -enabled true \
    -innerFirstId 1 \
    -increment 1 \
    -priority 1 \
    -firstId 1 \
    -innerIncrementStep 1 \
    -idIncrMode 2 \
    -innerEnable false \
    -innerPriority 1

```



Step 8: Define the NetworkRange

This step involves the creation of IP and MAC addresses.

This involves:

- Creation of an `ixNetworkRange` instance
- Configuring the network range used in a network

```
set Network_Range_1_in_client_network__198_18_0_1_100_ [::IxLoad new ixNetworkRange]
$Network_Range_1_in_client_network__198_18_0_1_100_ config \
-rangeType "Ethernet" \
-vlanPriority 0 \
-vlanEnable 0 \
-innerVlanPriority 0 \
-innerVlanUniqueCount 4094 \
-innerVlanIncrStep 1 \
-networkMask "255.255.0.0" \
-vlanIncrStep 1 \
-gateway "0.0.0.0" \
-vlanIncrementMode "inner-first" \
-gatewayIncrStep "None" \
-mssEnable 0 \
-mss 1460 \
-enableStats false \
-firstMac "00:C6:12:00:01:00" \
-ipType 1 \
-type 0 \
-firstIp "198.18.0.1" \
-enable 1 \
-vlanId 1 \
-vlanCount 1 \
-ipCount 100 \
-vlanUniqueCount 4094 \
-macIncrStep "00:00:00:00:01:00" \
-name "Network Range 1 in client network
(198.18.0.1+100)" \
-innerVlanCount 1 \
-ipIncrStep "0.0.0.1" \
-innerVlanId 1 \
-innerVlanEnable false \
-rxBandwidthLimit $my_ixBandwidthLimit \
-txBandwidthLimit $my_ixBandwidthLimit1
```

Step 9: Define the ixTimeline

This object configures the time in the test when the activities in the NetTraffics come online, and how long they stay up for. It is also used to configure the test's objectives. This involves:

- Creating an instance of ixTimeline object
- Configuring the timeline and objectives

```
#####  
# Timeline1 for activities HTTPClient1  
#####  
set Timeline1 [::IxLoad new ixTimeline]  
$Timeline1 config \  
-rampUpValue           10 \  
-rampUpType            0 \  
-offlineTime          0 \  
-rampDownTime         20 \  
-standbyTime          0 \  
-iterations           1 \  
-rampUpInterval       1 \  
-sustainTime          20 \  
-timelineType         0 \  
-name                 "Timeline1"
```



Step 10: Prepare to Run the Test

In this step, we will perform all operations necessary before starting the actual test. This involves:

- Creating an instance of the `ixTestController`, defining where the results should be placed.
- Initializing the `statCollectorUtils`, by using its `Initialize` command.
- Clear all statistics with `ClearStats`.
- Add statistics that we are interested in via the `AddStat` command.
- Define a callback command to receive statistics update. A trivial routine is included in this example.

```
#-----
# Create a test controller bound to the previously allocated
# chassis chain. This will eventually run the test
# we created earlier.
#-----
set testController [::IxLoad new ixTestController -outputDir 1]
$testController setResultDir "[pwd]/RESULTS/simplehttpclientandserver"

#####
# Create the test controller to run the test
#####
set testController [::IxLoad new ixTestController -outputDir True]

$testController setResultDir "[pwd]/RESULTS/simpleHTTP_3"

set NS statCollectorUtils

set test_server_handle [$testController getTestServerHandle]
${NS}::Initialize -testServerHandle $test_server_handle

${NS}::ClearStats
$Test1 clearGridStats

set HTTP_Client_Per_URL_StatList { \
{"HTTP Client Per URL" "HTTP Aborted After Request" "kMax"} \
{"HTTP Client Per URL" "HTTP Aborted Before Request" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (400)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (401)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (403)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (404)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (407)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (408)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (4xx other)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (4xx)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (505)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (5xx other)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (5xx)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Aborted)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Bad Header)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Read)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Timeout)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Write)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Sent" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Successful" "kMax"} \
{"HTTP Client Per URL" "HTTP Responses Received With Match" "kMax"} \
{"HTTP Client Per URL" "HTTP Responses Received Without Match" "kMax"} \
}
set HTTP_Client_StatList { \
{"HTTP Client" "Client Hello Sent" "kMax"} \
{"HTTP Client" "HTTP Aborted After Request" "kMax"} \
{"HTTP Client" "HTTP Aborted Before Request" "kMax"} \
```

```

{"HTTP Client" "HTTP Bytes" "kMax"} \
{"HTTP Client" "HTTP Bytes Received" "kMax"} \
{"HTTP Client" "HTTP Bytes Sent" "kMax"} \
{"HTTP Client" "HTTP Concurrent Connections" "kMax"} \
{"HTTP Client" "HTTP Connect Time (ms)" "kAverageRate"} \
{"HTTP Client" "HTTP Connection Attempts" "kMax"} \
{"HTTP Client" "HTTP Connections" "kMax"} \
{"HTTP Client" "HTTP Content Bytes Received" "kMax"} \
{"HTTP Client" "HTTP Content Bytes Sent" "kMax"} \
{"HTTP Client" "HTTP Cookie headers Rejected - (Memory Overflow)" "kMax"} \
{"HTTP Client" "HTTP Cookies Received" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected - (Cookiejar Overflow)" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected - (Domain Match Failed)" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected - (Path Match Failed)" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected - (Probabilistic Reject)" "kMax"} \
{"HTTP Client" "HTTP Cookies Sent" "kMax"} \
{"HTTP Client" "HTTP Requests Failed" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (400)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (401)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (403)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (404)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (407)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (408)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (4xx other)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (4xx)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (505)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (5xx other)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (5xx)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Aborted)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Bad Header)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Read)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Timeout)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Write)" "kMax"} \
{"HTTP Client" "HTTP Requests Sent" "kMax"} \
{"HTTP Client" "HTTP Requests Successful" "kMax"} \
{"HTTP Client" "HTTP Session Timeouts (408)" "kMax"} \
{"HTTP Client" "HTTP Sessions Rejected (503)" "kMax"} \
{"HTTP Client" "HTTP Simulated Users" "kSum"} \
{"HTTP Client" "HTTP Time To First Byte (ms)" "kAverageRate"} \
{"HTTP Client" "HTTP Time To Last Byte (ms)" "kAverageRate"} \
{"HTTP Client" "HTTP Transactions" "kMax"} \
{"HTTP Client" "HTTP Transactions Active" "kMax"} \
{"HTTP Client" "HTTP Users Active" "kMax"} \
{"HTTP Client" "SSL Alerts Received" "kMax"} \
{"HTTP Client" "SSL Alerts Received (access_denied)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (bad_certificate)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (bad_record_mac)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (certificate_expired)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (certificate_revoked)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (certificate_unknown)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (close_notify)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (decode_error)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (decompression_failure)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (decrypt_error)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (decryption_failed)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (export_restriction)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (handshake_failure)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (illegal_parameter)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (insufficient_security)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (internal_error)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (no_certificate)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (no_renegotiation)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (protocol_version)" "kMax"} \
{"HTTP Client" "SSL Alerts Received (record_overflow)" "kMax"} \

```

```
{ "HTTP Client" "SSL Alerts Received (unexpected_message)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (unknown_ca)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (unsupported_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (user_canceled)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (access_denied)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (bad_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (bad_record_mac)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (certificate_expired)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (certificate_revoked)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (certificate_unknown)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (close_notify)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (decode_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (decompression_failure)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (decrypt_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (decryption_failed)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (export_restriction)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (handshake_failure)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (illegal_parameter)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (insufficient_security)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (internal_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (no_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (no_renegotiation)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (protocol_version)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (record_overflow)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (unexpected_message)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (unknown_ca)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (unsupported_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (user_canceled)" "kMax" } \  
{ "HTTP Client" "SSL Bytes Received" "kMax" } \  
{ "HTTP Client" "SSL Bytes Sent" "kMax" } \  
{ "HTTP Client" "SSL Concurrent Sessions" "kMax" } \  
{ "HTTP Client" "SSL Errors Received" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (bad certificate)" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (no certificate)" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (no cipher)" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (undefined error)" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (unsupported certificate)" "kMax" } \  
{ "HTTP Client" "SSL Errors Sent" "kMax" } \  
{ "HTTP Client" "SSL Errors Sent (bad certificate)" "kMax" } \  
{ "HTTP Client" "SSL Errors Sent (no certificate)" "kMax" } \  
{ "HTTP Client" "SSL Errors Sent (no cipher)" "kMax" } \  
{ "HTTP Client" "SSL Errors Sent (undefined error)" "kMax" } \  
{ "HTTP Client" "SSL Errors Sent (unsupported certificate)" "kMax" } \  
{ "HTTP Client" "SSL Negotiation Finished Successfully" "kMax" } \  
{ "HTTP Client" "SSL Session Reuse Failed" "kMax" } \  
{ "HTTP Client" "SSL Session Reuse Success" "kMax" } \  
{ "HTTP Client" "SSL Throughput Bytes" "kMax" } \  
{ "HTTP Client" "Server Hello Received" "kMax" } \  
{ "HTTP Client" "TCP Accept Queue Entries" "kMax" } \  
{ "HTTP Client" "TCP Connection Requests Failed" "kMax" } \  
{ "HTTP Client" "TCP Connections Established" "kMax" } \  
{ "HTTP Client" "TCP Connections in CLOSE STATE" "kMax" } \  
{ "HTTP Client" "TCP Connections in CLOSE-WAIT State" "kMax" } \  
{ "HTTP Client" "TCP Connections in CLOSING State" "kMax" } \  
{ "HTTP Client" "TCP Connections in ESTABLISHED State" "kMax" } \  
{ "HTTP Client" "TCP Connections in FIN-WAIT-1 State" "kMax" } \  
{ "HTTP Client" "TCP Connections in FIN-WAIT-2 State" "kMax" } \  
{ "HTTP Client" "TCP Connections in LAST-ACK State" "kMax" } \  
{ "HTTP Client" "TCP Connections in LISTENING State" "kMax" } \  
{ "HTTP Client" "TCP Connections in SYN-RECEIVED State" "kMax" } \  
{ "HTTP Client" "TCP Connections in SYN-SENT State" "kMax" } \  
{ "HTTP Client" "TCP Connections in TIME-WAIT State" "kMax" } \  
{ "HTTP Client" "TCP FIN Received" "kMax" } \  
{ "HTTP Client" "TCP FIN Sent" "kMax" } \  

```

```

{"HTTP Client" "TCP FIN-ACK Received" "kMax"} \
{"HTTP Client" "TCP FIN-ACK Sent" "kMax"} \
{"HTTP Client" "TCP Listen Queue Drops" "kMax"} \
{"HTTP Client" "TCP Resets Received" "kMax"} \
{"HTTP Client" "TCP Resets Sent" "kMax"} \
{"HTTP Client" "TCP Retries" "kMax"} \
{"HTTP Client" "TCP SYN Failed" "kMax"} \
{"HTTP Client" "TCP SYN Sent" "kMax"} \
{"HTTP Client" "TCP SYN-ACK Sent" "kMax"} \
{"HTTP Client" "TCP SYN_SYN-ACK Received" "kMax"} \
{"HTTP Client" "TCP Timeouts" "kMax"} \
}

set HTTP_Server_Per_URL_StatList { \
{"HTTP Server Per URL" "HTTP Requests Failed" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Failed (404)" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Failed (50x)" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Failed (Write Error)" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Received" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Successful" "kMax"} \
}

set HTTP_Server_StatList { \
{"HTTP Server" "Client Hello Received" "kMax"} \
{"HTTP Server" "HTTP Bytes Received" "kMax"} \
{"HTTP Server" "HTTP Bytes Sent" "kMax"} \
{"HTTP Server" "HTTP Content Bytes Received" "kMax"} \
{"HTTP Server" "HTTP Content Bytes Sent" "kMax"} \
{"HTTP Server" "HTTP Cookies Received" "kMax"} \
{"HTTP Server" "HTTP Cookies Received With Matching ServerID" "kMax"} \
{"HTTP Server" "HTTP Cookies Received With Non-matching ServerID" "kMax"} \
{"HTTP Server" "HTTP Cookies Sent" "kMax"} \
{"HTTP Server" "HTTP Requests Failed" "kMax"} \
{"HTTP Server" "HTTP Requests Failed (404)" "kMax"} \
{"HTTP Server" "HTTP Requests Failed (50x)" "kMax"} \
{"HTTP Server" "HTTP Requests Failed (Write Error)" "kMax"} \
{"HTTP Server" "HTTP Requests Received" "kMax"} \
{"HTTP Server" "HTTP Requests Successful" "kMax"} \
{"HTTP Server" "HTTP Session Timeouts (408)" "kMax"} \
{"HTTP Server" "HTTP Sessions Rejected (503)" "kMax"} \
{"HTTP Server" "HTTP Transactions Active" "kMax"} \
{"HTTP Server" "SSL Alerts Received" "kMax"} \
{"HTTP Server" "SSL Alerts Received (access_denied)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (bad_certificate)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (bad_record_mac)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (certificate_expired)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (certificate_revoked)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (certificate_unknown)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (close_notify)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (decode_error)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (decompression_failure)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (decrypt_error)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (decryption_failed)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (export_restriction)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (handshake_failure)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (illegal_parameter)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (insufficient_security)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (internal_error)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (no_certificate)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (no_renegotiation)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (protocol_version)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (record_overflow)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (unexpected_message)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (unknown_ca)" "kMax"} \
{"HTTP Server" "SSL Alerts Received (unsupported_certificate)" "kMax"} \
}

```

```
{ "HTTP Server" "SSL Alerts Received (user_canceled)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (access_denied)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (bad_certificate)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (bad_record_mac)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (certificate_expired)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (certificate_revoked)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (certificate_unknown)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (close_notify)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (decode_error)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (decompression_failure)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (decrypt_error)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (decryption_failed)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (export_restriction)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (handshake_failure)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (illegal_parameter)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (insufficient_security)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (internal_error)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (no_certificate)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (no_renegotiation)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (protocol_version)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (record_overflow)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (unexpected_message)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (unknown_ca)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (unsupported_certificate)" "kMax" } \  
{ "HTTP Server" "SSL Alerts Sent (user_canceled)" "kMax" } \  
{ "HTTP Server" "SSL Bytes Received" "kMax" } \  
{ "HTTP Server" "SSL Bytes Sent" "kMax" } \  
{ "HTTP Server" "SSL Concurrent Sessions" "kMax" } \  
{ "HTTP Server" "SSL Errors Received" "kMax" } \  
{ "HTTP Server" "SSL Errors Received (bad certificate)" "kMax" } \  
{ "HTTP Server" "SSL Errors Received (no certificate)" "kMax" } \  
{ "HTTP Server" "SSL Errors Received (no cipher)" "kMax" } \  
{ "HTTP Server" "SSL Errors Received (undefined error)" "kMax" } \  
{ "HTTP Server" "SSL Errors Received (unsupported certificate)" "kMax" } \  
{ "HTTP Server" "SSL Errors Sent" "kMax" } \  
{ "HTTP Server" "SSL Errors Sent (bad certificate)" "kMax" } \  
{ "HTTP Server" "SSL Errors Sent (no certificate)" "kMax" } \  
{ "HTTP Server" "SSL Errors Sent (no cipher)" "kMax" } \  
{ "HTTP Server" "SSL Errors Sent (undefined error)" "kMax" } \  
{ "HTTP Server" "SSL Errors Sent (unsupported certificate)" "kMax" } \  
{ "HTTP Server" "SSL Negotiation Finished Successfully" "kMax" } \  
{ "HTTP Server" "SSL Session Reuse Failed" "kMax" } \  
{ "HTTP Server" "SSL Session Reuse Success" "kMax" } \  
{ "HTTP Server" "SSL Throughput Bytes" "kMax" } \  
{ "HTTP Server" "Server Hello Sent" "kMax" } \  
{ "HTTP Server" "TCP Accept Queue Entries" "kMax" } \  
{ "HTTP Server" "TCP Connection Requests Failed" "kMax" } \  
{ "HTTP Server" "TCP Connections Established" "kMax" } \  
{ "HTTP Server" "TCP Connections in CLOSE STATE" "kMax" } \  
{ "HTTP Server" "TCP Connections in CLOSE-WAIT State" "kMax" } \  
{ "HTTP Server" "TCP Connections in CLOSING State" "kMax" } \  
{ "HTTP Server" "TCP Connections in ESTABLISHED State" "kMax" } \  
{ "HTTP Server" "TCP Connections in FIN-WAIT-1 State" "kMax" } \  
{ "HTTP Server" "TCP Connections in FIN-WAIT-2 State" "kMax" } \  
{ "HTTP Server" "TCP Connections in LAST-ACK State" "kMax" } \  
{ "HTTP Server" "TCP Connections in LISTENING State" "kMax" } \  
{ "HTTP Server" "TCP Connections in SYN-RECEIVED State" "kMax" } \  
{ "HTTP Server" "TCP Connections in SYN-SENT State" "kMax" } \  
{ "HTTP Server" "TCP Connections in TIME-WAIT State" "kMax" } \  
{ "HTTP Server" "TCP FIN Received" "kMax" } \  
{ "HTTP Server" "TCP FIN Sent" "kMax" } \  
{ "HTTP Server" "TCP FIN-ACK Received" "kMax" } \  
{ "HTTP Server" "TCP FIN-ACK Sent" "kMax" } \  
{ "HTTP Server" "TCP Listen Queue Drops" "kMax" } \  

```

```

{"HTTP Server" "TCP Resets Received" "kMax"} \
{"HTTP Server" "TCP Resets Sent" "kMax"} \
{"HTTP Server" "TCP Retries" "kMax"} \
{"HTTP Server" "TCP SYN Failed" "kMax"} \
{"HTTP Server" "TCP SYN Sent" "kMax"} \
{"HTTP Server" "TCP SYN-ACK Sent" "kMax"} \
{"HTTP Server" "TCP SYN_SYN-ACK Received" "kMax"} \
{"HTTP Server" "TCP Timeouts" "kMax"} \
}

set statList [concat \
$HTTP_Client_Per_URL_StatList \
$HTTP_Client_StatList \
$HTTP_Server_Per_URL_StatList \
$HTTP_Server_StatList \
]

set count 1
foreach statItem $statList {
set caption [format "Watch_Stat_%s" $count]
set statSourceType [lindex $statItem 0]
set statName [lindex $statItem 1]
set aggregationType [lindex $statItem 2]

${NS}::AddStat \
    -caption $caption \
    -statSourceType $statSourceType \
    -statName $statName \
    -aggregationType $aggregationType \
    -filterList {}

incr count
}

proc ::my_stat_collector_command {args} {
puts "======"
puts "INCOMING STAT RECORD >>> $args"
puts "Len = [llength $args]"
puts [lindex $args 0]
puts [lindex $args 1]
puts "======"
}
${NS}::StartCollector -command ::my_stat_collector_command

```



Step 11: Start the Test

In this step, we'll actually start and stop the test. The steps involved are:

- Start the statistics collector using `StartCollector`.
- Use the `ixTestController` instance to run the test.
- Wait for the test complete.
- Stop the statistics collector using `StopCollector`.
- Disconnect from the remote server. See `Initial Overhead` for more details.

```

${NS}::StartCollector -command ::my_stat_collector_command

#-----
# Run the test
#-----
$testController run $test

#-----
# have the script (v)wait until the test is over
#-----
vwait ::ixTestControllerMonitor;
puts $::ixTestControllerMonitor

#-----
# Stop the collector (running in the tcl event loop)
#-----
${NS}::StopCollector

#-----
# Cleanup
#-----
$testController generateReport -detailedReport 1 -format "PDF;HTML"

$testController releaseConfigWaitFinish
::IxLoad delete $chassisChain
::IxLoad delete $clnt_network
::IxLoad delete $svr_network
::IxLoad delete $clnt_traffic
::IxLoad delete $svr_traffic
::IxLoad delete $clnt_t_n_mapping
::IxLoad delete $svr_t_n_mapping
::IxLoad delete $test
::IxLoad delete $testController
::IxLoad delete $logger
::IxLoad delete $logEngine

#-----
# Disconnect
#-----
}] {
    puts $errorInfo
}
#
# Disconnect/Release application lock
#
::IxLoad disconnect
```

Stopping a Test by Pressing Enter

You can configure a test to stop when the ENTER key. See below is the sample code.

For an example of a complete script that stops when ENTER is pressed, see the sample script `C:\Program Files\Ixia\IxLoad\Client\TclApi\Samples\simplehttp-abortrun.tcl`.

```
-----
# configure stdin for polling
#-----
fconfigure stdin -blocking 0 -buffering none
# wait for the first sample or test stop
while {${:ixTestControllerMonitor} == "" && [read stdin] == ""} {
    after 100 set wakeup 1
    # the script must call vwait or update while test runs
    # to keep TCL event loop going. Otherwise, no stat collector
    # callbacks will be made, and ixTestControllerMonitor will
    # never be set.
    vwait wakeup
}
#-----
# if aborted, then stop test gracefully
#-----
if {${:ixTestControllerMonitor} == ""} {
    puts ""
    puts "!!!Aborting test at earliest opportunity!!!"
    puts ""
    # stop the run
    $testController stopRun
    #
    # (v)wait until the test really stops
    #
    vwait ::ixTestControllerMonitor
    puts ${:ixTestControllerMonitor}
}
#
# Stop the collector
#
${NS}::StopCollector
#-----
# Cleanup
#-----
```

Running an IxLoad Tcl Script

The following sections describe how to run an IxLoad Tcl script test.

Windows (on page 66) describes how to run a script on Windows.

Unix / Linux (on page 66) describes how to run a script on Unix/Linux.



Windows

To run an IxLoad Tcl script, you can use either of the following Tcl shells:

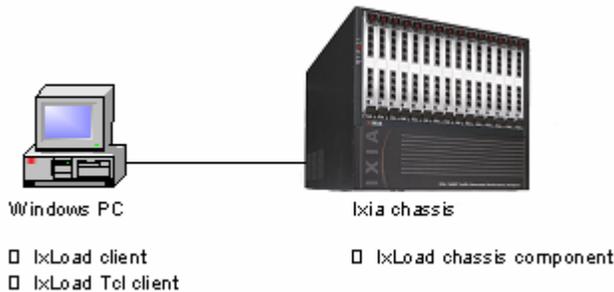
- Wish shell: C:\Program Files\Ixia\Tcl\<version>\bin\wish.exe
- Tcl shell: C:\Program Files\Ixia\Tcl\<version>\bin\tclsh.exe

The IxLoad TCL code resides under C:\Program Files\Ixia\IxLoad\<version>\TclScripts. The code in the `setup_ixload_paths.tcl` script used in earlier releases is no longer used. Instead, the current method used by all Ixia applications is to source `TclScripts\bin\IxiaWish.tcl` for the application, and follow that with a `package require` command.



Note: If more than one version of IxLoad is installed, the `package require` command uses the highest-numbered version. To select a different version, include the complete version number in the command. For example:

```
package require ixload 4.20.0.88
```



See Windows (on page 31) for an example of how to run a script on Windows.

Unix / Linux

To run an IxLoad Tcl script on Unix/Linux:

- You must use the `ixwish` shell or `IxTclsh` provided in the `bin` directory of the IxOS installation.
- You must install the IxOS Unix Tcl Client, and the IxLoad Unix Tcl client.

The `package require` command used in the sample scripts will only succeed if you have a version of IxLoad installed on the Unix/Linux machine that matches the one you request in the `package require` statement, and the environment is set up correctly

For multiversion support on Unix / Linux, the installer creates a `lib/IxLoad<ver>` directory with a `pkgIndex.tcl` for each version of IxLoad that you install. The version number is the same one reported to TCL when the package is required. All normal `package require` logic applies to this.



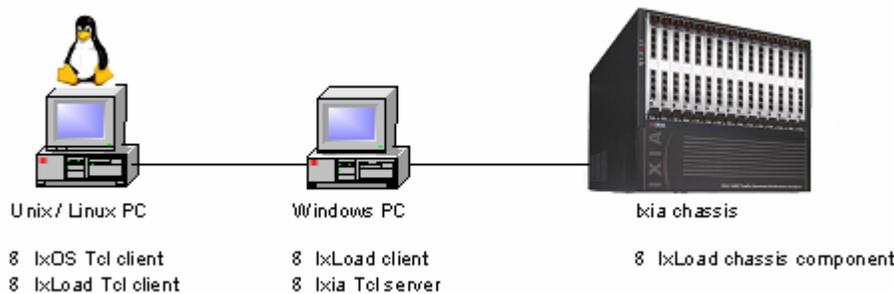
Note: If more than one version of IxLoad is installed, the `package require` command uses the highest-numbered version. To select a different version, include the complete version number in the command. For example:

```
package require ixload 4.20.0.88
```

- You must install and run the IxLoad client on a Windows machine. When you run the TCL scripts on the Unix/Linux host, the TCL scripts are sent to the Windows machine and executed there. The results are also saved on the Windows host.

Ixia Tcl Server must be running on a Windows-based host, not the chassis. The Tcl Server machine is specified in a call to `connect` in the `::IxLoad` command. The `::IxLoad connect` call will only succeed if the specified client is:

- Running a compatible Tcl Server (release notes will detail the IxOS version that is compatible with a particular IxLoad version),
- Has the identical version of IxLoad installed that was actually loaded by the `package require` statement on the Unix machine (i.e. returned by the Unix `package require IxLoad` command).
- The `::IxLoad connect` command also performs the IxOS `ixConnectToTclServer`, so a separate call is not necessary to access the `ixTclHal` commands on the client machine.



See Unix/Linux (on page 34) for an example of how to run a script on Unix/Linux.

Maximum Numbers of Scripts That Can Be Run

A maximum of four instances of IxLoad can run on a Windows client PC.

- Each copy of the IxLoad GUI counts as 2 instances.
- Each Tcl script counts as 1 instance.

If you receive the following error:

```
Error: exceptions.Exception: Already running maximum allowed copies of IxLoad.
```



the most likely cause is running more scripts than allowed (that is, from multiple shells or in the background).

Unix Tcl scripts are executed on the Windows client PC. If a Unix script is terminated (killed), the Windows client might take a few seconds notice and kill the corresponding `tclsh`, but it still counts as a copy of IxLoad until the `tclsh` is killed.

Scripts running on the Windows client do not launch their own `tclsh`, but still count as an instance. If a Windows Tcl script running in `wish` crashes during execution, it still counts as a running copy until the `wish` shell is killed.

Modifying Older Scripts

Multi-version support enables you to install multiple versions of IxLoad on the same client PC. Multi-version support was added to IxLoad beginning with release 3.40. If you want to run a non-multiversion (pre-3.40) script in a multi-version release, you must modify it.



To modify a multi-version script:

1. Open the script in an editor.
2. Remove following code from the old script:

```
if {$::tcl_platform(platform) == "windows"} {  
  package require registry 1  
  
  set ::_IXLOAD_INSTALL_ROOT [registry get {HKEY_LOCAL_MACHINE\Software\Ixia  
  Communications\IxLoad\InstallInfo} HOMEDIR]  
  
  set ::_IXLOAD_PKG_DIR [file join $::_IXLOAD_INSTALL_ROOT Client tclxt teepee  
  stage]  
  
  lappend ::auto_path $::_IXLOAD_PKG_DIR  
}
```

3. Replace the removed code with either of the following lines (replace `<version>` with the IxLoad version number):

```
source "C:\\Program  
Files\\Ixia\\IxLoad\\<version>\\TclScripts\\bin\\IxiaWish.tcl"  
  
source "C:\\Program  
Files\\Ixia\\IxLoad\\<version>\\TclScripts\\bin\\IxiaWish.tcl"
```

4. Save and close the file.

API Description

The following sections of this chapter are an overview of the Tcl API, by topic. They are described in the same order as the suggested steps in *Building an IxLoad Test*.

- `Network Commands`—Defines the client and server networks.
 - `ixNetworkGroup`—Configure the global network.
 - `ixChassisChain`—Indicates the chassis that are used in the test.

- DUT Commands
 - `ixDut`—Creates a DUT entry.
- Traffic Commands
 - `ixNetTraffic`—Configures client and server traffic.
 - `activityList`—Generates traffic for one side of a particular protocol.
 - `ixTimeline`—Configures the time in the test when the activities in the NetTraffics come online, and how long they stay up for. It is also used to configure the test's objectives.
- Test Structure Commands
 - `ixTimeline`—Configures the timeline and objectives for client and server.
 - `ixTest`—Creates a complete test structure.
 - `ixView`—Configures capture options in test repositories.
 - `ixTrafficFlow`—Lists the test scenario.
 - `ixTrafficColumn`—A container of `ixNetTraffic` and `ixDut` objects.
- Test Operation Commands
 - `ixTestController`—Starts and stops test.
 - `ixTestControllerMonitor`—A global variable to watch for test completion.
 - `statCollectorUtils`—Utilities for collecting statistics.

Reference pages for each of the IxLoad Tcl API commands are included in the following chapters:

- `IxLoad Tcl API Commands`. This includes a discussion of the most often used commands.
- `IxLoad Tcl API Internal Commands`. This includes a discussion of the behind-the-scenes commands on which most other commands are based.
- Each of the remaining chapters describes one of the supported protocols.

The remainder of this section is an overview of these commands, including brief descriptions of command operation, options, and subcommands.

Network Commands

The commands in this section are the high-level used to create the networks used to support client and server traffic. There are minor differences between client and server net

The bulk of the network-related commands are described in the Network Stack API section.



ixNetworkGroup

The `ixNetworkGroup` command is used to construct a client or server network, which is used as part of an `ixNetTraffic` object. A chassis chain object, as created in the `ixChassisChain` command, must be used in the construction of this object.

A list of network ranges, as defined in the `ixNetworkRange` object is associated with the client network. Network ranges are added to the client network through the use of the `networkRangeList.appendItem` command.

A list of Ixia ports is also associated with the network through the `portList` option.

If an emulated router is to be used, a list of IP ranges for the router is also associated with the network through the `emulatedRouterIpAddressPool` option. The pool is defined in the `ixEmulatedRouterIpAddressRange` object. These are added to the object through the use of the `em` command.

Refer to `ixNetworkGroup` on page 4-79 for a full description of this command. The important subcommands and options of this command are listed below.

ixClientNetwork Subcommands

Subcommand	Usage
<code>checkConfig</code>	Checks the configuration of this object, raising an exception in the case of an error.
<code>reset</code>	Disassociates the network from all of the Ixia ports used in this network.

ixClientNetwork Options

Option	Usage
<code>name</code>	The name associated with this object.
<code>networkRangeList</code>	The networks that are defined for this object—a list of <code>ixNetworkRange</code> objects.
<code>portList</code>	The Ixia ports that will carry traffic for this network.
<code>cardType</code>	The card type for all of the ports in this network. Cards of a similar type must be used for all ports in a network. The <code>cardType</code> option is now used only for error/diag messages, and is automatically selected. Please refer to <code>cardType</code> on page 4-93 for the list of card types.

Option	Usage
macMappingMode	Indicates whether one MAC address will be associated with each IP address or with each Ixia port. The use of the latter option indicates that an emulated router is to be used.
emulatedRouterIpAddressPool	If the macMappingMode indicates that one MAC is used per port, then this is a list of addresses ranges— a list of ixEmulatedRouterIpAddressRange objects.
emulatedRouterGateway	If the macMappingMode indicates that one MAC is used per port, then this is the gateway for the emulated router.
dnsParameters	An object of type ixDns, which defines DNS operation for this network.
arpSettings	An object of type ixArpSettings, which defines ARP operation for this network.
tcpParameters	An object of type ixTcpParameters, which defines TCP options for this network.
impairment	An object of type ixImpairment, which Impairs one or more types of traffic from a client and server network.

ixChassisChain

Before defining client and server networks, it is necessary to define a chassis chain. This command is used to construct a chain of Ixia chassis, whose ports may be used in the ixNetworkGroup command. Chassis are assigned chassis IDs starting at 1; these are used in the network commands to define the chassis associated with the port.

Refer to ixChassisChain on page 4-5 for a full description of this command. The important subcommands of this command are listed in the table below.

Subcommand	Usage
addChassis	Adds a chassis, by name or address, to the chassis chain.
setLoginName getLoginName	Sets and retrieves the user login name.
isValidChassisName	Checks to see whether a chassis name/address is valid.
getChassisNames	Returns the names of all of the chassis, ordered by their chassis IDs.
deleteChassisByName	Deletes a chassis, by name, from the list. The IDs of other chassis remain unaffected.



Subcommand	Usage
refresh refreshChassis	Rereads chassis information from one or all chassis.

ixEmulatedRouterIpAddressRange

The `ixEmulatedRouterIpAddressRange` command is used to construct an list of IP addresses assigned on a per-port basis for emulated routers, as used in the `ixNetworkGroup` command.

Refer to `ixEmulatedRouterIpAddressRange` on page 4-91 for a full description of this command. The important options of this command are listed below.

ixEmulatedRouterIpAddressRange Options

Option	Usage
enable	Enables the use of this address range.
ipType	Type of address (IPv4 or IPv6).
firstIp lastIp	Controls the range of IP addresses generated.
networkMask	The network mask for the IP addresses.

ixDns

The `ixDns` command is used to define DNS behavior on a network. A DNS object is set through the `dnsParameters` option of the `ixNetworkGroup` object.

Refer to `ixDns` on page 4-34 for a full description of this command. The important options of this command are listed below:

ixDns Options

Option	Usage
enable	Enables the use of this DNS specification.
serverList	A list of DNS servers to check at run time. This list consists of items of type <code>ixDnsServerItem</code> .
suffixList	A list of DNS suffixes to add to partial host names. This list consists of items of type <code>ixDnsSuffixList</code> .
cacheTimeout	The time-out value used for cached DNS lookups.

ixDnsServerItem

The `ixDnsServerItem` command is used to define a DNS server on a network. A DNS server item object is appended to the `serverList` option of the `ixDns` object. For example,

```
set dns [::IxLoad new ixDns options...]  
$test.dns.serverList appendItem \  
-data192.168.3.1
```

Refer to `ixDnsServerItem` on page 4-36 for a full description of this command. The important options of this command are listed below.

ixDnsServerItem Options

Option	Usage
data	The IP address of a DNS server.

ixDnsSuffixList

The `ixDnsSuffixItem` command is used to define a DNS suffix. A DNS suffix item object is appended to the `suffixList` option of the `ixDns` object. For example,

```
set dns [::IxLoad new ixDns options...]  
$test.dns.suffixList appendItem \  
-data".ixiacom.com"
```

Refer to `ixDnsServerItem` on page 4-36 for a full description of this command. The important options of this command are:

ixDnsSuffixItem Options

Option	Usage
data	A domain name suffix.

ixTcpParameters

The `ixTcpParameters` command is used to define TCP options on a network. A TCP parameters object is set in the `tcpParameters` option of an `ixNetworkGroup` object.

Refer to `ixTcpParameters` on page 4-47 for a full description of this command. A wide range of low level TCP options are exposed in this command.

ixEmulatedRouterIpAddressRange

The `ixEmulatedRouterIpAddressRange` command is used to construct an list of IP addresses assigned on a per-port basis for emulated routers, as used in the `ixNetworkGroup` command for both client and server.

Refer to `ixEmulatedRouterIpAddressRange` on page 4-91 for a full description of this command.



DUT Commands

In many cases, it is not necessary to define your DUT in an IxLoad test. Two cases are necessary, however:

- If your DUT is a Server Load Balancer (SLB) and the IP address of the DUT itself is the destination of client requests.
- If your DUT is a caching device, and direct server return is desired.

ixDut

The `ixDut` command is used to define a DUT used in the test. The DUTs are used to resolve symbolic references to them in traffic destinations in the various protoagents. It also controls several DUT specific features.

Refer to `ixDut` on page 4-81 for a full description of this command. The impropoptions of this command are listed below.

ixDut Options

Option	Usage
<code>name</code>	The name associated with the DUT.
<code>type</code>	The type of the DUT—external server, SLB or firewall.
<code>ipAddress</code>	The IP address, virtual IP address, or host name to be used to access the DUT.
<code>serverNetwork</code>	If the DUT is an SLB, this is the network that will be balanced.
<code>enableDirectServer Return</code>	If the DUT is an SLB, this option allows balanced servers to send their return traffic directly back to the source of the request.

Traffic Commands

The commands in this section relate to the generation of traffic by clients and the handling of traffic by servers.

ixNetTraffic

The `ixNetTraffic` command is used to configure client or server traffic. Two `sepaixNetTraffic` objects have to be created for client and server traffic. The `ixNetTraffic` configuration also declares the `ixNetworkGroup` object. The `activityList` is appended to the `ixNetTraffic` object.

Refer to `ixNetTraffic` on page 4-72 for a full description of this command.

activityList

Generates traffic for one side of a particular protocol. For example, an HTTP cliActivity generates HTTP client requests, simulating a web browser. The `activityList` is appended to the `ixNetTraffic` object.

Refer to `activityList` on page 4-76 for a full description of this command.

ixTimeline

Configures the time in the test when the activities in the `NetTraffics` come online, and how long they stay up for. It is also used to configure the test's objectives. The `ixTimeline` object is added to the `timeline` options of the `activityList` config.

Refer to `ixTimeline` on page 4-67 for a full description of this command.

Test Structure Commands

The commands in this section co-ordinate networks with traffic into communities and then communities into an entire test structure. These commands also define the operational parameters of the test.

ixTest

The `ixTest` command is used to construct a complete `IxLoad` test structure. It contains a list of client traffic-network and server traffic-network mappings, called communities. In addition to the two lists, several options control global operation. An `ixTest` command is used in conjunction with a `ixTestController` to operate the test and collect statistics.

A test is generally built via:

```
set test [::IxLoad new IxTest -name "my_test"]
$test clientCommunityList.appendItem -object $my_clients
$test serverCommunityList.appendItem -object $my_servers
```

Refer to `ixTest` on page 4-50 for a full description of this command.

ixView

Configures capture options in test repositories. It is added as an object instance to the `captureViewOptions` in `ixTest`.

Refer to `ixView` on page 4-89 for a full description of this command.

ixTrafficFlow

Lists the test scenario. The `ixTrafficFlow` command is used to list the test scenario. The `ixTrafficFlow` object is appended to the `ixTest` object.

Refer to `ixTrafficFlow` on page 4-77 for a full description of this command.

ixTrafficColumn

This is a container of `ixNetTraffic` and `ixDut` objects.

Refer to `ixTrafficColumn` on page 4-78 for a full description of this command.



Test Operation Commands

The commands in this section relate to the actual test and statistics gathering operations.

ixTestController

The `ixTestController` command is used to setup, start, and stop an IxLoad test. It references the `ixTest` object in its `run` subcommand.

Refer to `ixTestController` on page 4-53 for a full description of this command. The important subcommands and options of this command are listed below:

ixTestController Subcommands

Subcommand	Usage
<code>run</code>	Run the test. The name of an <code>ixTest</code> object is a required argument.
<code>setResultDir</code>	Specifies the location of where CSV files from the run are saved.
<code>isBusy</code>	Returns <code>true</code> while the test is running.
<code>getTestServerHandle</code>	Returns a value necessary for the statistics collection routines.
<code>releaseConfigWaitFinish</code>	Releases all IxLoad configurations and waits for it to complete.
<code>generateReport</code>	Generates report from TCL.

ixTestController Options

Options	Usage
<code>outputDir</code>	This should be set to a non-null value if you wish to save statistics in CSV files during the run. The actual directory used is set in the <code>setResultDir</code> subcommand.

ixTestControllerMonitor

This is a global variable whose state may be used in a `vwait` to determine when a test has completed. Refer to `ixTestControllerMonitor` on page 4-58 for a full description of this command.

statCollectorUtils

The `statCollectorUtils` is a library containing several commands to gather statistics during a test run. Refer to `statCollectorUtils` on page 4-60 for a full description of this library. The important commands of this library are:

statCollectorUtils Commands

Subcommand	Usage
Initialize	Initializes the statistics utility package. Requires the result of a call to <code>ixTest getTestServerHandle</code> .
AddStat	Adds a statistic to the list of desired statistics to follow.
AddNetworkStat	Adds a dynamic range network statistic to the list of desired statistics to follow.
ClearStats	Clears the statistics values from any previous run.
StartCollector	Starts the statistics gathering process. The name of a user's callback command is passed in here.
StopCollector	Stops the statistics gathering process.



Debugging

During the normal operation of the Tcl API, only errors and warnings are logged. To increase the level of debugging, you should use the following code fragment:

```
set logtag "IxLoad-api"
set logName "simplehttpclientandserver"
set logger [::CMD new ixLogger $logtag 1]
set logEngine [$logger getEngine]
$logEngine setLevels $::ixLogger(kLevelDebug) $::ixLogger(kLevelInfo)
$logEngine setFile $logName 2 256 1
```

The above fragment specifies that the log file name is prefixed with `simplehttpclientandserver`. The actual log file name is generated as fol

```
logName-<instance number>-<log file number>.log
```

where "instance number" is the number assigned to your session, with the first session being 1 up to a maximum of 4. "log file number" is a two digit number which is usually 00. Long or complicated tests may produce more log data than will fit in a single file, in which case a file ending with 01 will also exist. Extremely large logs may cause the sequence to start over, overwriting the original contents of log 00.

If the link is down on any of the ports in the test, the Tcl API logs the error in the log files but it does not display an error in the wish console. Although IxLoad allows the test to enter the "Configured" state with a link down, it will not allow the test to run.

Logging Levels

In the code snippet, the following line defines an example of the settings of the `setLevels` API on the logger object.

```
$logEngine setLevels $::ixLogger(kLevelDebug) $::ixLogger(kLevelInfo)
```

The log levels are accessed using `$::ixLogger(kLevelxxx)`. The first value is the `file` level and the second value is the `console` level.

File Level: The file level should always be `kLevelDebug`. Otherwise, the log files will not contain enough information, in the event of a problem with the script.

Console Level: The second level is typically `kLevelInfo`, but can be set to the other levels as desired. Setting it to `kLevelDebug` is not recommended as it is likely to flood the console with internal messages.

The following are some of the other options for the `Console Level`:

<code>::ixLogger</code> value	Messages Logged
<code>kLevelError</code>	Error messages only.
<code>kLevelWarning</code>	Error and warning messages.
<code>kLevelInfo</code>	Error, warning, and informational messages.

Log File Parameters

The following line defines the parameters of the log files:

```
$logEngine setFile $logName 2 256 1
```

2 is the number of log files to use before wrapping and overwriting the existing log files. The value 2 results in log files named \$logName-#-00.log and \$logName-#-01.log (the # is the session number and is determined dynamically by IxLoad. This also corresponds to the /S#/ in the login name for taking owner

256 is the size limit of each file, in KB.

1 is the truncate flag. 1 indicates to start the logging cycle over, using file -00, and deleting any previous log files. 0 causes logging to resume from where it left off.

Log File Locations

For Windows scripts, the log file is stored in the current working directory of the Tcl shell. For Unix scripts, the file is stored on the intermediate Windows client hosting your remote script, in the directory C:\Program Files\Ixia\Ixload\Client\tclxext\remoteScriptingService. To retrieve the log file from your Unix session, use the following script at the end of the test:

```
set fullLogName [file join "c:/Progra~1/IxLoad/Client/tclxext/remoteScriptingService"
[$logEngine getFileName]]
puts [::IxLoad retrieveFile $fullLogName]
```

Sample Scripts Shipped with IxLoad

The table below lists the files in the C:\Program Files\Ixia\Ixload\<version>\TclScripts\Samples directory, which are shipped with IxLoad. The sample files are grouped under four folders under Samples: Application FeaNetwork, Protocols, Stats.

File	Description
Application Features	
FTP_MixedTrafficMaps.tcl	Example of how to set up traffic on multiple (2) ports.
FTP_ModifyOnTheFly.tcl	Example on how to modify on the fly test objective value.
HTTP_AbortRun.tcl	Example of how to stop a test before completion.
HTTP_ActivityIpMapping.tcl	Example of how to configure IP addresses on a per-activity basis.
HTTP_Capture.tcl	Example of how capture test traffic.
HTTP_CaptureCustom.tcl	Similar to HTTP_Capture.tcl but with the default filter set to TCP.
HTTP_CaptureManual.tcl	Example of how capture test traffic by starting within the test script itself.
HTTP_ConfigStopRun.tcl	Example of how to stop and restart a test.

File	Description
HTTP_CustomTrafficMap.tcl	Example of how to set up a custom traffic pattern on a symbolic destination (IxLoad server or client).
HTTP_RetrieveResultsAPI.tcl	Example of how to retrieve the test results.
RepNewHTTP.tcl	Example of how to create a new repository and configure with a basic HTTP protocol test.
RepRun.tcl	Example of how to load a repository and start a test.
setup_ixload_paths.tcl	Example of how to set up paths to IxLoad tcl code relative to install directory.
setup_simple.tcl	Setup script used for simple*.tcl tests. This file is sourced by all IxLoad sample tcl test scripts, and provides a convenient central place to change the chassis, port, and card that the tests will run on.
SIP_RenamedObjective.tcl	Example of how to rename a Test Objective for a SIP protocol test.
Network	
HTTP_DHCP.tcl	Example of how to configure a DHCP network range.
HTTP_EmulatedRouter.tcl	Example of how to configure an emulated router.
HTTP_IPDHCPRelay.tcl	Example of how to configure a IPDHCPRelay network range.
HTTP_IPSec.tcl	Example of how to configure a IPSec network range.
HTTP_IPv6.tcl	Example of how to configure a IPv6 network range.
HTTP_PPPE.tcl	Example of how to configure a PPPoE network range.
HTTP_VLAN_Impairment.tcl	Example of how to configure impairment with VLANs.
setup_ixload_paths.tcl	Example of how to set up paths to IxLoad tcl code relative to install directory.
setup_simple.tcl	Setup script used for simple*.tcl tests. This file is sourced by all IxLoad sample tcl test scripts, and provides a convenient central place to change the chassis, port, and card that the tests will run on.
Protocols	
2.1.10_src_trace_http.cap	Capture file for use with Trace File Replay test.
ApplicationTest.pft	Sample .pft file for use with Application Test protocol.
ApplicationTest.tcl	Sample .tcl file for use with Application Test protocol.
DDoS.tcl	Example of a basic LDAP protocol test.
DHCP.tcl	Example of a basic DHCP protocol test.
DNS.tcl	Example of a basic DNS protocol test.
FTP.tcl	Example of a basic FTP protocol test.
FTP_POP3.tcl	Example of a basic FTP-POP3 protocol test.
HTTP.tcl	Example of a basic HTTP protocol test.

File	Description
HTTP_SSL.tcl	Example of a basic HTTP_SSL protocol test.
IMAP.tcl	Example of a basic IMAP protocol test.
LDAP.tcl	Example of a basic LDAP protocol test.
MGCP.tcl	Example of a basic MGCP protocol test.
MGCP_Signaling.tcl	Example of a basic MGCP_Signaling protocol test.
MGCP_Signaling_RTP.tcl	Example of a basic MGCP_Signaling_RTP protocol test.
POP3.tcl	Example of a basic POP3 protocol test.
QuickHTTP.tcl	Example of a basic QuickHTTP protocol test.
QuickTCP.tcl	Example of a basic QuickTCP protocol test.
RTSP.tcl	Example of a basic RTSP protocol test.
setup_ixoad_paths.tcl	Example of how to set up paths to IxLoad tcl code relative to install directory.
setup_simple.tcl	Setup script used for simple*.tcl tests. This file is sourced by all IxLoad sample tcl test scripts, and provides a convenient central place to change the chassis, port, and card that the tests will run on.
SIP.tcl	Example of a basic SIP protocol test.
sip_demo.wav	Audio file for SIP testing.
SIP_DTMF.tcl	Example of a SIP protocol test that uses DTMF tones.
SMTP.tcl	Example of a basic SMTP protocol test.
SPTS1-no_discontinuity	Video file for video testing.
TraceFileReplay.tcl	Example of a basic Trace File Replay (capture replay) test.
Video.tcl	Example of a basic Video protocol test.
Video_Configurable_Pid.tcl	Example of configuring Package Identifiers (PIDs) for Video protocol tests.
Video_Control_TS_Per_UDP.tcl	Example of configuring the number of transport stream (TS) packets contained in each UDP packet for Video protocol tests.
Video_I_Join_Latency.tcl	Example of how to measure the IGMP I Join latency in a Video protocol test.
Video_IGMPv1.tcl	Example of how to use IGMPv1 in a Video protocol test.
Video_MLDv1.tcl	Example of how to configure version 1 of Multicast Listener Discovery (MLD) in a Video protocol test.
Video_MLDv2.tcl	Example of how to configure version 2 of Multicast Listener Discovery (MLD) in a Video protocol test.
Video_Multicast_Profiles.tcl	Example of how to configure a multicast video test that uses profiles.
Video_Poisson.tcl	Example of how to configure the Poisson distribution in a Video protocol test.

File	Description
VulnerabilityAttacks.tcl	Example of a basic Vulnerability (Nessus) test.
Stats	
HTTP_PerInterfaceStats.tcl	Example of how to configure per-interface statistics for a HTTP protocol test.
HTTP_PerUrlPerIpStats.tcl	Example of how to configure per-url and per-ip statistics for a HTTP protocol test.
HTTP_RepRun_Stats.tcl	Example of how to load a repository, run an HTTP test, and retrieve the statistics.
HTTP_StateStats.tcl	Example of how to retrieve the Run State and Iteration Count statistics.
HTTP_StatFilter.tcl	Example of how to filter statistics by activity.
setup_ixload_paths.tcl	Example of how to set up paths to IxLoad tcl code relative to install directory.
setup_simple.tcl	Setup script used for simple*.tcl tests. This file is sourced by all IxLoad sample tcl test scripts, and provides a convenient central place to change the chassis, port, and card that the tests will run on.
SIP_PerStreamStats.tcl	Example of how to configure per-stream statistics for a SIP protocol test.
Video_PerStreamStats.tcl	Example of how to configure per-stream statistics for a Video protocol test.

Examples in the `Samples/...` directory should be run from that directory.

Example Program

The following is the complete example used in the Building an IxLoad Test sec of this chapter. This example is similar to, but not identical to the C:\PFiles\Ixia\IxLoad\Client\TclApi\Samples\simplehttpclientandserver.tcl file. This file is self-contained and omits some advanced usage features.

```
#####
# IxLoad ScriptGen created TCL script
# Test1 serialized using version 4.10.0.79
# simpleHTTP.tcl made on Aug 29 2008 15:03
#####

#####
# Copy content of setup_ixload_paths.tcl
#####

package require IxLoad

::IxLoad connect 1.2.3.4

if [catch {

set logtag "IxLoad-api"
set logName "simpleHTTP"
set logger [::IxLoad new ixLogger $logtag 1]
set logEngine [$logger getEngine]
$logEngine setLevels $::ixLogger(kLevelDebug) $::ixLogger(kLevelInfo)
$logEngine setFile $logName 2 256 1

global ixAppPluginManager
$ixAppPluginManager load "HTTP"

#####
# Build chassis chain
#####
set chassisChain [::IxLoad new ixChassisChain]

set my_ixViewOptions [::IxLoad new ixViewOptions]
$my_ixViewOptions config \
    -runMode 1 \
    -captureRunDuration 0 \
    -captureRunAfter 0 \
    -collectScheme 0 \
    -allocatedBufferMemoryPercentage 30

set Test1 [::IxLoad new ixTest]
$Test1 config \
    -comment "" \
    -csvInterval 4 \
    -networkFailureThreshold 0 \
    -name "Test1" \
    -statsRequired 1 \
    -enableResetPorts 0 \
    -enableNetworkStats false \
    -enableForceOwnership false \
    -enableReleaseConfigAfterRun 0 \
    -currentUniqueIDForAgent 2 \
    -allowMultipleIAGgregatedPorts false \
    -captureViewOptions $my_ixViewOptions
```



```
$Test1 scenarioList.clear

set TrafficFlow1 [::IxLoad new ixTrafficFlow]
$TrafficFlow1 config \
    -name                                "TrafficFlow1"

$TrafficFlow1 columnList.clear

set Client [::IxLoad new ixTrafficColumn]
$Client config \
    -name                                "Client"

$Client elementList.clear

set HTTP_client_client_network [::IxLoad new ixNetTraffic]

#####
# Activity newClientActivity1 of NetTraffic HTTP client@client network
#####
set Activity_newClientActivity1 [$HTTP_client_client_network activityList.appendItem \
    -protocolAndType                "HTTP Client" ]

#####
# Timeline1 for activities newClientActivity1
#####
set Timeline1 [::IxLoad new ixTimeline]
$Timeline1 config \
    -rampUpValue                    5 \
    -rampUpType                      0 \
    -offlineTime                    0 \
    -rampDownTime                   20 \
    -standbyTime                    0 \
    -iterations                     1 \
    -rampUpInterval                 1 \
    -sustainTime                    60 \
    -timelineType                   0 \
    -name                            "Timeline1"

$Activity_newClientActivity1 config \
    -enable                         1 \
    -name                           "newClientActivity1" \
    -userIpMapping                  "1:1" \
    -enableConstraint               false \
    -userObjectiveValue             100 \
    -constraintValue                100 \
    -userObjectiveType              "simulatedUsers" \
    -timeline                       $Timeline1

$Activity_newClientActivity1 agent.config \
    -vlanPriority                   0 \
    -enableDecompressSupport        false \
    -enableHttpsProxy              0 \
    -enableSsl                     0 \
    -enableUnidirectionalClose     0 \
    -uniqueID                       1 \
    -ipPreference                   2 \
    -loopValue                      1 \
    -maxPersistentRequests          1 \
    -enableEsm                     0 \
    -certificate                    "" \
    -sequentialSessionReuse         0 \
    -tos                            0 \
    -maxPipeline                    1 \
    -maxHeaderLen                   1024 \
    -maxSessions                    3
```

```

-enableHttpProxy          0 \
-enableTos                 false \
-cookieRejectProbability  0.0 \
-browserEmulation         1 \
-cookieJarSize            10 \
-privateKey                "" \
-commandTimeout           600 \
-enableIntegrityCheckSupport false \
-commandTimeout_ms       0 \
-privateKeyPassword       "" \
-urlStatsCount            10 \
-followHttpRedirects     0 \
-tcpCloseOption           0 \
-enableVlanPriority       0 \
-esm                      1460 \
-httpVersion              0 \
-sslVersion               3 \
-enableCookieSupport      0 \
-enableLargeHeader        0 \
-clientCiphers            "DEFAULT" \
-httpProxy                 ":80" \
-keepAlive                0 \
-enableCRCCheckSupport    false \
-httpsProxy                ":443"

$Activity_newClientActivity1 agent.actionList.clear

set my_ixHttpAction [::IxLoad new ixHttpAction]
$my_ixHttpAction config \
    -profile          -1 \
    -namevalueargs   "" \
    -destination     "HTTP server_newServerActivity1:80" \
    -abort            "None" \
    -command          "GET" \
    -arguments        "" \
    -pageObject       "/4k.html"

$Activity_newClientActivity1 agent.actionList.appendItem -object $my_ixHttpAction

$Activity_newClientActivity1 agent.headerList.clear

set my_ixHttpHeaderString [::IxLoad new ixHttpHeaderString]
$my_ixHttpHeaderString config \
    -data             "Accept: */*"

$Activity_newClientActivity1 agent.headerList.appendItem -object
$my_ixHttpHeaderString

set my_ixHttpHeaderString1 [::IxLoad new ixHttpHeaderString]
$my_ixHttpHeaderString1 config \
    -data             "Accept-Language: en-us"

$Activity_newClientActivity1 agent.headerList.appendItem -object
$my_ixHttpHeaderString1

set my_ixHttpHeaderString2 [::IxLoad new ixHttpHeaderString]
$my_ixHttpHeaderString2 config \
    -data             "Accept-Encoding: gzip, deflate"

$Activity_newClientActivity1 agent.headerList.appendItem -object
$my_ixHttpHeaderString2

set my_ixHttpHeaderString3 [::IxLoad new ixHttpHeaderString]
$my_ixHttpHeaderString3 config \

```



```
-data                                "User-Agent: Mozilla/4.0 (compatible;
MSIE 5.01; Windows NT 5.0)"

$Activity_newClientActivity1 agent.headerList.appendItem -object
$my_ixHttpRequestString3

$Activity_newClientActivity1 agent.profileList.clear

#####
# Network client network of NetTraffic HTTP client@client network
#####
set client_network [::IxLoad new ixNetworkGroup $chassisChain]
$client_network config \
    -comment                "" \
    -name                    "client network" \
    -macMappingMode         0 \
    -linkLayerOptions       0

$client_network globalPlugins.clear

set Filter [::IxLoad new ixNetFilterPlugin]
# ixNet objects needs to be added in the list before they are configured!
$client_network globalPlugins.appendItem -object $Filter

$Filter config \
    -all                     false \
    -pppoecontrol            false \
    -isis                    false \
    -name                     "Filter" \
    -auto                    true \
    -udp                      "" \
    -tcp                      "" \
    -mac                      "" \
    -pppoenetwork            false \
    -ip                       "" \
    -icmp                     ""

set GratarP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list before they are configured!
$client_network globalPlugins.appendItem -object $GratarP

$GratarP config \
    -enabled                 true \
    -name                     "GratarP"

set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list before they are configured!
$client_network globalPlugins.appendItem -object $TCP

$TCP config \
    -tcp_bic                 0 \
    -tcp_tw_recycle          true \
    -tcp_retries2            15 \
    -tcp_retries1            3 \
    -tcp_keepalive_time      9 \
    -tcp_moderate_rcvbuf     0 \
    -tcp_rfc1337             false \
    -tcp_ipfrag_time         30 \
    -tcp_rto_max             60000 \
    -tcp_vegas_alpha         2 \
    -tcp_ecn                 false \
    -tcp_westwood            0 \
    -tcp_rto_min             1000 \
    -tcp_reordering          3 \
    -tcp_vegas_cong_avoid    0 \
```

```

-tcp_keepalive_intvl          7200 \
-tcp_rmem_max                 262144 \
-tcp_orphan_retries          0 \
-tcp_max_tw_buckets          180000 \
-tcp_wmem_default            4096 \
-tcp_low_latency             0 \
-tcp_rmem_min                4096 \
-tcp_adv_win_scale           2 \
-tcp_wmem_min                4096 \
-tcp_port_min                1024 \
-tcp_stdurg                   false \
-tcp_port_max                65535 \
-tcp_fin_timeout             60 \
-tcp_no_metrics_save         false \
-tcp_dsack                    true \
-tcp_mem_high                49152 \
-tcp_frto                     0 \
-tcp_app_win                  31 \
-ip_no_pmtu_disc              false \
-tcp_window_scaling           false \
-tcp_max_orphans             8192 \
-tcp_mem_pressure            32768 \
-tcp_syn_retries             5 \
-name                         "TCP" \
-tcp_max_syn_backlog         1024 \
-tcp_mem_low                  24576 \
-tcp_fack                     true \
-tcp_retrans_collapse        true \
-tcp_rmem_default            4096 \
-tcp_keepalive_probes        75 \
-tcp_abort_on_overflow        false \
-tcp_tw_reuse                 false \
-tcp_wmem_max                 262144 \
-tcp_vegas_gamma              2 \
-tcp_synack_retries          5 \
-tcp_timestamps              true \
-tcp_vegas_beta              6 \
-tcp_sack                     true \
-tcp_bic_fast_convergence     1 \
-tcp_bic_low_window          14

set DNS [::IxLoad new ixNetDnsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$client_network globalPlugins.appendItem -object $DNS

$DNS config \
  -domain                    "" \
  -name                      "DNS" \
  -timeout                   30000

$DNS hostList.clear

$DNS searchList.clear

$DNS nameServerList.clear

set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$client_network globalPlugins.appendItem -object $Settings

$Settings config \
  -teardownInterfaceWithUser false \
  -name                      "Settings" \
  -interfaceBehavior         0

```



```
set Ethernet_1 [$client_network getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType           "master" \
    -negotiateMasterSlave      true

$Ethernet_1 config \
    -advertise10Full           true \
    -name                       "Ethernet-1" \
    -autoNegotiate             true \
    -advertise100Half          true \
    -advertise10Half           true \
    -speed                      "k100FD" \
    -advertise1000Full         true \
    -advertise100Full          true \
    -cardElm                   $my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

set MAC_VLAN_1 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_1

$MAC_VLAN_1 config \
    -name                       "MAC/VLAN-1"

$MAC_VLAN_1 childrenList.clear

set IP_1 [::IxLoad new ixNetIPv4V6Plugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_1 childrenList.appendItem -object $IP_1

$IP_1 config \
    -name                       "IP-1"

$IP_1 childrenList.clear

$IP_1 extensionList.clear

$MAC_VLAN_1 extensionList.clear

$Ethernet_1 extensionList.clear

#####
# Setting the ranges starting with the plugin on top of the stack
#####
$IP_1 rangeList.clear

set IP_R1 [::IxLoad new ixNetIPv4V6Range]
# ixNet objects needs to be added in the list before they are configured!
$IP_1 rangeList.appendItem -object $IP_R1

$IP_R1 config \
    -count                      100 \
    -name                       "IP-R1" \
    -gatewayAddress             "0.0.0.0" \
    -enabled                    true \
    -autoMacGeneration          true \
    -mss                        1460 \
    -incrementBy                "0.0.0.1" \
    -prefix                     16 \
    -gatewayIncrement           "0.0.0.0" \
    -gatewayIncrementMode       "perSubnet" \
    -generateStatistics          false \
```

```

        -ipAddress                "198.18.0.1" \
        -ipType                    "IPv4"

set MAC_R1 [$IP_R1 getLowerRelatedRange "MacRange"]

$MAC_R1 config \
    -count                        100 \
    -name                          "MAC-R1" \
    -enabled                       true \
    -mtu                            1500 \
    -mac                            "00:C6:12:00:01:00" \
    -incrementBy                    "00:00:00:00:00:01"

set VLAN_R1 [$IP_R1 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
    -incrementStep                100 \
    -uniqueCount                  4094 \
    -name                          "VLAN-R1" \
    -innerIncrement                1 \
    -innerUniqueCount              4094 \
    -enabled                       false \
    -innerFirstId                 1 \
    -increment                     1 \
    -priority                      0 \
    -firstId                       1 \
    -innerIncrementStep            1 \
    -idIncrMode                   1 \
    -innerEnable                   false \
    -innerPriority                  0

$HTTP_client_client_network config \
    -enable                        1 \
    -network                       $client_network

$HTTP_client_client_network traffic.config \
    -name                          "HTTP client"

$Client elementList.appendItem -object $HTTP_client_client_network

$TrafficFlow1 columnList.appendItem -object $Client

set DUT [::IxLoad new ixTrafficColumn]
$DUT config \
    -name                          "DUT"

$DUT elementList.clear

$TrafficFlow1 columnList.appendItem -object $DUT

set Server [::IxLoad new ixTrafficColumn]
$Server config \
    -name                          "Server"

$Server elementList.clear

set HTTP_server_server_network [::IxLoad new ixNetTraffic]

#####
# Activity newServerActivity1 of NetTraffic HTTP server@server network
#####
set Activity_newServerActivity1 [$HTTP_server_server_network activityList.appendItem \
    -protocolAndType                "HTTP Server" ]

set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]

```



```
$Activity_newServerActivity1 config \  
-enable 1 \  
-name "newServerActivity1" \  
-timeline $_Match_Longest_  
  
$Activity_newServerActivity1 agent.config \  
-vlanPriority 0 \  
-maxResponseDelay 0 \  
-uniqueID 2 \  
-enableEsm 0 \  
-certificate "" \  
-tos 0 \  
-enableMD5Checksum false \  
-httpPort "80" \  
-httpsPort "443" \  
-esm 1460 \  
-enableTos false \  
-integrityCheckOption "Custom MD5" \  
-privateKey "" \  
-privateKeyPassword "" \  
-urlStatsCount 10 \  
-tcpCloseOption 0 \  
-enableVlanPriority 0 \  
-docrootfile "" \  
-dhParams "" \  
-requestTimeout 300 \  
-ServerCiphers "DEFAULT" \  
-acceptSslConnections 0 \  
-enablePerServerPerURLstat 0 \  
-enableDHsupport 0 \  
-minResponseDelay 0  
  
$Activity_newServerActivity1 agent.webPageList.clear  
  
set 200_OK [::IxLoad new ResponseHeader]  
$200_OK config \  
-mimeType "text/plain" \  
-expirationMode 0 \  
-code "200" \  
-name "200_OK" \  
-lastModifiedMode 1 \  
-lastModifiedIncrementEnable false \  
-lastModifiedDateTimeValue "2005/02/02 21:55:04" \  
-lastModifiedIncrementFor 1 \  
-expirationDateTimeValue "2005/03/04 21:55:04" \  
-expirationAfterRequestValue 3600 \  
-expirationAfterLastModifiedValue 3600 \  
-lastModifiedIncrementBy 5 \  
-description "OK"  
  
$200_OK responseList.clear  
  
set my_PageObject [::IxLoad new PageObject]  
$my_PageObject config \  
-Md5Option 0 \  
-payloadSize "1-1" \  
-customPayloadId -1 \  
-payloadType "range" \  
-payloadFile "<specify file>" \  
-page "/lb.html" \  
-response $200_OK  
  
$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject
```

```
set my_PageObject1 [::IxLoad new PageObject]
$my_PageObject1 config \
    -Md5Option          0 \
    -payloadSize        "4096-4096" \
    -customPayloadId    -1 \
    -payloadType        "range" \
    -payloadFile        "<specify file>" \
    -page               "/4k.html" \
    -response           $200_OK

$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject1

set my_PageObject2 [::IxLoad new PageObject]
$my_PageObject2 config \
    -Md5Option          0 \
    -payloadSize        "8192-8192" \
    -customPayloadId    -1 \
    -payloadType        "range" \
    -payloadFile        "<specify file>" \
    -page               "/8k.html" \
    -response           $200_OK

$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject2

set my_PageObject3 [::IxLoad new PageObject]
$my_PageObject3 config \
    -Md5Option          0 \
    -payloadSize        "16536-16536" \
    -customPayloadId    -1 \
    -payloadType        "range" \
    -payloadFile        "<specify file>" \
    -page               "/16k.html" \
    -response           $200_OK

$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject3

set my_PageObject4 [::IxLoad new PageObject]
$my_PageObject4 config \
    -Md5Option          0 \
    -payloadSize        "32768" \
    -customPayloadId    -1 \
    -payloadType        "range" \
    -payloadFile        "<specify file>" \
    -page               "/32k.html" \
    -response           $200_OK

$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject4

set my_PageObject5 [::IxLoad new PageObject]
$my_PageObject5 config \
    -Md5Option          0 \
    -payloadSize        "65536" \
    -customPayloadId    -1 \
    -payloadType        "range" \
    -payloadFile        "<specify file>" \
    -page               "/64k.html" \
    -response           $200_OK

$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject5

set my_PageObject6 [::IxLoad new PageObject]
$my_PageObject6 config \
    -Md5Option          0 \
    -payloadSize        "131072" \
    -customPayloadId    -1 \
```



```
-payloadType           "range" \  
-payloadFile           "<specify file>" \  
-page                  "/128k.html" \  
-response              $200_OK  
  
$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject6  
  
set my_PageObject7 [::IxLoad new PageObject]  
$my_PageObject7 config \  
-Md5Option            0 \  
-payloadSize          "262144" \  
-customPayloadId      -1 \  
-payloadType          "range" \  
-payloadFile          "<specify file>" \  
-page                  "/256k.html" \  
-response              $200_OK  
  
$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject7  
  
set my_PageObject8 [::IxLoad new PageObject]  
$my_PageObject8 config \  
-Md5Option            0 \  
-payloadSize          "524288" \  
-customPayloadId      -1 \  
-payloadType          "range" \  
-payloadFile          "<specify file>" \  
-page                  "/512k.html" \  
-response              $200_OK  
  
$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject8  
  
set my_PageObject9 [::IxLoad new PageObject]  
$my_PageObject9 config \  
-Md5Option            0 \  
-payloadSize          "1048576" \  
-customPayloadId      -1 \  
-payloadType          "range" \  
-payloadFile          "<specify file>" \  
-page                  "/1024k.html" \  
-response              $200_OK  
  
$Activity_newServerActivity1 agent.webPageList.appendItem -object $my_PageObject9  
  
$Activity_newServerActivity1 agent.cookieList.clear  
  
set UserCookie [::IxLoad new CookieObject]  
$UserCookie config \  
-mode                 3 \  
-type                  2 \  
-name                  "UserCookie" \  
-description           "Name of User"  
  
$UserCookie cookieContentList.clear  
  
set firstName [::IxLoad new ixCookieContent]  
$firstName config \  
-domain                "" \  
-name                  "firstName" \  
-maxAge                 "" \  
-value                 "Joe" \  
-other                  "" \  
-path                  ""  
  
$UserCookie cookieContentList.appendItem -object $firstName
```

```

set lastName [::IxLoad new ixCookieContent]
$lastName config \
    -domain          "" \
    -name            "lastName" \
    -maxAge          "" \
    -value           "Smith" \
    -other           "" \
    -path            ""

$UserCookie cookieContentList.appendItem -object $lastName

$Activity_newServerActivity1 agent.cookieList.appendItem -object $UserCookie

set LoginCookie [::IxLoad new CookieObject]
$LoginCookie config \
    -mode            2 \
    -type            2 \
    -name            "LoginCookie" \
    -description     "Login name and password"

$LoginCookie cookieContentList.clear

set name [::IxLoad new ixCookieContent]
$name config \
    -domain          "" \
    -name            "name" \
    -maxAge          "" \
    -value           "joesmith" \
    -other           "" \
    -path            ""

$LoginCookie cookieContentList.appendItem -object $name

set password [::IxLoad new ixCookieContent]
$password config \
    -domain          "" \
    -name            "password" \
    -maxAge          "" \
    -value           "foobar" \
    -other           "" \
    -path            ""

$LoginCookie cookieContentList.appendItem -object $password

$Activity_newServerActivity1 agent.cookieList.appendItem -object $LoginCookie

$Activity_newServerActivity1 agent.customPayloadList.clear

set AsciiCustomPayload [::IxLoad new CustomPayloadObject]
$AsciiCustomPayload config \
    -repeat          false \
    -name            "AsciiCustomPayload" \
    -asciiPayloadValue "Ixia-Ixload-Http-Server-Custom-Payload"
\
    -payloadmode     0 \
    -offset          1 \
    -hexPayloadValue "" \
    -payloadPosition "Start With" \
    -id              0

$Activity_newServerActivity1 agent.customPayloadList.appendItem -object
$AsciiCustomPayload

set HexCustomPayload [::IxLoad new CustomPayloadObject]
$HexCustomPayload config \

```



```
-repeat 0 \  
-name "HexCustomPayload" \  
-asciiPayloadValue "" \  
-payloadmode 1 \  
-offset 1 \  
-hexPayloadValue "49 78 69 61 2d 49 78 6c 6f 61 64 2d 48  
74 74 70 2d 53 65 72 76 65 72 2d 43 75 73 74 6f 6d 2d 50 61 79 6c 6f 61 64" \  
-payloadPosition "Start With" \  
-id 1
```

```
$Activity_newServerActivity1 agent.customPayloadList.appendItem -object  
$HexCustomPayload
```

```
$Activity_newServerActivity1 agent.responseHeaderList.clear
```

```
set 200_OK1 [::IxLoad new ResponseHeader]  
$200_OK1 config \  
-mimeType "text/plain" \  
-expirationMode 0 \  
-code "200" \  
-name "200_OK" \  
-lastModifiedMode 1 \  
-lastModifiedIncrementEnable false \  
-lastModifiedDateTimeValue "2005/02/02 21:55:04" \  
-lastModifiedIncrementFor 1 \  
-expirationDateTimeValue "2005/03/04 21:55:04" \  
-expirationAfterRequestValue 3600 \  
-expirationAfterLastModifiedValue 3600 \  
-lastModifiedIncrementBy 5 \  
-description "OK"
```

```
$200_OK1 responseList.clear
```

```
$Activity_newServerActivity1 agent.responseHeaderList.appendItem -object $200_OK1
```

```
set 404_PageNotFound [::IxLoad new ResponseHeader]  
$404_PageNotFound config \  
-mimeType "text/plain" \  
-expirationMode 0 \  
-code "404" \  
-name "404_PageNotFound" \  
-lastModifiedMode 1 \  
-lastModifiedIncrementEnable false \  
-lastModifiedDateTimeValue "2005/02/02 21:55:04" \  
-lastModifiedIncrementFor 1 \  
-expirationDateTimeValue "2005/03/04 21:55:04" \  
-expirationAfterRequestValue 3600 \  
-expirationAfterLastModifiedValue 3600 \  
-lastModifiedIncrementBy 5 \  
-description "Page not found"
```

```
$404_PageNotFound responseList.clear
```

```
$Activity_newServerActivity1 agent.responseHeaderList.appendItem -object  
$404_PageNotFound
```

```
#####  
# Network server network of NetTraffic HTTP server@server network  
#####  
set server_network [::IxLoad new ixNetworkGroup $chassisChain]  
$server_network config \  
-comment "" \  
-name "server network" \  
-macMappingMode 0 \  
-linkLayerOptions 0
```

```

$server_network globalPlugins.clear

set Filter_1 [::IxLoad new ixNetFilterPlugin]
# ixNet objects needs to be added in the list before they are configured!
$server_network globalPlugins.appendItem -object $Filter_1

$Filter_1 config \
    -all false \
    -pppoecontrol false \
    -isis false \
    -name "Filter-1" \
    -auto true \
    -udp "" \
    -tcp "" \
    -mac "" \
    -pppoenetwork false \
    -ip "" \
    -icmp ""

set GratARP_1 [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list before they are configured!
$server_network globalPlugins.appendItem -object $GratARP_1

$GratARP_1 config \
    -enabled true \
    -name "GratARP-1"

set TCP_1 [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list before they are configured!
$server_network globalPlugins.appendItem -object $TCP_1

$TCP_1 config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 15 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 9 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \

```



```
-tcp_app_win          31 \
-ip_no_pmtu_disc     false \
-tcp_window_scaling  false \
-tcp_max_orphans     8192 \
-tcp_mem_pressure    32768 \
-tcp_syn_retries     5 \
-name                "TCP-1" \
-tcp_max_syn_backlog 1024 \
-tcp_mem_low         24576 \
-tcp_fack            true \
-tcp_retrans_collapse true \
-tcp_rmem_default    4096 \
-tcp_keepalive_probes 75 \
-tcp_abort_on_overflow false \
-tcp_tw_reuse        false \
-tcp_wmem_max        262144 \
-tcp_vegas_gamma     2 \
-tcp_synack_retries  5 \
-tcp_timestamps     true \
-tcp_vegas_beta      6 \
-tcp_sack            true \
-tcp_bic_fast_convergence 1 \
-tcp_bic_low_window  14

set DNS_1 [::IxLoad new ixNetDnsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$server_network globalPlugins.appendItem -object $DNS_1

$DNS_1 config \
    -domain          "" \
    -name            "DNS-1" \
    -timeout         30000

$DNS_1 hostList.clear

$DNS_1 searchList.clear

$DNS_1 nameServerList.clear

set Settings_1 [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$server_network globalPlugins.appendItem -object $Settings_1

$Settings_1 config \
    -teardownInterfaceWithUser false \
    -name                    "Settings-1" \
    -interfaceBehavior       0

set Ethernet_2 [$server_network getL1Plugin]

set my_ixNetEthernetELMPlugin1 [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin1 config \
    -negotiationType      "master" \
    -negotiateMasterSlave true

$Ethernet_2 config \
    -advertise10Full      true \
    -name                 "Ethernet-2" \
    -autoNegotiate        true \
    -advertise100Half     true \
    -advertise10Half      true \
    -speed                 "k100FD" \
    -advertise1000Full    true \
    -advertise100Full     true \
    -cardElm              $my_ixNetEthernetELMPlugin1
```

```
$Ethernet_2 childrenList.clear

set MAC_VLAN_2 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Ethernet_2 childrenList.appendItem -object $MAC_VLAN_2

$MAC_VLAN_2 config \
    -name "MAC/VLAN-2"

$MAC_VLAN_2 childrenList.clear

set IP_2 [::IxLoad new ixNetIpV4V6Plugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_2 childrenList.appendItem -object $IP_2

$IP_2 config \
    -name "IP-2"

$IP_2 childrenList.clear

$IP_2 extensionList.clear

$MAC_VLAN_2 extensionList.clear

$Ethernet_2 extensionList.clear

#####
# Setting the ranges starting with the plugin on top of the stack
#####
$IP_2 rangeList.clear

set IP_R2 [::IxLoad new ixNetIpV4V6Range]
# ixNet objects needs to be added in the list before they are configured!
$IP_2 rangeList.appendItem -object $IP_R2

$IP_R2 config \
    -count 1 \
    -name "IP-R2" \
    -gatewayAddress "0.0.0.0" \
    -enabled true \
    -autoMacGeneration true \
    -mss 1460 \
    -incrementBy "0.0.0.1" \
    -prefix 16 \
    -gatewayIncrement "0.0.0.0" \
    -gatewayIncrementMode "perSubnet" \
    -generateStatistics false \
    -ipAddress "198.18.1.1" \
    -ipType "IPv4"

set MAC_R2 [$IP_R2 getLowerRelatedRange "MacRange"]

$MAC_R2 config \
    -count 1 \
    -name "MAC-R2" \
    -enabled true \
    -mtu 1500 \
    -mac "00:C6:12:01:01:00" \
    -incrementBy "00:00:00:00:00:01"

set VLAN_R2 [$IP_R2 getLowerRelatedRange "VlanIdRange"]

$VLAN_R2 config \
    -incrementStep 1 \
```



```
-uniqueCount          4094 \  
-name                 "VLAN-R1" \  
-innerIncrement      1 \  
-innerUniqueCount    4094 \  
-enabled             false \  
-innerFirstId        1 \  
-increment           1 \  
-priority            0 \  
-firstId             1 \  
-innerIncrementStep  1 \  
-idIncrMode          1 \  
-innerEnable         false \  
-innerPriority        0  
  
$HTTP_server_server_network config \  
-enable              1 \  
-network             $server_network  
  
$HTTP_server_server_network traffic.config \  
-name                "HTTP server"  
  
$Server elementList.appendItem -object $HTTP_server_server_network  
  
$TrafficFlow1 columnList.appendItem -object $Server  
  
$TrafficFlow1 links.clear  
  
$Test1 scenarioList.appendItem -object $TrafficFlow1  
  
#####  
# Destination newServerActivity1 for newClientActivity1  
#####  
set destination [$HTTP_client_client_network getDestinationForActivity  
"newClientActivity1" "HTTP server_newServerActivity1"]  
$destination config \  
-portMapPolicy      "portMesh"  
  
#####  
# Session Specific Settings  
#####  
set my_ixNetMacSessionData [$Test1 getSessionSpecificData "L2EthernetPlugin"]  
$my_ixNetMacSessionData config \  
-duplicateCheckingScope      2  
  
set my_ixNetIpSessionData [$Test1 getSessionSpecificData "IPv4V6Plugin"]  
$my_ixNetIpSessionData config \  
-duplicateCheckingScope      2  
  
#####  
# Create the test controller to run the test  
#####  
set testController [::IxLoad new ixTestController -outputDir True]  
  
$testController setResultDir "RESULTS/simpleHTTP"  
set NS statCollectorUtils  
  
set test_server_handle [$testController getTestServerHandle]  
${NS}::Initialize -testServerHandle $test_server_handle  
  
${NS}::ClearStats  
$Test1 clearGridStats  
  
set HTTP_Client_Per_URL_StatList { \  
{"HTTP Client Per URL" "HTTP Aborted After Request" "kMax"} \  
{"HTTP Client Per URL" "HTTP Aborted Before Request" "kMax"} \  
}
```

```

{"HTTP Client Per URL" "HTTP Requests Failed" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (400)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (401)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (403)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (404)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (407)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (408)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (4xx other)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (4xx)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (505)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (5xx other)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (5xx)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Aborted)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Bad Header)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Read)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Timeout)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Failed (Write)" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Sent" "kMax"} \
{"HTTP Client Per URL" "HTTP Requests Successful" "kMax"} \
{"HTTP Client Per URL" "HTTP Responses Received With Match" "kMax"} \
{"HTTP Client Per URL" "HTTP Responses Received Without Match" "kMax"} \
}

set HTTP_Client_StatList { \
{"HTTP Client" "Client Hello Sent" "kMax"} \
{"HTTP Client" "HTTP Aborted After Request" "kMax"} \
{"HTTP Client" "HTTP Aborted Before Request" "kMax"} \
{"HTTP Client" "HTTP Bytes" "kMax"} \
{"HTTP Client" "HTTP Bytes Received" "kMax"} \
{"HTTP Client" "HTTP Bytes Sent" "kMax"} \
{"HTTP Client" "HTTP Concurrent Connections" "kMax"} \
{"HTTP Client" "HTTP Connect Time (ms)" "kAverageRate"} \
{"HTTP Client" "HTTP Connection Attempts" "kMax"} \
{"HTTP Client" "HTTP Connections" "kMax"} \
{"HTTP Client" "HTTP Content Bytes Received" "kMax"} \
{"HTTP Client" "HTTP Content Bytes Sent" "kMax"} \
{"HTTP Client" "HTTP Cookie headers Rejected - (Memory Overflow)" "kMax"} \
{"HTTP Client" "HTTP Cookies Received" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected - (Cookiejar Overflow)" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected - (Domain Match Failed)" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected - (Path Match Failed)" "kMax"} \
{"HTTP Client" "HTTP Cookies Rejected - (Probabilistic Reject)" "kMax"} \
{"HTTP Client" "HTTP Cookies Sent" "kMax"} \
{"HTTP Client" "HTTP Requests Failed" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (400)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (401)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (403)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (404)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (407)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (408)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (4xx other)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (4xx)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (505)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (5xx other)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (5xx)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Aborted)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Bad Header)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Read)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Timeout)" "kMax"} \
{"HTTP Client" "HTTP Requests Failed (Write)" "kMax"} \
{"HTTP Client" "HTTP Requests Sent" "kMax"} \
{"HTTP Client" "HTTP Requests Successful" "kMax"} \
{"HTTP Client" "HTTP Session Timeouts (408)" "kMax"} \
{"HTTP Client" "HTTP Sessions Rejected (503)" "kMax"} \
}

```

```
{ "HTTP Client" "HTTP Simulated Users" "kSum" } \  
{ "HTTP Client" "HTTP Time To First Byte (ms)" "kAverageRate" } \  
{ "HTTP Client" "HTTP Time To Last Byte (ms)" "kAverageRate" } \  
{ "HTTP Client" "HTTP Transactions" "kMax" } \  
{ "HTTP Client" "HTTP Transactions Active" "kMax" } \  
{ "HTTP Client" "HTTP Users Active" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (access_denied)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (bad_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (bad_record_mac)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (certificate_expired)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (certificate_revoked)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (certificate_unknown)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (close_notify)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (decode_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (decompression_failure)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (decrypt_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (decryption_failed)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (export_restriction)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (handshake_failure)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (illegal_parameter)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (insufficient_security)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (internal_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (no_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (no_renegotiation)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (protocol_version)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (record_overflow)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (unexpected_message)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (unknown_ca)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (unsupported_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Received (user_canceled)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (access_denied)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (bad_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (bad_record_mac)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (certificate_expired)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (certificate_revoked)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (certificate_unknown)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (close_notify)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (decode_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (decompression_failure)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (decrypt_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (decryption_failed)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (export_restriction)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (handshake_failure)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (illegal_parameter)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (insufficient_security)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (internal_error)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (no_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (no_renegotiation)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (protocol_version)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (record_overflow)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (unexpected_message)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (unknown_ca)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (unsupported_certificate)" "kMax" } \  
{ "HTTP Client" "SSL Alerts Sent (user_canceled)" "kMax" } \  
{ "HTTP Client" "SSL Bytes Received" "kMax" } \  
{ "HTTP Client" "SSL Bytes Sent" "kMax" } \  
{ "HTTP Client" "SSL Concurrent Sessions" "kMax" } \  
{ "HTTP Client" "SSL Errors Received" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (bad certificate)" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (no certificate)" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (no cipher)" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (undefined error)" "kMax" } \  
{ "HTTP Client" "SSL Errors Received (unsupported certificate)" "kMax" } \  

```

```

{"HTTP Client" "SSL Errors Sent" "kMax"} \
{"HTTP Client" "SSL Errors Sent (bad certificate)" "kMax"} \
{"HTTP Client" "SSL Errors Sent (no certificate)" "kMax"} \
{"HTTP Client" "SSL Errors Sent (no cipher)" "kMax"} \
{"HTTP Client" "SSL Errors Sent (undefined error)" "kMax"} \
{"HTTP Client" "SSL Errors Sent (unsupported certificate)" "kMax"} \
{"HTTP Client" "SSL Negotiation Finished Successfully" "kMax"} \
{"HTTP Client" "SSL Session Reuse Failed" "kMax"} \
{"HTTP Client" "SSL Session Reuse Success" "kMax"} \
{"HTTP Client" "SSL Throughput Bytes" "kMax"} \
{"HTTP Client" "Server Hello Received" "kMax"} \
{"HTTP Client" "TCP Accept Queue Entries" "kMax"} \
{"HTTP Client" "TCP Connection Requests Failed" "kMax"} \
{"HTTP Client" "TCP Connections Established" "kMax"} \
{"HTTP Client" "TCP Connections in CLOSE STATE" "kMax"} \
{"HTTP Client" "TCP Connections in CLOSE-WAIT State" "kMax"} \
{"HTTP Client" "TCP Connections in CLOSING State" "kMax"} \
{"HTTP Client" "TCP Connections in ESTABLISHED State" "kMax"} \
{"HTTP Client" "TCP Connections in FIN-WAIT-1 State" "kMax"} \
{"HTTP Client" "TCP Connections in FIN-WAIT-2 State" "kMax"} \
{"HTTP Client" "TCP Connections in LAST-ACK State" "kMax"} \
{"HTTP Client" "TCP Connections in LISTENING State" "kMax"} \
{"HTTP Client" "TCP Connections in SYN-RECEIVED State" "kMax"} \
{"HTTP Client" "TCP Connections in SYN-SENT State" "kMax"} \
{"HTTP Client" "TCP Connections in TIME-WAIT State" "kMax"} \
{"HTTP Client" "TCP FIN Received" "kMax"} \
{"HTTP Client" "TCP FIN Sent" "kMax"} \
{"HTTP Client" "TCP FIN-ACK Received" "kMax"} \
{"HTTP Client" "TCP FIN-ACK Sent" "kMax"} \
{"HTTP Client" "TCP Listen Queue Drops" "kMax"} \
{"HTTP Client" "TCP Resets Received" "kMax"} \
{"HTTP Client" "TCP Resets Sent" "kMax"} \
{"HTTP Client" "TCP Retries" "kMax"} \
{"HTTP Client" "TCP SYN Failed" "kMax"} \
{"HTTP Client" "TCP SYN Sent" "kMax"} \
{"HTTP Client" "TCP SYN-ACK Sent" "kMax"} \
{"HTTP Client" "TCP SYN_SYN-ACK Received" "kMax"} \
{"HTTP Client" "TCP Timeouts" "kMax"} \
}

set HTTP_Server_Per_URL_StatList { \
{"HTTP Server Per URL" "HTTP Requests Failed" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Failed (404)" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Failed (50x)" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Failed (Write Error)" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Received" "kMax"} \
{"HTTP Server Per URL" "HTTP Requests Successful" "kMax"} \
}

set HTTP_Server_StatList { \
{"HTTP Server" "Client Hello Received" "kMax"} \
{"HTTP Server" "HTTP Bytes Received" "kMax"} \
{"HTTP Server" "HTTP Bytes Sent" "kMax"} \
{"HTTP Server" "HTTP Content Bytes Received" "kMax"} \
{"HTTP Server" "HTTP Content Bytes Sent" "kMax"} \
{"HTTP Server" "HTTP Cookies Received" "kMax"} \
{"HTTP Server" "HTTP Cookies Received With Matching ServerID" "kMax"} \
{"HTTP Server" "HTTP Cookies Received With Non-matching ServerID" "kMax"} \
{"HTTP Server" "HTTP Cookies Sent" "kMax"} \
{"HTTP Server" "HTTP Requests Failed" "kMax"} \
{"HTTP Server" "HTTP Requests Failed (404)" "kMax"} \
{"HTTP Server" "HTTP Requests Failed (50x)" "kMax"} \
{"HTTP Server" "HTTP Requests Failed (Write Error)" "kMax"} \
{"HTTP Server" "HTTP Requests Received" "kMax"} \
{"HTTP Server" "HTTP Requests Successful" "kMax"} \
}

```



```

{"HTTP Server" "SSL Errors Sent (no cipher)" "kMax"} \
{"HTTP Server" "SSL Errors Sent (undefined error)" "kMax"} \
{"HTTP Server" "SSL Errors Sent (unsupported certificate)" "kMax"} \
{"HTTP Server" "SSL Negotiation Finished Successfully" "kMax"} \
{"HTTP Server" "SSL Session Reuse Failed" "kMax"} \
{"HTTP Server" "SSL Session Reuse Success" "kMax"} \
{"HTTP Server" "SSL Throughput Bytes" "kMax"} \
{"HTTP Server" "Server Hello Sent" "kMax"} \
{"HTTP Server" "TCP Accept Queue Entries" "kMax"} \
{"HTTP Server" "TCP Connection Requests Failed" "kMax"} \
{"HTTP Server" "TCP Connections Established" "kMax"} \
{"HTTP Server" "TCP Connections in CLOSE STATE" "kMax"} \
{"HTTP Server" "TCP Connections in CLOSE-WAIT State" "kMax"} \
{"HTTP Server" "TCP Connections in CLOSING State" "kMax"} \
{"HTTP Server" "TCP Connections in ESTABLISHED State" "kMax"} \
{"HTTP Server" "TCP Connections in FIN-WAIT-1 State" "kMax"} \
{"HTTP Server" "TCP Connections in FIN-WAIT-2 State" "kMax"} \
{"HTTP Server" "TCP Connections in LAST-ACK State" "kMax"} \
{"HTTP Server" "TCP Connections in LISTENING State" "kMax"} \
{"HTTP Server" "TCP Connections in SYN-RECEIVED State" "kMax"} \
{"HTTP Server" "TCP Connections in SYN-SENT State" "kMax"} \
{"HTTP Server" "TCP Connections in TIME-WAIT State" "kMax"} \
{"HTTP Server" "TCP FIN Received" "kMax"} \
{"HTTP Server" "TCP FIN Sent" "kMax"} \
{"HTTP Server" "TCP FIN-ACK Received" "kMax"} \
{"HTTP Server" "TCP FIN-ACK Sent" "kMax"} \
{"HTTP Server" "TCP Listen Queue Drops" "kMax"} \
{"HTTP Server" "TCP Resets Received" "kMax"} \
{"HTTP Server" "TCP Resets Sent" "kMax"} \
{"HTTP Server" "TCP Retries" "kMax"} \
{"HTTP Server" "TCP SYN Failed" "kMax"} \
{"HTTP Server" "TCP SYN Sent" "kMax"} \
{"HTTP Server" "TCP SYN-ACK Sent" "kMax"} \
{"HTTP Server" "TCP SYN_SYN-ACK Received" "kMax"} \
{"HTTP Server" "TCP Timeouts" "kMax"} \
}

set statList [concat \
$HTTP_Client_Per_URL_StatList \
$HTTP_Client_StatList \
$HTTP_Server_Per_URL_StatList \
$HTTP_Server_StatList \
]

set count 1
foreach statItem $statList {
set caption [format "Watch_Stat_%s" $count]
set statSourceType [lindex $statItem 0]
set statName [lindex $statItem 1]
set aggregationType [lindex $statItem 2]

${NS}::AddStat \
    -caption $caption \
    -statSourceType $statSourceType \
    -statName $statName \
    -aggregationType $aggregationType \
    -filterList {}

incr count
}

proc ::my_stat_collector_command {args} {
puts "=====
puts "INCOMING STAT RECORD >>> $args"
puts "Len = [llength $args]"
}

```



```
puts [lindex $args 0]
puts [lindex $args 1]
puts "======"
}
${NS}::StartCollector -command ::my_stat_collector_command
$testController run $Test1

vwait ::ixTestControllerMonitor
puts $::ixTestControllerMonitor

${NS}::StopCollector

#####
# Cleanup
#####
# Release config is only strictly necessary if enableReleaseConfigAfterRun is 0.
$testController releaseConfigWaitFinish

::IxLoad delete $chassisChain
::IxLoad delete $Test1
::IxLoad delete $my_ixViewOptions
::IxLoad delete $TrafficFlow1
::IxLoad delete $Client
::IxLoad delete $HTTP_client_client_network
::IxLoad delete $Activity_newClientActivity1
::IxLoad delete $Timeline1
::IxLoad delete $my_ixHttpAction
::IxLoad delete $my_ixHttpHeaderString
::IxLoad delete $my_ixHttpHeaderString1
::IxLoad delete $my_ixHttpHeaderString2
::IxLoad delete $my_ixHttpHeaderString3
::IxLoad delete $client_network
::IxLoad delete $Filter
::IxLoad delete $GratARP
::IxLoad delete $TCP
::IxLoad delete $DNS
::IxLoad delete $Settings
::IxLoad delete $Ethernet_1
::IxLoad delete $my_ixNetEthernetELMPlugin
::IxLoad delete $MAC_VLAN_1
::IxLoad delete $IP_1
::IxLoad delete $IP_R1
::IxLoad delete $MAC_R1
::IxLoad delete $VLAN_R1
::IxLoad delete $DUT
::IxLoad delete $Server
::IxLoad delete $HTTP_server_server_network
::IxLoad delete $Activity_newServerActivity1
::IxLoad delete $_Match_Longest_
::IxLoad delete $my_PageObject
::IxLoad delete $200_OK
::IxLoad delete $my_PageObject1
::IxLoad delete $my_PageObject2
::IxLoad delete $my_PageObject3
::IxLoad delete $my_PageObject4
::IxLoad delete $my_PageObject5
::IxLoad delete $my_PageObject6
::IxLoad delete $my_PageObject7
::IxLoad delete $my_PageObject8
::IxLoad delete $my_PageObject9
::IxLoad delete $UserCookie
::IxLoad delete $firstName
::IxLoad delete $lastName
::IxLoad delete $LoginCookie
::IxLoad delete $name
```

```
::IxLoad delete $password
::IxLoad delete $AsciiCustomPayload
::IxLoad delete $HexCustomPayload
::IxLoad delete $200_OK1
::IxLoad delete $404_PageNotFound
::IxLoad delete $server_network
::IxLoad delete $Filter_1
::IxLoad delete $GratARP_1
::IxLoad delete $TCP_1
::IxLoad delete $DNS_1
::IxLoad delete $Settings_1
::IxLoad delete $Ethernet_2
::IxLoad delete $my_ixNetEthernetELMPlugin1
::IxLoad delete $MAC_VLAN_2
::IxLoad delete $IP_2
::IxLoad delete $IP_R2
::IxLoad delete $MAC_R2
::IxLoad delete $VLAN_R2
::IxLoad delete $destination
::IxLoad delete $my_ixNetMacSessionData
::IxLoad delete $my_ixNetIpSessionData
::IxLoad delete $testController

#####
# Disconnect / Release application lock
#####
}] {
    puts $errorInfo
}

::IxLoad disconnect
```



4

IxLoad Tcl API Commands

This section describes the commands used to create the test infrastructure.

In this chapter:

::IxLoad	109
ixChassisChain	112
ixCustomPortMap	115
ixPort	119
Submap	122
ixSubmapRange	124
ixIntRange	125
ixRepository	127
statCollectorUtils	129
ixStatCatalogItem	131
ixStatFilter	133
ixStatSpec	135
ixTest	137
ixTestController	141
ixTestControllerMonitor	148
statCollectorUtils	150
ixScriptGen	160
ixTimeline	163
ixSubscriberNetTraffic	170
ixNetTraffic	172
activityList	177
ixTrafficFlow	179
ixTrafficColumn	180
ixNetworkGroup	182
ixDut	184
ixDutConfigVirtual	186
ixDutNetworkRange	189
ixDutProtocolPortRange	192
ixDutConfigVip	194
ixDutConfigSLB	195
ixView	197
ixClientNetwork	199
ixClientTraffic	204
ixClientTrafficNetworkMapping	207
ixNetworkRange	217
ixServerNetwork	221
ixServerTraffic	226
ixServerTrafficNetworkMapping	230
statCollectorUtils	233



::IxLoad

::IxLoad-Top level IxLoad utility.

SYNOPSIS

```
set object [::IxLoad new ixObject options]
```

DESCRIPTION

The `ixLoad` command is the means by which other top level objects are created. Its `new` subcommand is documented in each of the created objects' commands. In addition, the `connect` and `disconnect` commands are used to connect to a remote server when running from a non-Windows client.

Although the `connect` operation is not needed for Windows clients, the `disoperation` is required. It is best to always use the following structure:

```
::IxLoad connect <remote server>
catch {
    ... remainder of program ...
} connectResults
::IxLoad disconnect
```

When operating on a Windows client, the use of `localhost` as a `<remote server>` is a convenient placeholder.

When using a Unix host to run IxLoad Tcl API programs, the Windows-based host referred to in the `connect` subcommand must have installed the following software installed:

- The Tcl run-time environment from the IxOS installation.
- The IxLoad client component from the IxLoad client installation.

SUBCOMMANDS

::IxLoad connect `server`

On non-Windows client, connect to a remote IxTcl server process at `server`. This command has no effect on Windows clients.

::IxLoad disconnect

Disconnect from the last remote server used in a `connect` subcommand. This statement must be executed before exiting any IxLoad Tcl script.

::IxLoad level `command`

Evaluates the `command` in the context of IxLoad. When running on a Windows system, this evaluates locally. When run on a Unix system, it is evaluated on the target system.

::IxLoad retrieveFile `path`

This subcommand is intended to be used by a Unix/Linux client to retrieve files from a Windows host.



The Windows host that is the target of this subcommand is the host that the Unix/Linux client connected to in its most recent connect subcommand.

`retrieveFile` returns the contents of the file as a string.

```
::IxLoad retrieveFileCopy sourcePath destPath
```

This subcommand is intended to be used by a Unix/Linux client to retrieve files from a Windows host. `retrieveFileCopy` copies a file from the Windows host, and creates (or overwrites) it on the Unix/Linux host.

The Windows host that is the target of this subcommand is the host that the Unix/Linux client connected to in its most recent connect subcommand.

`sourcePath` is the file name and path on the Windows host.

`destPath` is the file name and path on the Unix/Linux host.

```
::IxLoad retrieveResults path
```

This subcommand is intended to be used by a Unix/Linux client to retrieve .csv files from a Windows host. `retrieveResults` tracks the path of the windows files internally, fetches the files, and places them in the unix machine; in the folder mentioned along with the `retrieveResults` subcommand.

The Windows host that is the target of this subcommand is the host that the Unix/Linux client connected to in its most recent connect subcommand.

`path` is the folder name and path on the Unix/Linux host.

```
puts "*****UnixResultDir = $UnixResultDir"  
#::IxLoad retrieveResults $::IxLoadPrivate::SimpleSettings::RESULTDIR  
::IxLoad retrieveResults $UnixResultDir
```

```
::IxLoad sendFileCopy sourcePath destPath
```

This subcommand is intended to be used by a Unix/Linux client to send files to a Windows host for use in an IxLoad test. For example, you can use this subcomto send files such as HTTP server pages and FTP server files.

The Windows host, which is the target of this subcommand, is the host that the Unix/Linux client connected to in its most recent connect subcommand.

`sourcePath` is the file name and path on the Unix/Linux host.

`destPath` is the file name and path on the Windows host.

OPTIONS

None.

EXAMPLE

See above.



ixChassisChain

ixChassisChain-Builds a set of Ixia chassis.

SYNOPSIS

```
set chassisChain [::IxLoad new ixChassisChain]
$chassisChain subcommand options...
```

DESCRIPTION

The `ixChassisChain` command is used to construct a chain of Ixia chassis, whose ports may be used in the `ixNetworkGroup` command for both client and server networks. Chassis are assigned chassis IDs starting at 1; these are used in the netcommands to define the chassis associated with the port.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition the following commands are available. Unless otherwise described, no value returned and an exception is raised for any error found.

addChassis chassisName

Adds a new chassis to the chassis chain. `chassisName` is the IP address or host name of a chassis. Each new chassis is assigned a `c`, starting at 1, which must be used to identify ports on that chassis.

deleteChassisByName chassisName

Deletes the chassis whose name is `chassisName` from the chassis chain. All other chassis IDs remain unaffected.

getChassisNames

Returns a list of all of the chassis names, ordered by their `chassisIDs`.

getLoginName

Returns the user's login name.

isValidChassisName chassisName

Checks to see whether `chassisName` is a valid IP address or host name. `True` is returned if the name is valid and `false` otherwise.

refresh

Refreshes all of the chassis in the chassis chain-retrieving current card and port configuration.

refreshChassis chassisName

Refreshes the chassis whose name is `chassisName` - retrieving current card and port configuration.

setCardAggregation aggregationMode

On a card that supports aggregating ports such as the ASM1000XMV12X, this option sets the port aggregation to `aggregationMode`. The following values are supported for `aggregationMode`:

Aggregation Mode	Description
kNonAggregated	Not aggregated
k1GAggregated	1G Aggregated
k10GAggregated	10G Aggregated

For example:

```
$chassisChain setCardAggregation [list $clientCard $serverCard]
$::ixCard(k1GAggregated)
```

For an example of how to use an aggregating load module in a script, see the example in the Tcl API \ Samples \ Network directory.

setLoginName name

Sets the user's login name to `name`. If this call is not made, then the name of the chassis is used when port ownership is taken.

OPTIONS

None.

EXAMPLE

```
#set chassisChain [::IxLoad new ixChassisChain]
#$chassisChain addChassis $chassisName

#####

# Build chassis chain

#####

set chassisChain [::IxLoad new ixChassisChain]
$chassisChain addChassis 10.205.29.101

set client_network [::IxLoad new ixNetworkGroup $chassisChain]
$client_network config \
    -comment          "" \
    -name             "client network" \
    -emulatedRouterSubnetIPv6
"FFFF:FFFF:FFFF:FFFF:FFFF:FFFF::0" \
    -linkLayerOptions 0 \
    -ipSourcePortFrom 1024 \
```



```
-emulatedRouterGatewayIPv6      ":::" \  
-cardType                        "ALM1000T8-1GB" \  
-emulatedRouterGateway          "0.0.0.0" \  
-ipSourcePortTo                 65535 \  
-emulatedRouterSubnet           "255.255.255.0" \  
-macMappingMode                 0 \  
-dnsParameters                  $my_ixDns \  
-tcpParameters                  $my_ixTcpParameters \  
-impairment                      $my_ixImpairment \  
-arpSettings                     $my_ixArpSettings  
  
$client_network portList.appendItem \  
  -chassisId 1 \  
  -cardId 3 \  
  -portId 7
```

SEE ALSO

`ixNetworkGroup`

ixCustomPortMap

ixCustomPortMap-Customizes the order and frequency, by which client IPs will access server IPs.

SYNOPSIS

```
$destination1 config -portMapPolicy $ixPortMap(kPortMapCustom)
```

```
set customPortMap [$destination1 cget customPortMap]
```

```
$customPortMap subcommand options
```

DESCRIPTION

The `ixCustomPortMap` command is used to map a range of client and server traffic is used to map client IPs onto server IPs or client VLANs onto server VLANs.

A custom port map is associated with a specific symbolic destination.

To create a Custom traffic map, the client and server network ranges, `rangeType` parameter can be anything, except IPsec. For DHCP and PPPoE ranges, VLAN must be enabled on both the client and server networks to use a custom traffic map.

SUBCOMMANDS

None.

OPTIONS

```
submapsIPv4
```

This is an `ixConfigSequenceContainer` holding a list of `Submap` objects.

```
submapsIPv6
```

This is an `ixConfigSequenceContainer` holding a list of `Submap` objects.

Steps for Custom Traffic Mapping

 To set up a Custom Traffic Map:

1. Set up the custom Traffic Map for symbolic destination. After creating the test object and assigning traffic-network mappings, set up the custom traffic map for the symbolic destination.

```
set destination1 [$clnt_t_n_mapping getDestinationForActivity my_http_client  
svr_traffic_my_http_server]
```



2. Set up the client or server traffic-network mapping. Set the client or server traffic-network mapping for a custom traffic. Set the `port` for a destination to `kPortMapCustom`. Now it is possible to access the `customPortMap` property on the destination object.

```
$destination1 config -portMapPolicy $ixPortMap(kPortMapCustom)
```

3. Include the custom port map object. This includes the custom port map object into a local variable for convenience of scripting.

```
set customMap [$destination1 cget -customPortMap]
```

4. Set the submaps. A submap is a portion of a `customPortMap` that describes a simple relationship between a set of source addresses and a set of destination addresses. Complex relationships can be described using multiple `ixPort` objects.

```
set submap [$customMap submapsIPv4.getItem 0]
```

5. Set the submap's mesh type to be IP range pairs:

```
$submap config -meshType $ixSubmap(kMeshTypeIpRangePairs)
```

IP mesh types start out with `ixSubmapRange` objects that correspond to network ranges in the client and server networks for the symbolic destination. In this mode, `ixSubmapRange` IDs are the row numbers of the corresponding ranges in the networks. `ixSubmapRange` can be split into smaller, equal subranges using the `split` command. Refer to `Split` and `Merge Submaps`.

vLAN mesh types start out with `ixSubmapRange` objects that correspond to vLAN IDs (one `ixSubmapRange` per vLAN) in the client and server networks for the symbolic destination. In this mode, `ixSubmapRange` IDs are the same as the vLAN IDs they represent. Each `ixSubmapRange` can potentially span portions of many network ranges, depending on how the vLANs are specified on those ranges.

6. Specify the interconnections. You can now specify which server submap range that each client submap range communicates with. In the following example, the numbers next to the source range and the destination range show the mapping pattern.

```
# wire second source range to first destination range
# and vise versa
$submap sourceRanges(0).config -destinationId 3
$submap sourceRanges(1).config -destinationId 1
$submap sourceRanges(2).config -destinationId 1
$submap sourceRanges(3).config -enable 0
```

7. Split and merge subIP meshes, you can split a range in the list into subranges by calling the `split` method on that range. Once split, a range can be merged by calling `merge` on it. `Merge` doesn't need a parameter because it removes all of the child nodes origicreated by using the `split`.

```
# split some ranges
$submap sourceRanges(0).split 2
$submap destinationRanges(0).split 2
```

EXAMPLE

```
#-----
# Set up the custom traffic map for the symbolic destination.
# This must be done after creating the test object and assigning
```

```
# traffic-network mappings

#-----

set destination1 [$clnt_t_n_mapping getDestinationForActivity my_http_client
svr_traffic_my_http_server]

$destination1 config -portMapPolicy $ixPortMap(kPortMapCustom)

# setting custom port map creates and initializes the custom port map object
# get it so we can modify it

set customMap [$destination1 cget -customPortMap]

# the default has a single submap range available. Modify it
set submap [$customMap submapsIPv4.getItem 0]

# set it to an IP range pair type
$submap config -meshType $ixSubmap(kMeshTypeIpRangePairs)

# split some ranges
#$submap sourceRanges(0).split 2
#$submap destinationRanges(0).split 2

# wire second source child to first destination child
# and vise versa

$submap sourceRanges(0).config -destinationId 3
$submap sourceRanges(1).config -destinationId 1
$submap sourceRanges(2).config -destinationId 1
$submap sourceRanges(3).config -enable 0
```

SEE ALSO

`ixClientTrafficNetworkMapping`



ixPort

ixPort - retrieves the ID of an Ixia port and controls the port capture.

SYNOPSIS

```
$network portList.appendItem -chassisId 1 -cardId 1 -portId 1
```

```
set port [$network portList.getItem 0]
```

DESCRIPTION

The `ixPort` command is used to define and retrieve the attributes of an Ixia port that is a member of a `portList` object. For example:

```
puts "Added card [$clnt_network portList(0).getId]"
```

SUB-COMMANDS

None.

OPTIONS

`getId`

Returns a string indicating the chassis ID, card ID, and port ID of a port, in the following format:
`chassisID.cardID.portID`

`getOwner`

Returns a string indicating the current owner of the port. Returns an empty string if there is no owner.

`isLinkUp`

Returns a flag indicating, whether a cable is connected to another live port.

`isPortCaptureEnabled`

This returns a flag indicating the capture is enabled on the port.

`setPortCaptureEnable`

This enables the port capture. It is also enabled during the traffic-network map

`setPortCaptureFile Name`

This enables the port capture and saves the details in a file on the hard disk.

EXAMPLE

```
#-----  
# Build Chassis Chain and add a Chassis  
#-----  
  
set chassisChain [::IxLoad new ixChassisChain]  
$chassisChain addChassis myChassis  
#-----  
# Build client Network  
#-----  
  
set clnt_network [::IxLoad new ixClientNetwork $chassisChain]  
#-----  
# Add a port  
#-----  
  
$clnt_network portList.appendItem \  
-chassisId 1 \  
-cardId 1\  
-portId 1  
#-----  
# Get the port back and check its ID  
#-----  
  
puts [$clnt_network portList(0).getId]
```

SEE ALSO

ixChassisChain

Submap

SYNOPSIS

```
set submap [$customMap submapsIPv4.getItem 0]
$submap config -meshType $ixSubmap(kMeshTypeIpRangePairs)
```

DESCRIPTION

A portion of a `customportmap` that describes a relationship between a set of source addresses and destination addresses. Arbitrarily complex relationships can be described using multiple `ixSubmaps`.

Options

`name`

This is the user-defined name for the submap.

`destinationRanges`

This is a list of `ixSubmapRange` objects representing the server IPs.

`sourceRanges`

This is a list of `ixSubmapRange` objects representing the server IPs.

`ipType`

This can be either, IPv4 or IPv6. This is read-only.

`allowsIpMesh`

This is a read-only value, and returns 1 if the IP `meshTypes` are allowed.

`allowsVlanMesh`

This is a read-only value, and returns 1 if the vLAN `meshTypes` are allowed.

`meshType`

This defines the relationship between the `sourceRanges` and `destinationRanges`. Can be one of:

Option	Usage
\$::ixSubmap(kMeshor "ipRangeMesh")	A pattern based on IP addresses, where each enabled client range communicates to all enabled server ranges.
\$::ixSubmap(kMeshor "ipRangePairs")	A pattern based on IP addresses, where each enabled client range communicates with a single server range, as specified by the client range's destination (see ixSubmapRange command).
\$::ixSubmap(kMeshType Vlanor "vlanRange")	A pattern based on vLAN IDs, where each enabled client range communicates with all enabled server ranges.
\$::ixSubmap(kMeshType Vlanor "vlanRange")	A pattern based on vLAN IDs, where each enabled client range communicates with all enabled server ranges.

ixSubmapRange

DESCRIPTION

A group of IPs, specified by either VLAN or IP (as determined by the `ixSubmap meshType` option). A submap range is the smallest unit of client or server IPs for specifying the traffic flow between clients and servers.

OPTION

`id`

This is the IxLoad-assigned ID for the submap. This is read-only.

`enable`

This enables or disables traffic for the submap range. In `full mesh` modes, `enable` applies to both the client and server submap ranges. In `range pair` modes, `enable` affects the submap ranges only. All enabled client submap ranges will talk to their specified destination submap range, whether enabled or not.

`destinationId`

This is enabled for client submap ranges in a range pair `meshType` mode. It specifies the destination submap range to be communicated with. It can handle a list of destination IDs

`childRanges`

This is for IP `meshTypes` only. This is an `ixConfigSequenceContainer` with a list of `ixSubmapRanges` for nodes created via the `split` command. This list cannot be extended manually via `appendItem`.

ixIntRange

DESCRIPTION

This will hold the items of comma separated list of ports defined in `portRanges` of `ixDutProtocolPortRange`. These items can either be a single integer value or a range of integers.

```
set my_ixIntRange [::IxLoad new ixIntRange]
$my_ixIntRange config \
-intRange                "16-80"
```

OPTION

`intRange`

The value of `portRanges` of `ixDutProtocolPortRange`.

EXAMPLE

```
set destination [$Traffic1_Network1 getDestinationForActivity "HTTPClient1"
"DUT1:custom"]
```

```
$destination config \
-portMapPolicy                "customMesh"
```

```
$destination portRangeList.clear
```

```
set my_ixIntRange [::IxLoad new ixIntRange]
```

```
$my_ixIntRange config \
-intRange                "16"
```

```
$destination portRangeList.appendItem -object $my_ixIntRange
```

```
set my_ixIntRange1 [::IxLoad new ixIntRange]
```

```
$my_ixIntRange1 config \
-intRange                "18"
```



```
$destination portRangeList.appendItem -object $my_ixIntRange1
```

```
set my_ixCustomPortMap [$destination cget -customPortMap]
```

```
set Submap1 [$my_ixCustomPortMap submapsIPv6.getItem 0]
```

```
$Submap1 config \
```

```
    -name                "Submap1" \
```

```
    -meshType            "ipRangeMesh"
```

ixRepository

ixRepository-Creates a repository object (RXF file).

SYNOPSIS

```
set ::repository [::IxLoad new ixRepository -name path]
```

DESCRIPTION

The repository (.RXF file) object is a set of lists that represents the tree shown in the IxLoad GUI. There are six lists, one for each top-level node in the GUI tree: clientNetworkList, serverNetworkList, dutList, clientTrafficList, serverTraffiand testList.

In order to create a repository, all test components (networks, traffic, activities, traffic-network mappings, and tests) to be saved in a repository must be created in these lists.

Similarly, the contents of an existing repository can be manipulated by maniputhe objects in these lists. The lists are of type `ixConfigSortedNamedItemList`.

For examples of repository usage, see the following scripts in the \Samples direc

- `reprun.tcl` - Runs all tests in a repository.
- `repNewHTTP.tcl` - Creates a new repository.
- `reprunhttpstats.tcl` - Runs all tests in a repository and collects http stats.

SUBCOMMANDS

The options for this command are configured and read using the subcommands defined in the `ixConfigSortedNamedItemList` command.

OPTIONS

`name`

Specifies the path to the file.

`activeTest`

The name of the active test in the repository. This test should be selected when the repository is loaded into the IxLoad GUI.

`clientNetworkList`

List of the client networks in the repository. This is a list of type `ixConfigSortedNamedItemList`.

`serverNetworkList`



List of the server networks in the repository. This is a list of type `ixConfigSortedNamedItemList`.

`dutList`

List of the DUTs in the repository. This is a list of type `ixConfigSortedNamedItemList`.

`clientTrafficList`

List of the client activities in the repository. This is a list of type `ixConfigSortedNamedItemList`.

`serverTrafficList`

List of the server activities in the repository. This is a list of type `ixConfigSortedNamedItemList`.

`testList`

List of the test configurations (traffic-network mappings, timelines, port selectin the repository. This is a list of type `ixConfigSortedNamedItemList`.

`write`

Save the repository to a file. `-write` takes the following arguments:

- `-destination` The path to the file. Can be omitted to rewrite an existing repository opened with the `-name` option.
- `-overwrite` If `true`, overwrites an existing file, provided it is accessible and not write-protected. (Defaults = `false`).

EXAMPLE

```
#Create an empty repository and save it
set ::newRepository [::IxLoad new ixRepository ]
$::newRepository write -destination newRepository -overwrite 1
```

SEE ALSO

statCollectorUtils

ixSendEventCommand - trigger a waiting command

SYNOPSIS

```
$my_ixSendEventCommand config \  
    -option      value
```

DESCRIPTION

`ixSendEventCommand` and `ixWaitEventCommand` synchronize the command lists of two or more activities within a Subscriber NetTraffic. `ixWaitEventCommand` stops command list execution until an `ixSendEventCommand` with a matching `eventId` is called. `ixSendEventCommand` causes all command lists within a Subscriber NetTraffic that are currently stopped by an `ixWaitEventCommand` with a matching `eventId` to resume execution.

`ixSendEventCommand` and `ixWaitEventCommand` are added to an `actionList` using the `appendItem` command.

For example, if Command2 must be executed only after Command1 has been executed:

1. An `ixWaitEventCommand` is inserted preceding Command2.
2. A `ixSendEventCommand` is added after Command1, with the same `eventId` as in the `ixWaitEventCommand`.

When Command1 finishes executing, the `ixSendEventCommand` ends the `ixWaitEventCommand` for Command2, causing Command2 to be executed.

`ixSendEventCommand` and `ixWaitEventCommand` can only be used with Subscriber activities.

OPTIONS

`commandType`

Command type. The only value is "SendEventCommand".

`eventID`

ID of the corresponding `ixWaitEventCommand`. Default value = 1.

EXAMPLE

```
set my_ixSendEventCommand [::IxLoad new ixSendEventCommand]
$my_ixSendEventCommand config \
  -commandType "SendEventCommand" \
  -eventId 1

$Subscriber_Activity_HTTPClient1 agent.actionList.appendItem -object
$my_ixSendEventCommand
.
.
.
$Subscriber_Activity_FTPClient1 agent.actionList.clear

set my_ixWaitEventCommand [::IxLoad new ixWaitEventCommand]
$my_ixWaitEventCommand config \
  -commandType "WaitEventCommand" \
  -eventId 1

$Subscriber_Activity_FTPClient1 agent.actionList.appendItem -object
$my_ixWaitEventCommand
```

SEE ALSO

[ixWaitEventCommand](#)

ixStatCatalogItem

ixStatCatalogItem-Describes a single item in a stat catalog.

SYNOPSIS

```
set statCatalog [$ixTestObject getStatCatalog]

puts [[lindex $statCatalog 0] statSpecList(0).cget -name]
```

DESCRIPTION

The `ixStatCatalogItem` object is a returned element of a list from the `ixTest g` command. It describes a statistics source and all of the statistics and filters available from that source.

SUBCOMMANDS

The options for this command are read using the standard `cget` and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`statFilterList`

(Read Only). The list of all filters available from the agent. Each item of the list is of type `ixStatFilter`. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list.

`statSourceType`

(Read Only). The agent from which statistics originate, of the form:

Protocol Client/Server

Where `Protocol` is one of the supported protocols-for example, HTTP or FTP, and `Client/Server` is one of those two values. Some examples are:

"HTTP Client"

"FTP Server"

`statSpecList`

(Read Only). The list of all statistics available from the agent. Each item of the list is of type `ixStatSpec`. Refer to `ixConfigSequenceContainer` for a list of comthat may be used to manipulate this list.

EXAMPLE

```
puts [[lindex $statCatalog 0] statSpecList(0).cget -name]
```

SEE ALSO

`ixTest`, `ixStatFilter`, `ixStatSpec`

ixStatFilter

ixStatFilter-Describes a single statistics filter in a stat catalog.

SYNOPSIS

```
set statCatalog [$ixTestObject getStatCatalog]

puts [[lindex $statCatalog 0] statFilterList(0).cget -type]
```

DESCRIPTION

The `ixStatFilter` object is one element of the `statFilterList` option of the `ixStatCatalogItem` object. It describes a single filter available for a protocol agent (`statSourceType`).

SUBCOMMANDS

The options for this command are read using the standard `cget` and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`type`

(Read Only). The type of the filter available. One of:

Option	Usage
<code>\$.ixStatFilter(kTypePort)</code> or "Port"	A filter operation may be performed across the port described in the <code>value</code> field.
<code>\$.ixStatFilter(kTypeCard)</code> or "Card"	A filter operation may be performed across the card described in the <code>value</code> field.
<code>\$.ixStatFilter(kTypeChassis)</code> or "Chassis"	A filter operation may be performed across the chassis described in the <code>value</code> field.
<code>\$.ixStatFilter(kTypeActivity)</code> or "Activity"	A filter operation may be performed across the activity described in the <code>value</code> field.
<code>\$.ixStatFilter(kTypeCommunity)</code> or "Traffic-NetworkMapping"	A filter operation may be performed across the community described in the <code>value</code> field.

`value`

(Read Only). A value corresponding to the value of the `type` option. One of:



Option	Usage
<code>\$.ixStatFilter(kTypePort)</code> or "Port"	A port specification in the form: <code>chassis/card/port</code>
<code>\$.ixStatFilter(kTypeCard)</code> or "Card"	A port specification in the form: <code>chassis/card</code>
<code>\$.ixStatFilter(kTypeChassis)</code> or "Chassis"	A port specification in the form: <code>chassis</code>
<code>\$.ixStatFilter(kTypeActivity)</code> or "Activity"	The name associated with an <code>ixCustomPortMap</code> or <code>ixServerTraffic</code> object in the test.
<code>\$.ixStatFilter(kTypeCommunity)</code> or "Traffic-NetworkMapping"	The name associated with an <code>ixClientTrafficNetworkMapping</code> or <code>ixServerTrafficNetworkMapping</code> object in the test.

EXAMPLE

```
puts [[lindex $statCatalog 0] statFilterList(0).cget -type]
```

SEE ALSO

`ixTest`, `ixStatCatalogItem`

ixStatSpec

ixStatSpec-Describes a single statistic in a stat catalog.

SYNOPSIS

```
set statCatalog [$ixTestObject getStatCatalog]

puts [[lindex $statCatalog 0] statSpecList(0).cget -type]
```

DESCRIPTION

The `ixStatSpec` object is one element of the `statSpecList` option of the `ixStatCatalogItem` object. It describes a single statistic available for a protocol agent (`statSourceType`).

SUBCOMMANDS

The options for this command are read using the standard `cget` and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`aggregationFunctionCode`

(Read Only). The type of statistic which this represents. One of:

Option	Usage
"Raw"	
"Interpolated"	
"Interpolated Rate"	
"Rate"	
"Smooth"	
"Interval Maximum"	
"Interval Minimum"	
"Interval Average"	
"Interval Weighted Average"	
"Sum over ports"	
"Maximum over ports"	
"Minimum over ports"	
"Average over ports"	



Option	Usage
"Weighted Average over ports"	

`enablePortAggregation`

(Read Only). If `true`, then it is possible to aggregate this statistic for all agents on a port.

`name`

(Read Only). The name of the statistic. This is the same name that is used in the `name` field of the `::statCollectorUtils::AddStat -statName` argument.

`path`

(Read Only). The internal full-path name of the statistic.

EXAMPLE

```
puts [[lindex $statCatalog 0] statSpecList(0).cget -type]
```

SEE ALSO

`ixTest`, `ixStatCatalogItem`

ixTest

ixTest-Builds a complete IxLoad test.

SYNOPSIS

```
set Test1 [::IxLoad new ixTest]
```

```
$test subcommand options...
```

DESCRIPTION

The `ixTest` command is used to construct a complete IxLoad test structure. It conof a list of client traffic-network and server traffic-network mappings, called communities. In addition to the two lists, several options control global operaAn `ixTest` command is used in conjunction with a `ixTestController` to operthe test and collect statistics.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addithe following commands are available. Unless otherwise described, no valare returned and an exception is raised for any error found.

getStatCatalog

This subcommand returns a list of objects of type `ixStatCatalogItem` that define all of the statistics available, along with all possible filters. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to maniputhis list.

getCommunityList

This subcommand returns all the communities in the test in no particular order. It is provided for convenience. It is essentially equivalent to getting the `clientComand serverCommunityList` and concatenating them.

```
# set the chassis chain on the repository
# since there is no chassisChain clear,
# it's easiest to start with a new one

myChassisChain = new ixChassisChain
myChassisChain.addChassis("myChassis")
repository.chassisChain = myChassisChain

# set ports on all the networks in the tests
# for test in repository.testList:
# or pick a specific test

for community in test.getCommunityList():
community.network.portList.clear()

# update x & y with next card and port to assign
# (assuming single chassis)
```



```
community.network.portList.appendItem(chassisId = 1, \ cardId = x, portId = y)
```

For an example of how to load a repository, see `RepRun.tcl` in the `Samples` directory.

OPTIONS

`clientCommunityList`

A list of objects of type `ixClientTrafficNetworkMapping` that define the client agent to network mappings used to generate client traffic. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list. (Default = `{}`).

`comment`

A comment associated with the test. (Default = `""`).

`csvInterval`

The interval, in seconds, at which the CSV statistics files are updated. In the GUI, this parameter is on the Test Options pane and is labeled `CSV Polling Interval`. This parameter does not set the statistics callback interval, which you must define manually for each script (see `statCollectorUtils` on page 4-60). (Default = `4`).

`enableForceOwnership`

If `true`, at the beginning of the test, any ports that are selected for the test but owned by another user are rebooted and their previous ownership cleared. This parameter corresponds to the GUI option "Forcefully Take Ownership." (Default = `false`).

`enableReleaseConfigAfterRun`

If `true`, purges the test configuration from the ports after a test completes, releases ownership of them, and the ports will no longer respond to ARPs and PINGs from the DUT. (Default = `false`).

`enableResetPorts`

If `true`, `IxLoad` reboots the ports before downloading the test configuration to them. To ensure the integrity of your testing, it is always safest to reboot the ports before running a test. However, rebooting the ports does increase the amount of time required to prepare the ports for a test.

If you are developing a test and making incremental changes to it and then `runit` to see the effect of your changes, it may be safe to save time by not rebooting the ports before each run.

If you do not want to reboot the ports for every test, you should at least reboot the ports the first time you load a repository; this will ensure that any software structures remaining from a previous test or other application are properly removed. (Default = `false`).

name

The name associated with this object. (Default = "TestEnvelope").

serverCommunityList

A list of objects of type `ixServerTrafficNetworkMapping` that define the server agent to network mappings used to simulate network servers. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list. (Default = {}).

statsRequired

If true, statistics will be collected for the test. (Default = 1).

EXAMPLE

```
#-----  
# Create the test  
#-----  
set Test1 [::IxLoad new ixTest]  
$Test1 config \  
    -comment                "" \  
    -csvInterval            4 \  
    -name                    "Test1" \  
    -statsRequired          1 \  
    -enableResetPorts       0 \  
    -enableForceOwnership   false \  
    -enableReleaseConfigAfterRun 0 \  
    -captureViewOptions     $my_ixViewOptions  
  
$Test1 scenarioList.clear  
  
$Test1 scenarioList.appendItem -object $TrafficFlow1
```

SEE ALSO

`ixTestController`, `ixDut`



ixTestController

ixTestController-Controls execution of an IxLoad test.

SYNOPSIS

```
set testController [::IxLoad new ixTestController options]
```

```
$testController subcommand options...
```

DESCRIPTION

The `ixTestController` command is used to control the execution of an IxLoad test. The `ixTest` object is referenced in this command's `run` subcommand.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition, the following commands are available. Unless otherwise described, no value is returned and an exception is raised for any error found.

applyConfig

Downloads the test configuration to the Ixia ports. The syntax is the same as for the `run` subcommand.

applyObjectiveValues

Applies the new objective values that are configured on the activity when the test is running. See the example for `canSetObjectiveValue`.

autorepository

Automatically creates a repository that is used as the source of data for the "Test Configuration" section in a generated report. The repository is created in the results (`$resultDir`) directory.

The `autorepository` and `repository` subcommands can both be used to create repositories that are the source of data for the "Test Configuration" section of reports.

- `autorepository` creates a repository based on IxLoad's internal, ephemeral repository.
- `repository` creates a copy of the repository specified by `$reposit`(created using `ixRepository`).

The choice of which one to use depends on whether or not you are using an `$repository` object in your test:

- If you are using `$repository`, use `repository`.
- If you are not using `$repository`, use `autorepository`.

`autorepository` requires the repository file name as an argument.

For example, the following generates a repository named `My_Rep.rxf`:



```
$testController run $test -autorepository "My_Rep.rxf"
```

canSetObjectiveValue

Checks whether the objective value can be set on the activity when the test is running. The objective value can be changed only during the Ramp-up and Sustain phases of the test.

```
# Modify objective value on the fly every 40s (3rd one should give a warning)
```

```
set maxObjectiveValue [$Activity_newClientActivity1 getMaxObjectiveValue]
```

```
puts "Max objective value - $maxObjectiveValue"
```

```
set objectiveValue 733100
```

```
for {set j 0} {$j < 3} {incr j} {
```

```
    sleep 40
```

```
    puts "Trying to change objective to $objectiveValue..."
```

```
    $Activity_newClientActivity1 config -objectiveValue $objectiveValue
```

```
    set canSetObjectiveValue [$Activity_newClientActivity1  
canSetObjectiveValue]
```

```
    if { $canSetObjectiveValue } {
```

```
        $Activity_newClientActivity1 applyObjectiveValues
```

```
    }
```

```
    incr objectiveValue 100
```

```
}
```

copyFile

Copies files from Windows to Linux. You can use `copyFile` to retrieve results files.

```
set resultDir [$testController getRunResultDirFull]
```

```
set remoteFile [file join $resultDir IxLoadDetailedReport.pdf]
```

```
::IxLoad remoteCopyFile $remoteFile /root/Report1.pdf
```

collectDebugLogs

This command places the debug logs in the `DebugInfo\Logs` directory of the configured results directory of the test configuration on the PC running the IxLoad client GUI. For example:

```
<ResultsDirectory>\ DebugInfo\Logs
```

Example:

```
$testController collectDebugLogs
```

getTestServerHandle

This subcommand returns a string used for statistics collection using the `statCollectorUtils::Initialize` command. IOR stands for Interoperable Object Refname given to a network-addressable reference as defined by CORBA.

getMaxObjectiveValue

Fetches the maximum objective value that can be configured on the activity when the test is running. This value is shown as the maximum value that can be set using the Objective slider on the Run tab of the application. See the example below and the example for `canSetObjectiveValue`.

Example:

```
proc ::my_stat_collector_command {args} {
    set ::ixStatCollectorMonitor "statsReceived"
}
${NS}::StartCollector -command ::my_stat_collector_command -interval 4

set ::ixTestControllerMonitor ""
$testController run $Test1

# wait till we get stats, indicating test is starting to run
vwait ::ixStatCollectorMonitor

set maxObjectiveValue [$Activity_newClientActivity1 getMaxObjectiveValue]
puts "Max objective value - $maxObjectiveValue"
```

generateReport

This command performs the report generation from TCL. The `generateReport` function is called after a test is run and completed. The test run generates certain CSV files. These files are stored in the result directory and contain the test `statisxReporter` processes the resulted CSVs and generates the PDF file. (Default = 1).

detailed report

To generate a detailed report, it is required to get the `.rxf` file for the test, and specify the detailed report option 1.

```
$testController run $test -autorepository <rxf Name>

vwait ::ixTestControllerMonitor

$testController releaseConfigWaitFinish

$testController generateReport -detailedReport 1
```

summary report

To generate a summary report, specify the summary report option 0.



```
$testController generateReport -detailedReport 0
```

getRunResultDirFull

The `getRunResultDirFull` returns the directory into which the generated report has been placed.

```
$testController generateReport -detailedReport 1  
  
set resultDir [$testController getRunResultDirFull]
```

isBusy

Following a call to the `run` subcommand, this subcommand returns `true` while the test is still running.

repository

Creates the repository that is used as the source of data for the “Test Configura” section in a generated report. The repository is created in the results (`$resultDir`) directory.

For example, the following generates a repository using the configuration specify the `ixRepository` object:

```
$testController run $test -repository $repository
```

The `autorepository` and `repository` subcommands can both be used to create repositories that are the source of data for the “Test Configuration” section of reports. For a description of the differences between them, see `autorepository`.

run \$test

This command causes the test specified in `$test`, which must be an object of type `ixTest`, to start. After calling the `TestController run` function, your script must call `vwait`

`::ixTestControllerMonitor` to ensure that the Tcl event loop is processed. Otherwise, `IxLoad` will not call your statistics callback command, and you will not be able to tell when the test ends.

If you have a lot of processing to do after calling `run`, but before the test ends, your code may be executing when `IxLoad` sets the `::ixTestController monitor` variable. Example 2 (see below) shows how to correctly handle detecting the end of test if this possibility exists.

setResultDir \$dir

Specifies the location of where the execution results will be kept. If this subcomis not called, no results will be stored. When running from a Unix client, this is a directory on the intermediate Windows host that the client connected to using the `connect` subcommand of `::IxLoad`. The `retrieveFile` or `retrieveFileCopy` subcommands of `::IxLoad` can be used to retrieve the files from the Windows host.

Within the directory you specify for `setResultDir`, `IxLoad` stores the following files for the current test:

```
Test_Client.csv  
  
Test_Server.csv  
  
<Protocol>_<Client|Server>.csv
```

```
<Protocol>_<Client|Server>_-_Default_CSV_Logs_<activity name>_<traffic name>@<network name>.csv
```

```
Test_Client.csv
```

```
Test_Server.csv
```

<Protocol> is the name of the protocol (for example, HTTP). There will be a set of files for each protocol used in the test.

<Client|Server> is the side of the connection, client or server. There will be a set of files for each side used in the test.

<activity name> is the name of the activities (agents) appended to the agentList of the traffic.

<traffic name> is the name of the ixCustomPortMap or ixServerTraffic element created in the test.

<network name> is the name of the ixDHCP or ixStatCatalogItem element created in the test.

stopRun

Stops the test.

OPTIONS

outputDir

If this is empty (""), then no result CSV files are saved. If this is not empty (for example, "1"), then CSV files are saved. (Default = "").

EXAMPLE

```
# Example 1: First method of using vwait

set testController [::IxLoad new ixTestController -outputDir 1]

$testController setResultDir \
    "[pwd]/RESULTS/simplehttpclientandserver"

# Run the test

$testController run $test

vwait ::ixTestController
```

#Example 2: Second method of using vwait. This method is useful if you have processing you wish to do while the test is running.

```
# Code to set up and define test and testController
```



```
# ...

# The following function is useful to delay while running
# the Tcl event loop.
proc sleep {duration} {
    after $duration {set wakeUp 1}
    vwait wakeUp
}

set ::ixTestControllerMonitor ""
$testController run $test

# Other activities here. While waiting you must call
# either vwait or update to ensure your statCollector command
# is called.

#
# wait, if necessary, until the test is over
#
while {[lsearch $::ixTestControllerMonitor TEST_STOPPED] == -1} {
    sleep 1000
}

puts $::ixTestControllerMonitor
```

SEE ALSO

ixTest

ixTestControllerMonitor

ixTestControllerMonitor-Global variable to wait on for test completion.

SYNOPSIS

```
vwait ::ixTestControllerMonitor
```

DESCRIPTION

The global variable `ixTestControllerMonitor` is maintained by `ixTestController` while a test is running. Its value may be `vwait`'d to determine when the test is complete.

`ixTestControllerMonitor` is set by `IxLoad` either at the end of the last `testControlCommand` (using either the `applyConfig` or `run` options). `ixTestControllerMon` will only be set while inside a `vwait` command or an `update` command.

The reason you should initialize `ixTestControllerMonitor` prior to issuing the `testCommand` is because it is `vwaiting` on something other than `ixTestConmonitor`, so you need to be able to detect the end of the test by examining the value of `::ixTestControllerMonitor`. Also, because it is not set by `IxLoad` prior to the end of the test (or `applyConfig`), it will be undefined otherwise.

Usually, you can use `vwait` or `ixTestControllerMonitor` directly, but if the script needs to do some other processing while the test is running, the following examfrom the `simplehttpconfigstoprun.tcl` sample script shows how this can be done.

In this example, the code waits for the first statistic to arrive, and then falls through if the test stops or the event occurs:

```
set ::ixTestControllerMonitor "" # initialize to known value

$testController run $test# do the command

# wait for the first sample or test stop

while {$::ixTestControllerMonitor == "" && $::gotOneStat == 0} {

    after 1000 set wakeup 1

    vwait wakeup          # you have to call vwait (or update)
                        # periodically to allow IxLoad to run

}
```

While waiting for the test to finish, the script must call either `vwait` (as in the example) or `update` to allow the Tcl event loop to function.

`ixTestControllerMonitor` returns one of the following values:

```
{eventType TEST_STOPPED status OK}

{eventType TEST_STOPPED status ERROR description {1}}
```

If an error occurs, refer to the log file to determine the cause.

EXAMPLE

See the example under `statCollectorUtils`.

SEE ALSO

`statCollectorUtils`, `ixTestController`

statCollectorUtils

statCollectorUtils-Handles statistics gathering.

SYNOPSIS

```
package require statCollectorUtils
```

```
::statCollectorUtils::command args
```

DESCRIPTION

The `statCollectorUtils` is a library containing several commands to gather statistics during a test run. The model for usage of these commands is:

- `Initialize` -Initializes the statistics utilities.
- `ClearStats` -Clears statistics from a previous run.
- `AddStat`-Adds a statistic to the list of statistics to be retrieved. Call this once per statistic.
- `AddNetworkStat`-Adds a network statistic to the list of network statistics to be retrieved. Call this once per statistic.
- `SetCsvVersion` -Allows the stat names written to the CSV to be the same as would be the case if generated by the given `buildNumber` or special constant.
- `SetCsvThroughputUnits` -Defines the units used for throughput statistics written to the CSV files.
- `StartCollector` `-command callbackCommand` -Starts the statistics collection process and indicates a callback command to invoke when statistics are delivered.
- Use `ixTestController run` to run the test.
- Use `vWait ::ixTestControllerMonitor` to wait for the test to end. During the run, the callback command indicated in `StartCollector` is called.
- `StopCollector` -Stops the statistics collection process.



Note: Conditional stats and the Network overview with their associated drill-downs are not supported in the Tcl API.

COMMANDS

Unless otherwise described, no values are returned and an exception is raised for any error found.

AddStat arguments

Adds a statistic to the list of desired responses. The arguments to this command are `-option value` pairs:

Option	Usage
aggregationType	<p>Specifies how statistics for multiple ports, as indicated in the <code>filter</code> argument, are combined. One of:</p> <ul style="list-style-type: none"> ▪ “kSum”-Adds all of the statistics together. ▪ “kMax” -Determines the maximum value. ▪ “kMin”-Determines the minimum value. ▪ “kAverage”-Determines the average value. ▪ “kWeightedAverage”-This type is for use with weighted statistics. The statistics descriptions indicate whether they are weighted or not. ▪ “kRate” -Determines the rate of change of the sum of all the statistics. ▪ “kMaxRate” -Determines the maximum rate. ▪ “kMinRate”-Determines the minimum rate. ▪ “kAverageRate”-Determines the average rate. ▪ “kString”-Treats as a string.
caption	The caption associated with the statistic. This is not currently used by the Tcl API, but a comment must be supplied.
enumerated	<p>If <code>true</code>, returns a list of stats as follows:</p> <ul style="list-style-type: none"> ▪ HTTP Client: Returns one stat in the callback for each <code>difURL</code> in the client's command list. ▪ HTTP Server: Returns one stat in the callback for each defined server page. <p>If <code>false</code> (default), returns a single stat for all URLs.</p>
filterList	<p>A list of filter items of type <code>ixStatFilter</code> that specify how to filter the statistics. Elements of the <code>filterList</code> should be of the form: <code>Chassisn/Cardm/Portp</code>, where <code>n</code> is the chassis number, <code>m</code> is the card on the chassis, and <code>p</code> is the port on the card.</p>
statName	<p>A particular statistics name, as listed in the <code>Statistics</code> topic of the manual page for the protocol client or server agent. For example, the list of statistics for HTTP Clients can be found in the <code>Statistics</code> topic at <code>HTTP Client Agent</code>.</p>
statSourceType	<p>The agent type that generates the statistics. This is a two part name of the form:</p> <pre>Protocol Client Server</pre> <p>Where <code>Protocol</code> is one of the supported protocols-for examHTTP or FTP, and <code>Client/Server</code> is one of those two values. Some examples are:</p> <pre>"HTTP Client"</pre> <pre>"FTP Server"</pre>

AddNetworkStat arguments

Adds a dynamic range network statistic to the list of network statistics to be retrieved.



Note: If you want to create a sample script using ScriptGen, the network statistics are not available until the test configuration has been downloaded to the Ixia ports. You can use the Apply Config command to download the test configuration to the ports without starting the test.

The arguments to this command are similar to those for AddStat. The arguments to this command are - option value pairs:

Option	Usage
aggregationType	See AddStat for description.
caption	See AddStat for description.
filterList	See AddStat for description.
statName	See AddStat for description.
statSourceType	The network plugin that generates the statistics. For example: "IPSec" The list of network plugin names is: <ul style="list-style-type: none">▪ WebAuth▪ 802.1x▪ EAPoUDP▪ IPSec▪ L2TP_PPP/PPP▪ L2TP_PPP/PPPoE▪ GTP▪ 3GPP▪ IMPAIR

ClearStats

Clears all statistical data from a previous or aborted run.

AddPerInterfaceStat arguments

This is the utility for per range interface statistics.

To activate this statistics for the client and the server, enable IP interface (`enableStats`) statistics in the client and the server network.

Option	Usage
statSourceType	<p>The agent type that generates the per interface statistics. This is a two part name of the form:</p> <pre>Interface Protocol - Client Server</pre> <p>Where Protocol is IPV4 or IPV6 and Client Server is one of those two values. Some examples are:</p> <pre>"Interface IPv4 Client" "Interface IPv4 Server" "Interface IPv6 Client" "Interface IPv6 Server"</pre>
statList	<p>This is a list of statistical names and aggregations function pair.</p> <p>An example of IPV4:</p> <pre>-statList {{ "Packets Sent" "kSum" } { "Packets Received" "kSum" }} \</pre>
ipList	<p>This is the list of IPs specified for the client and the server for collecting the statistics.</p> <pre>-ipList { "198.18.2.1" "198.18.2.2" }</pre>

For each address that you specify in ipList, IxLoad records the statistics specified in statList.

For example, if you specify an ipList and statList as follows:

```
-statList {{ "Packets Sent" "kSum" } { "Packets Received" "kSum" }} \
-ipList { "198.18.2.1" "198.18.2.2" }
```

and a returned list of statistics contains the following:

```
{{kInt 28112} {kInt 0} {kInt 31973} {kInt 0}}
```

{kInt 28112} is the statistic for Packets Sent for address 198.18.2.1.

{kInt 0} is the statistic for Packets Received for address 198.18.2.1.

{kInt 31973} is the statistic Packets Sent for address 198.18.2.2.

{kInt 0} is the statistic for Packets Received for address 198.18.2.2.

Because IxLoad adds the statistics in the order specified by statList for every address in ipList, you can parse the list of statistics returned by callback (see the StartCollector command) to obtain any specific statistic.

Initialize -testIOR \$testIOR

Initializes the statistics utility package. \$testIOR is the value returned from a call to ixTestController's getTestServerHandle subcommand. See the following example:

```
set tc [::IxLoad new ixTestController]
::statCollectorUtils::Initialize -testIOR [$tc getTestServerHandle]
```

SetCsvVersion <buildNumber>

Allows the stat names written to the CSV to be the same as would be the case if generated by the given `buildNumber` or special constant. The build number must be in dotted-quad notation (a.b.c.d). The build number is part of the installation path when IxLoad is installed. `buildNumber` must be 5.0.117.0 or greater. If set to anything less than 5.0.117.0, the value is ignored.

Special Constant	Description
rx	Uses the build number of the version of IxLoad that most recently saved the repository. If the TCL API does not load a repository, then it uses the build number of the current instance of IxLoad.
current	Uses the build number of current instance of IxLoad in all cases.

```
set tc [::IxLoad new ixTestController]
::statCollectorUtils::SetCsvVersion 5.0.280.0
```

SetCsvThroughputUnits <throughputUnits>

Defines the units used for throughput statistics written to the CSV files. `throughputUnits` can be one of: Bps (bytes per second, the legacy unit), Kbps, Mbps, or Gbps.

This overrides any IxAppOption.ini entry, allowing allowing your script to determine the units used for throughput statistics written to the CSV files.

StartCollector -command tclCommand -interval value

Initiates the operation of the statistics collection process, registering the name of a user supplied command (`tclCommand`), which will be called at `-interval` when new statistics are received.

Callback Command Invocation

The statistics callback interval (`-interval`) must be set manually. It is not set by the `csvInterval` parameter (see `ixTest`). To invoke the statistics callback, define the statistics as a set of name-value pair arguments of the form:

```
{timestamp 1102900690000 stats {{kInt 1659316} {kInt 58998232}}}
```

The pairs are:

Option	Usage
timestamp	The number of milliseconds from the time that the test started.
stats	A list of pairs, one per statistic registered with <code>AddStat</code> in the order registered. The first member of each pair indicates the data type of the value, one of: <ul style="list-style-type: none"> ▪ <code>kInt</code> -an integer value. ▪ <code>kStr</code> -a string. For example: {this is a string}.

StopCollector

Stops the operation of the statistics collector.

OPTIONS

None.

EXAMPLE

```
#-----  
# Set up stat Collection  
#-----  
set NS statCollectorUtils  
  
set ::test_server_handle [$testController getTestServerHandle]  
  
${NS}::Initialize -testServerHandle $::test_server_handle  
  
#-----  
# Clear any stats that may have been registered previously  
#-----  
  
${NS}::ClearStats  
  
#-----  
# Define the stats we would like to collect  
#-----  
  
${NS}::AddStat \  
-caption "Watch_Stat_1" \  
-statSourceType "HTTP Client" \  
-statName "HTTP Bytes Sent" \  
-aggregationType kSum \  
-filterList {}  
  
  
${NS}::AddStat \  
-caption "Watch_Stat_2" \  
-statSourceType "HTTP Client" \  
-statName "HTTP Bytes Received" \  

```



```
-aggregationType kSum \  
-filterList {}  
  
${NS}::AddStat \  
-caption "Watch_Stat_3" \  
-statSourceType "HTTP Client" \  
-statName "HTTP Time To Last Byte (ms)" \  
-aggregationType kWeightedAverage \  
-filterList {}  
  
    ${NS}::AddStat \  
-caption "Watch_Stat_4" \  
-statSourceType "HTTP Client" \  
-statName "HTTP Bytes Sent" \  
-aggregationType kRate \  
-filterList {}  
  
    ${NS}::AddStat \  
-caption "Watch_Stat_5" \  
-statSourceType "HTTP Client" \  
-statName "HTTP Bytes Received" \  
-aggregationType kRate \  
-filterList {}  
#-----  
# Define the network stats we would like to collect  
#-----  
  
set ::netstatList { \  

```

```

{"IPSec" "Interface ID" "kString"} \
{"IPSec" "Status" "kString"} \
{"IPSec" "NAT-T" "kString"} \
{"IPSec" "DPD" "kString"} \
{"IPSec" "Total Retries" "kSum"} \
{"IPSec" "Total Latency" "kSum"} \
{"IPSec" "Encapsulation Protocols" "kString"} \
{"IPSec" "Encapsulation Mode" "kString"} \
{"IPSec" "Initiator Subnet" "kString"} \
{"IPSec" "Initiator IP Address" "kString"} \
{"IPSec" "Responder IP Address" "kString"} \
{"IPSec" "Responder Subnet" "kString"} \
}

foreach statItem $::netstatList {

    set caption          [format "Watch_Stat_%s" $count]

    set statSourceType   [lindex $statItem 0]
    set statName         [lindex $statItem 1]
    set aggregationType [lindex $statItem 2]

    ${NS}::AddNetworkStat \
    -caption             $caption \
    -statSourceType      $statSourceType \
    -statName            $statName \
    -aggregationType     $aggregationType \
    -filterList          {}
}

```



```
        incr count
    }

# Start the collector (runs in the tcl event loop)
#
proc ::my_stat_collector_command {args} {
    puts "====="
    puts "INCOMING STAT RECORD >>> $args"
    puts "Len = [llength $args]"
    puts [lindex $args 0]
    puts [lindex $args 1]
    puts "====="
}

${NS}::StartCollector -command ::my_stat_collector_command -interval 2

#-----
# Run the test
#-----
$testController run $test

#-----
# have the script (v)wait until the test is over
#-----
    vwait ::ixTestControllerMonitor;
puts $::ixTestControllerMonitor

#-----
-----
```

```
# Stop the collector (running in the tcl event loop)
#-----
----
${NS}::StopCollector
```

SEE ALSO

`ixTestController`, `ixTestControllerMonitor`, `ixTest`

ixScriptGen

ixScriptGen-Generates a tcl script (TCL file).

SYNOPSIS

```
set scriptGenObj [::IxLoad new ixScriptGen]
```

DESCRIPTION

A scriptGen object is created and configured. scriptGen can generate a Tcl script for the following:

- Complete test
 - NetTraffics
 - Activities
 - Networks
-

SUBCOMMANDS

None.

OPTIONS

fileName

Specifies the name and path of the script to be generated.

includeStats

If `true`, the script includes code to record the default statistics for each activity in the test. If `false`, the script does not include any code to record statistics.

configSetting

This option determines whether or not the generated script includes code that sets the test control options to their default values.

Option	Usage
kConfigWriteAll	generates a script that includes all the test control code, includcode that sets the configuration options to their default val
kConfigCommen t	generates a script that comments out test control code that sets options to their default values.
kConfigOmit	generates a script that only includes test control code for options set to non-default values.

EXAMPLE

```
# Scriptgen for a complete script

#-----

"if {$::tcl_platform(platform) == "windows"} {

package require registry 1

set ::_IXLOAD_INSTALL_ROOT [registry get {HKEY_LOCAL_MACHINE\Software\Ixia
Communications\IxLoad\InstallInfo} HOMEDIR]

set ::_IXLOAD_PKG_DIR [file join $::_IXLOAD_INSTALL_ROOT Client tclext teepee
stage]

lappend ::auto_path $::_IXLOAD_PKG_DIR

}

package require IxLoad

::IxLoad connect 1.2.3.4

if [catch {

set logtag "IxLoad-api"

set logName "scriptgen"

set logger [::IxLoad new ixLogger $logtag 1]

set logEngine [$logger getEngine]

$logEngine setLevels $::ixLogger(kLevelDebug) $::ixLogger(kLevelInfo)

$logEngine setFile $logName 2 256 1

#-----

# Create a test controller bound to the previously allocated
# chassis chain. This will eventually run the test we created earlier.
#-----

set testController [::IxLoad new ixTestController -outputDir 1]

$testController setResultDir "[pwd]/RESULTS/reprun"
```



```
#

# Load the repository

#

set repository [::IxLoad new ixRepository -name
{E:\ixweb\ixweb\3.20\automation\B2B_310_IMAP_RTSP_TELNET_POP\Repository\IMAP_
dns_all_atomic_level_cmd_ipv4.rxf}]

set testName [$repository testList(0).cget -name]

set test [$repository testList.getItem $testName]

set scriptGenObj [::IxLoad new ixScriptGen]

$scriptGenObj config \

    -fileName
    {E:\ixweb\ixweb\3.20\automation\B2B_310_IMAP_RTSP_TELNET_POP\Repository\IM
AP_dns_all_atomic_level_cmd_ipv4_new.tcl} \

    -includeStats      False \

    -configSetting     $::ixScriptGen(kConfigWriteAll)

$scriptGenObj scriptGen $test

}] {

puts $errorInfo

}

::IxLoad disconnect
```

SEE ALSO

`ixNetTraffic`

ixTimeline

ixTimeline-Configures the time in the test when the activities in the NetTraffics come online, and how long they stay up for. It is also used to configure the test's objectives.

SYNOPSIS

```
set Activity_HTTPClient1 [$Traffic1_Network1 activityList.appendItem options...]  
  
set Timeline [::IxLoad new ixTimeline] options...
```

DESCRIPTION

The `ixTimeline` command is used to create a test scenario. It controls the times and rates at which Activities come online (rampUp), the length of time they stay up for (sustainTime), and the rate at which they go offline (rampDown).

There are two types of Timelines:

Basic: A Basic timeline controls activities linearly -- the rampUp, sustain, and rampDown phases are straight lines, and the rampUp, sustain, and rampDown occur at steady rates, either increasing (rampUp), static (sustain) or decreasing (rampDown).

Advanced: An advanced timeline allows you to plan the traffic shape to the objectValue, such as pulses or bursts. An Advanced timeline displays the ramsustain, and Rampdown phases as segments. There are five types of seg

Linear: a constant-slope segment that starts with the current objective value and ends at the End Objective Value value.

Step: a classic stair step pattern that starts with the current objective value and ending after a number of fixed deltas.

Burst: a burst segment starts with the current objective value and ends to the same objective value. Burst segments produce a symmetrical triangular shape fluctuation.

Pulse: a pulse segment starts with the current objective value and ends to the same objective value. Pulse segments produce a symmetrical pulse shape fluctuaan increase in rate,a duration of time spent at the new peak and then drop to the starting value.

Poisson: a poisson segment introduces a logarithmic noise element into the objective value.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS (BASIC TIMELINE)

rampUpValue



Value applied to `rampUpType` to either bring up users at a certain rate (Users per second) or to maintain a pool of users waiting to establish connections (MaxUsers).

`rampUpType`

Ramp-up type used by timeline. One of:

Value	Description
-2	Mixed (may be displayed when ramp-up value is retrieved for a community of mixed activities)
-1	N/A
0	Users per second
1	Maximum pending users
2	Smooth users

`offlineTime`

The amount of time agents are idle between iterations. (Default = 0). This is also applicable to advanced timeline.

`rampDownTime`

Amount of time used for closing any TCP connections that are still open after all transactions are complete. `rampDownTime` applies only to client activities.

`standbyTime`

The amount of time, expressed in seconds, that elapses between the time the test is started and the time that the traffic-network pair become active. The valid range is from 0 to 1,000 hours (3,600,000). (Default = 0). This is also applicable to advanced timeline.

`iterations`

The number of times that the traffic-network pair perform their functions (establish TCP connections, retrieving FTP files, and so forth) in the test. (Default = 1). This is also applicable to advanced timeline.

`rampUpInterval`

This field accepts integer values. The value for this option will be considered only when `rampUpType` is `usersPerSecond`. You can edit the value to increment or decrement the number of users to be started at every `rampUpInterval`. (Default = 1).

`sustainTime`

Amount of time when all users are up and performing the central test objectives, such as retrieving or serving pages (HTTP), or sending or receiving files (FTP).

timelineType

Denotes the type of phase in a section of the timeline. This is also used with the advanced timeline options.

name

Name of the Timeline.

OPTIONS (ADVANCED TIMELINE)

ixLinearTimeSegment

duration

The length of time that the segment lasts.

noiseAmplitudeScale

Amount of Gaussian noise added during the segment.

No noise is added to the last point in the segment so that the segment can end at the specified End Objective Value

endObjectiveScale

The value of the objective at the end the segment.

ixPoissonTimeSegment

duration

The length of time that the segment lasts

averageScale

Number used to compute the Poisson distribution for the segment

noiseAmplitudeScale

Amount of Gaussian noise added during the segment.

No noise is added to the last point in the segment so that the segment can end at the specified End Objective Value

ixPulsesTimeSegment

`amplitudeIncrementStepScale`

Additional gain in height (amplitude) from one pulse to the next.

`pulseRampDownDuration`

Amount of time allocated to the ramping-down of the pulse.

`pulseRampUpDuration`

Amount of time allocated to the rising edge of the pulse.

`numberOfRepetitions`

Number of steps. Minimum of 1.

`pulseOfflineDuration`

Time between pulses.

`pulseSustainDuration`

Length of time that the pulse occupies at the new peak value.

`noiseAmplitudeScale`

Amount of Gaussian noise added during the segment.

No noise is added to the last point in the segment so that the segment can end at the specified End Objective Value.

`startingPulseAmplitudeScale`

Height (amplitude) of the first pulse.

ixBurstsTimeSegment

`noiseAmplitudeScale`

Amount of Gaussian noise added during the segment.

No noise is added to the last point in the segment so that the segment can end at the specified End Objective Value

numberOfRepetitions

Number of steps. Minimum of 1

startingBurstHeightScale

Height (amplitude) of the first burst.

burstIncrementStepScale

Additional gain in height (amplitude) from one burst to the next.

burstDuration

Length of time that the burst occupies

burstSkew

Bias applied to the burst curve:

Symmetric: No bias (curve has identical slopes on both sides).

Left: Curve is biased to the left (left side of the curve is steeper than the right).

Right: Curve is biased to the right. (right side of the curve is steeper than the right)

burstOfflineDuration

Time between bursts.

ixStepsTimeSegment

stepHeightScale

Height of the step.

noiseAmplitudeScale

Amount of Gaussian noise added during the segment.

No noise is added to the last point in the segment so that the segment can end at the specified End Objective Value

stepSustainDuration

Length of time spent at the new peak objective value.



stepRampDuration

Length of time allocated to the rise to the new peak value.

numberOfRepetitions

Number of steps. Minimum of 1.

EXAMPLE

```
set Steps_Segment_0 [::IxLoad new ixStepsTimeSegment]

$Steps_Segment_0 config \

    -stepHeightScale           0.1 \
    -noiseAmplitudeScale       0.0 \
    -stepSustainDuration       20 \
    -stepRampDuration          20 \
    -numberOfRepetitions       3

$my_ixAdvancedIteration appendSegment $Steps_Segment_0

set Bursts_Segment_1 [::IxLoad new ixBurstsTimeSegment]

$Bursts_Segment_1 config \

    -noiseAmplitudeScale       0.0 \
    -numberOfRepetitions       3 \
    -startingBurstHeightScale  0.1 \
    -burstIncrementStepScale   0.1 \
    -burstDuration             20 \
    -burstSkew                 0 \
    -burstOfflineDuration      20

$my_ixAdvancedIteration appendSegment $Bursts_Segment_1

set Pulses_Segment_2 [::IxLoad new ixPulsesTimeSegment]
```

```
$Pulses_Segment_2 config \  
-amplitudeIncrementStepScale          0.1 \  
-pulseRampDownDuration                t
```

SEE ALSO

`ixNetTraffic`



ixSubscriberNetTraffic

ixSubscriberNetTraffic-Special type of NetTraffic that simulates the traffic patcreated by residential customers that receive voice, video, and data service (Triple-play) over a single physical connection (usually a cable or DSL connec

SYNOPSIS

```
set Subscriber1_Network1 [::IxLoad new ixSubscriberNetTraffic]
```

DESCRIPTION

Configuring an ixSubscriberNetTraffic is similar to configuring an ixNetTraffic. However, there are some differences:

Network and Protocols: Configuring a Subscriber is similar to configuring a NetTraffic. However, only the following protocols are supported:

DHCP	FTP	HTTP	IMAP
IPTV/ Video	LDAP	MGCP	POP3
RADIUS	RTSP	SIP	SMTP
SSH	TraceFileRepl ay	Telnet	DNS

objectiveType: The only objectiveType available for a Subscriber is simulatedUsers.

OPTIONS

Refer ixNetTraffic for information on the options.

EXAMPLE

```
set Subscriber1_Network1 [::IxLoad new ixSubscriberNetTraffic]

set Subscriber_Activity_HTTPClient1 [$Subscriber1_Network1
activityList.appendItem \

-protocolAndType                                "HTTP Client" ]

$Subscriber1_Network1 config \

-enable                                          true \

-network                                        $Network1

$Subscriber1_Network1 traffic.config \

-name                                           "Subscriber1"
```

SEE ALSO

`ixNetTraffic, ixBandwidthLimit`



ixNetTraffic

ixNetTraffic-Define client and server traffic.

SYNOPSIS

```
set HTTP_client_client_network [::IxLoad new ixNetTraffic]
```

DESCRIPTION

The `ixNetTraffic` command is used to configure client or server traffic. Two separate `ixNetTraffic` objects have to be created for client and server traffic. The `ixNetTraffic` configuration also declares the `ixNetworkGroup` object. The activity options are also configured.

OPTIONS

Enabling Options

`enable`

This enables the client or server network.

`network`

This specifies the name of the client or server network object.

activityList Configuration Options

`enable`

If true, this mapping is included in the `IxLoad` test. (Default = true).

`name`

Name of the `activityList` config object. Default = "newClientActivity1".

`enableConstraint`

Currently, constraints can be set on activities that run rate-based objectives, like `connectionRate`, `transactionRate`, throughput objectives. This option enables the constraint. Default = false.

`constraintValue`

If `enableConstraint` is true, this option mentions the constraint value. Default = 100.

timeline

Represents the name of the ixTimeline object.

userObjectiveType

userObjectiveType is the recommended way to set the objective. This is the Objective Type that is displayed in the GUI, and should be the most meaningful. Changing the userObjectiveType will result in an automatic change to the objectiveType.

For most protocols, the userObjectiveType and the objectiveType are the same, but protocols can define their own userObjectiveTypes when it makes sense to do so. For example, SIP defines the *channels* userObjectiveType that corresponds to an underlying objectiveType of simulatedUsers. See the individual protocols for a description of the userObjectiveTypes they accept and how they are translated to the objectiveType.

Option	Usage
userAgents	The objective is to sustain some number of SIP calls simultaneously. Specify the desired number of UserAgents in the objectiveValue option.
callsPerSecond	The objective is to establish a certain number of SIP calls per second. Specify the desired number of calls to establish per second in the objectiveValue option.
bhca	The objective is to establish a certain number of SIP calls per hour. Specify the desired number of calls to establish per hour in the objectiveValue option. Busy hour call attempts (BHCA) is a standard meaof the number of calls completed during a busy hour, the 60-minute period when the maximum traffic load occurs within a given 24-hour period.
registrationsinitiated	The objective is to establish a certain number of call registrations of SIP. Specify the desired number of registrations in the objectiveValue option.
redirectionsinitiated	The objective is to establish a certain number of call redirections of SIP. Specify the desired number of redirections in the objectiveValue option.
transactionAttemptRate	The objective is to issue some number of DNS query per second. The number of DNS query is mentioned in the userObjectiveValue option.
queriespersecond	The desired number of DNS query per second.



Option	Usage
connectionAttemptRate	The objective is achieved if IxLoad succeeds in making the specified number of attempts to connect to the HTTP server or DUT. Specify the desired number of attempted connections per second in the userObjectiveValue option.
streams	The objective is to monitor some number of multicast or unicast video or audio streams. Specify the desired number of streams in the userObjectiveValue option.

userObjectiveValue

The test objective value applied to the userObjectiveType. Default=100.

Note that some protocol-specific objectiveTypes apply scaling values to the value.

- bhca is mapped to transactionRate with a scaling factor of 3600.
- callsPerSec is mapped to transactionRate with a scaling factor of 1.
- userAgents is mapped to simulatedUsers with a scaling factor of 1.
- registrationsinitiated is mapped to transactionRate with a scaling factor of 1.
- redirectionsinitiated is mapped to transactionRate with a scaling factor of 1.

Traffic Map Setup Options

portMapPolicy

This option controls the sequence in which the client ports connect to the server ports. One of:

Option	Usage
\$.ixPortMap(kPortMapRoundRobin) or "portPairs"	(Default). Client agents connect to server agents on a one-to-one basis.
\$.ixPortMap(kPortMapFullMesh) or "portMesh"	Agents on every client port connect to every server port.
\$.ixPortMap(kPortMapIpPair) or "ipPair"	Each simulated user on the client side connects with only one server IP address. This choice is only valid for SIP agents.
\$.ixPortMap(kPortMapCustom) or "custom"	Each custom port map has a list of IPv4 subnets and IPv6 subnets. You can create a Custom traffic map. In a Custom traffic map, you select the client and server IP address ranges that will send traffic to each other. To create a Custom traffic map, the subnet's rangeType parameter must be set to IP Only (Ethernet).

For large numbers of ports, the Port Pair sequence scales performance better than the Port Mesh sequence.

The operation of Port Pairs can be described by three scenarios:

- If the number of client ports is equal to the number of server agents, client ports will establish connections to server ports on a one-to-one basis.
- If the number of client ports is less than the number of server ports, the client ports will establish connections to the server ports on a one-to-one basis until all client ports are paired with server ports. The remaining server ports will not be used.
- If the number of client ports is greater than the number of server ports, the client ports will establish connections to the server ports on a one-to-one basis until all server ports are paired with client ports. Then, the remaining client ports will return to the first server port and continue pairing themselves with server ports.

The `ixCustomPortMap` customizes the order and frequency, by which client IPs will access server IPs.

Each custom port map has a list of IPv4 submaps and IPv6 submaps. There will be a list for the appropriate IP type if any ranges of that type appear in the net for the symbolic destination. When a submap list is initialized, it will have a single submap that will be a full IP mesh, if that type is available. If only vLAN maps are allowed, then it will be a vLAN pairs map instead. If a submap is appended to the list, by default it will be a copy of the last submap in the list, unless values are passed in.

```
set destination [$HTTP_client_client_network getDestinationForActivity  
"newClientActivity1" "HTTP server_newServerActivity1"]
```

```
$destination config \
```

```
    -portMapPolicy                               "portMesh"
```

Configuring Traffic

```
name
```

The configuration that is set in the `protocolAndType` option for `activityList`.

```
$HTTP_client_client_network traffic.config \
```

```
    -name                                         "HTTP client"
```

```
$Client elementList.appendItem -object $HTTP_client_client_network
```

EXAMPLE

```
set HTTP_client_client_network [::IxLoad new ixNetTraffic]
$HTTP_client_client_network config \
    -enable 1 \
    -network $client_network
$HTTP_client_client_network traffic.config \
    -name "HTTP client"
$Activity_newAgent1 config \
    -enable 1 \
    -name "newClientActivity1" \
    -enableConstraint false \
    -userObjectiveValue 100 \
    -constraintValue 100 \
    -userObjectiveType "simulatedUsers" \
    -timeline $Timeline1
$Client elementList.appendItem -object $HTTP_client_client_network
#####
# Destination newServerActivity1 for newClientActivity1
#####
set destination [$HTTP_client_client_network getDestinationForActivity
    "newClientActivity1" "HTTP server_newServerActivity1"]
$destination config \
    -portMapPolicy "portMesh"
```

SEE ALSO

`ixSubscriberNetTraffic`

activityList

activityList-Generates traffic for one side of a particular protocol. For example, an HTTP client Activity generates HTTP client requests, simulating a web browser.

SYNOPSIS

```
set HTTP_client_client_network [::IxLoad new ixNetTraffic]
```

```
set Activity_newAgent1 [$HTTP_client_client_network  
activityList.appendoptions..]
```

DESCRIPTION

The `activityList` is used to generate traffic for one side of a particular protocol.

An Activity is added to the `ixNetTraffic` object using `appendItem` subcommand. Agents are added to the activity using the `agent.config` subcommand.

The `protocolAndType` is a required field. These define a particular type of agent; and the side of the communication. The agent definition should include options which are specific to a particular protocol, and defined in their respective appendix.

OPTIONS

`protocolAndType`

Protocol is the name of the protocol (for example, HTTP). Type denotes the side of the connection, that is, client or server.

EXAMPLE

```
set HTTP_client_client_network [::IxLoad new ixNetTraffic]
```

```
#-----
```

```
# Activity newAgent1 of NetTraffic HTTP client@client network
```

```
#-----
```

```
set Activity_newAgent1 [$HTTP_client_client_network activityList.appendItem \  
    -protocolAndType                "HTTP Client" ]
```

SEE ALSO

`ixNetTraffic`



ixTrafficFlow

ixTrafficFlow-Lists the test scenario.

SYNOPSIS

```
set TrafficFlow1 [::IxLoad new ixTrafficFlow]

$TrafficFlow1 config \ options...
```

DESCRIPTION

The `ixTrafficFlow` command is used to list the test scenario. Traffic Flow object is appended to the `ixTest` object.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

name

This represents the name of the `trafficflow` object.

EXAMPLE

```
set TrafficFlow1 [::IxLoad new ixTrafficFlow]

$TrafficFlow1 config \

-name      "TrafficFlow1"

$TrafficFlow1 columnList.clear
```

SEE ALSO

`ixDHCP`, `ixStatCatalogItem`

ixTrafficColumn

ixTrafficFlow-A container of ixNetTraffic and ixDut objects.

SYNOPSIS

```
set TrafficFlow1 [::IxLoad new ixTrafficFlow]

$TrafficFlow1 config \ options...
```

DESCRIPTION

The `ixTrafficColumn` command is used to define and configure client, server and DUT objects. The client, server and DUT objects are appended to the `ixTrafficFlow` object.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

name

This represents the name of the `trafficcolumn` object.

EXAMPLE

```
set Client [::IxLoad new ixTrafficColumn]

$Client config \
-name      "Client"

$Client elementList.clear

$TrafficFlow1 columnList.appendItem -object $Client

set DUT [::IxLoad new ixTrafficColumn]

$DUT config \
-name      "DUT"

$DUT elementList.clear

$TrafficFlow1 columnList.appendItem -object $DUT

set Server [::IxLoad new ixTrafficColumn]

$Server config \
```

```
-name "Server"  
  
$Server elementList.clear  
  
$TrafficFlow1 columnList.appendItem -object $Server
```

SEE ALSO

ixTrafficFlow



ixNetworkGroup

ixNetworkGroup-Configures the client and server network.

SYNOPSIS

```
set network [::IxLoad new ixNetworkGroup options]

$network config \ options...
```

DESCRIPTION

The `ixNetworkGroup` object is used to configure the client and server network. The client or server network is used by the `ixNetTraffic` object to map to the nettraffic.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

The options for this command are described in `ixDHCP` and `ixStatCatalogItem`.

aggregation

(Read-only) For load modules that support aggregating ports, this option returns the aggregation mode currently in use. For a list of aggregation modes, see `setCardAggregation` under `ixChassisChain`.

EXAMPLE

```
#-----
# Network client network of NetTraffic HTTP client@client network
#-----set
client_network [::IxLoad new ixNetworkGroup $chassisChain]

$client_network config \

    -comment                " " \
    -name                    "client network" \
    -emulatedRouterSubnetIPv6
"FFFF:FFFF:FFFF:FFFF:FFFF:FFFF::0" \
    -linkLayerOptions        0 \
    -ipSourcePortFrom        1024 \
```

```
-emulatedRouterGatewayIPv6      ":::" \  
-cardType                        "ALM1000T8-1GB" \  
-emulatedRouterGateway          "0.0.0.0" \  
-ipSourcePortTo                 65535 \  
-emulatedRouterSubnet           "255.255.255.0" \  
-macMappingMode                  0 \  
-dnsParameters                   $my_ixDns \  
-tcpParameters                   $my_ixTcpParameters \  
-impairment                       $my_ixImpairment \  
-arpSettings                      $my_ixArpSettings
```

SEE ALSO

`ixNetworkRange`

ixDut

ixDut-Defines a DUT.

SYNOPSIS

```
set DUT1 [::IxLoad new ixDut]
$DUT1 subcommand options...
```

DESCRIPTION

The `ixDut` command is used to define a DUT used in the test. The DUTs are used to resolve symbolic references in traffic destinations in the various protocol agents. It also controls several DUT specific features. DUTs are added to the `ixTest` object using `appendItem`. For example,

```
set DUT1 [::IxLoad new ixDut]

$DUT1 config \
    -comment      "" \
    -type          "VirtualDut" \
    -name          "DUT1" \
    -dutConfig    $my_ixDutConfigVirtual
```

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`comment`

A comment associated with this DUT. (Default = "").

`name`

The name associated with the DUT. (Default = "DUT1").

`type`

The type of DUT in use. One of:

Option	Usage
ExternalServer	The DUT is a protocol server.

Option	Usage
ServerLoadBalancer	The DUT is a server load balancer.(Default)
Firewall	The DUT is a firewall.
VirtualDut	The DUT is a virtual DUT

dutConfig

The object instance of the DUT type.

EXAMPLE

```
set DUT1 [::IxLoad new ixDut]
$DUT1 config \
    -comment                "" \
    -type                    "VirtualDut" \
    -name                    "DUT1" \
    -dutConfig                $my_ixDutConfigVirtual
$DUT elementList.appendItem -object $DUT1
$New_Traffic_Flow columnList.appendItem -object $DUT
```

SEE ALSO

ixTest, ixStatCatalogItem

ixDutConfigVirtual

ixDutConfigVirtual-Configures a virtual DUT.

SYNOPSIS

```
set my_ixDutConfigVirtual [::IxLoad new ixDutConfigVirtual]
$my_ixDutConfigVirtual subcommand options...
```

DESCRIPTION

The `ixDutConfigVirtual` command is used to:

- Define a range of IP addresses for the DUT, instead of the single address that the other DUT Types allow.
- Specify the TCP/UDP ports that the Virtual DUT listens on, on a per-protobasis.

Virtual DUTs are added to the `ixDut` object as an option. For example,

```
set my_ixDutConfigVirtual [::IxLoad new ixDutConfigVirtual]
$my_ixDutConfigVirtual config
$my_ixDutConfigVirtual networkRangeList.clear
set DUT1 [::IxLoad new ixDut]
$DUT1 config \
    -comment      "" \
    -type          "VirtualDut" \
    -name          "DUT1" \
    -dutConfig    $my_ixDutConfigVirtual
```

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`ixDutNetworkRange` and `ixDutProtocolPortRange` are appended to the `ixDutConfigVirtual` object.

EXAMPLE

```
set my_ixDutConfigVirtual [::IxLoad new ixDutConfigVirtual]
$my_ixDutConfigVirtual config
```

```
$my_ixDutConfigVirtual networkRangeList.clear

set Network_Range_1_in_DUT1__1_1_1_1_100_ [::IxLoad new ixDutNetworkRange]

$Network_Range_1_in_DUT1__1_1_1_1_100_ config \

    -vlanUniqueCount          4094 \

    -firstIp                   "1.1.1.1" \

    -enable                    true \

    -name                       "Network Range 1 in DUT1
(1.1.1.1+100)" \

    -vlanEnable                true \

    -vlanId                    1 \

    -innerVlanEnable           false \

    -ipIncrStep                "0.0.0.1" \

    -networkMask               "255.255.0.0" \

    -ipType                    1 \

    -vlanIncrStep              1 \

    -vlanCount                 1 \

    -ipCount                   100

$my_ixDutConfigVirtual networkRangeList.appendItem -object
$Network_Range_1_in_DUT1__1_1_1_1_100_

$my_ixDutConfigVirtual protocolPortRangeList.clear

set my_ixDutProtocolPortRange [::IxLoad new ixDutProtocolPortRange]

$my_ixDutProtocolPortRange config \

    -portRanges                "1001,1002,1003-1006" \

    -protocol                   "HTTP"

$my_ixDutConfigVirtual protocolPortRangeList.appendItem -object
$my_ixDutProtocolPortRange

set DUT1 [::IxLoad new ixDut]

$DUT1 config \

    -comment                    "" \

    -type                       "VirtualDut" \
```



```
-name "DUT1" \  
-dutConfig $my_ixDutConfigVirtual  
$DUT elementList.appendItem -object $DUT1
```

SEE ALSO

ixDut

ixDutNetworkRange

ixDutNetworkRange-Defines the one range of IP addresses (a subnet) that the Virtual DUT will have. Subnets defined here should match the subnets configon the actual DUT.

SYNOPSIS

```
set Network_Range_1_in_DUT1__1_1_1_1_100_ [::IxLoad new ixDutNet
$Network_Range_1_in_DUT1__1_1_1_1_100_ subcommand options...
```

DESCRIPTION

Defines the one range of IP addresses (a subnet) that the Virtual DUT will have. Subnets defined here should match the subnets configured on the actual DUT.

The Range of IP addresses are added to the ixRepository object. For example,

```
set Network_Range_1_in_DUT1__1_1_1_1_100_ [::IxLoad new ixDutNetworkRange]
$Network_Range_1_in_DUT1__1_1_1_1_100_ config
```

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`vlanUniqueCount`

Specifies the number of VLAN IDs to create.

`firstIp`

This is the First IP address on the subnet, and subnet mask. Enter the subnet in /<bits> format, following the IP address.

For example, to specify an address of 198.162.0.1 with a subnet of 255.255.0.0, enter: 198.162.0.1/16 (Default = "1.1.1.1").

`enable`

If true this makes a subnet active. Only traffic from active subnets can be meshed meshed; inactive subnets are not used. Default = true.

`name`



Specifies the name of the Network Range.

`vlanEnable`

Enable this if the actual DUT uses VLANs. `Default = false`.

`vlanId`

Value of first 802.1Q VLAN tag.

`ipIncrStep`

Amount of increase in the IP address used to create additional IP addresses on the subnet, and octet that will be incremented. `Default = "0.0.0.1"`.

`networkMask`

This specifies the subnet mask. `Default = "255.255.0.0"`.

`ipType`

Specifies the type of addressing for the subnet: IPv4 or IPv6. IxLoad supports all forms of IPv6 addressing except `::dotted-quad` notation (for example, `::1.2.3.4`).

You must select the same type of addressing used on the corresponding subnet on the actual DUT. `Default = 1`.

`vlanIncrStep`

Amount of increase in the VLAN ID. IxLoad applies this value to the ID to create the complete list of VLAN IDs that will be meshed. `Default = 1`.

`vlanCount`

Number of VLAN IDs to create. `Default = 1`.

`ipCount`

Number of IP addresses on this subnet.

EXAMPLE

```
set Network_Range_1_in_DUT1__1_1_1_1_100_ [::IxLoad new ixDutNetworkRange]
$Network_Range_1_in_DUT1__1_1_1_1_100_ config \
    -vlanUniqueCount 4094 \
```

```
-firstIp          "1.1.1.1" \  
-enable          true \  
-name            "Network Range 1 in DUT1  
(1.1.1.1+100)" \  
-vlanEnable      true \  
-vlanId          1 \  
-innerVlanEnable false \  
-ipIncrStep      "0.0.0.1" \  
-networkMask     "255.255.0.0" \  
-ipType          1 \  
-vlanIncrStep    1 \  
-vlanCount       1 \  
-ipCount         100
```

SEE ALSO

`ixDut`, `ixDutConfigVirtual`

ixDutProtocolPortRange

ixDutProtocolPortRange-defines a protocol that the Virtual DUT listens for, and the ports that it listens of for that protocol.

SYNOPSIS

```
set my_ixDutProtocolPortRange [::IxLoad new ixDutProtocolPortRange]
$my_ixDutProtocolPortRange subcommand options...
```

DESCRIPTION

Defines a protocol that the Virtual DUT listens for, and the ports that it listens of for that protocol.

The ProtocolPortRange object is appended to the ixDutConfigVirtual object. For example,

```
$my_ixDutConfigVirtual protocolPortRangeList.appendItem -object
$my_ixDutProtocolPortRange
```

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

portRanges

Specifies the port numbers that the Virtual DUT listens on for the protocols in the `protocol` field.

protocol

Defines a protocol to listen for. A virtual dut supports the following protocols:

FTP	•HTTP	•IMAP	•IPTV/Video
LDAP	•POP3	•RADIUS	•RTSP
SMTP	•SSH	•DNS	•All

EXAMPLE

```
set my_ixDutProtocolPortRange [::IxLoad new ixDutProtocolPortRange]
$my_ixDutProtocolPortRange config \
    -portRanges                " " \
    -protocol                   "All"
```

SEE ALSO

`ixDut, ixDutConfigVirtual`



ixDutConfigVip

ixDutConfigVip-DUT Configuration class for firewall and external server.

SYNOPSIS

```
set my_ixDutConfigVip [::IxLoad new ixDutConfigVip]
$my_ixDutConfigVip subcommand options...
```

DESCRIPTION

This class is associated with `ixDut` for DUT types - Firewall and External Server.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`ipAddress`

Specifies the IP address used to access the DUT.

EXAMPLE

```
set my_ixDutConfigVip [::IxLoad new ixDutConfigVip]
$my_ixDutConfigVip config \
    -ipAddress "1.1.1.1"
```

SEE ALSO

`ixDut`

ixDutConfigSLB

ixDutConfigSLB-DUT Configuration class for server load balancer.

SYNOPSIS

```
set my_ixDutConfigSLB [::IxLoad new ixDutConfigSLB]
$my_ixDutConfigSLB subcommand options...
```

DESCRIPTION

This class is associated with `ixDut` for DUT types - Server Load Balancer.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enableDirectServerReturn`

Enables the Direct Server Run. In a topology using Direct Server Return, the responses are sent directly from the servers to the clients; they do not go through the SLB. `Default = false`.

`ipAddress`

Specifies the IP address used to access the DUT.

SUB-OBJECTS

`serverNetwork`

If type is "Server Load Balancer (SLB)" and the SLB is balancing Ixia emulated servers, set this option to the server network that is being balanced. This must be an object of type `ixStatCatalogItem`. (`Default = {}`).

Note: Ixia Server Network is not supported in SLB options.

EXAMPLE

```
set my_ixDutConfigSLB [::IxLoad new ixDutConfigSLB]
$my_ixDutConfigSLB config \
    -enableDirectServerReturn          false \
    -ipAddress                          "198.18.0.101"
```

SEE ALSO

`ixDut`

ixView

ixView-Configures capture options in test repositories.

SYNOPSIS

```
set my_ixViewOptions [::IxLoad new ixViewOptions]
```

```
$my_ixViewOptions config options...
```

DESCRIPTION

The `ixViewoptions` commands is configured using the `ixconfig` subcommand. It is added as an object instance to the `captureViewOptions` in `ixTest`.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`runMode`

Specifies when capture starts, and how long it continues for. Values are:

- | | |
|---|-----------------------------|
| 1 | Automatic |
| 2 | Manual |
| 3 | Start Capture after and for |

`captureRunDuration`

If runmode type is 3, the capture starts after the specified number of seconds, minutes, or hours mentioned in this field; after the test begins.

`captureRunAfter`

If runmode type is 3, the capture continues for the specified number of seconds, minutes, or hours mentioned in this field.

`collectScheme`

Specifies whether or not packets are displayed as they are captured during a test. Values are:

- | | |
|---|--------|
| 0 | Stream |
| 1 | Upload |



allocatedBufferMemoryPercentage

Percentage of the available memory on the Ixia port allocated for capturing pack

The memory available for capturing packets is the total amount of memory availon the port, less the amount required for the IxLoad test configuration. Of this remaining amount, you can reserve up to 70% for capturing packets.

EXAMPLE

```
set my_ixViewOptions [::IxLoad new ixViewOptions]

$my_ixViewOptions config \

    -runMode 1 \
    -captureRunDuration 0 \
    -captureRunAfter 0 \
    -collectScheme 0 \
    -allocatedBufferMemoryPercentage 30

set Test1 [::IxLoad new ixTest]

$Test1 config \

    -comment "" \
    -csvInterval 4 \
    -name "Test1" \
    -statsRequired 1 \
    -enableResetPorts 0 \
    -enableForceOwnership false \
    -enableReleaseConfigAfterRun 0 \
    -captureViewOptions $my_ixViewOptions
```

SEE ALSO

ixTest

ixClientNetwork

ixClientNetwork-Defines a network for client agents.

SYNOPSIS

```
set clientNetwork [::IxLoad new ixClientNetwork $chassisChain options]
$clientNetwork subsubcommandcommand options...
```

DESCRIPTION

The `ixClientNetwork` command is used to construct a client network, which is used as part of an `ixClientTrafficNetworkMapping` object. A chassis chain object, as created in the `ixChassisChain` command, must be used in the construction of this object.

A list of network ranges, as defined in the `ixRepository` object is associated with the client network. Network ranges are added to the client network through the use of the `networkRangeList.appendItem` command.

A list of ports is also associated with the network through the `portList` option.

If an emulated router is to be used, a list of IP ranges for the router is also associated with the network through the `emulatedRouterIpAddressPool` option. The pool is defined in the `ixEmulatedRouterIpAddressRange` object. These are added to the object through the use of the `em` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition, the following commands are available. Unless otherwise described, no value is returned and an exception is raised for any error found.

checkConfig

Checks the configuration of the client network object.

reset

Disassociates the network from all of the Ixia ports currently in the `portList` option. Ownership of the ports is cleared.

OPTIONS

`chassisChain`

This must be a chassis chain object, as created in the `ixChassisChain` command. It represents the set of chassis used in the test and defines the chassis IDs used in the `portList` component. This option should not be changed after `portList` is set. (Default = None).



comment

A commentary string for the object. (Default = "").

emulatedRouterGateway

If `macMappingMode` is set to `kMacMappingModePort`, then an emulated router is inserted between the clients and the external port. This is the gateway to be used for that router. (Default = 0.0.0.0).

emulatedRouterGatewayIPv6

If `macMappingMode` is set to `kMacMappingModePort` and `ipType` in `ixEmulat` is set to "IPv6" for any addresses, then an IPv6 address is also required for the emulated router inserted between the clients and the external port. This is the IPv6-format address of the gateway to be used for that router. `IxLoad` supports all forms of IPv6 addressing except `::dotted-quad` notation (for example, `:::1.2.3.4`). (Default = `:::C212:0001`).

emulatedRouterSubnetIPv6

Subnet mask applied to `emulatedRouterGatewayIPv6` address. (Default = `"FFFF:FFFF:FFFF:FFFF:FFFF:FFFF::0"`)

emulatedRouterIpAddressPool

If `macMappingMode` is set to `kMacMappingModePort`, then an emulated router is inserted between the clients and the external port. This option is a list of `ixEmulatedRouterIpAddressRange` objects which define the routers' source addresses that will be used. One IP address is taken from the list and used for each Ixia port. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list. (Default = `{}`)

emulatedRouterSubnet

If `macMappingMode` is set to `kMacMappingModePort`, then an emulated router is inserted between the clients and the external port. This is the network mask to be used for that router. (Default = 255.255.255.0).

ipSourcePortFrom

Defines the beginning of the range of ephemeral port numbers used to establish connections to the server. The end of the range is specified by `ipSourcePortTo`.

The first port in the range that `IxLoad` uses for traffic is 1 greater than the value you specify for `ipSourcePortFrom`. For example, if you specify 1,024, traffic originates from port 1025; no traffic originates from port 1,024. The minimum value for `ipSourcePortFrom` is 1024. (Default = 1,024).

ipSourcePortTo

Defines the end of the range of ephemeral port numbers used to establish connecto the server. The beginning of the range is specified by `ipSourcePortFrom`. (Default = 65,535).

`linkLayerOptions`

The link layer options to be associated with the ports associated with this client network. Only Ethernet options are currently supported. (Default = `kLink`)

`macMappingMode`

The mapping between IP addresses and MAC addresses. One of:

Option	Usage
<code>ixClientNetwork(kMacMappingModeIp)</code>	(Default) One MAC address is associated with each IP address.
<code>ixClientNetwork(kMacMappingModePort)</code>	One MAC address is used for all IP addresses on the port.

`name`

The name associated with this object. (Default = "newNetwork").

`networkRangeList`

A list of `ixRepository` objects that define the networks from which addresses will be associated with the clients. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list. (Default = {}).

`portList`

A list of ports associated with the client network. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to maniputhis list. Ports are added directly into this object; see the following example:

```
$clientNetwork portList.appendItem \
    -chassisId 1 \
    -cardId 2 \
    -portId 2
```

SUB-OBJECTS

`arpSettings`

This is an object of type `ixArpSettings`, which specifies the manner in which ARP is handled on this network. (Default = <see `ixArpSettings`>). The options of this object should be set directly via:



```
$clientNetwork arpSettings.config options...
```

dnsParameters

This is an object of type `ixDns`, which specifies the manner in which specifies the DNS operation associated with clients on this network. (Default = <see `ixDns`>). The options of this object should be set directly via:

```
$clientNetwork dnsParameters.config options...
```

```
$clientNetwork dnsParameters.serverList.appendItem options...
```

tcpParameters

This is an object of type `ixTcpParameters` that specifies the manner in which TCP traffic is handled on this network. (Default = <see `ixTcpParameters`>). The options of this object should be set directly via:

```
$clientNetwork tcpParameters.config options...
```

EXAMPLE

```
set clnt_network [::IxLoad new ixClientNetwork $chassisChain]
```

```
$clnt_network config -name "clnt_network" \
```

```
    -cardType $::ixCard(kCard1000Txs4)
    -ipSourcePortFrom 1024 \
    -ipSourcePortTo 65536 \
```

```
$clnt_network networkRangeList.appendItem \
```

```
    -name          "clnt_range" \
    -enable        1 \
    -firstIp       "198.18.2.1" \
    -ipCount       100 \
    -networkMask   "255.255.0.0" \
    -gateway       "0.0.0.0" \
    -firstMac      "00:C6:12:02:01:00" \
    -vlanEnable    0 \
    -vlanId        1 \
    -mssEnable     0 \
    -mss           100
```

```
$clnt_network portList.appendItem \  
    -chassisId 1 \  
    -cardId    2 \  
    -portId    1
```

SEE ALSO

`ixClientTrafficNetworkMapping`, `ixChassisChain`, `ixRepository`,
`ixEmulatedRouterIpAddressRange`, `ixDns`, `ixArpSettings`, `ixTcpParameters`

ixClientTraffic

ixClientTraffic-Builds a list of client agents to generate client traffic.

SYNOPSIS

```
set clientTraffic [::IxLoad new ixClientTraffic options]
```

```
$clientTraffic subcommand options...
```

DESCRIPTION

The `ixClientTraffic` command is used to construct the model for client traffic to be applied during a test. It is used in the `ixClientTrafficNetworkMapping` comto co-ordinate networks with client agent traffic. Its primary option is the `agentList` list of agents which will generate client traffic.

Agents are added to the `agentList` using the `appendItem` subcommand and may be otherwise manipulated using the commands supported by the `ixConfigSequenceContainer` command. All agents are added in the same manner:

```
set clientTraffic [::IxLoad new ixClientTraffic \  
    -name "Traffic" ]  
  
$clientTraffic agentList.appendItem \  
    -name      "my_protocol_traffic" \  
    -protocol  "<PROTOCOL>" \  
    -type      "Client" \  
    <other per-protocol options>
```

The `name`, `protocol`, and `type` are required fields. These define a particular type of agent; the `protocol` field should be drawn from the table above. In addition to the required fields, the agent definition should include options which are specific to a particular protocol, and defined in their respective appendix.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition, the following commands are available. Unless otherwise described, no valare returned and an exception is raised for any error found.

checkConfig

Checks the configuration of the client traffic object.

OPTIONS

name

The name associated with the `agentList` object. (Default = "newActivityModel").

agentList

A list of agent objects which define the agents that will be used to generate client traffic. Refer to the various appendixes listed above to determine the options that the agents offer. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list. (Default = {}).

EXAMPLE

```
#-----  
# Construct Client Traffic  
#-----  
set clnt_traffic [::IxLoad new ixClientTraffic \  
    -name "client_traffic"]  
  
#-----  
# Create a HTTP client agent  
#-----  
$clnt_traffic agentList.appendItem \  
    -name                "my_http_client" \  
    -protocol            "HTTP" \  
    -type                "Client" \  
    -maxSessions         3 \  
    -httpVersion         $::HTTP_Client(kHttpVersion10) \  
    -keepAlive           0 \  
    -maxPersistentRequests 3 \  
    -followHttpRedirects 0 \  
    -enableCookieSupport 0 \  
    -enableHttpProxy     0
```



```
-enableHttpsProxy      0 \  
-browserEmulation      $::HTTP_Client(kBrowserTypeIE5) \  
-enableSsl             0
```

```
#-----  
# Add actions to this client agent  
#-----  
foreach {pageObject destination} {  
    "/4k.htm" "svr_traffic_my_http_server"  
    "/8k.htm" "svr_traffic_my_http_server"  
} {  
    $clnt_traffic agentList(0).actionList.appendItem \  
        -command      "GET" \  
        -destination  $destination \  
        -pageObject   $pageObject  
}
```

SEE ALSO

`ixClientTrafficNetworkMapping, protocol support appendices`

ixClientTrafficNetworkMapping

ixClientTrafficNetworkMapping-Ties a client network to traffic model.

SYNOPSIS

```
set clientMapping [::IxLoad new ixClientTrafficNetworkMapping options]
$clientMapping subcommand options...
```

DESCRIPTION

The `ixClientTrafficNetworkMapping` command is used to map a set of agents that generate client traffic (in an `ixCustomPortMap` object) to the set of networks, which will carry the traffic (in an `ixDHCP` object).

A number of additional options control the manner in which the client traffic is applied to the networks.

The `objectiveType` and `objectValue` options allow the application of traffic to achieve a particular objective-for example, connections per second.

The `setObjectiveTypeForActivity` and `setObjectiveValueForActivity` options allow you set objectives and values for individual activities within a traffic-network mapping.

The `rampUpType`, `rampUpValue`, `rampDownTime`, `standbyTime`, `offlineTime`, `sustainTime`, and `totalTime` options determine the timeline for application of traffic.

`portMapPolicy` controls the manner in which client traffic is sent to servers.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

If `true`, this mapping is included in the `IxLoad` test. (Default = `true`).

`getUserObjectiveTypeForActivity`

Objective type for user objective activity within a traffic-network mapping. You must specify the activity name. See the following example:

```
set objType [$clnt_t_n_mapping
getUserObjectiveTypeForActivity("my_sip_client")]
```

`getUserObjectiveValueForActivity`



Objective type for user objective value within a traffic-network mapping. You must specify the activity name. See the following example:

```
set objValue [$clnt_t_n_mapping getUserObjectiveValueForActiv
```

```
iterations
```

The number of times that the traffic-network pair perform their functions (estabTCP connections, retrieving FTP files, and so forth) in the test. (Default = 1).

```
name
```

The name associated with this object. This is read-only and cannot be set from the API. (Default = "NetworkTrafficMapping").

```
objectiveType
```

The objective to be achieved for this traffic to network mapping. One of:

Option	Usage
"N/A"	(Default).
\$.:ixObjective(kObjectiveTypeSimulatedUsers) or "simulatedUsers"	The objective is to simulate some number of users. If you select this objective, remember that a 'user' does not necessarily mean one human user. For example, a Web browser used by one permay open several connections to a Web site simultaneously; each connection counts as one 'user,' because each connection was initiated by the same source simultaneously. Specify the desired number of users in the objectiveValue option.
\$.:ixObjective(kObjectiveTypeConcurrentConnections) or "concurrentConnections"	The objective is to sustain some number of connections simultaSpecify the desired number of connections in the objec option.
\$.:ixObjective(kObjectiveTypeConcurrentSessions) or "concurrentSessions"	The objective is to sustain some number of sessions simultaSpecify the desired number of connections in the objec option.
\$.:ixObjective(kObjectiveTypeConnectionRate) or "connectionRate"	The objective is to create connections at a certain rate. For example, if the traffic in this mapping is HTTP client traffic, this objective will attempt to generate the specified number of HTTP connections per second. Specify the desired number of connecper second in the objectiveValue option.
\$.:ixObjective(kObjectiveTypeThroughputMbps) or "throughputMbps"	As of IxLoad 5.00, this option has been deprecated. Use ThroughputMbps instead.

Option	Usage
<code>\$.ixObjective(kObjectiveTypeThroughputMbps)</code> or <code>"throughputMbps"</code>	The objective is to achieve a certain level of throughput, mean megabits per second (Mbps). Specify the amount of throughput in the <code>objectiveValue</code> option.
<code>\$.ixObjective(kObjectiveTypeThroughputKBps)</code> or <code>"throughputKBps"</code>	As of IxLoad 5.00, this option has been deprecated. Use <code>ThroughputKbps</code> instead.
<code>\$.ixObjective(kObjectiveTypeThroughputKbps)</code> or <code>"throughputKbps"</code>	The objective is to achieve a certain level of throughput, mean kilobits per second (Kbps). Specify the amount of throughput in the <code>objectiveValue</code> option.
<code>\$.ixObjective(kObjectiveTypeThroughputGbps)</code> or <code>"throughputGbps"</code>	The objective is to achieve a certain level of throughput, mean gigabits per second (Gbps). Specify the amount of throughput in the <code>objectiveValue</code> option.
<code>\$.ixObjective(kObjectiveTypeTransactionRate)</code> or <code>"transactionRate"</code>	<p>The objective is to complete transactions at a certain rate. For example, if the traffic in this mapping is HTTP client traffic, this objective will attempt to complete the specified number of transactions per second. The definition of what constitutes one complete HTTP transaction depends on whether you select HTTP 1.0 or 1.1:</p> <p>HTTP 1.0: open socket - issue GET - GET response - close socket. HTTP 1.1: Open socket (if closed) - send request - Get response.</p> <p>Specify the desired number of transactions per second in the <code>objectiveValue</code> option.</p>

`objectiveValue`

Value for the choice made in the `objectiveType` option.

`objectiveConstraints`

Currently, constraints can be set on activities that run rate-based objectives, like `connectionRate`, `transactionRate`, `throughput` objectives.

The following API can be used to set the constraint value. The constraint needs to be enabled on the activity.

```
$clnt_t_n_mapping setconstraints "my_http_client" 200 true
```

This sets the constraint value to 200 and `true` enables the constraint. If the activity is running a rate based activity, then the number of simulated users will be limited to 200.

```
$clnt_t_n_mapping setconstraints "my_http_client" 200 false
```



This sets the constraint value to 200 and `false` does not enable the constraint. The number of simulated users will not be limited here.

`offlineTime`

The amount of time agents are idle between iterations. (Default = 0).

`portMapPolicy`

This option controls the sequence in which the client ports connect to the server ports. One of:

Option	Usage
<code>ixPortMap(kPortMapRoundRobin)</code> or "portPairs"	(Default). Client agents connect to server agents on a one-to-one basis.
<code>ixPortMap(kPortMapFullMesh)</code> or "portMesh"	Agents on every client port connect to every server port.
<code>ixPortMap(kPortMapIpPair)</code> or "ipPair"	Each simulated user on the client side comwith only one server IP address. This choice is only valid for SIP agents.
<code>ixPortMap(kPortMapCustom)</code> or "custom"	Each custom port map has a list of IPv4 subband IPv6 submaps. You can create a Custom traffic map. In a Custom traffic map, you select the client and server IP address ranges that will send traffic to each other. To create a Custom traffic map, the subnet's <code>rangeType</code> parameter must be set to <code>IP Only (Ethernet)</code> .

For large numbers of ports, the Port Pair sequence scales performance better than the Port Mesh sequence.

The operation of Port Pairs can be described by three scenarios:

- If the number of client ports is equal to the number of server agents, client ports will establish connections to server ports on a one-to-one basis.
- If the number of client ports is less than the number of server ports, the client ports will establish connections to the server ports on a one-to-one basis until all client ports are paired with server ports. The remaining server ports will not be used.
- If the number of client ports is greater than the number of server ports, the client ports will establish connections to the server ports on a one-to-one basis until all server ports are paired with client ports. Then, the remaining client ports will return to the first server port and continue pairing themselves with server ports.

The `ixCustomPortMap` customizes the order and frequency, by which client IPs will access server IPs.

Each custom port map has a list of IPv4 submaps and IPv6 submaps. There will be a list for the appropriate IP type if any ranges of that type appear in the netfor the symbolic destination. When a submap list is initialized, it will have a single submap that will be a full IP mesh, if that type is available. If only vLAN maps are allowed, then it will be a vLAN pairs map instead. If a submap is appended to the list, by default it will be a copy of the last submap in the list, unless values are passed in.

rampDownTime

The amount of time used for closing any TCP connections that are still open after all transactions are complete. When the ramp downtime expires, IxLoad terminates users.

If IxLoad terminates any client users that are still running after the ramp downtime expires, statistics for servers and clients that should match may not. This is an indication that the ramp downtime may be too short. (Default = 20).

rampUpTime

(Read-only). The amount of time that the test will spend bringing users online and initiating their first TCP connections. IxLoad calculates this time based on the number of users and the rampUpType option.

rampUpType

The method used to apply the rampUpValue. One of:

Option	Usage
\$.ixTimeline(kRampUpType UsersPerSecond)	(Default). IxLoad applies the rampUpValue to bring up the specified number of users per second. For example, if you select Users/Second and you specify 10 for the rampUpValue, IxLoad brings up 10 new users every second until all the users are up and running.
\$.ixTimeline(kRampUpType MaxPendingUsers)	IxLoad applies the rampUpValue to maintain a pool of users waiting to create connections. Regardless of how quickly the servers complete connections, IxLoad will always be ready with one or more new clients waiting to connect. As each user successfully creates a connection, IxLoad adds a new user to the pending pool until all the users are up and running. For example, if you select Max. Pending Users and you specify 10 for the rampUpValue, IxLoad maintains 10 users waiting to establish connections until all the users are up and running.

rampUpValue

A value dependent on the setting of rampUpType. One of:

Option	Usage
\$.ixTimeline(kRampUpType UsersPerSecond)	The specified number of users per second to bring up.
\$.ixTimeline(kRampUpType MaxPendingUsers)	The size of the pool of pending users awaiting con

rampUpInterval



This field accepts integer values. The value for this option will be considered only when `rampUpType` is `usersPerSecond`. You can edit the value to increment or decrement the number of users to be started at every `rampUpInterval`. (Default = 1).

```
#-----  
# Create a client and server mapping and bind into the  
# network and traffic that they will be employing  
#-----  
set clnt_t_n_mapping [::IxLoad new ixClientTrafficNetworkMapping \  
    -network          $clnt_network \  
    -traffic          $clnt_traffic \  
    -objectiveType    $::ixObjective(kObjectiveTypeSimulatedUsers) \  
    -objectiveValue   20 \  
    -rampUpValue      5 \  
    -rampUpInterval   10 \  
    -sustainTime      20 \  
    -rampDownTime     20
```

In this example, 5 simulated users will be started every 10 seconds until the contotal number of simulated users are started.

```
setObjectiveTypeForActivity
```

Objective type for a single activity within a traffic-network mapping. You must specify the activity and the objective type. The objectives available are the same as for `objectiveType`. See the following example:

```
setObjectiveTypeForActivity "my_http_client" \  
$::ixObjective(kObjectiveTypeConnectionRate)
```

```
setObjectiveValueForActivity
```

Objective value for a single activity within a traffic-network mapping. You need to specify the activity and the value. See the following example:

```
setObjectiveTypeForActivity setObjectiveValueForActivity \  
"my_http_client" 200
```

```
setPortMapForActivity
```

Port mapping for a single activity within a traffic-network mapping. You need to specify the activity and the `portMapPolicy`. See the following example:

```
setObjectiveTypeForActivitysetPortMapForActivity \ "my_http_client"  
$:ixPortMap(kPortMapFullMesh)
```

```
setUserObjectiveTypeForActivity
```

Objective type for user objective activity within a traffic-network mapping. You need to specify the activity name and the `userObjectiveType`. See the following example:

```
$clnt_t_n_mapping setUserObjectiveTypeForActivity("my_sip_client", "bhca")
```

```
setUserObjectiveValueForActivity
```

Objective type for user objective value within a traffic-network mapping. You need to specify the activity name and the `userObjectiveType`. See the following example:

```
$clnt_t_n_mapping setUserObjectiveValueForActivity("my_sip_client", 3600)
```

```
standbyTime
```

The amount of time, expressed in seconds, that elapses between the time the test is started and the time that the traffic-network pair become active. If you have multiple traffic-network pairs in your test, you can use this parameter to stagger their start times. A value of 0 causes the test to begin immediately. The valid range is from 0 to 1,000 hours (3,600,000). (Default = 0).

```
sustainTime
```

The amount of time, in seconds, when all users are up and performing the central test objectives, such as establishing and closing connections (TCP), retrieving or serving pages (HTTP), or sending or receiving files (FTP). The valid range is from 0 to 1,000 hours (3,600,000). (Default = 20).

```
totalTime
```

The total time required to run the test, including Ramp Up, Ramp Down, Sustain, and Offline times for all iterations. (Default = 60).

```
userObjectiveType
```

UserObjectivetypes are basically alternate representations of the basic `objectiveType` - `simulatedUsers`, `transactionRate`, `concurrentSessions`, `concurconnectionsPerSecond`, `throughputmbps`, `throughputkpbs`. They can have a scaling factor associated with them. For example, `bhca` has a scaling factor of 3,600. This means that, 3,600 busy hour call attempts (BHCA) `userOb`represents 1 `transactionRate` `objectiveValue`.

`userAgents` represents `simulatedUsers` with scaling factor of 1.

`callsPerSec` represents `transactionRate` with scaling factor of 1.



Registrationsinitiated represents transactionRate with scaling factor of 1.

Redirectionsinitiated also represents transactionRate with scaling factor of 1.

```
set clnt_t_n_mapping [::IxLoad new ixClientTrafficNetworkMapping \  
  
-network          $clnt_network \  
-traffic          $clnt_traffic \  
-standbyTime     30 \  
-userObjectiveType "bhca" \  
-userObjectiveValue 3600 \  
-rampUpValue     1 \  
-sustainTime     40 \  
-rampDownTime    20
```

Option	Usage
userAgents"	The objective is to sustain some number of SIP calls simultaneously. Specify the desired number of UserAgents inthe objectiveValue option.
callsPerSecond	The objective is to establish a certain number of SIP calls per second. Specify the desired number of calls to establish per second in the objectiveValue option.
bhca	The objective is to establish a certain number of SIP calls per hour. Specify the desired number of calls to establish per hour in the objectiveValue option. Busy hour call attempts (BHCA) is a standard meaof the number of calls completed during a busy hour, the 60-minute period when the maximum traffic load occurs within a given 24-hour period.
registrationsinitiated	The objective is to establish a certain number of call registrations of SIP. Specify the desired number of registrations in the objectiveValue option.
redirectionsinitiated	The objective is to establish a certain number of call redirections of SIP. Specify the desired number of redirections in the objectiveValue option.

Option	Usage
transactionAttemptRate	The objective is to issue some number of DNS query per second. The number of DNS query is mentioned in the userObjectiveValue option.
connectionAttemptRate	The objective is achieved if IxLoad succeeds in makthe specified number of attempts to connect to the HTTP server or DUT. Specify the desired number of attempted connections per second in the userObjectiveValue option.

userObjectiveValue

A value related to the choice made in the userObjectiveType option. One of:

Option	Usage
Calls	The desired number of calls.
Callspersecond	The desired number of calls to establish per second.
Bhca	The desired number of calls to establish per hour.
Useragents	The desired number of user agents to be simulated.
Registrationinitiated	The desired number of registrations to be initiated during the test.
Redirectionsinitiated	The desired number of call redirections initiated durthe test.
Queriespersecond	The desired number of DNS query per second.

SUB-OBJECTS

network

An object instance of type `ixDHCP`, which provides the networks from which the traffic will be generated. (Default = `{}`).

traffic

An object of type `ixCustomPortMap`, which provides the model of traffic to be generated. (Default = `{}`).

EXAMPLE

```
#-----
# Create the client traffic to network mapping
```



```
#-----  
set clnt_mapping [::IxLoad new ixClientTrafficNetworkMapping \  
    -network          $clnt_network \  
    -traffic          $clnt_traffic \  
    -objectiveType $::ixObjective(kObjectiveTypeSimulatedUsers)\ \  
    -objectiveValue  20 \  
    -rampUpValue     5 \  
    -sustainTime     20 \  
    -rampDownTime    20  
]
```

SEE ALSO

`ixTest`, `ixDHCP`, `ixCustomPortMap`

ixNetworkRange

ixNetworkRange-Defines a range of IP and MAC addresses.

Note: This item has been deprecated.

SYNOPSIS

```
set networkRange [::IxLoad new ixNetworkRange options]
```

```
$networkRange subcommand options...
```

DESCRIPTION

The `ixNetworkRange` command is used to construct a network range consisting of a set of IP, MAC, and vLAN addresses. This is used in the `ixDHCP` and `ixStatCatalogItem` commands. If the `ixDHCP/ixStatCatalogItem` command specifies "MACPerPort" mapping mode in its `macMappingMode` option, then the `gateway`, `firstMac`, and `macIncrStep` options are not relevant; all network ranges route to the emulated router and a single MAC addresses emanates from each Ixia port.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition, the following commands are available. Unless otherwise described, no value returned and an exception is raised for any error found.

```
checkConfig
```

Checks the configuration of the client network object.

```
set range
```

Helps to select the activities (protocol agents) that each `networkRange` will run.

```
set rangel [$clnt_network networkRangeList.getItem 0]
```

```
$clnt_t_n_mapping setActivityAvailableForRange $rangel "my_http_client" true
```

```
set isAvailable [$clnt_t_n_mapping isActivityAvailableForRange $rangel  
"my_http_client"]
```

```
puts "==== Activity-IP Mapping for Http Agent ====="
```

```
puts $isAvailable
```

OPTIONS

```
enable
```



If true, enables the use of this network range. (Default = true).

enableStats

This is enabled to value 1, to collect per interface statistics (- AddPerInterfaceStat arguments). (Default = 0).

firstIp

The first IP address for the range. If ipType is set to "IPv4," this must be an IPv4 address. If ipType is set to "IPv6," this must be an IPv6 address. Only HTTP and FTP agents support IPv6 addressing. If there is a mixture of IPv4 and IPv6 addresses, other protocols will use only the IPv4 addresses. IxLoad supports all forms of IPv6 addressing except ::dotted-quad notation (for example, "::1.2.3.4"). (Default = 198.18.0.1).

firstMac

The first MAC address for the range. This is not used if the value of macMin the containing ixDHCP/ixStatCatalogItem object is set to "MACPerPort." (Default = 00:C6:12:00:01:00).

gateway

The gateway associated with all IP addresses in the network range. (Default = 0.0.0.0).

ipCount

The number of unique IP addresses in the network range. (Default = 100).

ipIncrStep

Indicates the increment to be applied between generated IP addresses. The format of this option is a dotted-quad IP address, in which only one of the octets may be nonzero. For example, 0.0.0.1, 0.0.2.0, 0.22.0.0 and 4.0.0.0 are valid values which will increment a different octet each time. Values that use more than one octet, for example 0.0.1.1, are illegal. (Default = 0.0.0.1). Some useful constants are:

Constant	Value
\$.ixNetworkRange(kIpIncrOctetFirst)	1.0.0.0
\$.ixNetworkRange(kIpIncrOctetSecond)	0.1.0.0
\$.ixNetworkRange(kIpIncrOctetThird)	0.0.1.0
\$.ixNetworkRange(kIpIncrOctetForth)	0.0.0.1

ipType

Type of IP address. This parameter indicates whether the address range is a range of IPv4 addresses or a range of IPv6 addresses. Only HTTP and FTP agents supIPV6. If there is a mixture of IPv4 and IPv6 addresses, other protocols will use only IPv4 addresses. IxLoad supports all forms of IPv6 addressing except ::dotted-quad notation (for example, "::1.2.3.4"). The choices are: "IPv4" and "IPv6." (Default = "IPv4").

macIncrStep

Indicates the increment to be applied between generated MAC addresses. The format of this option is a colon separated MAC address, in which only one of the octets may be nonzero. For example, 00:00:00:00:00:01, 00:00:00:00:22:00, 00:00:00:33:00:00, 00:00:44:00:00:00, 00:AA:00:00:00:00, and C:00:00:00:00:00 are valid values that will increment a different octet each time. Values that use more than one octet, for example 00:00:00:00:01:01, are illegal. This is not used if the value of macMappingMode in the containing ixDHCP/ixStatCatalogItem object is set to "MACPerPort." (Default = 00:00:00:00:00:01). Some useful constants are:

Constant	Value
\$.ixNetworkRange(kMacIncrOctetFirst)	01:00:00:00:00:00
\$.ixNetworkRange(kMacIncrOctetSecond)	00:01:00:00:00:00
\$.ixNetworkRange(kMacIncrOctetThird)	00:00:01:00:00:00
\$.ixNetworkRange(kMacIncrOctetForth)	00:00:00:01:00:00
\$.ixNetworkRange(kMacIncrOctetFifth)	00:00:00:00:01:00
\$.ixNetworkRange(kMacIncrOctetSixth)	00:00:00:00:00:01

mss

If mssEnable is true, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, IxLoad clients or servers advertise their TCP MaxiSegment Size as 1,460 bytes. (Default = 1,460).

mssEnable

If true, the use of the mss option is enabled. (Default = false).

networkMask

The subnet mask associated with the IP range. (Default = 255.255.0.0).

rangeType

Type of IP range configured on the subnet.

Value
Ethernet (default)



Value
DHCP
IPSec
PPPoE
DHCP-PD
DHCP-PD Client

`vlanEnable`

If `true`, vLAN IDs are inserted.

`vlanId`

If `vlanEnable` is `true`, this is the vLAN ID used. (Default = None).

EXAMPLE

See example in `ixDHCP`.

SEE ALSO

`ixDHCP`, `ixStatCatalogItem`

ixServerNetwork

ixServerNetwork-Defines a network for server agents.

SYNOPSIS

```
set serverNetwork [::IxLoad new ixServerNetwork $chassisChain options]
$serverNetwork subcommand options...
```

DESCRIPTION

The `ixServerNetwork` command is used to construct a server network, which is used as part of an `ixServerTrafficNetworkMapping` object. A chassis chain object, as created in the `ixChassisChain` command, must be used in the construction of this object.

A list of network ranges, as defined in the `ixRepository` object is associated with the server network. Network ranges are added to the server network through the use of the `networkRangeList.appendItem` command.

A list of ports is also associated with the network through the `portList` option.

If an emulated router is to be used, a list of IP ranges for the router is also associated with the network through the `emulatedRouterIpAddressPool` option. These are added to the object through the use of the `em` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition the following commands are available. Unless otherwise described, no value is returned and an exception is raised for any error found.

checkConfig

Checks the configuration of the server network object.

reset

Disassociates the network group from all of the Ixia ports currently in the `portList` option. Ownership of the ports is cleared.

OPTIONS

`chassisChain`

This must be a chassis chain object, as created in the `ixChassisChain` command. It represents the set of chassis used in the test and defines the chassis IDs used in the `portList` component. This option should not be changed after `portList` is set. (Default = None).

`comment`



A commentary string for the object. (Default = "").

emulatedRouter
Gateway

If `macMappingMode` is set to `kMacMappingModePort`, then an emulated router is inserted between the servers and the external port. This is the gateway to be used for that router. (Default = 0.0.0.0).

emulatedRouterIp
AddressPool

If `macMappingMode` is set to `kMacMappingModePort`, then an emulated router is inserted between the servers and the external port. This option is a list of `ixEmulatedRouterIpAddressRange` objects that define the routers' source addresses that will be used. One IP address is taken from the list and used for each Ixia port. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list. (Default = {}).

emulatedRouter
Subnet

If `macMappingMode` is set to `kMacMappingModePort`, then an emulated router is inserted between the clients and the external port. This is the network mask to be used for that router. (Default = 255.255.255.0).

emulatedRouter
GatewayIPv6

If `macMappingMode` is set to `kMacMappingModePort` and `ipType` in `ixEmulat` is set to "IPv6" for any addresses, then an IPv6 address is also required for the emulated router inserted between the clients and the external port. IxLoad supports all forms of IPv6 addressing except `::dotted-quad` notation (for example, `:::1.2.3.4`). This is the IPv6-format address of the gateway to be used for that router. (Default = `:::C212:0001`).

emulatedRouter
SubnetIPv6

Subnet mask applied to `emulatedRouterGatewayIPv6` address. (Default = `FFFF:FFFF:FFFF:FFFF:FFFF:FFFF::0`).

impairment

If enabled, this option helps to intentionally degrade the traffic transmitted by the network. You can cause it to drop or duplicate packets, or delay them for certain lengths of time. Refer to `ixImpairment` for a description of all the options.

ipSourcePortFrom

Defines the beginning of the range of ephemeral port numbers used to establish connections. The end of the range is specified by `ipSourcePortTo`.

The first port in the range that IxLoad uses for traffic is 1 greater than the value you specify for `ipSourcePortFrom`. For example, if you specify 1,024, traffic originates from port 1,025; no traffic originates from port 1,024. The minimum value for `ipSourcePortFrom` is 1,024. (Default = 1,024).

`ipSourcePortTo`

Defines the end of the range of ephemeral port numbers used to establish connecto the server. The beginning of the range is specified by `ipSourcePortFrom`. (Default = 65,535).

`linkLayerOptions`

The link layer options to be associated with the ports associated with this server network. Only Ethernet options are currently supported. (Default = `kLink`

`macMappingMode`

The mapping between IP addresses and MAC addresses. One of:

Option	Usage
<code>\$.ixServerNetwork(kMacMappingModelp)</code>	(Default) One MAC address is associated with each IP address.
<code>\$.ixServerNetwork(kMacMappingModePort)</code>	One MAC address is used for all IP addresses on the port.

`name`

The name associated with this object. (Default = "newNetwork").

`networkRangeList`

A list of `ixRepository` objects that define the networks from which addresses will be associated with the servers. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list. (Default = `{}`).

`portList`

A list of ports associated with the server network. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to maniputhis list. Ports are added directly into this object; see the following example:

```
$serverNetwork portList.appendItem \
    -chassisId 1 \
```



```
-cardId      2 \  
-portId      2
```

rangeType

Type of IP range configured on the subnet.

Value
Ethernet (default)
DHCP
IPSec
PPPoE
DHCP-PD
DHCP-PD Client

You can insert the same parameters for the `ixStatCatalogItem`.

SUB-OBJECTS

arpSettings

This is an object of type `ixArpSettings`, which specifies the manner in which ARP is handled on this network. (Default = <see `ixArpSettings`>). The options of this object should be set directly via:

```
$serverNetwork arpSettings.config options...
```

tcpParameters

This is an object of type `ixTcpParameters` that specifies the manner in which TCP traffic is handled on this network. (Default = <see `ixTcpParameters`>). The options of this object should be set directly via:

```
$serverNetwork tcpParameters.config options...
```

EXAMPLE

```
set svr_network [::IxLoad new ixServerNetwork $chassisChain]  
  
$svr_network config -name "svr_network" \  
    -cardType $::ixCard(kCard1000Txs4) \  
    -ipSourcePortFrom 1024 \  
    -ipSourcePortTo 65536 \  
  
$svr_network networkRangeList.appendItem \  
    -name          "svr_range" \  
    -rangeType     Ethernet
```

```
-enable          1 \  
-firstIp         "198.18.200.1" \  
-ipCount         1 \  
-networkMask    "255.255.0.0" \  
-gateway         "0.0.0.0" \  
-firstMac        "00:C6:12:02:02:00" \  
-vlanEnable      0 \  
-vlanId          1 \  
-mssEnable       0 \  
-mss             100
```

```
$svr_network impairment.config\  
-enable True\  
-addDrop True\  
-drop 5
```

```
$svr_network portList.appendItem \  
-chassisId 1 \  
-cardId 2 \  
-portId 2
```

SEE ALSO

`ixServerTrafficNetworkMapping`, `ixChassisChain`, `ixRepository`,
`ixEmulatedRouterIpAddressRange`, `ixArpSettings`, `ixTcpParameters`

ixServerTraffic

ixServerTraffic-Builds a list of server agents to handle server traffic.

SYNOPSIS

```
set serverTraffic [::IxLoad new ixServerTraffic options]
$serverTraffic subcommand options...
```

DESCRIPTION

The `ixServerTraffic` command is used to construct the model for server network traffic to be handled during a test. It is used in the `ixServerTrafficNetworkMapping` command to co-ordinate networks with server agents.

Its primary option is the `agentList` list of agents that will handle server traffic. The agents that exist for a number of protocols are documented in the subsequent chapters.

Agents are added to the agent list using the `appendItem` subcommand and may be otherwise manipulated using the commands supported by the `ixConfigSequenceContainer` command. All agents are added in the same manner:

```
set serverTraffic [::IxLoad new ixServerTraffic \  
    -name "Servers"]  
  
$serverTraffic agentList.appendItem \  
    -name      "my_protocol_server" \  
    -protocol  "<PROTOCOL>" \  
    -type      "Server" \  
    <other per-protocol options>
```

The `name`, `protocol`, and `type` are required fields. These define a particular type of agent; the `protocol` field should be drawn from the table above. In addition to the required fields, the agent definition should include options that are specific to a particular protocol, and defined in their respective appendix.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition the following commands are available. Unless otherwise described, no value returned and an exception is raised for any error found.

checkConfig

Checks the configuration of the server traffic object.

OPTIONS

agentList

A list of agent objects that define the agents, which will be used to handle server traffic. Refer to the various appendixes listed above to determine the options that the agents offer. Refer to `ixConfigSequenceContainer` for a list of commands that may be used to manipulate this list. (Default = {}).

name

The name associated with this object. (Default = "newActivityModel").

EXAMPLE

```
#-----  
# Construct the Server Traffic  
#-----  
set svr_traffic [::IxLoad new ixServerTraffic \  
                -name "svr_traffic"]  
  
#-----  
# Create a server agent -- no actions are involved in this agent  
#-----  
$svr_traffic agentList.appendItem \  
    -name      "my_http_server" \  
    -protocol  "HTTP" \  
    -type      "HTTP" \  
    -agent     "HTTP"
```



```
-type          "Server" \  
-httpPort     80  
  
for {set idx 0} {$idx < \  
[$svr_traffic agentList(0).responseHeaderList.indexCount]}\  
{incr idx} {  
    set response [$svr_traffic \  
        agentList(0).responseHeaderList.getItem $idx]  
    if {[$response cget -name] == "200_OK"} {  
        set response200ok $response  
    }  
    if {[$response cget -name] == "404_PageNotFound"} {  
        set response404_PageNotFound $response  
    }  
}  
  
#-----  
# Clear pre-defined web pages, add new web pages  
#-----  
$svr_traffic agentList(0).webPageList.clear  
  
$svr_traffic agentList(0).webPageList.appendItem \  
    -page          "/4k.html" \  
    -payloadType   "range" \  
    -payloadSize   "4096-4096" \  
    -response      $response200ok  
  
$svr_traffic agentList(0).webPageList.appendItem \  
    -page          "/8k.html" \  
    -payloadType   "range" \  
    -payloadSize   "8192-8192" \  

```

-response \$response404_PageNotFound

SEE ALSO

ixServerTrafficNetworkMapping, protocol support appendices

ixServerTrafficNetworkMapping

ixServerTrafficNetworkMapping-Ties a server network to traffic model.

SYNOPSIS

```
set serverMapping [::IxLoad new ixServerTrafficNetworkMapping options]
$serverMapping subcommand options...
```

DESCRIPTION

The `ixServerTrafficNetworkMapping` command is used to map a set of server agents that receive traffic (in an `ixServerTraffic` object) to the set of networks that will carry the traffic (in an `ixStatCatalogItem` object).

A number of additional options control the manner in which the server traffic applied to the networks.

The `standbyTime`, `offlineTime`, `sustainTime`, and `totalTime` options determine the timeline for server agents.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

If `true`, this mapping is included in the `IxLoad` test. (Default = `true`).

`iterations`

The number of times that the traffic-network pair perform their functions (establishing TCP connections, retrieving FTP files, and so forth) in the test. (Default = `1`).

`matchClientTotalTime`

If `true`, the servers on this mapping will stay online for the same length of time as the longest-running client agent.

If you do not check this box and a server's duration is shorter than one of the connecting to it, the server will go offline while the client is connected; if this is not what you intend to happen during testing, the test results for that client may be misleading.

If `false`, `IxLoad` calculates agent run times independently for each server activity. (Default = `true`).

`name`

The name associated with this object. (Default = "NetworkTrafficMapping").

offlineTime

Amount of time agents are idle between iterations. (Default = 0).

standbyTime

The amount of time, expressed in seconds, that elapses between the time the test is started and the time that the traffic-network pair become active. If you have multiple traffic-network pairs in your test, you can use this parameter to stagger their start times. A value of 0 causes the test to begin immediately. The valid range is from 0 to 1,000 hours (3,600,000). (Default = 0).

sustainTime

The amount of time, in seconds, during which all users are up and performing the central test objectives, such as establishing and closing connections (TCP), retrieving or serving pages (HTTP), or sending or receiving files (FTP). (Default = 20).

totalTime

The total time required to run the test, including Standby, Sustain, and Offline times for all iterations. (Default = 60).

SUB-OBJECTS

network

An object of type `ixStatCatalogItem` that provides the networks associated with the server agents. (Default = {}).

traffic

An object of type `ixServerTraffic` that provides the model of traffic to be gener (Default = {}).

EXAMPLE

```
#-----  
# Create the server traffic to network mapping  
#-----  
set svr_mapping [::IxLoad new ixServerTrafficNetworkMapping \  
    -network          $svr_network \  
    -traffic          $svr_traffic \  
    -matchClientTotalTime 1
```



]

SEE ALSO

`ixTest`, `ixStatCatalogItem`, `ixServerTraffic`

statCollectorUtils

`ixWaitEventCommand` - cause a command to wait for another to execute

SYNOPSIS

```
$my_ixWaitEventCommand config \  
    -option      value
```

DESCRIPTION

`ixWaitEventCommand` causes one command to wait for another to finish executing for it is itself executed. `ixSendEventCommand` is used to trigger the waiting command. `ixSendEventCommand` and `ixWaitEventCommand` are added to an `actionList` using the `appendItem` command.

For example, if `Command2` must be executed only after `Command1` has been executed:

1. An `ixWaitEventCommand` is inserted preceding `Command2`.
2. An `ixSendEventCommand` is added after `Command1`, with the same `eventID` as in the `ixWaitEventCommand`.

When `Command1` finishes executing, the `ixSendEventCommand` ends the `ixWaitEventCommand` for `Command2`, causing `Command2` to be executed.

`ixSendEventCommand` and `ixWaitEventCommand` can only be used with Subscriber activities.

OPTIONS

`commandType`

Command type. The only value is "WaitEventCommand".

`eventID`

Unique value identifying this `ixWaitEventCommand`. Default value = 1.

EXAMPLE

```
set my_ixSendEventCommand [::IxLoad new ixSendEventCommand]
$my_ixSendEventCommand config \
  -commandType "SendEventCommand" \
  -eventId 1

$Subscriber_Activity_HTTPClient1 agent.actionList.appendItem -object
$my_ixSendEventCommand
.
.
.
$Subscriber_Activity_FTPClient1 agent.actionList.clear

set my_ixWaitEventCommand [::IxLoad new ixWaitEventCommand]
$my_ixWaitEventCommand config \
  -commandType "WaitEventCommand" \
  -eventId 1

$Subscriber_Activity_FTPClient1 agent.actionList.appendItem -object
$my_ixWaitEventCommand
```

SEE ALSO

[ixSendEventCommand](#)



Internal Commands

This section lists the IxLoad Tcl API's internal commands.

In this chapter:

ixConfig	236
ixConfigSequenceContainer	238
ixConfigSortedNamedItemList	241

ixConfig

ixConfig—Allows options to be configured for an object.

SYNOPSIS

```
set anyIxLoadObject [$::IxLoad new ixLoadObject options]
$anyIxLoadObject subcommand options...
```

DESCRIPTION

The `ixConfig` object provides the means by which command options are set and read. It is never used directly. The commands that are based on `ixConfig` support the subcommands described below.

SUBCOMMANDS

The following subcommands are available to handle options:

cget option

This subcommand is used to obtain the current value of any option. The `option` must begin with a hyphen (-). The return value is of a type appropriate for the option.

config option value option value...

The `config` subcommand may be used to set the value of one or more options in a command. The `option` must begin with a hyphen (-). The `value` must be of a type appropriate for the option.

getOptions

This subcommand returns a Tcl list with all of the options available for a command including an initial hyphen for each option.

OPTIONS

None.

EXAMPLE

```
$object cget -name

$object config -name "media" -value "mp3"

set optionList [$object getOptions]
```

ixConfigSequenceContainer

ixConfigSequenceContainer—Handles a list of objects.

SYNOPSIS

```
set anyIxLoadObject [$::IxLoad new ixLoadObject options]

$anyIxLoadObject option.subcommand sub-options...
```

DESCRIPTION

The `ixConfigSequenceContainer` object provides a list in which commands contheir options.

See the following example:

- `$anIxLoadCommand` is an instance of an `ixLoadCommand`.
- `ix` has an option `listOfIxStuff`.
- `listOfIxStuff` is a list, each of whose elements is of type `ixStuff`, with options `firstIp` and `lastIp`.

In order to create a new instance of `ixLoadCommand` and add an item to its `li` you should use the following sequence:

```
set $anIxLoadCommand [$::IxLoad new ixLoadCommand]
$anIxLoadCommand listOfIxStuff.appendItem \
    -firstIp 192.18.0.1 \
    -lastIp 192.18.0.100
```

The first item in a sequence container has index 0. Negative indexes may be used to indicate positions from the last item in the container. -1 corresponds to the last item in the list, -2 to the one before that, and so forth.

SUBCOMMANDS

The following subcommands are available to handle options. Except where noted, no value is returned; an exception is raised in the case of an error. In all cases where they are used the `option` must begin with a hyphen (-). The `value` must be of a type appropriate for the option.

```
appendItem option value option value...
```

The `appendItem` subcommand may be used to add an item to a list. Any number of options in the listed item may be set as part of the append.

```
configItem index option value option value...
```

The `configItem` subcommand may be used to configure a particular item in a list. Any number of options in the list item may be set. The `index` argument is used to indicate which item in the list is to be configured.

clear

The `clear` subcommand may be used to delete all listed items from a list.

deleteItem index

The `deleteItem` subcommand may be used to delete a listed item from a list. The `index` argument is used to indicate which item in the list is to be configured.

find mode option value option value...

The `find` subcommand may be used to search a list for matching criterion. The `mode` argument may be one of:

Option	Usage
<code>exact</code>	Match the <code>value</code> fields exactly.
<code>regexp</code>	Use regular expressions in the matching.
<code>uppercase</code>	Perform a caseless match.

Any number of options may be used in the match. The `find` subcommand searches for all items in the list, whose keyworded options match the values. A list of indexes of matching items is returned.

getItem index

The `getItem` subcommand may be used to retrieve an item from a list. The `index` argument is used to indicate which item in the list is to be retrieved. This subcommand returns the object from the list.

indexCount

The `indexCount` subcommand returns the number of objects in the list.

insertItem index option value option value...

The `insertItem` subcommand may be used to insert an item in a list. Any number of options in the list item may be set. The `index` argument is used to indicate the insertion point in the list. The new item will be inserted before the `index`'th item in the list.

OPTIONS

None.

EXAMPLE

```
$list_object.clear
$list_object.appendItem -name "sample"
$list_object.insertItem 1 -name "sample2"
$list_object.configItem -value "mp4"
$list_object.deleteItem -1
```



```
set found_list [$list_object.find regexp \  
    -speed "\d*[Mm]bps"]  
  
$list_object.getItem 3  
  
set numObjects [$list_object.indexCount]
```

ixConfigSortedNamedItemList

ixConfigSortedNamedItemList—Handles a list of objects that is in sorted order.

SYNOPSIS

```
set anyIxLoadObject [${::IxLoad} new ixLoadObject options]
${anyIxLoadObject} option.subcommand sub-options...
```

DESCRIPTION

`ixConfigSortedNamedItemList` behaves similar to `ixConfigSequenceContainer`, except that `getItem` requires the name of an item rather than its index. The `list(index)` notation still works for positional indexing with the `deleteItem` and `configItem` options.

`insertItem` and `appendItem` are not supported; instead an `addItem` method is supwhich has the same syntax as `appendItem`. This difference is required because an item's position in the list is controlled by the automatic sorting and cannot be specified by the user.

As with `ixConfigSequenceContainer`, the first item in an `ixConfigSortedNamed` has index 0. Negative indexes indicate positions from the last item in the list. For example, -1 corresponds to the last item in the list, -2 to the one before that, and so forth.

SUBCOMMANDS

The following subcommands are available to handle options, which are lists. Except where noted, no value is returned; an exception is raised in the case of an error. In all cases where they are used the `option` must begin with a hyphen (-). The `value` must be of a type appropriate for the option.

```
addItem name option value option value...
```

The `addItem` subcommand adds an item to a list. Any number of options in the list item may be set as part of the addition. Items added with the `addItem` method should always include the `-name` option so that the item can be referenced later. If you do not specify a name, `IxLoad` will assign a default name, but you should not rely on default names because future releases of `IxLoad` may assign different default names. After `addItem` has been executed, it returns the object that has been added so that you can use the `config` subcommand to configure it.

```
configItem index option value option value...
```

The `configItem` subcommand configures a particular item in a list. You can pass multiple option/value pairs in one command, so that the command configures multiple options at the same time. The `index` argument specifies the list item to be configured. To determine the index number of an item, use the `find` subcommand.

```
clear
```

The `clear` subcommand deletes all items from a list.

deleteItem index

The `deleteItem` subcommand deletes an item from a list. The `index` argument specifies the list item to be deleted. To determine the index number of an item, use the `find` subcommand. To delete an item by name, use the `removeItem` sub-command.

find mode option value option value...

The `find` subcommand searches a list for item that matches its search criteria. The `mode` argument may be one of:

Option	Usage
<code>exact</code>	Match the <code>value</code> fields exactly.
<code>regexp</code>	Use regular expressions in the matching.
<code>uppercase</code>	Perform a caseless match.

Any number of options may be used in the match. The `find` subcommand searches for all items in the list, whose keyworded options match the values. A list of indexes of matching items is returned.

getItem name

The `getItem` subcommand may be used to retrieve an item from a list. The `name` argument is used to indicate which item in the list is to be retrieved. This subcommand returns the object from the list.

indexCount

The `indexCount` subcommand returns the number of objects in the list.

removeItem name

The `removeItem` subcommand deletes an item from a list. The `name` argument specifies the list item to be deleted. To delete an item by its index, use the `deleteItem` subcommand.

OPTIONS

None.

EXAMPLE

```
$list_object.clear
$list_object.addItem -name "sample"
$list_object.configItem -value "mp4"
set found_list [$list_object.find regexp \
    -speed "\d*[Mm]bps"]
$list_object.getItem "sample"
set numObjects [$list_object.indexCount]
```

`$list_object.deleteItem -1`

SEE ALSO

`ixConfigSequenceContainer, ixRepository`



Network Stack API

Beginning with release 4.10, IxLoad uses an object-oriented model for its network stack. TCL scripts created with previous releases of IxLoad will still function, but any scripts created using ScriptGen will use the object-oriented network stack.

The following sections describe the object model.

In this chapter:

Setting the Global Plug-ins List	245
Creating the Stack	245
List Programming in the Network Stack	246
Creating the Ranges	247
Network- and Session-Specific Data	248
Network Stack Configuration in pre-4.10 Releases	250
Network Stack Configuration in 4.10 and Later Releases	251
Network Stack Object Model	253
Physical Layer	258
Layer 2 Protocols	270
Emulated Router Plugin	288
IP Plugin	300
DHCP Client and Server	318
Authentication Extension Plugins	378
Impair Plugin	482
IPSec Plugin	514
PPPoX Plugin	564
L2TP Plugin	602
GTP Plugin	641
Global Plugins	692
Mobile Subscribers Plugins	730

Setting the Global Plug-ins List

The Global plug-ins are: Filter, GratArp, TCP, DNS and the Settings plug-in. In the GUI, the Global Plug-ins are displayed to the right of the network configuration window (see the figure below).



In a Tcl script, you must add them to the Global Plug-in list, then configure them.

Creating the Stack

To create a network stack, you begin with a Layer 1 plug-in, which for IxLoad is the Ethernet Plug-in. You can retrieve this plugin by using `getL1Plugin` function called on the network object. For example:

```
set Ethernet__PHY_1 [$clnt_network getL1Plugin]
```

Next, you continue constructing the stack by adding the L2 plug-in (MAC/VLAN) to the children list of the Ethernet Plug-in:

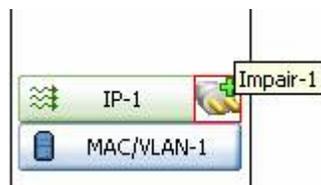
```
$Ethernet__PHY_1 childrenList.appendItem -object $MAC_VLAN_1
```

Continue adding plugins until the stack is complete.



In addition to the main stack plug-ins, there are also extension plugins for protocols that provide additional functionality for the stack plug-ins. For example, the Impair plug-in (see below) impairs traffic according parameters that you set. You add these extension plug-ins to the extension list. For example:

```
$IP_1 extensionList.appendItem -object $Impair_1
```





List Programming in the Network Stack

Lists in the IxLoad 4.10 network stack function slightly differently from previous releases.

For any network stack object (that is, any object that begins with `IxNet`, you first create the object, then append it to the list, then configure it. This sequence was different in pre-4.10 releases:

pre-4.10: set - config - append

4.10 and later: create - append - config

You can use all of the existing list operators on a 4.10 network stack list. See `ixConfigSequenceContainer` (on page 238) for information on the operators.

Creating the Ranges

When you add a range to a stack plug-in, IxLoad automatically adds related ranges to the other plug-ins (except for the Emulated Router plug-in). Therefore, you start with the plug-in at the top of the stack, and work your way down to configure each address range. When you create a range for that top-most plug-in and add it to the `rangeList`, IxLoad automatically creates the ranges for all the other plug-ins (again, except for the Emulated Router plug-in).

For example:

```
set ip_1 [::IxLoad new ixNetIPv4V6Range]
# ixNet objects needs to be added in the list before they are configured!
$IP_1 rangeList.appendItem -object $ip_1
```

You can get the remaining ranges using the `getLowerRelatedRange` method:

```
set MAC_R4 [$ER_R3 getLowerRelatedRange "MacRange"]
```

Following that, you configure the range:

```
$MAC_R4 config \
-count                1 \
-name                 "MAC-R4" \
-enabled              true \
-mtu                  1500 \
-mac                  "00:0A:0A:00:08:00" \
-incrementBy         "00:00:00:00:00:01"
```

To get the ranges of the extension plug-ins, pass the name of the extension plug-in to the `getExtensionRange` function:

```
set impair_2 [$ip_2 getExtensionRange $Impair_2]
```

For the Emulated Router plugin, instead of using `getLowerRelatedRange`, you must add the ranges manually. For example:

```
set ER_R3 [::IxLoad new ixNetEmulatedRouterRange]
# ixNet objects needs to be added in the list before they are configured!
$Emulated_Router_3 rangeList.appendItem -object $ER_R3
```

```
$ER_R3 config \
-count                1 \
-name                 "ER-R3" \
-gatewayIncrementMode "perSubnet" \
-enabled              true \
-autoMacGeneration   true \
-publishStats         false \
-incrementBy         "0.0.0.1" \
-prefix              16 \
-gatewayIncrement    "0.0.0.0" \
-generateStatistics   false \
-mss                  1460 \
-gatewayAddress       "0.0.0.0" \
-ipAddress            "10.10.0.8" \
-ipType               "IPv4"
```

There is also a `getUpperRelatedRange` method that you can use to configure the range of the next *higher* plug-in in the stack:

```
set ER_R4 [$MAC_R4 getUpperRelatedRange "EmulatedRouter"]
```

```
$ER_R4 config \
-count                1 \
-name                 "ER-R4" \
-gatewayIncrementMode "perSubnet" \
```



```
-enabled true \  
-autoMacGeneration true \  
-publishStats false \  
-incrementBy "0.0.0.1" \  
-prefix 16 \  
-gatewayIncrement "0.0.0.0" \  
-generateStatistics false \  
-mss 1460 \  
-gatewayAddress "0.0.0.0" \  
-ipAddress "10.10.0.14" \  
-ipType "IPv4"
```

Network- and Session-Specific Data

In addition to adding plug-ins to the network stack, there are network- and session-specific options that can be set. The session- and network- settings are related to one or more of the plug-ins in the network stack.

In the GUI, you access the session-specific options by clicking the “Global Options” in the network configuration window. To access the network-specific options, you use the “Network Group Settings” tab.



To set these options in a TCL script, use code as follows:

```
#####
# Session Specific Settings
#####
set my_ixNetImpairSessionData [$Test1 getSessionSpecificData "ImpairPlugin"]
$my_ixNetImpairSessionData config
set my_ixNetPppoxSessionData [$Test1 getSessionSpecificData "PppoxPlugin"]
$my_ixNetPppoxSessionData config
    -teardownRateInitial          300
    -maxOutstandingRequests       300
    -maxOutstandingReleases      300
    -setupRateInitial            300

#####
# Network Specific Settings(overrides some of the
# session specific settings)
#####
# If the 'override' property is set to true the specific
# settings for this network will override the session specific settings.

set my_ixNetPppoxPortGroupData [$server network getNetworkSpecificData "PppoxPlugin"]
$my_ixNetPppoxPortGroupData config
    -useWaitForCompletionTimeout  false
    -maxOutstandingRequests       300
    -perSessionStatFilePrefix     "MY_PREFIX"
    -enablePerSessionStatGeneration false
    -waitForCompletionTimeout      120
    -maxOutstandingReleases      300
    -overrideGlobalRateControls   false
    -activityID                   "0"
    -role                          "client"
    -teardownRateInitial          300
    -setupRateInitial            300
```



Network Stack Configuration in pre-4.10 Releases

The code example below shows an example of how a network was configured in releases prior to 4.10. Compare this example with the post-4.10 example that follows.

```
set clnt_network [::IxLoad new ixClientNetwork $chassisChain]

$clnt_network config \
name          "clnt_network"

$clnt_network networkRangeList.appendItem \

-name          "clnt_range" \
-enable        1 \
-firstIp       "198.18.2.1" \
-ipIncrStep    $::ixNetworkRange(kIpIncrOctetForth) \
-ipCount       100 \
-networkMask   "255.255.0.0" \
-gateway       "0.0.0.0" \
-firstMac      "00:C6:12:02:01:00" \
-macIncrStep   $::ixNetworkRange(kMacIncrOctetSixth) \
-vlanEnable    0 \
-vlanId        1 \
-mssEnable     0 \
-mss           100

$clnt_network portList.appendItem \
-chassisId     1 \
-cardId        $::IxLoadPrivate::SimpleSettings::clientPort(CARD_ID)\
-portId        $::IxLoadPrivate::SimpleSettings::clientPort(PORT_ID)
```

Network Stack Configuration in 4.10 and Later Releases

The code example below shows an example of how a network is configured starting with release 4.10.

```
set clnt_network [::IxLoad new ixNetworkGroup $chassisChain]

$clnt_network config \
  -name          "clnt_network"

$clnt_network globalPlugins.clear

set Filter [::IxLoad new ixNetFilterPlugin]
# ixNet objects needs to be added in the list before they are configured!
$clnt_network globalPlugins.appendItem -object $Filter\
$Filter config \
  -name          "Filter"
set GratArp [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list before they are configured!
$clnt_network globalPlugins.appendItem -object $GratArp
$GratArp config \
  -enabled       true \
  -name          "GratArp"

set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list before they are configured!
$clnt_network globalPlugins.appendItem -object $TCP
$TCP config \
  -tcp_tw_recycle      true \
  -tcp_keepalive_time  75 \
  -tcp_keepalive_intvl 7200 \
  -tcp_wmem_default    4096 \
  -tcp_port_min        1024 \
  -tcp_port_max        65535 \
  -tcp_window_scaling  false \
  -name                "TCP" \
  -tcp_rmem_default    4096

set DNS [::IxLoad new ixNetDnsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$clnt_network globalPlugins.appendItem -object $DNS

$DNS config \
  -name          "DNS"

set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$clnt_network globalPlugins.appendItem -object $Settings

$Settings config \
  -name          "Settings"

set Ethernet__PHY_1 [$clnt_network getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config

$Ethernet__PHY_1 config \
  -name          "Ethernet /PHY-1" \
  -cardElm      $my_ixNetEthernetELMPlugin
```



```
$Ethernet__PHY_1 childrenList.clear

set MAC_VLAN_1 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Ethernet__PHY_1 childrenList.appendItem -object $MAC_VLAN_1

$MAC_VLAN_1 config \
  -name          "MAC/VLAN-1"

$MAC_VLAN_1 childrenList.clear

set IP_1 [::IxLoad new ixNetIpV4V6Plugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_1 childrenList.appendItem -object $IP_1

$IP_1 config \
  -name          "IP-1"

#####
# Setting the ranges starting with the plugin on top of the stack
#####
$IP_1 rangeList.clear

set ip_1 [::IxLoad new ixNetIpV4V6Range]
# ixNet objects needs to be added in the list before they are configured!
$IP_1 rangeList.appendItem -object $ip_1

$ip_1 config \
  -count          100 \
  -name           "ip-1" \
  -gatewayAddress "0.0.0.0" \
  -autoMacGeneration true \
  -prefix         16 \
  -mss            100 \
  -ipAddress      "198.18.2.1"

set mac_1 [$ip_1 getRange "macRange"]

$mac_1 config \
  -count          100 \
  -mac            "00:C6:12:02:01:00" \
  -name           "mac-1"

set vlan_1 [$mac_1 getRange "vlanRange"]

$vlan_1 config \
  -name           "vlan-1" \
  -idIncrMode     1 \
  -priority       0 \
  -innerPriority   0

$clnt_network portList.appendItem \
  -chassisId     1 \
  -cardId        $::IxLoadPrivate::SimpleSettings::clientPort(CARD_ID)\
  -portId        $::IxLoadPrivate::SimpleSettings::clientPort(PORT_ID)
```

Network Stack Object Model

The following sections describe the higher-level objects in the network stack object model. Many of these objects are not directly configurable; instead, they are base classes for lower-level classes for objects that can be configured in the Tcl API.

Network Stack Object Model - Base Classes beginning with 'A-DH'

Activity	<i>Base class for Activity plug-ins</i>
└ GTPSAActivity	<i>GTP SSGN Activity plug-in</i>
Associate	<i>Associate object</i>
└ IPSecAssociate	<i>IPSec associate</i>
DHCPRange	<i>DHCP address range</i>

Network Stack Object Model - Base Classes beginning with 'DN'

DNplug-inBase	<i>Base class for NAC and 8021X plug-ins</i>
└ Nacl3plug-in	<i>NAC Layer 3 plug-in</i>
└ Dot1xplug-in	<i>8021X plug-in</i>
DNRangeBase	<i>Base class for NAC and 8021X address ranges</i>
└ Nacl3Range	<i>NAC Layer 3 address range</i>
└ Dot1xRange	<i>8021X address range</i>
DNPortGroupBase	<i>Base class for NAC and 8021X port group options</i>
└ Nacl3PortGroupData	<i>NAC Layer 3 port group options</i>
└ Dot1xPortGroupData	<i>8021X port group options</i>
DNSessionBase	<i>Base class for NAC and 8021X rate control and protocol options</i>
└ Nacl3SessionData	<i>NAC Layer 3 rate control and protocol options</i>
└ Dot1xSessionData	<i>8021X rate control and protocol options</i>

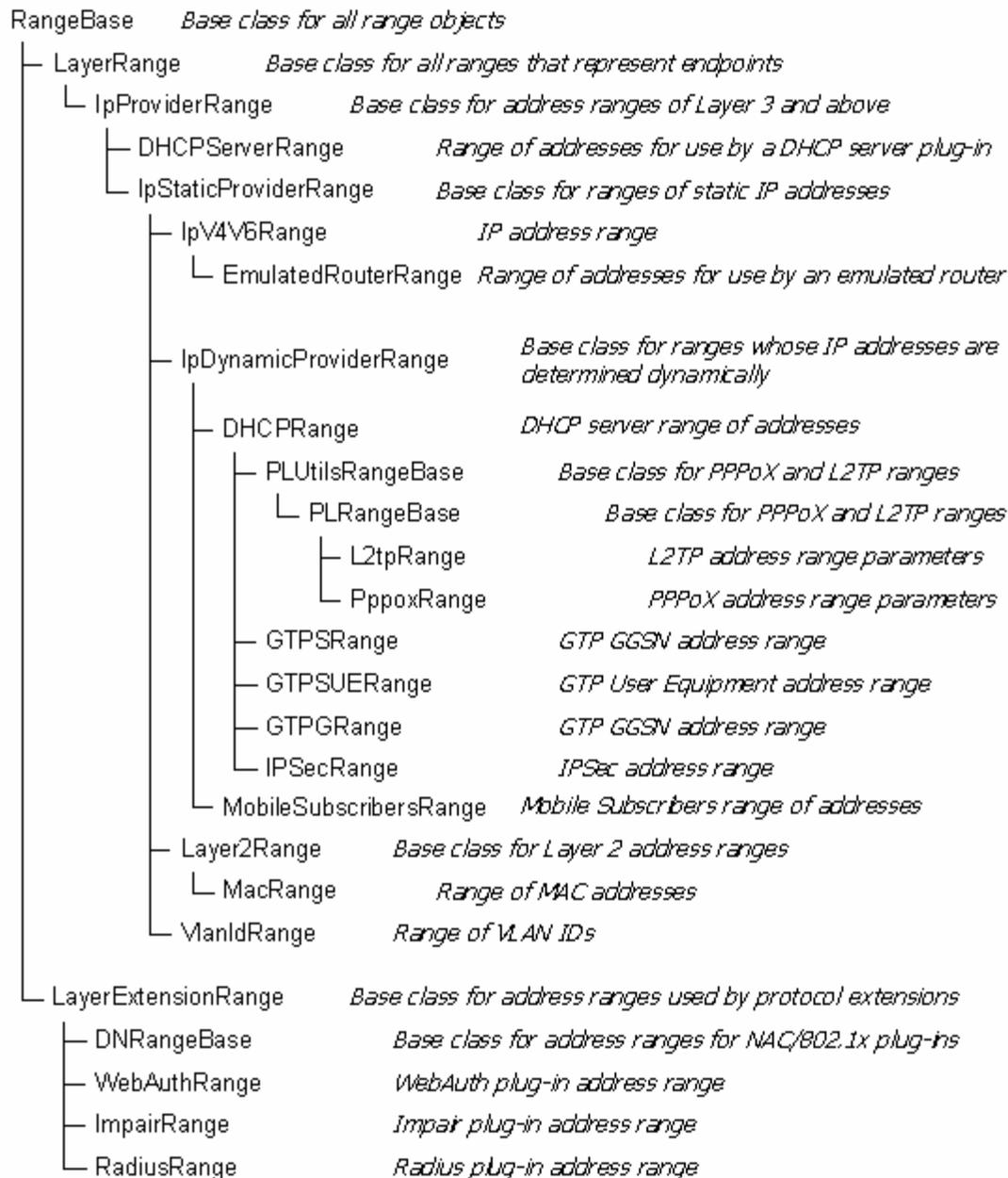
Network Stack Object Model - Base Classes beginning with 'I'

IPSecEapTuple	<i>Base class for IPSec EAP and AKA tuples</i>
└ IPSecEapTuple	<i>IPSec EAP tuple</i>
└ IPSecAkaTuple	<i>IPSec AKA tuple</i>

Network Stack Object Model - Base Classes Beginning with 'P'

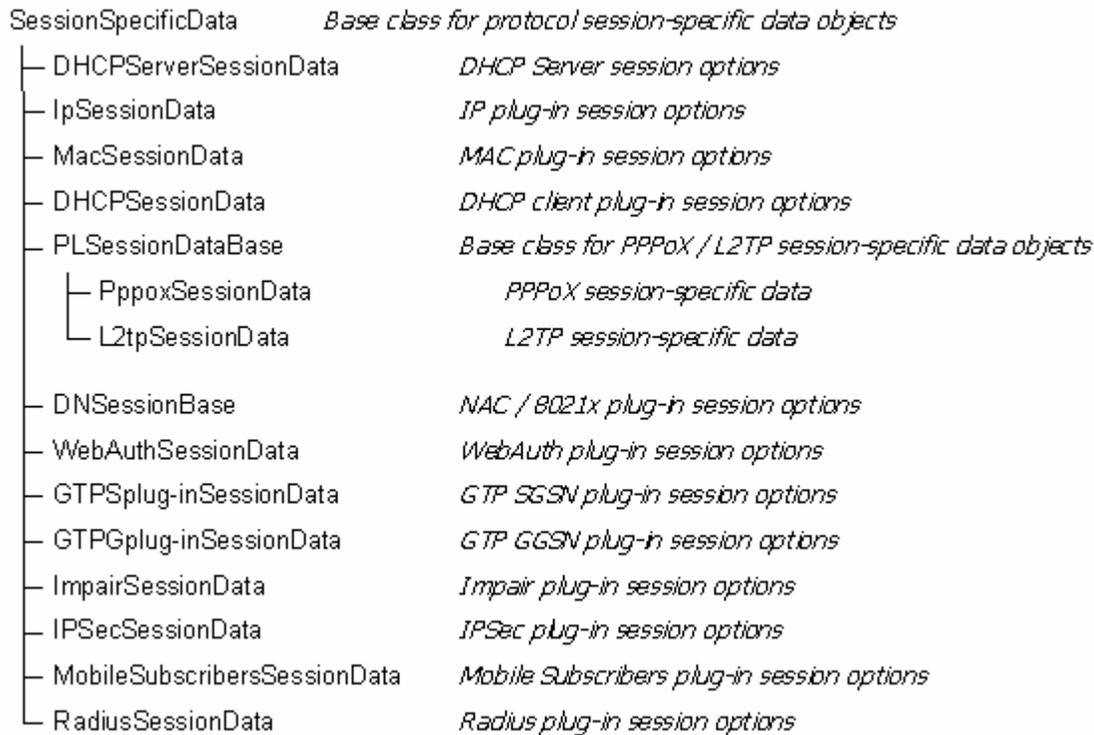
PortGroupSpecificData	<i>Base class for plug-in specific port group data classes</i>
├─ PortGroupDataAssociates	<i>Base class for plug-in specific port group data classes</i>
│ └─ DHCPPortGroupData	<i>DHCP client port group options</i>
│ └─ PLPortGroupDataBase	<i>PPPoX / L2TP port group options</i>
│ └─ GTPSplug-inPortGroupData	<i>GTP GGSN port group options</i>
│ └─ IPSecPortGroupData	<i>IPSec port group options</i>
│ └─ MobileSubscribersPortGroupData	<i>Mobile Subscribers port group options</i>
├─ L2EthernetPortGroupData	<i>Layer 2 Ethernet port group options</i>
├─ IpRangePortGroupData	<i>IP address range port group options</i>
├─ DNPortGroupBase	<i>NAC / 8021x port group options</i>
├─ WebAuthPortGroupData	<i>WebAuth port group options</i>
└─ GTPGplug-inPortGroupData	<i>GTP GGSN port group options</i>

Network Stack Object Model - Base Classes Beginning with 'R'

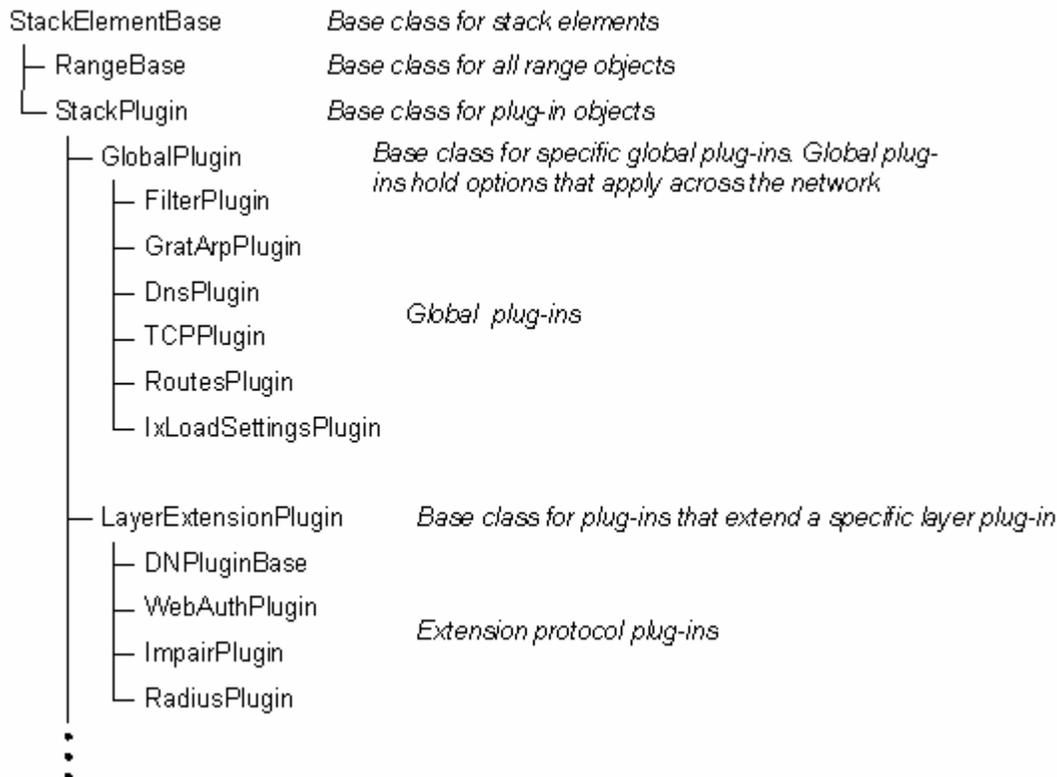




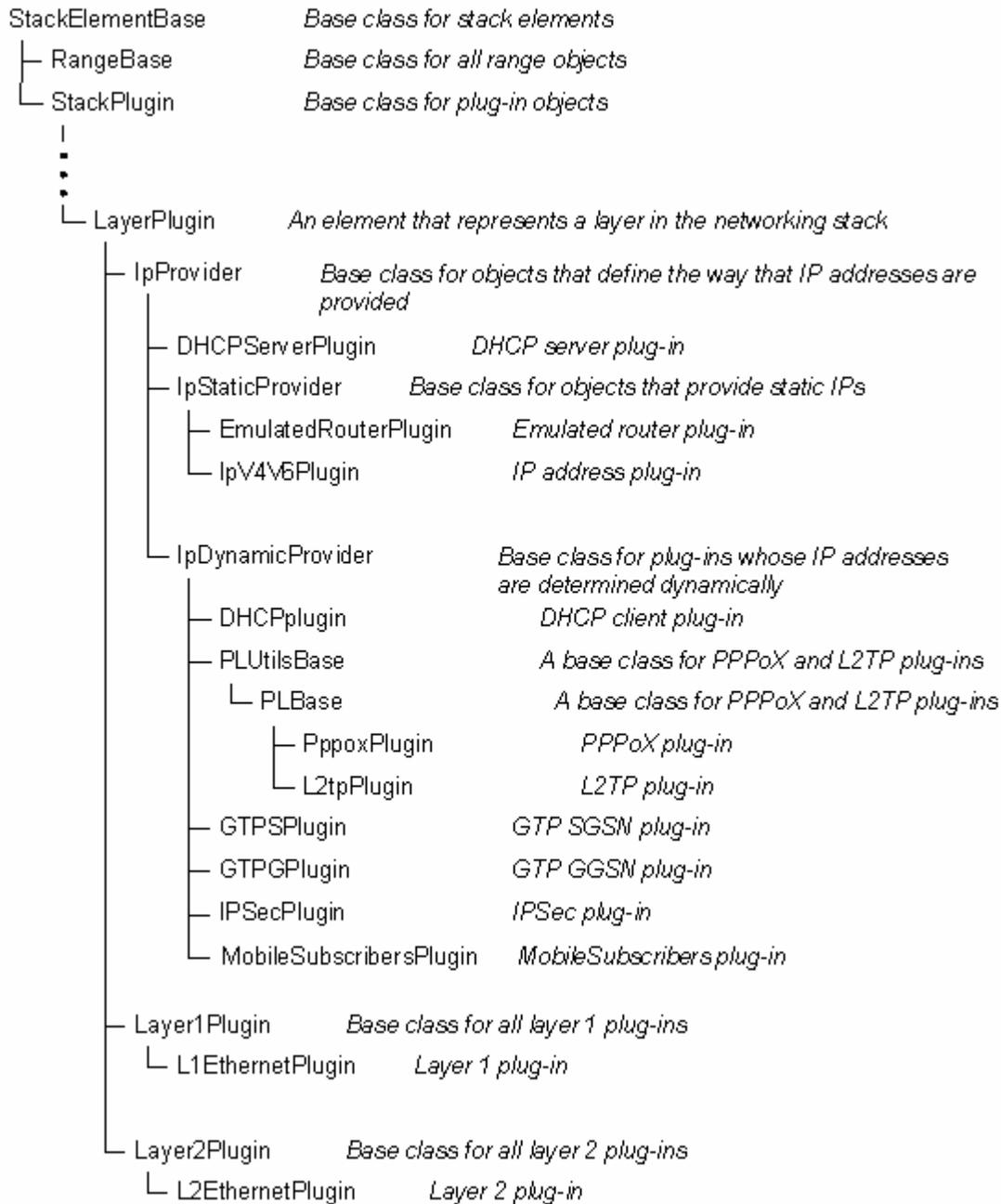
Network Stack Object Model - Base Classes Beginning with 'Se'



Network Stack Object Model - Base Classes Beginning with 'St'



Network Stack Object Model - Base Classes Beginning with 'St' (cont.)





Physical Layer

This section describes the Physical Layer plugin.

L1EthernetPlugin

SYNOPSIS

DESCRIPTION

First plugin for all Ethernet stacks. This element is preconfigured to be the first element of the stack in the Network Group. You can get this item from the network by calling `get NetworkPlugin`.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`autoNegotiate`

If `true`, the Ixia port auto-negotiates its speed and duplex operation with the DUT, using the values that you select for the `Speed` parameter. If `false`, the Ixia port uses the speed that you select for the `Speed` parameter.

Default value = "True"

`speed`

If `autoNegotiate` is `true`, this parameter lists the speeds that the Ixia port advertises. See `SpeedChoices` for the list of speeds.



Default value = "k100FD"

advertise10Half

If true, the Ixia port advertises 10 Mbps half duplex speed.

Default value = "True"

name="advertise10Full"

If true, the Ixia port advertises 10 Mbps full duplex speed.

Default value = "True"

name="advertise100Half"

If true, the Ixia port advertises 100 Mbps half duplex speed.

Default value = "True"

name="advertise100Full"

If true, the Ixia port advertises 100 Mbps full duplex speed.

Default value = "True"

name="advertise1000Full"

If true, the Ixia port advertises 1 Gbps full duplex speed.

Default value = "True"

name="cardElm"

Default value = "None"

EXAMPLE

SEE ALSO



SpeedChoices

SYNOPSIS

DESCRIPTION

List of valid speeds advertised for auto negotiation by the `speed` option of `LlEthernetPlugin`.

SUBCOMMANDS

OPTIONS

Value	Description
k10FD	10Mbit Full Duplex
k10HD	10Mbit Half Duplex
k100FD	100Mbit Full Duplex
k100HD	100Mbit Half Duplex
k1000	1 Gigabit
k10000	10 Gigabit

EXAMPLE

SEE ALSO

MediumChoices

SYNOPSIS

DESCRIPTION

List of valid physical media types for DualPhy Ethernet cards. See the `autoNegotiate` option of `LlEthernetPlugin`.

SUBCOMMANDS

OPTIONS

Value	Description
copper	Use copper mode.
fiber	Use fiber mode.
auto	Automatically select the media type .

EXAMPLE

SEE ALSO



EthernetELMPlugin

SYNOPSIS

DESCRIPTION

Defines the parameters for using an Encrypting Load Module (ELM) in an IxLoad test.

SUBCOMMANDS

OPTIONS

`enabled`

If `true`, an ELM port will be configured as an ELM port.

If `false`, an ELM port will be configured as a generic Ethernet port.

Default value = "False"

`negotiateMasterSlave`

If `true`, the master/slave relationship of all ports will be auto-negotiated.

If `false`, the `negotiationType` value determines the master/slave relationship.

Default value = "True"

`negotiationType`

If `negotiateMasterSlave` is `false`, this value determines the role (master or slave) of the ELM port. See `NegotiationTypeChoices` for the list of choices.

Default value = "master"

EXAMPLE

```
set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
  -negotiationType           "master" \
  -negotiateMasterSlave     true

$Ethernet_1 config \
  -advertise10Full          true \
  -name                     "Ethernet-1" \
  -autoNegotiate            true \
  -advertise100Half         true \
  -advertise10Half          true \
  -speed                    "k100FD" \
  -advertise1000Full        true \
  -advertise100Full         true \
  -cardElm                  $my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO



NegotiationTypeChoices

SYNOPSIS

DESCRIPTION

List of valid negotiation types for the `negotiationType` parameter in `EthernetELMPlugin`.

SUBCOMMANDS

OPTIONS

Value	Description
master	Port is master.
slave	Port is slave.

EXAMPLE

SEE ALSO

Physical Layer Example

This section shows an example of how to create a physical layer plugin in the Tcl API.

Physical Layer Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.10.0.79
# Basic.tcl made on Aug 14 2008 14:58
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              Create a network group. "Network1" \
    -macMappingMode   0 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear Clear the global plugins list.

Begin appending items to global plugin list.

set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
    -enabled          true \
    -name             "GratARP"
```



Physical Layer Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP

$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
    -name "TCP" \
```

Configure the TCP
portion of the stack.

Physical Layer Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior              0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                "master" \
    -negotiateMasterSlave           true

$Ethernet_1 config \
    -advertise10Full                true \
    -name                            "Ethernet-1" \
    -autoNegotiate                  true \
    -advertise100Half               true \
    -advertise10Half                true \
    -speed                          "k100FD" \
    -advertise1000Full              true \
    -advertise100Full               true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

$Ethernet_1 extensionList.clear

#####
# Setting the ranges starting with the plugin on top of the stack
#####
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.



Layer 2 Protocols

This section describes the Layer 2 protocol plugins.

L2EthernetPlugin

SYNOPSIS

DESCRIPTION

Plugin that describes the MAC and VLAN settings. This object appears as MAC/VLAN in the GUI.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`macRangeList`

Name of the list of MAC address ranges used by this plugin. The list must be a `MacRangeList` object.

Default value = "None"

`vlanRangeList`

Name of the list of VLAN tag ranges used by this plugin. The list must be a `VlanIdRangeList` object.

Default value = "None"

EXAMPLE

```
set MAC_VLAN_3 [::IxLoad new ixNetL2EthernetPlugin]

# ixNet objects needs to be added in the list before they are configured!

$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_3

$MAC_VLAN_3 config \
    -name "MAC/VLAN-3"

$MAC_VLAN_3 childrenList.clear
```

SEE ALSO

L2EthernetPortGroupData

SYNOPSIS

DESCRIPTION

Options for Layer 2 Ethernet port groups.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

EXAMPLE

SEE ALSO



MacSessionData

SYNOPSIS

DESCRIPTION

Global MAC settings for the `L2EthernetPlugin`.

SUBCOMMANDS

OPTIONS

`duplicateCheckingScope`

Value used to scope of check to determine whether IP is unique within the session, within the port group, or disabled.

See `eMacValidationOptions` of the choices.

Default value = "None"

EXAMPLE

SEE ALSO

eMacValidationOptions

SYNOPSIS

DESCRIPTION

List of values for `duplicateCheckingScope` option of `MacSessionData`.

SUBCOMMANDS

OPTIONS

Value	Description
0	Disabled
1	PortGroup
2	Session

EXAMPLE

SEE ALSO



MacRangeList

SYNOPSIS

DESCRIPTION

SUBCOMMANDS

OPTIONS

List of objects of `MacRange` type.

EXAMPLE

SEE ALSO

MacRange

SYNOPSIS

DESCRIPTION

Range of MAC addresses.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`mac`

The base value used when the network stack element creates MAC addresses. This address will be associated with the first interface on the port.

This parameter is available for editing only when the *AutoGenerate MAC* option is disabled (in the IP, DHCP Client, or Emulated Router configuration).

The default value is `aa:bb:cc:00:00:00`.

When you require a range of multiple MAC addresses, the network stack element uses this base address plus the *Increment By* value to create the range of addresses.

Default value (for API)=" " (none)

`incrementBy`

The value that is used (in conjunction with the base MAC address) to create a range of multiple MAC addresses.

The parameter is set to `00:00:00:00:00:01`. It is not configurable.



Default value=" '00:00:00:00:00:01' "

mtu

Maximum Transmission Unit (MTU) is the largest packet that a given network medium can carry.

Ethernet, for example, has a standard MTU of 1500 bytes, ATM has a fixed MTU of 48 bytes, and PPP has a negotiated MTU that is usually between 500 and 2000 bytes.

The default value is 1500, the minimum value is 500, and the maximum value is 9500.

Default value="1500".

count

Number of MAC addresses to create.

Default value="1" .

vlanRange

Name of the VLAN range associated with the MAC address.

Default value="None".

EXAMPLE

```
set MAC_R2 [${DHCP_R1 getLowerRelatedRange "MacRange"}

$MAC_R2 config \
  -count          1 \
  -name          "MAC-R2" \
  -enabled       true \
  -mtu           1500 \
  -mac           "22:73:F7:4E:00:00" \
  -incrementBy  "00:00:00:00:00:01"
```

SEE ALSO

VlanIdRangeList

SYNOPSIS

DESCRIPTION

List of objects of `VlanIdRange` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



VlanIdRange

SYNOPSIS

DESCRIPTION

Range of VLAN IDs.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`enable`

When enabled, the outer VLAN range is included in the configuration.

Default value="False"

`firstId`

The first VLAN ID to be used for the outer VLAN tag.

Default value="1"

`incrementStep`

The value to be added to the outer VLAN ID for each new assignment. The maximum value is 4093.

Default value="1"

increment

How often a new outer VLAN ID is generated. For example, a value of 10 will cause a new VLAN ID to be used in blocks of 10 IP addresses.

When using Inner First increment mode, this parameter determines how many inner cycles must be completed before a new outer VLAN ID is generated. For example if *Increment every...* is 2, a new outer VLAN ID is generated following two inner VLAN ID cycles.

(A cycle is complete when the Unique Count has been reached for inner VLAN IDs.)

Default value="1"

uniqueCount

The number of unique outer VLAN IDs that will be created. The default value is 4094.

Default value="4094"

priority

The 802.1Q priority for the outer VLAN. The minimum value is zero; the maximum value is 7.

Default value="1"

innerEnable

When enabled, the inner VLAN range is included in the configuration.

Inner VLAN cannot be enabled unless Outer VLAN is enabled.

Default value="False"

innerFirstId

The first VLAN ID to be used for the inner VLAN tag.

Default value="1"

innerIncrementStep

The value to be added to the inner VLAN ID for each new assignment. The maximum value is 4093.

DefaultValue="1"

innerIncrement

How often a new inner VLAN ID is generated. For example, a value of 10 will cause a new VLAN ID to be used in blocks of 10 IP addresses.



When using Outer First increment mode, this parameter determines how many outer cycles must be completed before a new inner VLAN ID is generated. For example if *Increment every...* is 2, a new inner VLAN ID is generated following two outer VLAN ID cycles.

(A cycle is complete when the Unique Count has been reached for outer VLAN IDs.)

Default value="1"

innerUniqueCount

The number of unique inner VLAN IDs that will be created. The default value is 4094.

Default value="4094"

innerPriority

The 802.1Q priority for the inner VLAN. The minimum value is zero; the maximum value is 7.

Default value="1"

idIncrMode

The Method used to increment VLAN IDs:

- Outer VLAN first - The outer VLAN ID is incremented first. When the Unique Count is reached the number of times specified by the *Increment every...* parameter, the inner VLAN ID is incremented.
- Inner VLAN first - The inner VLAN ID is incremented first. When the Unique Count is reached the number of times specified by the *Increment every...* parameter, the outer VLAN ID is incremented.
- Both - Both VLAN IDs are incremented at the same time.

Refer to VLAN Increment Examples for more information about VLAN increment modes.

Default value="2"

EXAMPLE

```
set VLAN_R1 [$DHCP_R1 getLowerRelatedRange "VlanIdRange"]
```

```
$VLAN_R1 config \  
-incrementStep          1 \  
-uniqueCount            4094 \  
-name                   "VLAN-R1" \  
-innerIncrement         1 \  
-innerUniqueCount       4094 \  
-enabled                true \  
-innerFirstId           1 \  
-increment              1 \  
-priority               1 \  
-firstId                1 \  
-innerIncrementStep     1 \  
-idIncrMode             2 \  
-innerEnable            false \  
-innerPriority           1
```

SEE ALSO



Layer 2 Example

This section shows an example of how to create a layer 2 plugin in the Tcl API.

Layer 2 Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.10.0.79
# MAC VLAN.tcl made on Aug 14 2008 15:16
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              "Network1" \
    -macMappingMode   0 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear

Begin appending items to global plugin list.

set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
    -enabled          true \
    -name              "GratARP"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.

Layer 2 Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

Configure the TCP
portion of the stack.

```
$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
```



Layer 2 Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior               0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                 "master" \
    -negotiateMasterSlave            true

$Ethernet_1 config \
    -advertise10Full                 true \
    -name                             "Ethernet-1" \
    -autoNegotiate                    true \
    -advertise100Half                 true \
    -advertise10Half                  true \
    -speed                             "k100FD" \
    -advertise1000Full                 true \
    -advertise100Full                  true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.

Layer 2 Example

```
set MAC_VLAN_2 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_2

$MAC_VLAN_2 config \
    -name "MAC/VLAN-2"

$MAC_VLAN_2 childrenList.clear
$MAC_VLAN_2 extensionList.clear
$Ethernet_1 extensionList.clear

#####
# Setting the ranges starting with the plugin on top of the stack
#####
$MAC_VLAN_2 macRangeList.clear

$MAC_VLAN_2 vlanRangeList.clear
```

Configure the MAC addresses and VLAN tags.

Clear the lists of extension protocols.

Emulated Router Plugin

SYNOPSIS

DESCRIPTION

Used over `L2EthernetPlugin` to define an emulated router.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`EmulatedRouterRangeList`

List of `EmulatedRouterRange` objects.

Default value="None".

EXAMPLE

SEE ALSO



EmulatedRouterRangeList

SYNOPSIS

DESCRIPTION

List of emulated router ranges. The list contains items that are of `EmulatedRouterRange` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

EmulatedRouterRange

SYNOPSIS

DESCRIPTION

Defines a range of IP addresses that will be used by an emulated router.

You need to assign one port per address to this range.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`ipType`

Indicates the IP version for each range:

- IPv4
- IPv6

The default value is IPv4.

Must be one of the choices in `IpTypeChoices`. Default value=" 'IPv4' ".

`ipAddress`

The first IP address in the range. This is the base address used for enumerating all the addresses in the range.

The default IPv4 address is 10.10.10.2, and the default IPv6 address is ::A0A:A02.



You cannot assign IP addresses in the 10.0.0.0 /16 range: IxOS reserves this range of addresses for use in the Ixia chassis VNIC network. If you attempt to configure IP addresses from subnet 10.0.x.x, IxLoad will reject the address assignment.

Default value= " '10.10.0.1' " .

`prefix`

The number of one bits in the subnet mask. For example, a mask of 255.255.240.0 has a prefix of 20.

The default IPv4 value is 24, and the default IPv6 value is 96.

Default value= " 16 " .

`incrementBy`

The value used to enumerate all the addresses in the range.

The default IPv4 value is 0.0.0.1. The default IPv6 value is ::1.

Default value= " 1 " .

`gatewayAddress`

The gateway address to be associated with all the addresses in the range.

If the Protocol is IPv6, the Gateway field adds a default route to this gateway for the range (unless the gateway is 0 : : 0, in which case the route is not added).

The default IPv4 value is 10.10.10.1. The default IPv6 value is ::A0A:A01.

Note: When you configure an Emulated Router in an IP stack, the Emulated Router provides the gateway addresses for the IP ranges. In this case, the IP gateway parameters are not configurable.

Default value= " '0.0.0.0' " .

`gatewayIncrement`

Defines the address increment value that is used to generate each gateway address required in the network group. (The gateway addresses are incremented according to the *Gateway Increment Mode*.)

The default IPv4 value is 0.0.0.0, and the default IPv6 value is ::0. When the default value is used, the base gateway address will not be incremented. Rather, all gateway IPs will be the same for all interfaces generated by the plug-in range.

Default value= " 0.0.0.0 " .

`gatewayIncrementMode`

Determines when the gateway addresses are incremented. The options are:

- Increment every subnet: A new gateway address is created for each subnet defined in the network group. With this mode, the increment operation is triggered when a range IP increment operation creates an IP address that is in a new subnet.
- Increment every interface: A new gateway address is created for each interface, whether or not the next address is from the same subnet.

The default is *Increment Every Subnet*.

Refer to Static IP Plug-in Gateway Addresses for more information.

Must be one of the choices in `GatewayIncrementModeChoices`. Default value= " 'perSubnet' ".

`generateStatistics`

Determines whether or not the per-session statistics for this range will be published.

Values=True/False. Default value="False".

`mss`

The Maximum Segment Size. The MSS is the largest amount of data, specified in bytes, that the IP device can transmit as a single, unfragmented unit.

The TCP MSS equals the MTU minus the TCP header size minus the IP header size.

IxLoad supports jumbo frames. Therefore the maximum value is 9460 (9500 minus 40).

The default value is 1460.

Default value="1460".

`autoMacGeneration`

This parameter is used to automatically generate MAC addresses:

- If enabled, MAC addresses will be automatically generated based on the IP addresses, in which case the associated MAC range is ignored.

If disabled, the associated MAC range is used to create the MAC addresses.

Values=True/False. Default value="True".

`macRange`

Name of the MAC range. Must be one of the choices in `MacRange`.

Default value="None".

`vlanRange`

Name of the VLAN range. Must be one of the choices in `VlanIdRange`.

Must be one of the choices in `VlanIdRange`. Default value="None".

EXAMPLE

```
set ER_R1 [::IxLoad new ixNetEmulatedRouterRange]

# ixNet objects needs to be added in the list before they are configured!

$Emulated_Router_1 rangeList.appendItem -object $ER_R1

$ER_R1 config \

    -count                1 \

    -name                  "ER-R1" \

    -gatewayAddress        "0.0.0.0" \

    -enabled               true \

    -autoMacGeneration     true \

    -mss                   1460 \

    -incrementBy           "0.0.0.1" \

    -prefix                16 \

    -gatewayIncrement      "0.0.0.0" \

    -gatewayIncrementMode  "perSubnet" \

    -generateStatistics    false \

    -ipAddress             "10.10.0.3" \

    -ipType                 "IPv4"
```

SEE ALSO

Emulated Router Example

This section shows an example of how to create an Emulated Router in the Tcl API.

Emulated Router Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.10.0.79
# EmulatedRouter.tcl made on Aug 21 2008 14:01
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              "Network1" \
    -macMappingMode   1 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear
# Clear the global plugins list.

# Begin appending items to global plugin list.

set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
    -enabled          true \
    -name              "GratARP"
```



Emulated Router Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the
# list before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

Configure the TCP
portion of the stack.

```
$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
    -name "TCP" \
```

Emulated Router Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior               0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                 "master" \
    -negotiateMasterSlave            true

$Ethernet_1 config \
    -advertise10Full                 true \
    -name                             "Ethernet-1" \
    -autoNegotiate                    true \
    -advertise100Half                true \
    -advertise10Half                  true \
    -speed                             "k100FD" \
    -advertise1000Full                true \
    -advertise100Full                 true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

set MAC_VLAN_2 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_2

$MAC_VLAN_2 config \
    -name                             "MAC/VLAN-2"

$MAC_VLAN_2 childrenList.clear
```

Configure the Dynamic Control plane settings.

Configure the physical layer properties.

Configure the MAC addresses and VLAN tags.



Emulated Router Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior               0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                 "master" \
    -negotiateMasterSlave            true

$Ethernet_1 config \
    -advertise10Full                 true \
    -name                             "Ethernet-1" \
    -autoNegotiate                    true \
    -advertise100Half                 true \
    -advertise10Half                  true \
    -speed                             "k100FD" \
    -advertise1000Full                true \
    -advertise100Full                 true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

set MAC_VLAN_2 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_2

$MAC_VLAN_2 config \
    -name                             "MAC/VLAN-2"

$MAC_VLAN_2 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.

Configure the MAC
addresses and VLAN tags.

Emulated Router Example

```
set Emulated_Router_1 [::IxLoad new ixNetEmulatedRouterPlugin]
# ixNet objects needs to be added in the
# list before they are configured!
$MAC_VLAN_2 childrenList.appendItem -object $Emulated_Router_1

$Emulated_Router_1 config \
    -name "Emulated Router-1"

$Emulated_Router_1 childrenList.clear
$Emulated_Router_1 extensionList.clear
$MAC_VLAN_2 extensionList.clear
$Ethernet_1 extensionList.clear

#####
# Setting the ranges starting with the plugin on top of the stack
#####
$Emulated_Router_1 rangeList.clear
set ER_R1 [::IxLoad new ixNetEmulatedRouterRange]
# ixNet objects needs to be added in the list
# before they are configured!
$Emulated_Router_1 rangeList.appendItem -object $ER_R1

$ER_R1 config \
    -count 1 \
    -name "ER-R1" \
    -gatewayAddress "0.0.0.0" \
    -enabled true \
    -autoMacGeneration true \
    -mss 1460 \
    -incrementBy "0.0.0.1" \
    -prefix 16 \
    -gatewayIncrement "0.0.0.0" \
    -gatewayIncrementMode "perSubnet" \
    -generateStatistics false \
    -ipAddress "10.10.0.3" \
    -ipType "IPv4"
```

Configure an Emulated Router plugin.

Clear the lists of protocol extensions.

Configure an address range for the Emulated Router.



IP Plugin

This section describes the IP protocol plugin.

IpRangePortGroupData

SYNOPSIS

DESCRIPTION

Options for IP ranges within a specific port group.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

EXAMPLE

SEE ALSO



IpSessionData

SYNOPSIS

DESCRIPTION

Configures the IP global settings.

SUBCOMMANDS

OPTIONS

Same as `SessionSpecificData` plus the following:

`duplicateCheckingScope`

Value used to scope of check to determine whether IP is unique within the session, within the port group, or disabled.

See `eIPValidationOptions` for the list of values.

Default value="None"

EXAMPLE

SEE ALSO

elpValidationOptions

SYNOPSIS

DESCRIPTION

List of values for `duplicateCheckingScope` option of `IpSessionData`.

SUBCOMMANDS

OPTIONS

Value	Description
0	Disabled
1	Port Group
2	Session

EXAMPLE

SEE ALSO



IpV4V6Plugin

SYNOPSIS

DESCRIPTION

Layer 3 plugin that provides IPv4/IPv6 address ranges.

SUBCOMMANDS

OPTIONS

From `IpStaticProvider`:

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`rangeList`

Name of the IP range. This parameter is read-only. Default value="None".

EXAMPLE

```
set IP_2 [::IxLoad new ixNetIpV4V6Plugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_7 childrenList.appendItem -object $IP_2

$IP_2 config \
    -name "IP-2"

$IP_2 childrenList.clear

$IP_2 extensionList.clear

$MAC_VLAN_7 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO



IpV4V6RangeList

SYNOPSIS

DESCRIPTION

List of IP ranges. The list contains items that are of `IpV4V6Range` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

IpV4V6Range

SYNOPSIS

DESCRIPTION

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`ipType`

Indicates the IP version for each range:

- IPv4
- IPv6

The default value is IPv4.

Must be one of the choices in `IpTypeChoices`. Default value=" 'IPv4' ".

`ipAddress`

The first IP address in the range. This is the base address used for enumerating all the addresses in the range.

The default IPv4 address is 10.10.10.2, and the default IPv6 address is ::A0A:A02.

You cannot assign IP addresses in the 10.0.0.0 /16 range: IxOS reserves this range of addresses for use in the Ixia chassis VNIC network. If you attempt to configure IP addresses from subnet 10.0.x.x, IxLoad will reject the address assignment.

Default value=" " (none).



`prefix`

The number of one bits in the subnet mask. For example, a mask of 255.255.240.0 has a prefix of 20.

The default IPv4 value is 24, and the default IPv6 value is 96.

Default value="16".

`incrementBy`

The value used to enumerate all the addresses in the range.

The default IPv4 value is 0.0.0.1. The default IPv6 value is ::1.

Default value="0.0.0.1".

`count`

The number of addresses to be created for this range.

The default value is 1.

Default value="1".

`gatewayAddress`

The gateway address to be associated with all the addresses in the range.

If the Protocol is IPv6, the Gateway field adds a default route to this gateway for the range (unless the gateway is 0::0, in which case the route is not added).

The default IPv4 value is 10.10.10.1. The default IPv6 value is ::A0A:A01.

Note: When you configure an Emulated Router in an IP stack, the Emulated Router provides the gateway addresses for the IP ranges. In this case, the IP gateway parameters are not configurable.

Default value="0.0.0.0".

`gatewayIncrement`

Defines the address increment value that is used to generate each gateway address required in the network group. (The gateway addresses are incremented according to the *Gateway Increment Mode*.)

The default IPv4 value is 0.0.0.0, and the default IPv6 value is ::0. When the default value is used, the base gateway address will not be incremented. Rather, all gateway IPs will be the same for all interfaces generated by the plug-in range.

Default value="0.0.0.0"

`gatewayIncrementMode`

Determines when the gateway addresses are incremented. The options are:

- Increment every subnet: A new gateway address is created for each subnet defined in the network group. With this mode, the increment operation is triggered when a range IP increment operation creates an IP address that is in a new subnet.
- Increment every interface: A new gateway address is created for each interface, whether or not the next address is from the same subnet.

The default is *Increment Every Subnet*.

Refer to Static IP Plug-in Gateway Addresses for more information.

Must be one of the choices in `GatewayIncrementModeChoices`. Default value= " 'perSubnet' ".

`generateStatistics`

Determines whether or not the per-session statistics for this range will be published.

Values=True/False. Default value="False".

`mss`

The Maximum Segment Size. The MSS is the largest amount of data, specified in bytes, that the IP device can transmit as a single, unfragmented unit.

The TCP MSS equals the MTU minus the TCP header size minus the IP header size.

IxLoad supports jumbo frames. Therefore the maximum value is 9460 (9500 minus 40).

The default value is 1460.

Default value="1460".

`autoMacGeneration`

This parameter is used to automatically generate MAC addresses:

- If enabled, MAC addresses will be automatically generated based on the IP addresses, in which case the associated MAC range is ignored.

If disabled, the associated MAC range is used to create the MAC addresses.

Values=True/False. Default value="True".

`macRange`

Name of the MAC range. Must be one of the choices in `MacRange`.

Default value="None".

`vlanRange`

Name of the VLAN range. Must be one of the choices in `VlanIdRange`.

Must be one of the choices in `VlanIdRange`. Default value="None".

EXAMPLE

```
$IP_2 rangeList.clear

set IP_R2 [::IxLoad new ixNetIpV4V6Range]
# ixNet objects needs to be added in the list before they are configured!
$IP_2 rangeList.appendItem -object $IP_R2

$IP_R2 config \
  -count                1 \
  -name                  "IP-R2" \
  -gatewayAddress        "0.0.0.0" \
  -enabled                true \
  -autoMacGeneration     true \
  -mss                   1460 \
  -incrementBy           "0.0.0.1" \
  -prefix                16 \
  -gatewayIncrement      "0.0.0.0" \
  -gatewayIncrementMode  "perSubnet" \
  -generateStatistics    false \
  -ipAddress              "10.10.0.2" \
  -ipType                 "IPv4"
```

SEE ALSO

IpTypeChoices

SYNOPSIS

DESCRIPTION

Indicates the IP version for each range:

- IPv4
- IPv6

The default value is IPv4.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



GatewayIncrementModeChoices

SYNOPSIS

DESCRIPTION

Determines when the gateway addresses are incremented. The options are:

- Increment every subnet: A new gateway address is created for each subnet defined in the network group. With this mode, the increment operation is triggered when a range IP increment operation creates an IP address that is in a new subnet.
- Increment every interface: A new gateway address is created for each interface, whether or not the next address is from the same subnet.

The default is *Increment Every Subnet*.

Refer to Static IP Plug-in Gateway Addresses for more information.

For the API, the choices are:

- "perSubnet "
- "perInterface "

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

IP Plugin Example

This section shows an example of how to create a IP plugin in the Tcl API.

LOD_THY_IP Plugin Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.10.0.79
# IP.tcl made on Aug 21 2008 14:04
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              "Network1" \
    -macMappingMode   1 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear

set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
    -enabled          true \
    -name              "GratARP"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.



IP Plugin Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

Configure the TCP
portion of the stack.

```
$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
```

IP Plugin Example

```

set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the
# list before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior               0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new
ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                "master" \
    -negotiateMasterSlave           true

$Ethernet_1 config \
    -advertise10Full                true \
    -name                            "Ethernet-1" \
    -autoNegotiate                   true \
    -advertise100Half               true \
    -advertise10Half                true \
    -speed                           "k100FD" \
    -advertise1000Full              true \
    -advertise100Full               true \
    -cardElm                        

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

set MAC_VLAN_2 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_2

$MAC_VLAN_2 config \
    -name                            "MAC/VLAN-2"

$MAC_VLAN_2 childrenList.clear

```

Configure the Dynamic Control plane settings.

Configure the physical layer properties.

Configure the MAC addresses and VLAN tags.



IP Plugin Example

```
set IP_3 [::IxLoad new ixNetIpV4V6Plugin]
# ixNet objects needs to be added in the list
# before they are configured!
$MAC_VLAN_2 childrenList.appendItem -object $IP_3

$IP_3 config \
    -name "IP-3"

$IP_3 childrenList.clear
$IP_3 extensionList.clear

$MAC_VLAN_2 extensionList.clear

$Ethernet_1 extensionList.clear

#####
# Setting the ranges starting with the plugin on top of the stack
#####
$IP_3 rangeList.clear

set IP_R3 [::IxLoad new ixNetIpV4V6Range]
# ixNet objects needs to be added in the
# list before they are configured!
$IP_3 rangeList.appendItem -object $IP_R3

$IP_R3 config \
    -count 1 \
    -name "IP-R3" \
    -gatewayAddress "0.0.0.0" \
    -enabled true \
    -autoMacGeneration true \
    -mss 1460 \
    -incrementBy "0.0.0.1" \
    -prefix 16 \
    -gatewayIncrement "0.0.0.0" \
    -gatewayIncrementMode "perSubnet" \
    -generateStatistics false \
    -ipAddress "10.10.0.4" \
    -ipType "IPv4"
```

Clear the lists of extension protocols.

Configure an IP address range.

IP Plugin Example

```
set MAC_R2 [$IP_R3 getLowerRelatedRange "MacRange"]

$MAC_R2 config \
  -count          1 \
  -name           "MAC-R2" \
  -enabled        true \
  -mtu            1500 \
  -mac            "00:0A:0A:00:04:00" \
  -incrementBy    "00:00:00:00:00:01"

set VLAN_R1 [$IP_R3 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
  -incrementStep  1 \
  -uniqueCount    4094 \
  -name           "VLAN-R1" \
  -innerIncrement 1 \
  -innerUniqueCount 4094 \
  -enabled        true \
  -innerFirstId   1 \
  -increment      1 \
  -priority       1 \
  -firstId        1 \
  -innerIncrementStep 1 \
  -idIncrMode     2 \
  -innerEnable    false \
  -innerPriority   1
```

Configure the MAC addresses for the IP range.

Configure the VLAN tags for the IP range.



DHCP Client and Server

This section describes the DHCP client and server plugins.

DHCPPlugin

SYNOPSIS

DESCRIPTION

DHCP client.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`rangeList`

List of DHCP ranges. New elements can be added to the using `appendItem`. The elements of the list can be modified, but the list cannot be replaced.

Default value="None".

EXAMPLE

```
set DHCP_Client_1 [::IxLoad new ixNetDHCPlugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_3 childrenList.appendItem -object $DHCP_Client_1

$DHCP_Client_1 config \
    -name "DHCP Client-1"

$DHCP_Client_1 childrenList.clear
$DHCP_Client_1 extensionList.clear

$MAC_VLAN_3 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO

DHCPSessionData

SYNOPSIS

DESCRIPTION

Configures the DHCP client Global General parameters.

SUBCOMMANDS

OPTIONS

`waitForCompletion`

If true, the application waits for a specified amount of time for the sessions to negotiate before declaring a negotiation timeout.

Default value="True"

`acceptPartialConfig`

Default value = "False"

`optionSets`

Name of the list of options used by this plugin. The list must be a `DHCPOptionSetList` object.

Default value = "None"

`setupRateInitial`

Setup rate is the number of clients to start each second. This value is the initial setup rate.

The default is 10; the minimum is equal to the number of stacks defined for the ; and the maximum is 100,000.

Default value = "10"

`setupRateIncrement`

The number by which the request rate is incremented each second. Negative values are permitted.



The default is 0, the minimum is negative 100,000, and the maximum is 100,000.

Default value = "0"

`setupRateMax`

The maximum number of addresses to set up per second. The setup rate will not change after this value is reached.

The default is 50; the minimum is equal to the number of stacks defined for the network group; and the maximum is 100,000.

Default value = "50"

`maxOutstandingRequests`

The maximum number of requests to be sent by all DHCP clients during session startup. This is the threshold at which the DHCP Client plug-in begins throttling back the number of new clients being set up.

The default is 50, the minimum is 1, and the maximum is 100,000.

Default value = "50"

`teardownRateInitial`

Teardown rate is the number of clients to stop each second. This value is the initial teardown rate.

The default is 50; the minimum is equal to the number of stacks defined for the network group; and the maximum is 100,000.

Default value = "50"

`teardownRateIncrement`

The number by which the teardown rate is incremented each second. Negative values are permitted.

The default is 50, the minimum is negative 100,000, and the maximum is 100,000.

Default value = "50"

`teardownRateMax`

The maximum number of addresses to release per second. The teardown rate will not change after this value is reached.

The default is 500; the minimum is equal to the number of stacks defined for the network group; and the maximum is 100,000.

Default value = "500"

`maxOutstandingReleases`

The maximum number of requests to be sent by all DHCP clients during session teardown.

The default is 500, the minimum is 1, and the maximum is 100,000.

Default value = "500"

`dhcp4ResponseTimeout`

The initial time, in seconds, that the subnet waits to receive a response from a DHCP server.

The default value is 4, the minimum is 1, and the maximum is 65,535.

Default value = "4"

`dhcp4ResponseTimeoutFactor`

The value by which the timeout will be multiplied each time the response timeout has been reached. For example, if a DHCPDISCOVER times out in 4 seconds, and the Discover Timeout Factor is 2, the next time a DHCPDISCOVER will be sent, the timeout will be 8 seconds (4 times 2).

The default value is 2, the minimum is 1, and the maximum is 100.

Default value = "2"

`dhcp4NumRetry`

Number of times that the client will retransmit a request for which it has not received a response.

When the maximum number of retransmissions is reached, the port will increment the failure counter (DHCPSetupFail).

The default value is 3, the minimum is 0, and the maximum is 100.

Default value = "3"

`dhcp4ClientPort`

The UDP port on which the client listens for DHCP and BOOTP responses.

The default value port number is 68.

Default value = "68"

`dhcp4ServerPort`

The UDP port to which the client addresses its server requests.

The default value port number is 67.

Default value = "67"

`dhcp4MaxMsgSize`



The maximum size of a DHCP packet that the client will send or accept, including IP and UDP headers.

According to RFC 2131, the minimum message size that a client should accept is 576 octets.

The default value is 576, the minimum is 264, and the maximum is 1,500.

Default value = "576"

`dhcp4AddrLeaseTime`

Period of time (in seconds) for which an IP address is requested by DHCP client.

The actual lease time is specified by the DUT in the DHCPACK message.

The default value is 3600, the minimum is 10, and the maximum is 2,147,483,647.

Default value = "3600"

`dhcp6EchoIAInfo`

If `true`, the DHCPv6 client will send Request messages to any server that has the same IA information (addresses and options) as presented in the previous Advertisement message.

Default value = "False"

`dhcp6SolTimeout`

The Initial solicit timeout, in seconds.

The default value is 1, the minimum is 1, and the maximum is 100.

Default value = "1"

`dhcp6SolMaxRt`

The maximum solicit timeout value, in seconds.

The default value is 120, the minimum is 1, and the maximum is 10,000.

Default value = "120"

`dhcp6SolMaxRc`

The maximum solicit retry attempts.

The default value is 3, the minimum is 1, and the maximum is 100.

Default value = "3"

`dhcp6ReqTimeout`

The initial request timeout value, in seconds.

The default value is 1, the minimum is 1, and the maximum is 100.

Default value = "1"

`dhcp6ReqMaxRt`

The maximum request timeout value, in seconds.

The default value is 30, the minimum is 1, and the maximum is 10,000.

Default value = "30"

`dhcp6ReqMaxRc`

The maximum request retry attempts.

The default value is 10, the minimum is 1, and the maximum is 100.

Default value = "10"

`dhcp6RenTimeout`

The Initial Renew timeout, in seconds.

The default value is 10, the minimum is 1, and the maximum is 100.

Default value = "10"

`dhcp6RenMaxRt`

The Max Renew timeout value, in seconds.

The default value is 600, the minimum is 1, and the maximum is 10,000.

Default value = "600"

`dhcp6RebTimeout`

The initial rebind timeout., in seconds.

The default value is 10, the minimum is 1, and the maximum is 100.

Default value = "10"

`dhcp6RebMaxRt`

The maximum rebind timeout value, in seconds.



The default value is 600, the minimum is 1, and the maximum is 10,000.

Default value = "600"

`dhcp6RelTimeout`

The initial release timeout, in seconds.

The default value is 1, the minimum is 1, and the maximum is 100.

Default value = "1"

`dhcp6RelMaxRc`

The number of release attempts.

The default value is 5, the minimum is 1, and the maximum is 100.

Default value = "5"

EXAMPLE

SEE ALSO

DHCPPortGroupData

SYNOPSIS

DESCRIPTION

DHCP options for the port group.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`associates`

Name of the list of associates. This list must an `AssociateList` object.

This option is read only.

Default value = "None"

`overrideGlobalSetupRate`

If true, all the rate settings defined at the global level will be overridden by the rate settings defined on this port.

For example, if you have set the initial setup rate to 100 on the global level, and you have defined two ports, these 100 session setups will be evenly distributed across the ports (50 for each). If you then enable Override Global Setup Rate, you can modify the number of session setups for each of the ports (such as changing the distribution from 50-50 to 60-40).

Default value = "False"



`setupRateInitial`

Setup rate is the number of clients to start each second. This value is the initial setup rate.

The default is 10; the minimum is equal to the number of stacks defined for the ; and the maximum is 100,000.

Default value = "10"

`setupRateIncrement`

The number by which the request rate is incremented each second. Negative values are permitted.

The default is 0, the minimum is negative 100,000, and the maximum is 100,000.

Default value = "0"

`setupRateMax`

The maximum number of addresses to set up per second. The setup rate will not change after this value is reached.

The default is 50; the minimum is equal to the number of stacks defined for the network group; and the maximum is 100,000.

Default value = "10"

`maxOutstandingRequests`

The maximum number of requests to be sent by all DHCP clients during session startup. This is the threshold at which the DHCP Client plug-in begins throttling back the number of new clients being set up.

The default is 50, the minimum is 1, and the maximum is 100,000.

Default value = "50"

`overrideGlobalTeardownRate`

If selected, all the rate settings defined at the session level will be overridden by rate settings defined on this port.

For example, if you have set the initial teardown rate to 100 on the global level, and you have defined two ports, these 100 session teardowns will be evenly distributed across the ports (50 for each). If you then enable Override Global Teardown Rate, you can modify the number of session teardowns for each of the ports (such as changing the distribution from 50-50 to 75-25).

Default value = "False"

`teardownRateInitial`

Teardown rate is the number of clients to stop each second. This value is the initial teardown rate.

The default is 50; the minimum is equal to the number of stacks defined for the network group; and the maximum is 100,000.

Default value = "50"

`teardownRateIncrement`

The number by which the teardown rate is incremented each second. Negative values are permitted.

The default is 50, the minimum is negative 100,000, and the maximum is 100,000.

Default value = "50"

`teardownRateMax`

The maximum number of addresses to release per second. The teardown rate will not change after this value is reached.

The default is 500; the minimum is equal to the number of stacks defined for the network group; and the maximum is 100,000.

Default value = "500"

`maxOutstandingReleases`

The maximum number of requests to be sent by all DHCP clients during session teardown.

The default is 500, the minimum is 1, and the maximum is 100,000.

Default value = "500"

EXAMPLE

SEE ALSO



DHCPRangeList

SYNOPSIS

DESCRIPTION

List of DHCP ranges. The list contains items that are of `DHCPRange` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

DHCPRange

SYNOPSIS

DESCRIPTION

Configures an IP/DHCP range of addresses.

SUBCOMMANDS

OPTIONS

name

Name of the instance of the plugin.

Default value = "None"

enabled

If `True`, the range base is enabled.

Default value="True".

macRange

defaultValue="None"

vlanRange

Default Value = "None"

count

The total number of IP addresses to be created for the range.

The default value is 1, the minimum value is 1, and the maximum value is 1,000,000.

Default Value = "1"



`ipType`

Defines the IP address version to be used for the range:

- IPv4
- IPv6

The default value is IPv4.

Default Value = "IPv4"

`useVendorClassId`

Enables or disables the use of the Vendor Class Identifier. When enabled, you configure the Identifier in the Vendor Class Id field.

The default value is False.

Default Value = "False"

`vendorClassId`

DHCP clients use the Vendor Class Identifier (Option 60) to identify the vendor and functionality of a DHCP client. The identifier has a meaning specified by the vendor of the DHCP client.

The identifier is either a variable length string or a hexadecimal value (hexadecimal values must begin with 0x).

The default value is "Ixia DHCP Client".

Default Value = "Ixia DHCP Client"

`renewTimer`

When an address is allocated or reallocated, the client starts two timers that control the renewal process. The renewal process is designed to ensure that a client's lease can be extended before it is scheduled to end. When the Renewal Timer (T1) expires, the client starts the process of renewing the lease.

The DHCP client will use either its own Renewal Timer value or the Renewal Timer value specified by the DHCP Server, whichever is lower.

The default value is 0, the minimum value is 0, and the maximum value is 2,147,483,647.

This value is ignored if it is zero.

Default Value = "0"

`clientOptionSet`

Identifies a DHCP Option Set that contains the DHCP extensions that will be used by the clients created with this DHCP range.

You define DHCP Option Sets in the Network Plug-in Settings window.

Default Value = "None"

dhcp4ParamRequestList

A list of parameters that a DHCPv4 client wants to obtain from a server. The available client request options are:

- 1 - Subnet Mask
- 3 - Router
- 6 - Domain Name Server
- 15 - Domain Name
- 33 - Static Route
- 44 - NetBIOS over TCP/IP Name Server
- 46 - NetBIOS over TCP/IP Name Type
- 51 - IP Address Lease Time
- 54 - Server Identifier
- 58 - Renewal (T1) Time Value
- 59 - Rebinding (T2) Time Value
- 90 - Authentication (RFC 3118)

The default list comprises the following options:
1; 3; 58; 59.

Default Value = "1; 3; 58; 59"

dhcp4UseFirstServer

If enabled, the subnet accepts the IP addresses offered by the first server to respond to the DHCPDISCOVER message.

This parameter is enabled by default.

Default Value = "True"

dhcp4ServerAddress

The address of the DHCP server from which the subnet will accept IP addresses.

If you set the server address to 0.0.0.0, IxLoad uses the services of the first DHCP server that responds to the DHCP Lease Request message.

The default value is 10.0.0.1.

Default Value = "10.0.0.1"

dhcp6MasterRange

The DHCP-PD range whose negotiated prefix will be used by this range to configure its addresses.

This parameter is valid only when *IA Type* is set to IAPD. When this is the case, select the range (such as *dhcp-1*).

Default Value = "None"



dhcp6ParamRequestList

The Option Request field is used to identify a list of options in a message between a client and a server. The available client request options are:

- 2 - Server Identifier
- 7 - Preference
- 11 - Authentication
- 12 - Server Unicast
- 13 - Status Code
- 17 - Vendor-specific Information
- 18 - Interface-ID
- 23 - DNS Resolvers
- 24 - DNS List

The default list comprises the following options:
2; 7; 23; 24.

Default Value = "2; 7; 23; 24"

dhcp6DuidType

DHCP Unique Identifier (DUID) Type. Each DHCPv6 client and server has a DUID. DHCP servers use DUIDs to identify clients for the selection of configuration parameters and in the association of IAs with clients. DHCP clients use DUIDs to identify a server in messages in which a server must be identified.

The DUID types are:

- DUID-LLT (Link-layer address plus time)
- DUID-EN (Vendor-assigned unique ID based on Enterprise Number)
- DUID-LL (Link-layer address)

The default value is DUID-LLT.

Default Value = "DUID-LLT"

dhcp6DuidEnterpriseId

The vendor's registered Private Enterprise Number maintained by IANA.

This parameter is enabled only when the *DUID Type* is DUID-EN.

The default value is 10, the minimum is 1, and the maximum is 2147483647.

Default Value = "10"

dhcp6DuidVendorId

A unique identifier defined by the vendor. This ID is incremented automatically for each DHCP client.

This parameter is enabled only when the *DUID Type* is DUID-EN.

The default value is 10, the minimum is 1, and the maximum is 2147483647.

Default Value = "10"

dhcp6DuidVendorIdIncrement

The value by which the Vendor ID is incremented for each DHCP client.

This parameter is enabled only when the *DUID Type* is DUID-EN.

The default value is 1, the minimum is 1, and the maximum is 2147483647.

Default Value = "1"

dhcp6IaType

The Identity Association Type. An identify association is a construct through which a server and a client identify, group, and manage a set of related IPv6 addresses. Each IA has an associated IA ID. A client may have more than one IA assigned with it (such as one for each of its interfaces).

The IA types are:

- IANA – Identity Association for Non-temporary Addresses (IA_NA) is an IA that carries assigned addresses that are not temporary addresses.
- IATA – Identity Association for Temporary Addresses (IA_TA) is an IA that carries temporary addresses.
- IAPD – Identity Association for Prefix Delegation (IA_PD) is a collection of prefixes assigned to the requesting router. Each IA_PD has an associated IAID. A requesting router may have more than one IA_PD assigned to it (such as one for each of its interfaces). An IA_PD does not need to be associated with exactly one interface. One IA_PD can be associated with the requesting router, with a set of interfaces, or with exactly one interface. Refer to Notes for Configuring a DHCP/PD Test for an example.

The default is IANA.

Default Value = "IANA"

dhcp6IaId

Identity Association (IA) Unique Identifier. This ID is incremented automatically for each DHCP client.

The default value is 10, the minimum is 1, and the maximum is 2147483647.

Default Value = "10"

dhcp6IaIdIncrement

The value by which the IA-ID is incremented for each DHCP client.

The default is 1, the minimum is 1, and the maximum is 2147483647.

Default Value = "1"

dhcp6IaT1

The suggested time, in seconds, at which the client contacts the server to extend the lifetimes of the assigned addresses. (This refers to the server that allocated the addresses that are currently assigned.)

This parameter is disabled when the *IA Type* is IATA.

The default value is 302400, the minimum is 0, and the maximum is 2147483647.



Default Value = "302400"

dhcp6IaT2

The suggested time, in seconds, at which the client contacts any available server to extend the lifetimes of the addresses assigned.

This parameter is disabled when the *IA Type* is IATA.

The default value is 483840, the minimum is 0, and the maximum is 2147483647.

Default Value = "483840"

useRelayAgent

If enabled, the subnet will emulate a DHCP relay agent. If disabled, the DHCP server must be on the same subnet as the port on which the subnet will run.

The default value is Disabled.

Note that the Trusted Network Element feature implements a subset of the relay agent functionality. Therefore, you generally will not enable both *Use Relay Agent* and *Enable Trusted Network Elements*.

Default Value = "False"

relayCount

The number of relay agents to use in this range. Note that:

- The number of Ethernet or ATM interfaces used by a range has to be equal to the number of DHCP clients plus the number of relay agents.
- The relay agent count cannot exceed the number of DHCP clients defined on the Ixia port.

The relay agent addresses are distributed among physical ports. For example, if you define a range with ten clients and four relay agents, and assign two physical ports to its network group, each port will get two relay agents and five clients.

In some cases, the distribution may result in one or more ports ending up without any relay agents (for example, if you have one relay agent in the range and five physical ports), In this case, suboption 6 is not added and the circuit ID/remote ID options are not sent.

The default is 1, the minimum is 1, and the maximum is 100.

Default Value = "1"

relayDestination

The address to which the requests from DHCP clients are forwarded.

The default IPv4 value is 20.0.0.1. The default IPv6 value is ::1400:1.

Default Value = "20.0.0.1"

relayFirstAddress

The IP address of the first emulated DHCP Relay Agent.

The addresses used for the relay agents are created using the following parameters: *Relay First Address*, *Relay Agent Count*, and *Relay Address Increment*.

For network groups with multiple physical ports, the first address sent to each port is different, such that an ascending sequence of IP addresses is assigned to the ports in order. For example, if the network group comprises three ports, and you configure the Relay Agent Count as 5, the Relay First Address as 12.1.0.10, and the Relay Increment as 0.0.0.1, the addresses will be distributed to the ports as follows:

- Port1: two relay agents (12.1.0.10 and 12.1.0.11)
- Port2: two relay agents (12.1.0.12 and 12.1.0.13)
- Port3: one relay agent (12.1.0.14)

The default IPv4 value is 20.0.0.100. The default IPv6 value is ::1400:64.

Default Value = "20.0.0.100"

`relaySubnet`

The network mask (expressed as a prefix length) used for all relay agents.

The default IPv4 value is 24, and the default IPv6 value is 96.

Default Value = "24"

`relayGateway`

The gateway address used for all relay agents.

The default IPv4 value is 20.0.0.1. The default IPv6 value is ::1400:1.

Default Value = "20.0.0.1"

`relayAddressIncrement`

The value by which the IP address is incremented for each relay agent.

The default IPv4 value is 0.0.0.1. The default IPv6 value is ::1.

Default Value = "0.0.0.1"

`useTrustedNetworkElement`

If enabled, the plug-in emulates trusted network elements on the port, rather than emulating relay agents. In this case, the Ixia port sends DHCPv4 packets with option 82 set. This causes the port to emulate one or more DHCP clients behind one or more trusted network elements.

When you enable this option, you will also provide a Client ID and/or a Remote ID.

This parameter is disabled by default.

Note that the Trusted Network Element feature implements a subset of the relay agent functionality. Therefore, you generally will not enable both *Use Relay Agent* and *Enable Trusted Network Elements*.

Default Value = "False"



`relayUseCircuitId`

If enabled, the CIRCUIT-ID option is sent along with the other options.

This parameter is disabled by default.

Default Value = "False"

`relayCircuitId`

This sub-option can be added by DHCP relay agents that terminate switched or permanent circuits. It encodes an agent-local identifier of the circuit from which a DHCP client-to-server packet was received. It is intended for use by agents in relaying DHCP responses back to the proper circuit. Possible uses of this field include:

- Router interface number
- Switching Hub port number
- Remote Access Server port number
- Frame Relay DLCI
- ATM virtual circuit number
- Cable Data virtual circuit number

You can insert variables into this field to generate large numbers of unique Circuit IDs. Use the following variables for text substitution:

- \$P - Port identifier associated with the DHCP client
- \$R - Range identifier associated with the DHCP client
- \$I - Interface identifier associated with the DHCP client

Use two dollar signs (\$\$) to include the literal "\$" symbol in the string.

In addition to using \$ variables, you can also use decimal, hexadecimal, and string sequences to generate Circuit IDs and Remote IDs. For details, refer to the *Sequence Generators* topic in IxLoad online help.

The default value is CIRCUITID-\$P.

Default Value = "CIRCUITID-\$P"

`relayHostsPerCircuitId`

The Number of consecutive hosts with the same Circuit ID. The default is 1, the minimum is 1, and the maximum is 100.

Default Value = "1"

`relayUseRemoteId`

If enabled, the REMOTE-ID option is sent along with the other options.

This parameter is disabled by default.

Default Value = "False"

`relayRemoteId`

This sub-option can be added by DHCP relay agents that terminate switched or permanent circuits and have mechanisms to identify the remote host end of the circuit.

Possible uses of this field include:

- Caller ID telephone number for dial-up connection
- User name prompted for by a Remote Access Server
- Remote caller ATM address
- Modem ID of a cable data modem
- Remote IP address of a point-to-point link
- Remote X.25 address for X.25 connections

You can insert variables into this field to generate large numbers of unique Circuit IDs. Use the following variables for text substitution:

- \$P - Port identifier associated with the DHCP client
- \$R - Range identifier associated with the DHCP client
- \$I - Interface identifier associated with the DHCP client

Use two dollar signs (\$\$) to include the literal "\$" symbol in the string.

In addition to using \$ variables, you can also use decimal, hexadecimal, and string sequences to generate Circuit IDs and Remote IDs. The string can contain a sequence of values of the form [StartValue-EndValue]. For example:

- Decimals [11-22]
- Hexadecimals [0x00-0xFF]
- Characters [AA-ZZ]

The default value is REMOTEID-\$I.

Default Value = "REMOTEID-\$I"

```
relayHostsPerRemoteId
```

The number of consecutive hosts with the same Remote ID.

The default is 1, the minimum is 1, and the maximum is 100.

Default Value = "1"

```
relay6UseOptInterfaceId
```

When enabled, the relay INTERFACE-ID option is sent along with the other options.

This parameter is disabled by default.

Default Value = "False"

```
relay6OptInterfaceId
```

This option can be added by DHCP relay agents that terminate switched or permanent circuits and have mechanisms to identify the remote host-end of the circuit (see RFC 3315, section 22.18).

The string can contain a sequence of values of the form [StartValue-EndValue]. For example:



- Decimals [11-22]
- Hexadecimals [0x00-0xFF]
- Characters [AA-ZZ]

The default is *id-[001-900]*.

Default Value = "id-[001-900]"

`relay6HostsPerOptInterfaceId`

The number of consecutive hosts with the same interface ID option.

The default is 1, the minimum is 1, and the maximum is 100.

Default Value = "1"

`relayUseSuboption6`

When enabled, relay agents will add suboption 6 (RFC3993) to the DHCP packets they send to DHCP servers.

Suboption 6 defines an IP address to which replies from the DHCP server should be sent. Each relay agent gets its own suboption 6 address. You configure these addresses by specifying *Suboption 6 First Address* and *Suboption 6 Subnet* values. For example, if you define the number of relay agents as 5, set the *Suboption 6 First Address* to 10.200.0.1, and set the *Suboption 6 Subnet* to 24, IxLoad will generate the following addresses:

- Relay0: suboption6=10.200.0.1
- Relay1: suboption6=10.200.0.2
- Relay2: suboption6=10.200.0.3
- Relay3: suboption6=10.200.0.4
- Relay4: suboption6=10.200.0.5

This parameter is disabled by default.

Default Value = "False"

`suboption6FirstAddress`

The first suboption6 IP address to use. (See *Use Suboption6* for an example.)

The default is 20.0.0.100.

Default Value = "20.0.0.100"

`suboption6AddressSubnet`

The network mask used for all suboption 6 addresses. The default is 24, the minimum is 1, and the maximum is 32.

Default Value = "24"

relayOverrideVlanSettings

When enabled, the DHCP Client plug-in overrides the VLAN settings (using the *First Vlan ID*, *Number of VLANs for Relays*, and *VLAN Increment* parameters), thereby allowing you to specify how VLANs are assigned.

For example, if you have three relay agents, and *First Vlan ID* is 1, *Number of VLANs for Relays* is 2, and *VLAN Increment* is 7, the relay agents will be assigned to the VLANs as follows:

- relay agent 1 assigned to vlan 1
- relay agent 2 assigned to vlan 8
- relay agent 3 assigned to vlan 1

This parameter is disabled by default.

Default Value = "False"

relayFirstVlanId

The first (outer) VLAN ID to allocate to relay agent interfaces.

The default is 1, the minimum is 1, and the maximum is 4094.

Default Value = "1"

relayVlanCount

The number of different VLAN IDs to use.

The default is 1, the minimum is 1, and the maximum is 4094.

Note: If your VLAN increment operation results in a VLAN ID that exceeds 4094, the DHCP plug-in will silently set it to the *First Vlan ID* (that is, the increment operation rolls over to the start value).

Default Value = "1"

relayVlanIncrement

The VLAN increment to use for relay interfaces.

The default is 1, the minimum is 1, and the maximum is 4094.

Default Value = "1"

relayOptionSet

Identifies a DHCP Option Set that contains the DHCP extensions that will be used by the relay agents created with this DHCP range.

You define DHCP Option Sets in the Network Plug-in Settings window.

Default Value = "None"

EXAMPLE

```
$DHCP_Client_1 rangeList.clear

set DHCP_R1 [::IxLoad new ixNetDHCPRange]
# ixNet objects needs to be added in the list before they are configured!
$DHCP_Client_1 rangeList.appendItem -object $DHCP_R1

set DefaultOptionSet [::IxLoad new ixNetDHCPOptionSet]
$DefaultOptionSet config \
    -name "DefaultOptionSet" \
    -defaultp true \
    -ipType "IPv4"

$DefaultOptionSet optionTlvs.clear

$DHCP_R1 config \
    -dhcp6IaT2 483840 \
    -dhcp6IaT1 302400 \
    -relayAddressIncrement "0.0.0.1" \
    -relayFirstAddress "20.0.0.100" \
    -relayUseRemoteId false \
    -dhcp6ParamRequestList "2; 7; 23; 24" \
    -relayUseCircuitId false \
    -relayVlanIncrement 1 \
    -relay6HostsPerOptInterfaceId 1 \
    -suboption6FirstAddress "20.0.0.100" \
    -relayRemoteId "REMOTEID-\$I" \
    -relaySubnet 24 \
    -relay6UseOptInterfaceId false \
    -dhcp6DuidVendorId 10 \
    -relayCircuitId "CIRCUITID-\$P" \
    -relayCount 1 \
    -relayOverrideVlanSettings false \
    -dhcp4ServerAddress "10.0.0.1" \
    -vendorClassId "Ixia DHCP Client" \
    -useTrustedNetworkElement false \
    -dhcp6DuidType "DUID-LLT" \
    -relayHostsPerCircuitId 1 \
    -relay6OptInterfaceId "id-\[001-900\]" \
    -relayDestination "20.0.0.1" \
    -relayVlanCount 1 \
    -suboption6AddressSubnet 24 \
    -dhcp6IaType "IANA" \
    -dhcp4UseFirstServer true \
    -relayUseSuboption6 false \
    -relayHostsPerRemoteId 1 \
    -ipType "IPv4" \
    -count 1 \
    -useVendorClassId false \
    -name "DHCP-R1" \
    -relayGateway "20.0.0.1" \
    -enabled true \
    -useRelayAgent false \
    -dhcp6IaId 10 \
    -relayFirstVlanId 1 \
    -dhcp4ParamRequestList "1; 3; 58; 59" \
    -dhcp6IaIdIncrement 1 \
    -dhcp6DuidVendorIdIncrement 1 \
    -dhcp6DuidEnterpriseId 10 \
    -renewTimer 0 \
    -clientOptionSet $DefaultOptionSet
```

SEE ALSO



DHCPOptionSetList

SYNOPSIS

DESCRIPTION

List of objects of DHCPOptionSet type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

DHCPOptionSet

SYNOPSIS

DESCRIPTION

Configures a DHCP option set for use by an IP/DHCP range.

SUBCOMMANDS

OPTIONS

`name`

Unique name for this DHCP option set.

Default value = "None"

`ipType`

The IP version used with this option set: IPv4 or IPv6.

Default value = "IPv4"

`defaulttp`

If enabled, indicates that this DHCP Option Set will be your default option set. It will be automatically referenced by new DHCP ranges that you create

Default value = "False"

`optionTlvs`

List of Option TLVs used by this option set.

Default value = "None"

EXAMPLE

```
set DefaultOptionSet [::IxLoad new ixNetDHCPOptionSet]
$DefaultOptionSet config \
  -name                "DefaultOptionSet" \
  -defaulttp           true \
  -ipType              "IPv4"

$DefaultOptionSet optionTlvs.clear
```

SEE ALSO

DHCPOptionTLVList

SYNOPSIS

DESCRIPTION

List of DHCP type-length-value entities for DHCP options. The list contains items that are of `DHCPOptionTLV` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



DHCPOptionTLV

SYNOPSIS

DESCRIPTION

Configures a DHCP type-length-value entity for use by a `DHCPOptionTLVList`.

SUBCOMMANDS

OPTIONS

`name`

The option name.

Default value = "None"

`code`

The option code.

Default value = "1"

`type`

The option value format type. See `eValueType` (on page 350) for the list of values.

Default value = "None"

`value`

The string representation of the option value. The `IxLoad DHCP Client` plug-in parses and validates the string according to `type` field.

Default Value = "VALUE"

EXAMPLE

SEE ALSO



eValueType

SYNOPSIS

DESCRIPTION

List of option values used by a DHCP option TLV. See DHCPOptionTLV (on page 348).

SUBCOMMANDS

OPTIONS

Value	Description
10	String
20	Hexadecimal
30	Domain Name
40	Zero Length
50	Boolean
60	8 bit Integer
61	8 bit Integer List
70	16 bit Integer
71	16 bit Integer List
80	32 bit Integer
81	32 bit Integer List
90	IPv4 Prefix
100	IPv6 Prefix
110	IPv4 Address
111	IPv4 Address List
120	IPv6 Address
121	IPv6 Address List

EXAMPLE

SEE ALSO



Dhcp6DuidTypeChoices

SYNOPSIS

DESCRIPTION

DHCP Unique Identifier (DUID) Type. Each DHCPv6 client and server has a DUID. DHCP servers use DUIDs to identify clients for the selection of configuration parameters and in the association of IAs with clients. DHCP clients use DUIDs to identify a server in messages in which a server must be identified.

The DUID types are:

- DUID-LLT (Link-layer address plus time)
- DUID-EN (Vendor-assigned unique ID based on Enterprise Number)
- DUID-LL (Link-layer address)

The default value is DUID-LLT.

SUBCOMMANDS

OPTIONS

Value	Description
DUID-LLT	DUID-LLT
DUID-EN	DUID-EN
DUID-LL	DUID-LL

EXAMPLE

SEE ALSO

Dhcp6IaTypeChoices

SYNOPSIS

DESCRIPTION

The Identity Association Type. An identify association is a construct through which a server and a client identify, group, and manage a set of related IPv6 addresses. Each IA has an associated IA ID. A client may have more than one IA assigned with it (such as one for each of its interfaces).

The IA types are:

- IANA – Identity Association for Non-temporary Addresses (IA_NA) is an IA that carries assigned addresses that are not temporary addresses.
- IATA – Identity Association for Temporary Addresses (IA_TA) is an IA that carries temporary addresses.
- IAPD – Identity Association for Prefix Delegation (IA_PD) is a collection of prefixes assigned to the requesting router. Each IA_PD has an associated IAID. A requesting router may have more than one IA_PD assigned to it (such as one for each of its interfaces). An IA_PD does not need to be associated with exactly one interface. One IA_PD can be associated with the requesting router, with a set of interfaces, or with exactly one interface. Refer to Notes for Configuring a DHCP/PD Test for an example.

The default is IANA.

SUBCOMMANDS

OPTIONS

Value	Description
IANA	IANA
IATA	IATA
IAPD	IAPD

EXAMPLE

SEE ALSO



DHCP Client Example

This section shows an example of how to create a DHCP client plugin in the Tcl API.

DHCP Client Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.10.0.79
# DHCPclient.tcl made on Aug 14 2008 15:20
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              "Network1" \
    -macMappingMode   0 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear

Begin appending items to global plugin list.

set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
    -enabled          true \
    -name              "GratARP"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.



DHCP Client Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

Configure the TCP
portion of the stack.

```
$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
    -name "TCP" \
```

DHCP Client Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the
# list before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior               0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new
ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                 "master" \
    -negotiateMasterSlave            true

$Ethernet_1 config \
    -advertise10Full                 true \
    -name                             "Ethernet-1" \
    -autoNegotiate                    true \
    -advertise100Half                true \
    -advertise10Half                  true \
    -speed                             "k100FD" \
    -advertise1000Full                true \
    -advertise100Full                 true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.



DHCP Client Example

```
set MAC_VLAN_3 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the
# list before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_3

$MAC_VLAN_3 config \
    -name "MAC/VLAN-3"

$MAC_VLAN_3 childrenList.clear

set DHCP_Client_1 [::IxLoad new ixNetDHCPPugin]
# ixNet objects needs to be added in the
# list before they are configured!
$MAC_VLAN_3 childrenList.appendItem -object $DHCP_Client_1

$DHCP_Client_1 config \
    -name "DHCP Client-1"

$DHCP_Client_1 childrenList.clear
$DHCP_Client_1 extensionList.clear

$MAC_VLAN_3 extensionList.clear
$Ethernet_1 extensionList.clear
```

Configure the MAC addresses and VLAN tags.

Configure the DHCP client plugin.

Clear the lists of extension protocols.

DHCP Client Example

```
#####
# Setting the ranges starting with the plugin on top of the stack
#####
$DHCP_Client_1 rangeList.clear

set DHCP_R1 [::IxLoad new ixNetDHCPRange]
# ixNet objects needs to be added in the
# list before they are configured!
$DHCP_Client_1 rangeList.appendItem -object $DHCP_R1

set DefaultOptionSet [::IxLoad new ixNetDHCPOptionSet]
$DefaultOptionSet config \
    -name "DefaultOptionSet" \
    -defaultp true \
    -ipType "IPv4"

$DefaultOptionSet optionTlvs.clear

$DHCP_R1 config \
    -dhcp6IaT2 483840 \
    -dhcp6IaT1 302400 \
    -relayAddressIncrement "0.0.0.1" \
    -relayFirstAddress "20.0.0.100" \
    -relayUseRemoteId false \
    -dhcp6ParamRequestList "2; 7; 23; 24" \
    -relayUseCircuitId false \
    -relayVlanIncrement 1 \
    -relay6HostsPerOpt InterfaceId 1 \
    -suboption6FirstAddress "20.0.0.100" \
    -relayRemoteId "REMOTEID-{$I}" \
    -relaySubnet 24 \
    -relay6UseOpt InterfaceId false \
    -dhcp6DuidVendorId 10 \
    -relayCircuitId "CIRCUITID-{$P}" \
    -relayCount 1 \
    -relayOverrideVlanSettings false \
    -dhcp4ServerAddress "10.0.0.1" \
    -vendorClassId "Ixia DHCP Client" \
    -useTrustedNetworkElement false \
    -dhcp6DuidType "DUID-LLT" \
```

Configure the address ranges for the DHCP client.

Configure the option set for the DHCP client.



DHCP Client Example

```
#####  
# Setting the ranges starting with the plugin on top of the stack  
#####  
$DHCP_Client_1 rangeList.clear  
  
set DHCP_R1 [::IxLoad new ixNetDHCPRange]  
# ixNet objects needs to be added in the list before they are  
# configured!  
$DHCP_Client_1 rangeList.appendItem -object $DHCP_R1  
  
set DefaultOptionSet [::IxLoad new ixNetDHCPOptionSet]  
$DefaultOptionSet config \  
    -name                "DefaultOptionSet" \  
    -defaulttp           true \  
    -ipType              "IPv4"  
  
$DefaultOptionSet optionTlvs.clear  
  
$DHCP_R1 config \  
    -dhcp6IaT2           483840 \  
    -dhcp6IaT1           302400 \  
    -relayAddressIncrement "0.0.0.1" \  
    -relayFirstAddress   "20.0.0.100" \  
    -relayUseRemoteId    false \  
    -dhcp6ParamRequestList "2; 7; 23; 24" \  
    -relayUseCircuitId   false \  
    -relayVlanIncrement  1 \  
    -relay6HostsPerOpt InterfaceId 1 \  
    -suboption6FirstAddress "20.0.0.100" \  
    -relayRemoteId       "REMOTEID-\\$I" \  
    -relaySubnet         24 \  
    -relay6UseOpt InterfaceId false \  
    -dhcp6DuidVendorId   10 \  
    -relayCircuitId      "CIRCUITID-\\$P" \  
    -relayCount          1 \  
    -relayOverrideVlanSettings false \  
    -dhcp4ServerAddress  "10.0.0.1" \  
    -vendorClassId       "Ixia DHCP Client" \  
    -useTrustedNetworkElement false \  
    -dhcp6DuidType       "DUID-LLT" \  

```

DHCP Client Example

```

-relayHostsPerCircuitId      1 \
-relay6OptInterfaceId      "id-[001-900]" \
-relayDestination          "20.0.0.1" \
-relayVlanCount            1 \
-suboption6AddressSubnet   24 \
-dhcp6IaType               "IANA" \
-dhcp4UseFirstServer       true \
-relayUseSuboption6        false \
-relayHostsPerRemoteId     1 \
-ipType                    "IPv4" \
-count                     1 \
-useVendorClassId          false \
-name                      "DHCP-R1" \
-relayGateway              "20.0.0.1" \
-enabled                   true \
-useRelayAgent             false \
-dhcp6IaId                 10 \
-relayFirstVlanId          1 \
-dhcp4ParamRequestList    "1; 3; 58; 59" \
-dhcp6IaIdIncrement        1 \
-dhcp6DuidVendorIdIncrement 1 \
-dhcp6DuidEnterpriseId    10 \
-renewTimer                 0 \
-clientOptionSet           $DefaultOptionSet

```

```
set MAC_R2 [$DHCP_R1 getLowerRelatedRange "MacRange"]
```

```

$MAC_R2 config \
-count          1 \
-name          "MAC-R2" \
-enabled       true \
-mtu           1500 \
-mac          "22:73:F7:4E:00:00" \
-incrementBy  "00:00:00:00:00:01"

```

Configure the MAC addresses for the DHCP IP ranges.



DHCP Client Example

```
set VLAN_R1 [$DHCP_R1 getLowerRelatedRange "VlanIdRange"]
```

```
$VLAN_R1 config \  
  -incrementStep 1 \  
  -uniqueCount 4094 \  
  -name "VLAN-R1" \  
  -innerIncrement 1 \  
  -innerUniqueCount 4094 \  
  -enabled true \  
  -innerFirstId 1 \  
  -increment 1 \  
  -priority 1 \  
  -firstId 1 \  
  -innerIncrementStep 1 \  
  -idIncrMode 2 \  
  -innerEnable false \  
  -innerPriority 1
```

Configure the VLAN tags for the DHCP IP ranges.

DHCPServerPlugin

SYNOPSIS

DESCRIPTION

Configures a DHCP server.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`rangeList`

Name of the list of DHCP Server ranges. The list must be a `DHCPServerRangeList` object.

Default value = "None"

EXAMPLE

```
set DHCP_Server_1 [::IxLoad new ixNetDHCPServerPlugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_4 childrenList.appendItem -object $DHCP_Server_1

$DHCP_Server_1 config \
    -name "DHCP Server-1"

$DHCP_Server_1 childrenList.clear
$DHCP_Server_1 extensionList.clear

$MAC_VLAN_4 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO

DHCPServerRangeList

SYNOPSIS

DESCRIPTION

A list of IP addresses for the DHCP servers. The list contains items that are of `DHCPServerRange` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



DHCP Server Range

SYNOPSIS

DESCRIPTION

Configures a DHCP server for use in a `DHCP Server Range List`.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value = "True".

`macRange`

Name of the range of MAC addresses to be used by this DHCP server.

Default value = "None"

`vlanRange`

Name of the range of VLAN tags to be used by this DHCP server.

Default value = "None"

`count`

This value specifies the total number of IP addresses that will be allocated to the address pool(s).

The default is 1, the minimum is 1, and the maximum is 1,000,000.

Note that an address pool cannot span multiple subnets.

Default value = "1"

`ipType`

Defines the IP address version that will be used for the range.

The only option is IPv4.

Default value = "IPv4"

`ipAddress`

The starting IP address for this address pool.

Except in back-to-back Ixia port scenarios, these addresses do not need to be in the same subnet as that defined for the DHCP server address (on the DHCP Server tab).

The IPv4 default is 10.10.1.1.

The IPv6 default is ::A0A:101.

Default value = "10.10.1.1"

`ipPrefix`

The network prefix length associated with this address pool.

The IPv4 default is 16.

The IPv6 default is 112.

Default value = "16"

`ipGateway`

The gateway IP address associated with this address pool. This is the gateway that will be assigned to any client that is allocated an IP address from this pool.

The default is a NULL value (no gateway address will be assigned).

Default value = "" (null)

`ipDns1`

The first DNS server associated with this address pool. This is the first DNS address that will be assigned to any client that is allocated an IP address from this pool.

Note that IxLoad does not verify that this DNS address is reachable.

The default is a NULL value (no DNS server address will be assigned).

Default value = "" (null)

`ipDns2`



The second DNS server associated with this address pool. This is the second (of two) DNS addresses that will be assigned to any client that is allocated an IP address from this pool.

Note that IxLoad does not verify that this DNS address is reachable.

The default is a NULL value (no DNS server address will be assigned).

Default value = "" (null)

`serverAddress`

The first IP address of the interface on which the DHCP server listens.

The DHCP Server plug-in assigns a unique IP address for each range that you add to a . However, you can use the same address for multiple ranges. For example:

	Name	Status	Server Address	Server IP Prefix
▶ 1	DHCPServer-R1	Activated	10.10.0.1	16
2	DHCPServer-R2	Activated	10.10.0.1	16

This type of configuration is required when emulating a real-life scenario in which a DHCP server has multiple (and distinct) pools configured and can assign addresses from different subnets to the requesting clients (when these clients are behind emulated or real relay agents).

The IPv4 default is 10.10.0.1.

The IPv6 default is ::A0A:1.

Default value = "10.10.0.1"

`serverAddressIncrement`

The value by which to increment the IP address for each DHCP server.

The default is 0.0.0.1.

Default value = "0.0.0.1"

EXAMPLE

```
$DHCP_Server_1 rangeList.clear

set DHCPserver_R1 [::IxLoad new ixNetDHCPserverRange]
# ixNet objects needs to be added in the list before they are configured!
$DHCP_Server_1 rangeList.appendItem -object $DHCPserver_R1

$DHCPserver_R1 config \
    -count 1 \
    -ipDns2 "" \
    -name "DHCPserver-R1" \
    -ipGateway "" \
    -ipDns1 "" \
    -enabled true \
    -ipPrefix 16 \
    -serverAddressIncrement "0.0.0.1" \
    -ipAddress "10.10.1.1" \
    -serverAddress "10.10.0.1" \
    -ipType "IPv4"

set MAC_R3 [$DHCPserver_R1 getLowerRelatedRange "MacRange"]

$MAC_R3 config \
    -count 1 \
    -name "MAC-R3" \
    -enabled true \
    -mtu 1500 \
    -mac "4A:5B:5B:08:00:00" \
    -incrementBy "00:00:00:00:00:01"

set VLAN_R1 [$DHCPserver_R1 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
    -incrementStep 1 \
    -uniqueCount 4094 \
    -name "VLAN-R1" \
    -innerIncrement 1 \
    -innerUniqueCount 4094 \
    -enabled true \
    -innerFirstId 1 \
    -increment 1 \
    -priority 1 \
    -firstId 1 \
    -innerIncrementStep 1 \
    -idIncrMode 2 \
    -innerEnable false \
    -innerPriority 1
```

SEE ALSO



DHCPServerSessionData

SYNOPSIS

DESCRIPTION

Configures the DHCP server global settings.

SUBCOMMANDS

OPTIONS

`defaultLeaseTime`

`defaultValue="3600"`

`maxLeaseTime`

`defaultValue="86400"`

EXAMPLE

SEE ALSO



DHCP Server Example

This section shows an example of how to create a DHCP server plugin in the Tcl API.

DHCP Server Example

```
#####  
# IxLoad ScriptGen created TCL script  
# Network1 serialized using version 4.10.0.79  
# DHCPserver.tcl made on Aug 14 2008 15:21  
#####  
  
set Network1 [::IxLoad new ixNetworkGroup $chassisChain]  
$Network1 config \  
    -comment          "" \  
    -name              "Network1" \  
    -macMappingMode   0 \  
    -linkLayerOptions 0  
  
$Network1 globalPlugins.clear  
  
Begin appending items to global plugin list.  
  
set GratARP [::IxLoad new ixNetGratArpPlugin]   
# ixNet objects needs to be added in the   
# list before they are configured!   
$Network1 globalPlugins.appendItem -object $GratARP  
  
$GratARP config \  
    -enabled          true \  
    -name              "GratARP"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuituous ARP.

DHCP Server Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

Configure the TCP
portion of the stack.

```
$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -name "TCP" \
```



DHCP Server Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the
# list before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior              0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                "master" \
    -negotiateMasterSlave          true

$Ethernet_1 config \
    -advertise10Full                true \
    -name                            "Ethernet-1" \
    -autoNegotiate                  true \
    -advertise100Half               true \
    -advertise10Half                true \
    -speed                          "k100FD" \
    -advertise1000Full              true \
    -advertise100Full               true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear
```

Configure the Dynamic Control plane settings.

Configure the physical layer properties.

DHCP Server Example

```
set MAC_VLAN_4 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the
# list before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_4

$MAC_VLAN_4 config \
    -name "MAC/VLAN-4"

$MAC_VLAN_4 childrenList.clear

set DHCP_Server_1 [::IxLoad new ixNetDHCPServerPlugin]
# ixNet objects needs to be added in the
# list before they are configured!
$MAC_VLAN_4 childrenList.appendItem -object $DHCP_Server_1

$DHCP_Server_1 config \
    -name "DHCP Server-1"

$DHCP_Server_1 childrenList.clear
$DHCP_Server_1 extensionList.clear
$MAC_VLAN_4 extensionList.clear
$Ethernet_1 extensionList.clear
```

Configure the MAC addresses and VLAN tags.

Configure the DHCP server plugin.

Clear the lists of extension protocols.



DHCP Server Example

```
#####
# Setting the ranges starting with the plugin on top of the stack
#####
$DHCP_Server_1 rangeList.clear

set DHCPserver_R1 [::IxLoad new ixNetDHCPserverRange]
# ixNet objects needs to be added in the list Configure the IP address
# before they are configured! ranges for the DHCP server.
$DHCP_Server_1 rangeList.appendItem -object $DHCPserver_R1

$DHCPserver_R1 config \
    -count 1 \
    -ipDns2 "" \
    -name "DHCPserver-R1" \
    -ipGateway "" \
    -ipDns1 "" \
    -enabled true \
    -ipPrefix 16 \
    -serverAddressIncrement "0.0.0.1" \
    -ipAddress "10.10.1.1" \
    -serverAddress "10.10.0.1" \
    -ipType "IPv4"

set MAC_R3 [$DHCPserver_R1 getLowerRelatedRange "MacRange"]

$MAC_R3 config \
    -count 1 \
    -name Configure the MAC
addresses for the DHCP IP
ranges. "MAC-R3" \
    -enabled true \
    -mtu 1500 \
    -mac "4A:5B:5B:08:00:00" \
    -incrementBy "00:00:00:00:00:01"
```

DHCP Server Example

```
set VLAN_R1 [$DHCPserver_R1 getLowerRelatedRange "VlanIdRange"]
```

```
$VLAN_R1 config \  
  -incrementStep 1 \  
  -uniqueCount 4094 \  
  -name "VLAN-R1" \  
  -innerIncrement 1 \  
  -innerUniqueCount 4094 \  
  -enabled true \  
  -innerFirstId 1 \  
  -increment 1 \  
  -priority 1 \  
  -firstId 1 \  
  -innerIncrementStep 1 \  
  -idIncrMode 2 \  
  -innerEnable false \  
  -innerPriority 1
```

Configure the VLAN tags for the DHCP IP ranges.



Authentication Extension Plugins

This section describes the Authentication Extension plugins.

DNPluginBase

SYNOPSIS

DESCRIPTION

Base class for 802.1x classes.

SUBCOMMANDS

OPTIONS

name

Name of this instance of the plugin.

Default value = "None"

EXAMPLE

SEE ALSO



DNPortGroupBase

SYNOPSIS

DESCRIPTION

Base class for NAC L3 or 802.1x port group options.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`overrideGlobalSetupRate`

If selected, the setup and teardown parameters defined at the network group level will override those defined at the global level.

For example, if you have set the initial setup rate to 80 on the global level, and you have defined two ports, these 80 session setups will be evenly distributed across the ports (40 for each). If you then enable Override Global Rate Controls, you can modify the number of session setups for each of the ports (such as changing the distribution from 40-40 to 60-20).

Default value="False"

`maxClientsPerSecond`

The maximum number of 802.1X sessions to set up, per second.

Default value="10"

`maxOutstandingRequests`

The maximum number of 802.1X sessions that can be outstanding at any given time.

Default value="10"

EXAMPLE

SEE ALSO



DNSessionBase

SYNOPSIS

DESCRIPTION

Configures the EAPoUDP global rate control and protocol settings.

SUBCOMMANDS

OPTIONS

`waitForCompletion`

Wait for all interfaces to report a change in their state. If there is no change in the state of any interface for 100 seconds, this will time out. This does not mean there was a failure; it is a means of early detection of a potential problem.

Note: IxLoad ignores the *Wait For Completion* parameter when *Trigger Origin* is set to "Internal" and *Trigger Type* is set to one of the DHCP types (DHCP, DHCP_ARP, DHCP_GratArp, DHCP_ICMP).

Default value = "True"

`fragmentSize`

The maximum size of a fragment that can be sent on the wire for TLS fragments that comprise the phase 1 conversation (tunnel establishment).

The default is 1400, the minimum is 500, and the maximum is 1400.

Default value = "1400"

`waitBeforeRun`

Specifies a delay (in seconds) before EAPoUDP negotiations are started on the Ixia port.

This option is useful when the DUT does not immediately bring up the link and the protocol properties of the port.

Note that this option also has the effect of delaying the start of DHCP protocol negotiation on the port.

The default value is zero seconds, and the maximum is 600 seconds.

Default value="0"

`certInfo`

Name of object describing authentication certificate to be used. This object must be a CertInfo object.

Default value="None"

maxClientsPerSecond

The rate at which internal triggers are sent, per second.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value="10"

maxOutstandingRequests

The threshold at which IxLoad begins throttling back the number of triggers being sent.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value="10"

EXAMPLE

SEE ALSO



DNRangeBase

SYNOPSIS

DESCRIPTION

Configures the 802.1X Range Parameters.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value= "True".

`targetRange`

The MAC Range to authorize with this 802.1X Range.

Default value = "None"

`nacSequence`

The NAC Sequence used by this range.

Note: If a NAC Sequence has been selected for an 802.1X range, but you do not want a NAC Sequence associated with the range, specify `None` to remove the NAC Sequence from the range configuration.

Default value = "None"

`fastProvisionMode`

FAST provisioning mode specifies how the tunnel PAC is acquired. See `FastProvisionModeChoices` for the list of choices.

Default value="unauthenticated"

`fastInnerMethod`

FAST inner method See `FastInnerMethodChoices` for the list of choices.

Default value = "GTC"

`fastStatelessResume`

FAST stateless resume mode. See `FastStatelessResumeMethodChoices` for the list of choices.

Default value="no"

`userName`

The `UserName` used to authenticate the port.

Default value = "username_1_1_1_file"

`userPassword`

The User Password used to authenticate the port.

Default value = "userpass_1_1_1_file"

EXAMPLE

SEE ALSO



FastProvisionModeChoices

SYNOPSIS

DESCRIPTION

List of choices for the `FastProvisionMode` option of the `DNRRangeBase` object.

SUBCOMMANDS

OPTIONS

Value	Description
authenticated	Authenticated (certificate is required). PAC provisioning occurs inside an authenticated tunnel, using the server certificate.
unauthenticated	Unauthenticated (no certificate required). When using unauthenticated mode, the first session will acquire a special token named <i>tunnelPac</i> from the ACS server. By design, the server will send a plain EAP Failure message at the end of the successful session. The ACS logs will indicate for the failed session that the client was provisioned with a new tunnelPac. IxLoad will not re-initiate a new session with the obtained tunnelPac unless the DUT re-initiates full re-authentication or a new start test message is sent to the PCPU.
fromfile	Load from File A PAC is stored on the chassis. The share location is <code>C:\Program Files\ixia\nfs\rw\ports_x_y_</code> In this mode, the PAC file is loaded from the file and is directly presented to server. The client does not request a new PAC as part of its session. Phase 0 (provisioning) is skipped.
auth_save_pac	Authenticated and save to file This option is identical to the Authenticated option except that after receiving a tunnel PAC, the client saves it on the Ixia chassis share <code>C:\Program Files\ixia\nfs\rw\ports_x_y_</code> Note that any existing PACs will be overwritten.
unauth_save_pac	This option is identical to the Unauthenticated option except that after receiving a tunnel PAC, the client saves it on the Ixia chassis share <code>C:\Program Files\ixia\nfs\rw\ports_x_y_</code> Note that any existing PACs will be overwritten.

EXAMPLE

SEE ALSO



FastInnerMethodChoices

SYNOPSIS

DESCRIPTION

List of choices for the `FastInnerMethod` option of the `DNRangeBase` object.

SUBCOMMANDS

OPTIONS

Value	Description
GTC	GTC
MsChapv2	MS CHAP v2

EXAMPLE

SEE ALSO

FastStatelessResumeChoices

SYNOPSIS

DESCRIPTION

List of choices for the `FastStatelessResume` option of the `DNRangeBase` object.

SUBCOMMANDS

OPTIONS

Value	Description
yes	Yes: Within the secure tunnel established between the client and the server, based on the tunnel PAC, the client also requires a UserPAC which will be provided by the server (if it is enabled to do so). If the client is provided with the UserPAC, during the next authentication sessions, it will have to send the userPAC to the server, within the tunnel, for the authentication to complete.
no	No: FAST stateless resume mode is not used.

EXAMPLE

SEE ALSO



WebAuthPlugin

SYNOPSIS

DESCRIPTION

Configures a WebAuthx plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of this plugin.

`rangeList`

List of address ranges used by this plugin.

Default value = "None"

EXAMPLE

SEE ALSO

WebAuthPortGroupData

SYNOPSIS

DESCRIPTION

Configures WebAuth port parameters that apply to all of the address ranges defined in the port configuration.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`overrideGlobalArpResponse`

If `true`, the global *ARP Control* setting for this port is overridden. This option acts as a Boolean negation of the global setting.

Default value = "False"

`overrideGlobalSetupRate`

If `true`, the setup and teardown parameters defined at the network group level will override those defined at the global level.

For example, if you have set the initial setup rate to 80 on the global level, and you have defined two ports, these 80 session setups will be evenly distributed across the ports (40 for each). If you then enable Override Global Rate Controls, you can modify the number of session setups for each of the ports (such as changing the distribution from 40-40 to 60-20).

Default value = "False"



`maxClientsPerSecond`

The number of interfaces to set up, per second.

Default value = "10"

`maxOutstandingRequests`

The maximum number of sessions that can be outstanding at any time. If this maximum is reached, `IxLoad` begins throttling the number of interfaces being sent.

A session refers to all the events associated with an authentication attempt (connecting, timeout, success, failure). A session ends when the authentication request results in success, failure, or timeout.

Default value = "10"

EXAMPLE

SEE ALSO

WebAuthSessionData

SYNOPSIS

DESCRIPTION

Configures the WebAuth global parameters that are common to all ports in the network group and all ranges in the test.

SUBCOMMANDS

OPTIONS

`waitForCompletion`

If enabled, the configuration will end only after all interfaces are configured or a timeout occurs.

A timeout occurs if there is no change in the state of any interface for 100 seconds.

Note: IxLoad ignores the *Wait For Completion* parameter when the WebAuth extension is enabled for a DHCP stack.

`defaultValue="True"`

`protocol`

The protocol used by the host to access the network. The only supported protocol is HTTP.

`defaultValue="HTTP"`

`port`

The TCP port number for the protocol type.

The default is 80.

`defaultValue="80"`

`renewDhcp`

If selected, renew the DHCP address lease upon successful authentication (assuming that DHCP address allocation is enabled).

This setting has no effect if DHCP is not enabled.



Note: If the DHCP server allocates a different IP address to the client, that client will timeout while trying to reach the initially-requested page. This occurs because the new IP address is not authorized.

```
defaultValue="False"
```

```
requestUrl
```

The URL requested by the emulated clients at the start of authentication. The default value is `http://192.168.0.10`.

If *Perform Policy Check* is enabled, the host will try to reach the initially-requested URL after successful authentication.

The URL can have a maximum of 300 characters.

Note: Only IPv4 addresses are supported.

```
defaultValue="http://192.168.0.10"
```

```
redirectLoginURL
```

The Login page URL to which the DUT redirects the emulated clients, if the Login page is not sent directly from the DUT. (The DUT will send a redirect for the Login page only if it is configured for this.)

The URL can have a maximum of 300 characters.

```
defaultValue=""
```

```
redirectFailureURL
```

The Failure page URL to which the DUT redirects the emulated clients, if the Failure page is not sent directly from the DUT. (The DUT will send a redirect for the Failure page only if it is configured for this.)

The URL can have a maximum of 300 characters.

```
defaultValue=""
```

```
name="authSuccess"
```

The text string used to determine if authentication was successful. This must be a unique text string that appears somewhere on the success page that is returned by the DUT.

For example, if the value of the Success Text field is "Success", IxLoad will expect to find that string somewhere in the returned web page. The following HTML code will match the configured string:

```
<title>Login Success</title>
```

Note that IxLoad parses the entire web page. Therefore, it is important that the success web page not include the string used for the "Failure Text" field, and vice versa.

The maximum number of characters is 100.

```
defaultValue="Success"
```

```
name="authFailure"
```

The text string used to determine if authentication failed. This must be a unique text string that appears somewhere on the failure page that is returned by the DUT.

For example, if the value of the Failure Text field is "Failed", IxLoad will expect to find that string somewhere in the returned web page. The following HTML code will match the configured string:

```
<title>Login Failed</title>
```

Note that IxLoad parses the entire web page. Therefore, it is important that the failure web page not include the string used for the "Success Text" field, and vice versa.

The maximum number of characters is 100.

```
defaultValue="Failed"
```

```
name="authTimeout"
```

The amount of time to wait for the DUT to return a success or failure page.

The default: is 30 seconds, the minimum is 1 second, and the maximum is 300 seconds.

Note: Take into account the usual TCP timeout when changing this parameter. (When attempting to establish a connection, TCP sends as many as four SYN packets at three second intervals. If the connection is unsuccessful after this 12 second period, a TCP timeout occurs.)

```
defaultValue="30"
```

```
policyEnable
```

If `true`, the host will try to reach the initially-requested URL after successful authentication.

Default value = "False"

```
policySuccess
```

The text string used to determine if the client was successfully redirected to the originally-requested web page. This must be a unique text string that appears somewhere on the web page.

For example, if the value of the Success Text field is "Ixia", IxLoad will expect to find that string somewhere in the returned web page. The following HTML code would match the configured string:

```
<title>Ixia Downloads</title>
```

It is important to set this field to a correct value, even if Perform Policy Check is not selected.

If this string is found before the policy check, the *Bypass* statistic is incremented.

The default value is *Hit*. The maximum number of characters is 100.



Default value = "Hit"

`policyTimeout`

The amount of time to wait for the DUT to return the requested URL page.

Default value = "30"

`name="disableArpResponse"`

When the option is selected, the Ixia ports configured for WebAuth stop responding to ARP requests when the Stop command is issued. When the option is not selected, the Ixia ports configured for WebAuth continue to respond to ARP requests when the Stop command is issued.

Default value = "False"

`defaultFields`

Enables or disables the use of default tags that will be searched on the Login page for input fields 1, 2, and 3.

If this field is disabled, IxLoad uses the values specified in `inputField1`, `inputField2`, and `inputField3`.

Default value = "True"

`inputField1`

The label to use on the first field of the web authentication form. Usually, this represents the username of the client.

For example, if the value of Input Field 1 is "username", IxLoad will expect to find that label in the HTML form. The following HTML code would successfully match the configured string:

```
<Input type="text" name="username">
```

The default is *username*. The maximum number of characters is 100.

Default value = "username"

`inputField2`

The label to use on the second field of the web authentication form. Usually, this represents the password of the client.

See `inputField1` for an example.

The default is *password*. The maximum number of characters is 100.

Default value = "password"

`inputField3`

The label to use on the third field of the web authentication form. The third field typically represents the account type.

See `inputField1` for an example.

The default is *Account_type*. The maximum number of characters is 100.

Default value = "role"

`responseDelay`

The response delay in seconds. This is the amount of time that hosts will wait before responding to the DUT's request of authentication.

This delay simulates the time needed for a user to enter authentication data and send the page to the web server.

The default is 0 seconds (an immediate response). The range is from 0 to 300 seconds.

Default value = "0"

`numRetry`

The number of times the host can attempt web authentication, including failures and timeouts. IxLoad will retry the authentication process if the failure criteria are met or a timeout occurs.

A session is defined from the start of authentication process up to success, failure, or timeout. A retry will reset the state of the host to the initial state.

The default value is 1 retry. The minimum is 0, and the maximum is 300.

Default value = "1"

`waitBeforeRun`

Specifies a delay (in seconds) before WebAuth negotiations are started on the Ixia port.

This option is useful when the DUT does not immediately bring up the link and the protocol properties of the port.

Note that this option also has the effect of delaying the start of DHCP protocol negotiation on the port.

The default value is zero seconds, and the maximum is 600 seconds.

Default value = "0"

`maxClientsPerSecond`

The number of interfaces to set up, per second.



The default is 10, the minimum is 1, and the maximum is 1024.

Default value = "10"

`enableMaxOutstanding`

Enables control of outstanding sessions.

Default value = "False"

`maxOutstandingRequests`

The maximum number of sessions that can be outstanding at any time. If this maximum is reached, `IxLoad` begins throttling the number of interfaces being sent.

A session refers to all the events associated with an authentication attempt (connecting, timeout, success, failure). A session ends when the authentication request results in success, failure, or timeout.

This parameter is applicable only if `enableMaxOutstanding` is `true`.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value = "10"

EXAMPLE

SEE ALSO

WebAuthRangeList

SYNOPSIS

DESCRIPTION

List of `WebAuthRange` objects for use by a `WebAuthPlugin` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



WebAuthRange

SYNOPSIS

DESCRIPTION

Creates a range of names or passwords for use in a `WebAuthRangeList` object.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`targetRange`

The MAC Range to authorize with this 802.1X Range.

Default value = "None"

`inputValue1`

The first of three values to be returned in the HTTP POST message that is sent back to the DUT. *Input Field 1* is typically used to obtain the client's user name.

Default value = "client"

`inputValue2`

The second of three values to be returned in the HTTP POST message that is sent back to the DUT. *Input Field 2* is typically used to obtain the client's password.

Default value = "password"

inputValue3

The third of three values to be returned in the HTTP POST message that is sent back to the DUT. *Input Field 3* is typically used to obtain the client account type.

Default value = ""

expect

The expected result. The choices are:

- Success
- Failure
- Timeout

This value helps determine if the DUT is responding correctly to both valid and invalid authentication attempts.

This parameter applies for the entire session, without retries.

Default value = "SUCCESS"

EXAMPLE

SEE ALSO

WebAuth Example

This section shows an example of how to create a WebAuth extension plugin in the Tcl API.



WebAuth Example

```
#####  
# IxLoad ScriptGen created TCL script  
# Network1 serialized using version 4.10.0.79  
# WebAuth.tcl made on Aug 27 2008 13:31  
#####  
  
set Network1 [::IxLoad new ixNetworkGroup $chassisChain]  
$Network1 config \  
    -comment  
    -name  
    -macMappingMode  
    -linkLayerOptions  
  
$Network1 globalPlugins.clear  
  
set GratARP [::IxLoad new ixNetGratArpPlugin]  
# ixNet objects needs to be added in the list  
# before they are configured!  
$Network1 globalPlugins.appendItem -object $GratARP  
  
$GratARP config \  
    -enabled  
    -name
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuituous ARP.

```
    "" \  
    "Network1" \  
    0 \  
    0  
  
    true \  
    "GratARP"
```

WebAuth Example

```

set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP

$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
    -name "TCP" \

```

Configure the TCP portion of the stack.



WebAuth Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior              0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new
ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                "master" \
    -negotiateMasterSlave          true

$Ethernet_1 config \
    -advertise10Full                true \
    -name                            "Ethernet-1" \
    -autoNegotiate                  true \
    -advertise100Half               true \
    -advertise10Half                true \
    -speed                          "k100FD" \
    -advertise1000Full              true \
    -advertise100Full               true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

set MAC_VLAN_1 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_1

$MAC_VLAN_1 config \
    -name                            "MAC/VLAN-1"

$MAC_VLAN_1 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.

Configure the MAC
addresses and VLAN tags.

WebAuth Example

```
#####
# Setting the ranges starting with the plugin on top of the stack
#####
$IP_1 rangeList.clear

set IP_R1 [::IxLoad new ixNetIpV4V6Range]
# ixNet objects needs to be added in the list before they are
# configured!
$IP_1 rangeList.appendItem -object $IP_R1

$IP_R1 config \
    -count                1 \
    -name                 "IP-R1" \
    -gatewayAddress       "0.0.0.0" \
    -enabled              true \
    -autoMacGeneration    true \
    -mss                  1460 \
    -incrementBy          "0.0.0.1" \
    -prefix               16 \
    -gatewayIncrement     "0.0.0.0" \
    -gatewayIncrementMode "perSubnet" \
    -generateStatistics   false \
    -ipAddress            "10.10.0.1" \
    -ipType               "IPv4"

set WebAuth_R1 [$IP_R1 getExtensionRange $WebAuth_1]

$WebAuth_R1 config \
    -inputValue2          "password" \
    -inputValue3          "" \
    -inputValue1          "client" \
    -enabled              true \
    -expect               "SUCCESS" \
    -name                 "WebAuth-R1"
```

Configure the address ranges.

Configure the WebAuth plugin.



WebAuth Example

```
#####  
# Setting the ranges starting with the plugin on top of the stack  
#####  
$IP_1 rangeList.clear  
  
set IP_R1 [::IxLoad new ixNetIpV4V6Range] Configure the address  
ranges.  
# ixNet objects needs to be added in the list before they are  
configured!  
$IP_1 rangeList.appendItem -object $IP_R1  
  
$IP_R1 config \  
-count 1 \  
-name "IP-R1" \  
-gatewayAddress "0.0.0.0" \  
-enabled true \  
-autoMacGeneration true \  
-mss 1460 \  
-incrementBy "0.0.0.1" \  
-prefix 16 \  
-gatewayIncrement "0.0.0.0" \  
-gatewayIncrementMode "perSubnet" \  
-generateStatistics false \  
-ipAddress "10.10.0.1" \  
-ipType "IPv4"  
  
set WebAuth_R1 [$IP_R1 getExtensionRange $WebAuth_1]  
  
$WebAuth_R1 config \  
-inputValue2 Configure the WebAuth  
plugin. "password" \  
-inputValue3 "" \  
-inputValue1 "client" \  
-enabled true \  
-expect "SUCCESS" \  
-name "WebAuth-R1"
```

WebAuth Example

```

set MAC_R1 [$IP_R1 getLowerRelatedRange "MacRange"]

$MAC_R1 config \
  -count          1 \
  -name           "MAC-R1" \
  -enabled        true \
  -mtu            1500 \
  -mac            "00:0A:0A:00:01:00" \
  -incrementBy    "00:00:00:00:00:01"

set VLAN_R1 [$IP_R1 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
  -incrementStep  1 \
  -uniqueCount    4094 \
  -name           "VLAN-R1" \
  -innerIncrement 1 \
  -innerUniqueCount 4094 \
  -enabled        true \
  -innerFirstId   1 \
  -increment      1 \
  -priority       1 \
  -firstId        1 \
  -innerIncrementStep 1 \
  -idIncrMode     2 \
  -innerEnable    false \
  -innerPriority   1

```

Configure the MAC addresses for the IP range.

Configure the VLAN tags for the IP range.



Dot1xPlugin

SYNOPSIS

DESCRIPTION

Configures an 802.1x plugin.

SUBCOMMANDS

OPTIONS

name

Name of the instance of the plugin.

Default value = "None"

rangeList

Name of the list of address ranges for use by this plugin. This must be a Dot1xPlugin object.

Default value = "None"

EXAMPLE

```
set 802_1X_1 [::IxLoad new ixNetDot1xPlugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_13 extensionList.appendItem -object $802_1X_1

$802_1X_1 config \
    -name "802-1X-1"

$Ethernet_2 extensionList.clear
```

SEE ALSO



Dot1xPortGroupData

SYNOPSIS

DESCRIPTION

Configures an 802.1x port group.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`overrideGlobalSetupRate`

If selected, the setup and teardown parameters defined at the network group level will override those defined at the global level.

For example, if you have set the initial setup rate to 80 on the global level, and you have defined two ports, these 80 session setups will be evenly distributed across the ports (40 for each). If you then enable Override Global Rate Controls, you can modify the number of session setups for each of the ports (such as changing the distribution from 40-40 to 60-20).

Default value="False"

`maxClientsPerSecond`

The maximum number of 802.1X sessions to set up, per second.

Default value="10"

`maxOutstandingRequests`

The maximum number of 802.1X sessions that can be outstanding at any given time.

Default value="10"

`logoffMaxClientsPerSecond`

The number of interfaces to log off per second.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value = "10"

`logoffMaxOutstandingRequests`

Default value = "10"

EXAMPLE

SEE ALSO



Dot1xSessionData

SYNOPSIS

DESCRIPTION

Configures the 802.1X global settings for rate control, protocol parameters, and session parameters.

SUBCOMMANDS

OPTIONS

`waitForCompletion`

Wait for all interfaces to report a change in their state. If there is no change in the state of any interface for 100 seconds, this will time out. This does not mean there was a failure; it is a means of early detection of a potential problem.

Note: IxLoad ignores the *Wait For Completion* parameter when *Trigger Origin* is set to "Internal" and *Trigger Type* is set to one of the DHCP types (DHCP, DHCP_ARP, DHCP_GratArp, DHCP_ICMP).

Default value = "True"

`fragmentSize`

The maximum size of a fragment that can be sent on the wire for TLS fragments that comprise the phase 1 conversation (tunnel establishment).

The default is 1400, the minimum is 500, and the maximum is 1400.

Default value = "1400"

`waitBeforeRun`

Specifies a delay (in seconds) before EAPoUDP negotiations are started on the Ixia port.

This option is useful when the DUT does not immediately bring up the link and the protocol properties of the port.

Note that this option also has the effect of delaying the start of DHCP protocol negotiation on the port.

The default value is zero seconds, and the maximum is 600 seconds.

Default value="0"

`certInfo`

Name of object describing authentication certificate to be used. This object must be a `CertInfo` object.

Default value="None"

`maxClientsPerSecond`

The rate at which internal triggers are sent, per second.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value="10"

`maxOutstandingRequests`

The threshold at which `IxLoad` begins throttling back the number of triggers being sent.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value="10"

`dutTestMode`

Indicates the DUT's mode of authentication. The selected *DUT Test Mode* is applied to each physical port assigned to the test. See `DutTestModeChoices` for the list of options.

Default value = "MultiAuth"

`authOnNoResponse`

If `true`, if the DUT does not respond to EAPOL Start messages after the configured number of retries, declare the session a success.

Default value = "False"

`disableLogoff`

If `true`, do not send the EAPOL Logoff frame at the end of the authentication session.

Default value = "False"

`onlyMulticast`

If `true`, the destination address in all outgoing EAPOL messages from the Supplicants will be set to the multicast address 01:80:C2:00:00:03. Otherwise, once the authenticator MAC address is known, the Supplicants will use that MAC address.

When a frame is received with a destination address of 01:80:C2:00:00:03, it is distributed to active 802.1X supplicants, with respect to VLAN tagging:



- if the incoming frame is tagged, all corresponding VLAN-tagged active 802.1X clients will receive this frame.
- If the frame is not tagged, all active non-tagged supplicants will receive this frame.

Default value = "True"

authenticator

Default value = "01:80:c2:00:00:03"

authenticatorMCastAddressing

Default value = "False"

macAuthPrefix

When using machine authentication, a prefix is needed to differentiate between users and machines. The *Machine Auth Prefix* parameter is used by ACS to distinguish between users and machines (hosts). Note that an IAS server does not need this distinction, as a machine name always has a dollar sign (\$) at the end of the name (such as ixhost1\$).

Default value = "host/"

maxStart

The number of times to send EAPoL Start frames for which no response is received before declaring that the sessions have timed out. (A session timeout occurs if no frames are received from the authenticator).

Note, however, that if the *Authorized on No Response* global option is checked (see description below), session establishment will be declared a success.

This parameter can vary between 1 and 100.

Default value = "3"

successiveStart

The number of EAPoL Start messages sent when the supplicant starts the process of authentication.

This parameter can vary between 1 and 100, with a default of 1.

Default value = "1"

authPeriod

The maximum time interval, measured in seconds, that a supplicant will wait for an authenticator response. This interval is measured from the time that the supplicant sends an EAP-Response message. If the *Authentication Period* expires without an authenticator response, the authentication session is considered to have failed.

The default is 30, the minimum is 1, and the maximum is 3600 seconds.

Default value = "30"

`startPeriod`

The time interval between successive EAPOL Start messages sent by a Supplicant, if the Authenticator does not respond (that is, does not send Request-ID messages).

This parameter can vary between 1 and 3600 seconds, with a default of 30 seconds.

Default value = "30"

`heldPeriod`

Default value = "60"

`statsPeriod`

The time interval between successive EAPOL Start messages sent by a Supplicant, if the Authenticator does not respond (that is, does not send Request-ID messages).

This parameter can vary between 1 and 3600 seconds, with a default of 30 seconds. Default value = "5"

`nlogoffMaxClientsPerSecond`

The number of interfaces to log off per second.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value = "10"

`logoffMaxOutstandingRequests`

Default value = "10"

EXAMPLE

SEE ALSO



DutTestModeChoices

SYNOPSIS

DESCRIPTION

List of choices for the `dutTestMode` option of the `Dot1xSessionData` object.

SUBCOMMANDS

OPTIONS

Option	Description
SingleHost	Single-Host – one Supplicant is connected to one authenticator port.
MultiHost	Multi-Host – Multi-Host supports a configuration in which multiple hosts are connected to a single authenticator port. Only one supplicant performs authentication and the rest of the supplicants on that physical port are automatically enabled by the DUT. If the authenticated supplicant logs off, then the DUT will initiate a new authentication with a different supplicant.
MultiAuth	Multi-Auth – each of the supplicants on a physical port needs to be individually authenticated.

EXAMPLE

SEE ALSO



Dot1xRangeList

SYNOPSIS

DESCRIPTION

List of `Dot1xRange` objects for use by a `Dot1xPlugin` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

Dot1xRange

SYNOPSIS

DESCRIPTION

Creates a range of names and passwords for use in a `Dot1xRangeList` object.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`targetRange`

The MAC Range to authorize with this 802.1X Range.

Default value = "None"

`nacSequence`

The NAC Sequence used by this range.

Note: If a NAC Sequence has been selected for an 802.1X range, but you do not want a NAC Sequence associated with the range, specify `None` to remove the NAC Sequence from the range configuration.

Default value = "None"

`fastProvisionMode`

FAST provisioning mode specifies how the tunnel PAC is acquired. See `FastProvisionModeChoices` for the list of choices.



Default value="unauthenticated"

`fastInnerMethod`

FAST inner method See `FastInnerMethodChoices` for the list of choices.

Default value = "GTC"

`fastStatelessResume`

FAST stateless resume mode. See `FastStatelessResumeMethodChoices` for the list of choices.

Default value="no"

`userName`

The Username used to authenticate the port.

Default value = "username_1_1_1_file"

`userPassword`

The User Password used to authenticate the port.

Default value = "userpass_1_1_1_file"

`waitId`

defaultValue="False"

`protocol`

The Authentication Protocol that this 802.1X range will use.

The choices are:

- TLS
- PEAPv0
- PEAPv1
- MD5
- TTLS
- FAST

When you choose FAST as the protocol, you need to also configure three more options (described below): FAST Provisioning, FAST Inner Method, and FAST Stateless.

defaultValue="MD5"

hostAuthMode

Host (Machine) Authentication Method that this 802.1X range will use. See `HostAuthModeChoices` for the list of choices:

```
defaultValue="None"
```

hostName

The Machine Name used to authenticate the port.

```
defaultValue="hostname_1_1_1_file" />
```

hostPassword

The Machine Password used to authenticate the port.

```
defaultValue="hostpass_1_1_1_file" />
```

EXAMPLE

SEE ALSO



HostAuthModeChoices

SYNOPSIS

DESCRIPTION

List of host authentication mode choices for the `hostAuthMode` option of the `Dot1xRange` object.

SUBCOMMANDS

OPTIONS

Parameter	Description
None	None – No machine account is used.
HOST-ONLY	Host Only – When this mode is used, the Ixia port emulates only the host for each MAC address (but not the user). Nonetheless, the username certificate will be requested and transferred to the chassis.
HOST-USER-REAUTH	Host User-Reauth – When this mode is used, the Ixia port emulates both the host and the user for each MAC address. This emulates a case in which a domain machine that is started, the domain policies are applied (machine authentication), and multiple users then successfully log in. For each host/user pair, the host is authenticated first, followed by the user. Note that for user authentication to take place, the DUT must re-trigger authentication.
HOST-USER-BOTH	Host User-Both – When this mode is used, the Ixia port emulates both the host and the user for each MAC address. This emulates a case in which a domain machine is started, a domain policy is deployed, a user logs in, then a reboot occurs; the reboot starts the cycle again.

EXAMPLE

SEE ALSO

Dot1x Plugin Example

This section shows an example of how to create an 802.1x plugin in the Tcl API.

Dot1x Plugin Example

```
#####
# IxLoad ScriptGen created TCL script
# Network2 serialized using version 4.10.0.79
# 802.1x.tcl made on Aug 14 2008 15:35
#####

set Network2 [::IxLoad new ixNetworkGroup $chassisChain]
$Network2 config \
    -comment          "" \
    -name              "Network2" \
    -macMappingMode   0 \
    -linkLayerOptions 0

$Network2 globalPlugins.clear

set GratARP_1 [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network2 globalPlugins.appendItem -object $GratARP_1

$GratARP_1 config \
    -enabled          true \
    -name              "GratARP-1"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.



Dot1x Plugin Example

```
set TCP_1 [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network2 globalPlugins.appendItem -object $TCP_1

$TCP_1 config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
    -name "TCP-1" \
```

Configure the TCP
portion of the stack.

Dot1x Plugin Example

```

set Settings_1 [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network2 globalPlugins.appendItem -object $Settings_1

$Settings_1 config \
    -teardownInterfaceWithUser          false \
    -name                                "Settings-1" \
    -interfaceBehavior                   0

set Ethernet_2 [$Network2 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                    "master" \
    -negotiateMasterSlave                true

$Ethernet_2 config \
    -advertise10Full                     true \
    -name                                 "Ethernet-2" \
    -autoNegotiate                       true \
    -advertise100Half                    true \
    -advertise10Half                     true \
    -speed                                "k100FD" \
    -advertise1000Full                   true \
    -advertise100Full                    true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_2 childrenList.clear

set MAC_VLAN_13 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_2 childrenList.appendItem -object $MAC_VLAN_13

$MAC_VLAN_13 config \
    -name                                 "MAC/VLAN-13"

$MAC_VLAN_13 childrenList.clear

```

Configure the Dynamic Control plane settings.

Configure the physical layer properties.

Configure the MAC addresses and VLAN tags.



Dot1x Plugin Example

```
set IP_6 [::IxLoad new ixNetIpV4V6Plugin]
# ixNet objects needs to be added in the list
# before they are configured!
$MAC_VLAN_13 childrenList.appendItem -object $IP_6

$IP_6 config \
    -name "IP-6"

$IP_6 childrenList.clear
$IP_6 extensionList.clear

$MAC_VLAN_13 extensionList.clear

set 802_1X_1 [::IxLoad new ixNetDot1xPlugin]
# ixNet objects needs to be added in the list before they are
# configured!
$MAC_VLAN_13 extensionList.appendItem -object $802_1X_1

$802_1X_1 config \
    -name "802-1X-1"

$Ethernet_2 extensionList.clear
```

Create an IP
address range.

Clear the lists of
extension protocols.

Configure an
802.1x plugin.

Dot1x Plugin Example

```
set IP_R2 [::IxLoad new ixNetIpV4V6Range]
# ixNet objects needs to be added in the list
# before they are configured!
$IP_2 rangeList.appendItem -object $IP_R2

$IP_R2 config \
    -count 1 \
    -name "IP-R2" \
    -gatewayAddress "0.0.0.0" \
    -enabled true \
    -autoMacGeneration true \
    -mss 1460 \
    -incrementBy "0.0.0.1" \
    -prefix 16 \
    -gatewayIncrement "0.0.0.0" \
    -gatewayIncrementMode "perSubnet" \
    -generateStatistics false \
    -ipAddress "10.10.0.2" \
    -ipType "IPv4"
```

Configure the IP address range.



Dot1x Plugin Example

```
set 802_1X_R1 [$MAC_R12 getExtensionRange $802_1X_1]

$802_1X_R1 config \
  -userName
"username_1_1_1_file" \
  -fastProvisionMode          "unauthenticated" \
  -protocol                   "MD5" \
  -name                       "802-1X-R1" \
  -userPassword
"userpass_1_1_1_file" \
  -hostName
"hostname_1_1_1_file" \
  -enabled                    true \
  -fastInnerMethod            "GTC" \
  -hostAuthMode               "None" \
  -fastStatelessResume       "no" \
  -hostPassword
"hostpass_1_1_1_file" \
  -waitId                     false

set VLAN_R1 [$IP_R6 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
  -incrementStep              1 \
  -uniqueCount                4094 \
  -name                       "VLAN-R1" \
  -innerIncrement             1 \
  -innerUniqueCount           4094 \
  -enabled                    true \
  -innerFirstId               1 \
  -increment                  1 \
  -priority                   1 \
  -firstId                    1 \
  -innerIncrementStep         1 \
  -idIncrMode                 2 \
  -innerEnable                false \
  -innerPriority               1
```

Configure the 802.1x parameters.

Configure the VLAN tags for the IP range.

Nacl3Plugin

SYNOPSIS

DESCRIPTION

Configures an NAC Layer 3 plugin.

SUBCOMMANDS

OPTIONS

name

Name of the instance of the plugin.

Default value = "None"

rangeList

Name of the address range list used by this plugin. This must be a `Nacl3RangeList` object.

Default value = "None"

EXAMPLE

```
set EAPoUDP_1 [::IxLoad new ixNetNacl3Plugin]
# ixNet objects needs to be added in the list before they are configured!
$IIP_5 extensionList.appendItem -object $EAPoUDP_1

$EAPoUDP_1 config \
    -name "EAPoUDP-1"

$MAC_VLAN_11 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO



Nacl3SessionData

SYNOPSIS

DESCRIPTION

Configures the EAPoUDP Rate Control and Protocol Parameters.

SUBCOMMANDS

OPTIONS

`waitForCompletion`

Wait for all interfaces to report a change in their state. If there is no change in the state of any interface for 100 seconds, this will time out. This does not mean there was a failure; it is a means of early detection of a potential problem.

Note: IxLoad ignores the *Wait For Completion* parameter when *Trigger Origin* is set to "Internal" and *Trigger Type* is set to one of the DHCP types (DHCP, DHCP_ARP, DHCP_GratArp, DHCP_ICMP).

Default value = "True"

`fragmentSize`

The maximum size of a fragment that can be sent on the wire for TLS fragments that comprise the phase 1 conversation (tunnel establishment).

The default is 1400, the minimum is 500, and the maximum is 1400.

Default value = "1400"

`waitBeforeRun`

Specifies a delay (in seconds) before EAPoUDP negotiations are started on the Ixia port.

This option is useful when the DUT does not immediately bring up the link and the protocol properties of the port.

Note that this option also has the effect of delaying the start of DHCP protocol negotiation on the port.

The default value is zero seconds, and the maximum is 600 seconds.



Default value="0"

`certInfo`

Name of object describing authentication certificate to be used. This object must be a `CertInfo` object.

Default value="None"

`maxClientsPerSecond`

The rate at which internal triggers are sent, per second.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value="10"

`maxOutstandingRequests`

The threshold at which `IxLoad` begins throttling back the number of triggers being sent.

The default is 10, the minimum is 1, and the maximum is 1024.

Default value="10"

`triggerOrigin`

Specifies the trigger origination:

- Internal – The EAPoUDP protocol instance running on this port sends the trigger to the destination.
- External – Another protocol or application sends the trigger to the destination.

Default value = "External"

`triggerType`

When the trigger origin is *Internal*, this parameter specifies the type of trigger. The trigger types are:

- GratArp (Gratuitous ARP)
- ARP
- ICMP
- DHCP
- DHCP_ARP
- DHCP_GratArp
- DHCP_ICMP

For the `DHCP_ARP`, `DHCP_GratArp`, and `DHCP_ICMP` triggers, the message (ARP, GratArp, or ICMP) is sent following IP address allocation.

Default value = "gratArp"

triggerCount

When the trigger origin is *Internal*, this parameter specifies how many trigger messages are sent when a client is started.

In some cases, sending a single trigger message does not start the EAPoUDP process. This parameter allows you to send multiple messages for a trigger.

The maximum number of trigger messages that you can send is 1000.

Default value = "1"

port

The UDP port on which IxLoad listens for requests from the DUT.

The default is 21862 (0x5566 hex), the minimum is 1, and the maximum is 65535.

Default value = "21862"

cookieSize

When sending EAPoUDP Hello messages, this parameter specifies the length of the TLV named *cookie*.

Default value="4"

changeStatusQuery

The number of EAP-StatusQuery queries that will occur before IxLoad announces that a status change has occurred (a full posturing occurs).

The default is zero (meaning that a change will not be announced to the DUT), the minimum is zero, and the maximum is 65535.

Default value = "0"

timeout

The number of seconds to wait for the DUT to send a packet.

This field is used to indicate that the DUT has not sent an EAPoUDP message to an Ixia emulated client for a specified period of time. When the trigger origin is *External*, the timer is enabled following receipt of the first message from the DUT. If a timeout occurs, the timer is not restarted until the DUT sends another EAPoUDP message.

This field provides a method to determine whether or not the DUT is successfully tracking all the clients. If the timeout statistics are not zero, that usually indicates a problem on the authenticator side. It may indicate, for example, that the DUT does not have sufficient capacity to satisfy all the client authentication requests.

Default value="600"

EXAMPLE

SEE ALSO

Nacl3RangeList

SYNOPSIS

DESCRIPTION

List of `Nacl3Range` objects.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



Nacl3Range

SYNOPSIS

DESCRIPTION

Configures the EAPoUDP Range Parameters.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`targetRange`

The MAC Range to authorize with this 802.1X Range.

Default value = "None"

`nacSequence`

The NAC Sequence used by this range.

Note: If a NAC Sequence has been selected for an 802.1X range, but you do not want a NAC Sequence associated with the range, specify `None` to remove the NAC Sequence from the range configuration.

Default value = "None"

`fastProvisionMode`

FAST provisioning mode specifies how the tunnel PAC is acquired. See `FastProvisionModeChoices` for the list of choices.

Default value="unauthenticated"

`fastInnerMethod`

FAST inner method See `FastInnerMethodChoices` for the list of choices.

Default value = "GTC"

`fastStatelessResume`

FAST stateless resume mode. See `FastStatelessResumeMethodChoices` for the list of choices.

Default value="no"

`userName`

The `UserName` used to authenticate the port.

Default value = "username_1_1_1_file"

`userPassword`

The User Password used to authenticate the port.

Default value = "userpass_1_1_1_file"

`protocol`

The Authentication Protocol that this EAPoUDP range will use. The choices are:

- PEAPv1
- FAST

When you choose FAST as the protocol, you need to also configure three more options: Fast Provisioning, Fast Inner Method, and Fast Stateless.

Default value = "PEAPv1"

`responseType`

The types of EAPoUDP messages to which the range responds. See `ResponseTypeChoices` for the list of choices.

Default value = "RespondToAll"

`expectedSystemToken`



The expected system token. See `ExpectedSystemTokenChoices` for the list of choices.

Default value = "0; 10; 15; 20; 30; 100"

EXAMPLE

SEE ALSO

ResponseTypeChoices

SYNOPSIS

DESCRIPTION

List of choices for the `ResponseType` option of the `Nacl3Range` object.

SUBCOMMANDS

OPTIONS

Value	Description
<code>RespondToAll</code>	Respond To All – Respond to all EAPoUDP messages.
<code>NoStatusQuery</code>	No Status Query – Do not respond to EAP-StatusQuery messages. The lack of a response to these messages triggers a new full authorization exchange.
<code>IgnoreAll</code>	Ignore All – Do not respond to any EAPoUDP messages. This simulates a non-responsive host.

EXAMPLE

SEE ALSO



ExpectedSystemTokenChoices

SYNOPSIS

DESCRIPTION

List of choices for the `ExpectedSystemToken` option of the `Nacl3Range` object.

SUBCOMMANDS

OPTIONS

Value	Description
0	Healthy
10	Checkup
15	Transition
20	Quarantine
30	Infected
100	Unknown

EXAMPLE

SEE ALSO

NacSettings

SYNOPSIS

DESCRIPTION

Returns the range level EAPoUDP parameters.

SUBCOMMANDS

OPTIONS

`nacSequences`

The NAC sequence used by this range. All EAPoUDP ranges must set a NAC Sequence value.

Default value = "None"

`nacPostures`

List of NAC postures. See `NacPostureList` for the list of postures.

Default value = "None"

`nacTlvs`

List of NAC TLVs. See `NacTLVList` for the list of NacTLVs.

Default value = "None"

`vendorIdList`

List of vendor IDs. See `NacTlvVendorIdList` for the list of vendor IDs.

Default value = "None"

EXAMPLE

SEE ALSO

Nacl3PortGroupData

SYNOPSIS

DESCRIPTION

Configures an EAPoUDP port group.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`overrideGlobalSetupRate`

If selected, the setup and teardown parameters defined at the network group level will override those defined at the global level.

For example, if you have set the initial setup rate to 80 on the global level, and you have defined two ports, these 80 session setups will be evenly distributed across the ports (40 for each). If you then enable Override Global Rate Controls, you can modify the number of session setups for each of the ports (such as changing the distribution from 40-40 to 60-20).

Default value="False"

`maxClientsPerSecond`

The maximum number of 802.1X sessions to set up, per second.

Default value="10"

`maxOutstandingRequests`



The maximum number of 802.1X sessions that can be outstanding at any given time.

Default value="10"

`icmpTriggerTargetAddress`

Specifies one of the following:

- the IP address to which ICMP messages are sent, when ICMP internal triggers are used
- the target IP address, when ARP triggers are sent

Default value = "60.70.80.90"

`dutMac`

Specifies the MAC address of the trigger target IP address. The address is required when the trigger is ICMP and internal triggering is used. (Without this address, the DUT will drop the frame.)

Default value = "aa:bb:cc:dd:ee:ff"

EXAMPLE

SEE ALSO

NacSequenceList

SYNOPSIS

DESCRIPTION

List of NacSequence objects for use in the NacSettings object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



NacSequence

SYNOPSIS

DESCRIPTION

Defines a NAC sequence for use in a `NacSequenceList`.

SUBCOMMANDS

OPTIONS

`name`

Name of the NAC sequence.

Default value = "None"

`nacPostures`

Name of the `NacPostureList` object.

Default value = "None"

EXAMPLE

```
set Sequence_1 [::IxLoad new ixNetNacSequence]
$Sequence_1 config \
    -name                "Sequence-1"

$Sequence_1 nacPostures.clear
```

SEE ALSO

NacPostureList

SYNOPSIS

DESCRIPTION

List of `NacPosture` objects for use in a `NacSequence` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



NacPostureList2

SYNOPSIS

DESCRIPTION

List of NAC postures.

SUBCOMMANDS

OPTIONS

See NacPosture.

EXAMPLE

SEE ALSO

NacPosture

SYNOPSIS

DESCRIPTION

Defines a NAC posture for use in a `NacPostureList`.

SUBCOMMANDS

OPTIONS

`name`

Name of the NAC posture.

Default value = "None"

`expectedSystemToken`

The expected system token returned from the NAC Assessment Server. There are six predefined, non-configurable posture tokens.

Value	Description
-1	(Default) Not Set – The default value. It indicates that you have not selected one of the predefined tokens. You will use this token only if you are not interested in the expected system token values. When the token is <i>Not Set</i> , you will not be able to determine what is and is not matched.
0	Healthy – The endpoint device complies with the required credentials.
10	Checkup – The endpoint device is within the policy but does not have the latest security software; update recommended.
15	Transition – A posture check is in progress on the endpoint device. The device is given interim access pending a result from a full posture validation. This token is applicable during host startup where all services may not be running or while audit results are not yet available.
20	Quarantine – The endpoint device is out of policy and needs to be restricted to a remediation network. The device is not actively placing a threat on other hosts, but is susceptible to attack or infection.
30	Infected – The endpoint device is an active threat to other hosts; network access should be severely restricted and placed into remediation or totally denied all network access.



100	Unknown – The posture credentials of the endpoint device cannot be determined. Quarantine the host and audit, or remediate until a definitive posture can be determined.
-----	--

`nacTlvs`

Name of the `NacTLVList` for this posture.

Default value = "None"

`selected`

If true, this posture will be used.

Default value = "False"

EXAMPLE

SEE ALSO

NacTLVList

SYNOPSIS

DESCRIPTION

List of NacTLV objects for use in a NacPosture object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



NacTLVList2

SYNOPSIS

DESCRIPTION

List of NacTLV objects for use in a NacPosture object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

NacTLV

SYNOPSIS

DESCRIPTION

Defines a NacTLV object (Application State List) for use in a NacTLVList.

SUBCOMMANDS

OPTIONS

name

Name of the NAC Type-Length-Value attribute.

Default value = "None"

vendorRef

The name of the vendor that produced the plug-in.

Default value = "None"

vendorId

Vendor ID. See NacTlvVendorId for the list of IDs.

Default value="0"

appTypeRef

The type of agent in use.

Default value = "None"

appType

Application type. See NacTlvAppTypeList for the valid list of Application Types.

Default value = "0"

appCodeRef



Name of the list of Application Codes. See `NacTlvAppCodeList` for the list of application codes.

Default value = "None"

`appCode`

An attribute that the agent supports.

Default value = "0"

`avpType`

Value type. The value types can be any of the following:

- Array
- Integer
- Unsigned integer
- String
- IPv4
- IPv6
- Time (4 octets)
- Version
- Boolean

Default value = "1"

`value`

The value returned.

`defaultValue="Linux"`

`selected`

If true, this `NacTLVObject` is used.

Default value = "False"

EXAMPLE

SEE ALSO



NacTlvVendorIdList

SYNOPSIS

DESCRIPTION

List of NacTlvVendorId objects.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

NacTlvVendorId

SYNOPSIS

DESCRIPTION

Associates a vendor with a vendor ID.

SUBCOMMANDS

OPTIONS

`name`

Vendor name.

Default value = "None"

`value`

Value for vendor ID.

Default value = "1"

`appTypeList`

Name of the ApplicationTypeList associated with this vendor ID. This must be a `NacTlvAppTypeList` object.

Default value = "None"

EXAMPLE

SEE ALSO



NacTlvAppTypeList

SYNOPSIS

DESCRIPTION

List of `NacTlvAppType` objects for use in a `NacTlvVendorId` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

NacTlvAppType

SYNOPSIS

DESCRIPTION

Defines a `NacTlvAppType` object for use in a `NacTlvAppTypeList`.

SUBCOMMANDS

OPTIONS

`name`

The name of the application state.

Default value = "None"

`value`

Value for the application state.

Default value = "1"

`appCodeList`

Name of the list of application codes.

Default value = "None"

EXAMPLE

SEE ALSO



NacTlvAppCodeList

SYNOPSIS

DESCRIPTION

List of `NacTlvAppCode` objects for use in a `NacTlvVendorId` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

NacTlvAppCode

SYNOPSIS

DESCRIPTION

Defines a NacTlvAppCode object for use in a NacTlvAppCodeList.

SUBCOMMANDS

OPTIONS

name

Name of the application.

Default value = "None"

value

Value for the application.

Default value = "1"

EXAMPLE

SEE ALSO



Nacl3 (EAPoUDP) Plugin Example

This section shows an example of how to create an NAC Layer 3 / EAPoUDP plugin in the Tcl API.

Nacl3 (EAPoUDP) Plugin Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.10.0.79
# NAcl3_(EAPoUDP).tcl made on Aug 27 2008 16:31
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              "Network1" \
    -macMappingMode   0 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear

set GratARP_1 [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP_1

$GratARP_1 config \
    -enabled          true \
    -name              "GratARP-1"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.

Nacl3 (EAPoUDP) Plugin Example

```

set TCP_1 [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP_1

$TCP_1 config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
    -name "TCP-1" \
    -tcp_max_syn_backlog 1024 \

```

Configure the TCP portion of the stack.



Nacl3 (EAPoUDP) Plugin Example

```
set Settings_1 [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings_1

$Settings_1 config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings-1" \
    -interfaceBehavior              0

set Ethernet_2 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new
ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                "master" \
    -negotiateMasterSlave          true

$Ethernet_2 config \
    -advertise10Full                true \
    -name                            "Ethernet-2" \
    -autoNegotiate                  true \
    -advertise100Half               true \
    -advertise10Half                true \
    -speed                           "k100FD" \
    -advertise1000Full              true \
    -advertise100Full               true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_2 childrenList.clear

set MAC_VLAN_2 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_2 childrenList.appendItem -object $MAC_VLAN_2

$MAC_VLAN_2 config \
    -name                            "MAC/VLAN-2"

$MAC_VLAN_2 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.

Configure the MAC
addresses and VLAN tags.

Nacl3 (EAPoUDP) Plugin Example

```
set IP_2 [::IxLoad new ixNetIpV4V6Plugin]
# ixNet objects needs to be added in the list before they are
configured!
$MAC_VLAN_2 childrenList.appendItem -object $IP_2

$IP_2 config \
    -name "IP-2"

$IP_2 childrenList.clear

$IP_2 extensionList.clear

set EAPoUDP_1 [::IxLoad new ixNetNacl3Plugin]
# ixNet objects needs to be added in the list
# before they are configured!
$IP_2 extensionList.appendItem -object $EAPoUDP_1

$EAPoUDP_1 config \
    -name "EAPoUDP-1"

$MAC_VLAN_2 extensionList.clear

$Ethernet_2 extensionList.clear
```

Create an IP
address range.

Configure an
EAPoUDP plugin.

Clear the lists of
extension protocols.



Nacl3 (EAPoUDP) Plugin Example

```
#####
# Setting the ranges starting with the plugin on top of the stack
#####
$IP_2 rangeList.clear

set IP_R2 [::IxLoad new ixNetIpV4V6Range]
# ixNet objects needs to be added in the list
# before they are configured!
$IP_2 rangeList.appendItem -object $IP_R2

$IP_R2 config \
    -count                1 \
    -name                  "IP-R2" \
    -gatewayAddress        "0.0.0.0" \
    -enabled               true \
    -autoMacGeneration     true \
    -mss                   1460 \
    -incrementBy           "0.0.0.1" \
    -prefix                16 \
    -gatewayIncrement      "0.0.0.0" \
    -gatewayIncrementMode  "perSubnet" \
    -generateStatistics    false \
    -ipAddress             "10.10.0.2" \
    -ipType                "IPv4"
```

Configure the IP range.

NacI3 (EAPoUDP) Plugin Example

```
set EOU_R1 [$IP_R2 getExtensionRange $EAPoUDP_1]

set Sequence_1 [::IxLoad new ixNetNacSequence]
$Sequence_1 config \
    -name "Sequence-1"

$Sequence_1 nacPostures.clear

$EOU_R1 config \
    -userName
"username_1_1_1_file" \
    -responseType "RespondToAll" \
    -fastProvisionMode "unauthenticated" \
    -protocol "PEAPv1" \
    -name "EOU-R1" \
    -userPassword "userpass_1_1_1_file" \
    -enabled true \
    -fastInnerMethod "GTC" \
    -expectedSystemToken "0; 10; 15; 20; 30; 100" \
    -fastStatelessResume "no" \
    -nacSequence $Sequence_1
```

Configure the NAC /
EAPoUDP parameters.



Nacl3 (EAPoUDP) Plugin Example

```
set MAC_R2 [$IP_R2 getLowerRelatedRange "MacRange"]

$MAC_R2 config \
  -count 1 \
  -name "MAC-R2" \
  -enabled true \
  -mtu 1500 \
  -mac "00:0A:0A:00:02:00" \
  -incrementBy "00:00:00:00:00:01"

set VLAN_R1 [$IP_R2 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
  -incrementStep 1 \
  -uniqueCount 4094 \
  -name "VLAN-R1" \
  -innerIncrement 1 \
  -innerUniqueCount 4094 \
  -enabled true \
  -innerFirstId 1 \
  -increment 1 \
  -priority 1 \
  -firstId 1 \
  -innerIncrementStep 1 \
  -idIncrMode 2 \
  -innerEnable false \
  -innerPriority 1
```

Configure the MAC addresses for the IP range.

Configure the VLAN tags for the IP range.

CertInfo

SYNOPSIS

DESCRIPTION

Defines a certificate for use in `certInfo` option of a `DNSSessionBase` object.

SUBCOMMANDS

OPTIONS

`useCertServer`

Enables Run-Time Certificate Generation. Select this option if you want the application to automatically request certificates from a SCEP server. If you select this option, you must configure the parameters for the Certificate Server that will generate the certificates. (The generated certificates are in .pem format.)

Default value = "False"

`getCACertOnly`

When using SCEP, request only the root certificate (CA self-signed certificate) from CA and do not request supplicant certificates.

Default value = "False"

`sendCACertOnly`

When transferring the certificates to a port, send only the root.pem or root.cer certificate (do not transfer certificates for the supplicants).

This option is needed because some protocols may behave differently with or without supplicant certificates. IAS is known to request client certificates for PEAP, but ACS does not.

Default value = "False"

`certFormat`

The format for the certificates. See `CertFormatChoices` for the list choices.

Default value = ".pem"



sameKeyFile

Set this option to `true` if the certificate and the private key are in the same file. In this case, the file `client.pem` holds the certificate and the key, where `client` is the supplicant name.

Set this option to `false` if the certificate and the private key are in separate files. In this case, there are two files: `client.pem` that holds the certificate and `client_key.pem` that holds the key, where `client` is the supplicant name.

This option is not available if certificates are stored in DER files (certificate format of `.cer`). In this case, there is always a certificate file `client.cer` and a private key file `client_key.cer`.

Note: If this option state is changed, all certificates stored in Certificate Location must be erased.

Default value = "True"

certPath

The folder in which the file containing the certificate is stored. IxLoad obtains the certificate file from this folder and transfers it to the port CPU.

Default value = "C:/Program Files/Ixia/authfiles"

keyPath

The folder in which the file containing the private key is stored. IxLoad obtains the key file from this folder and transfers it to the port CPU.

This option is valid if the `sameKeyFile` option is `false`.

Default value = "C:/Program Files/Ixia/authfiles"

serverURL

The URL specifying the path of the Certificate Authority (CA) Server. The server must be configured with the SCEP protocol.

The default URL is `http://localhost/certsrv/mscep/mscep.dll`. This default URL assumes that the host on which the Ixia application is running is also a Microsoft Certificate Authority server. The URL references the Microsoft Simple Certificate Enrollment Protocol DLL, which generates the certificates for the Microsoft CA server.

Note: IxLoad does not support a chained CA PKI deployment. The SCEP server must be a standalone root CA or an SCEP server able to offer certificates with:

- all application policies, or
- Key Usage for critical, digitalSignature, keyEncipherment, nonRepudiation and both OIDs:
 - OID=1.3.6.1.5.5.7.3.1; Server Authentication
 - OID=1.3.6.1.5.5.7.3.2; Client Authentication

The Microsoft setup for SCEP is composed of a standalone root CA and the latest SCEP extension downloaded from the Microsoft Web site.

Default value = "http://localhost/certsrv/mscep/mscep.dll"

company

Company name attribute for the certificate request.

Default value = "IXIA"

department

Department attribute for the certificate request.

Default value = "Networking"

city

City attribute for the certificate request.

Default value = "Calabasas"

state

State attribute for the certificate request.

Default value = "California"

country

Country attribute for the certificate request. See CountryChoices for the list of countries.

Default value = "US"

keyUsage

Key usage attribute for the certificate request.

The default value is: critical, digitalSignature, keyEncipherment, nonRepudiation.

Default value = "critical,digitalSignature,keyEncipherment,nonRepudiation"

keySize

Bit length of key.

Default value = "512"

altName



Alternative Subject Name attribute for the certificate request.

Default value = "ixia"

EXAMPLE

SEE ALSO

CertFormatChoices

SYNOPSIS

DESCRIPTION

List of valid choices for the `certFormat` option of `CertInfo`.

The format for the certificates. If *Run-Time Certificate Generation* is enabled, the certificates will be generated with the SCEP protocol in `.pem` format.

SUBCOMMANDS

OPTIONS

Value	Description
<code>.pem</code>	<code>.pem</code> (base 64 encoded)
<code>.cer</code>	<code>cer</code> (binary encoded)

EXAMPLE

SEE ALSO



CountryChoices

SYNOPSIS

DESCRIPTION

List of valid choices for the `country` option of `CertInfo`.

SUBCOMMANDS

OPTIONS

Value	Description
AX	Aland Islands
AD	Andorra
AE	United Arab Emirates
AF	Afghanistan
AG	Antigua and Barbuda
AI	Anguilla
AL	Albania
AM	Armenia
AN	Netherlands Antilles
AO	Angola
AQ	Antarctica
AR	Argentina
AS	American Samoa
AT	Austria
AU	Australia
AW	Aruba
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BD	Bangladesh
BE	Belgium
BF	Burkina Faso

BG	Bulgaria
BH	Bahrain
BI	Burundi
BJ	Benin
BM	Bermuda
BN	Brunei Darussalam
BO	Bolivia
BR	Brazil
BS	Bahamas
BT	Bhutan
BV	Bouvet Island
BW	Botswana
BY	Belarus
BZ	Belize
CA	Canada
CC	Cocos Keeling Islands
CF	Central African Republic
CG	Congo
CD	Congo, Democratic Republic of the
CH	Switzerland
CI	Cote D' Ivoire / Ivory Coast
CK	Cook Islands
CL	Chile
CM	Cameroon
CN	China
CO	Colombia
CR	Costa Rica
CS	Czechoslovakia former
CU	Cuba
CV	Cape Verde
CX	Christmas Island
CY	Cyprus
CZ	Czech Republic
DE	Germany
DJ	Djibouti
DK	Denmark



DM	Dominica
DO	Dominican Republic
DZ	Algeria
EC	Ecuador
EE	Estonia
EG	Egypt
EH	Western Sahara
ER	Eritrea
ES	Spain
ET	Ethiopia
FI	Finland
FJ	Fiji
FK	Falkland Islands / Malvinas
FM	Micronesia
FO	Faroe Islands
FR	France
FX	France Metropolitan
GA	Gabon
GB	Great Britain / UK
GD	Grenada
GE	Georgia
GF	French Guiana
GG	Guernsey
GH	Ghana
GI	Gibraltar
GL	Greenland
GM	Gambia
GN	Guinea"
GP	Guadeloupe
GQ	Equatorial Guinea
GR	Greece
GS	S. Georgia and S. Sandwich Islands
GT	Guatemala
GU	Guam
GW	Guinea Bissau
GY	Guyana

HK	Hong Kong
HM	Heard and McDonald Islands
HN	Honduras
HR	Croatia Hrvatska
HT	Haiti
HU	Hungary
ID	Indonesia
IE	Ireland
IL	Israel
IM	Isle of Man
IN	India
IO	British Indian Ocean Territory
IQ	Iraq
IR	Iran
IS	Iceland
IT	Italy
JE	Jersey
JM	Jamaica
JO	Jordan
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KH	Cambodia
KI	Kiribati
KM	Comoros
KN	Saint Kitts and Nevis
KP	Korea North
KR	Korea South
KW	Kuwait
KY	Cayman Islands
KZ	Kazakhstan
LA	Laos
LB	Lebanon
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka



LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
LY	Libya
MA	Morocco
MC	Monaco
MD	Moldova
ME	Montenegro
MG	Madagascar
MH	Marshall Islands
MK	Macedonia
ML	Mali
MM	Myanmar
MN	Mongolia
MO	Macau
MP	Northern Mariana Islands
MQ	Martinique
MR	Mauritania
MS	Montserrat
MT	Malta
MU	Mauritius
MV	Maldives
MW	Malawi
MX	Mexico
MY	Malaysia
MZ	Mozambique
NA	Namibia
NC	New Caledonia
NE	Niger
NF	Norfolk Island
NG	Nigeria
NI	Nicaragua
NL	Netherlands
NO	Norway

NP	Nepal
NR	Nauru
NT	Neutral Zone
NU	Niue
NZ	New Zealand Aotearoa
OM	Oman
PA	Panama
PE	Peru
PF	French Polynesia
PG	Papua New Guinea
PH	Philippines
PK	Pakistan
PL	Poland
PM	St Pierre and Miquelon
PN	Pitcairn
PR	Puerto Rico
PS	Palestinian Territory
PT	Portugal
PW	Palau
PY	Paraguay
QA	Qatar
RE	Reunion
RO	Romania
RS	Serbia RU Russian Federation
RW	Rwanda
SA	Saudi Arabia
SB	Solomon Islands
SC	Seychelles
SD	Sudan
SE	Sweden
SG	Singapore
SH	St. Helena
SI	Slovenia
SJ	Svalbard and Jan Mayen Islands
SK	Slovak Republic
SL	Sierra Leone



SM	San Marino
SN	Senegal
SO	Somalia
SR	Suriname
ST	Sao Tome and Principe
SU	USSR former
SV	El Salvador
SY	Syria
SZ	Swaziland
TC	Turks and Caicos Islands
TD	Chad
TF	French Southern Territories
TG	Togo
TH	Thailand
TJ	Tajikistan
TK	Tokelau
TM	Turkmenistan
TN	Tunisia
TO	Tonga
TP	East Timor
TR	Turkey
TT	Trinidad and Tobago
TV	Tuvalu
TW	Taiwan
TZ	Tanzania
UA	Ukraine
UG	Uganda
UM	US Minor Outlying Islands
US	United States
UY	Uruguay
UZ	Uzbekistan
VA	Vatican City State / Holy See
VC	Saint Vincent and the Grenadines
VE	Venezuela
VG	Virgin Islands British "
VI	Virgin Islands U S

VN	Viet Nam
VU	Vanuatu
WF	Wallis and Futuna Islands
WS	Samoa
YE	Yemen
YT	Mayotte
ZA	South Africa
ZM	Zambia
ZW	Zimbabwe

EXAMPLE

SEE ALSO

Impair Plugin

SYNOPSIS

DESCRIPTION

Defines an Impair plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`rangeList`

Name of the Impair range.

Default value = "None"

EXAMPLE

```
set Impair_1 [::IxLoad new ixNetImpairPlugin]
# ixNet objects needs to be added in the list before they are configured!
$IP_5 extensionList.appendItem -object $Impair_1

$Impair_1 config \
    -name "Impair-1"

$MAC_VLAN_11 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO



ImpairRangeList

SYNOPSIS

DESCRIPTION

List of `ImpairRange` objects for use by an `ImpairRange` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

ImpairRange

SYNOPSIS

DESCRIPTION

Defines the properties of the Impair range.

SUBCOMMANDS

OPTIONS

name

Name of the instance of the plugin.

Default value = "None"

enabled

If `True`, the range base is enabled.

Default value="True".

targetRange

The target MAC range.

Default value = "None"

profile

The name of the `ImpairProfile` object that contains the impairment settings used by this range.

defaultValue="None"

EXAMPLE

```
$IP_5 rangeList.clear
```

```
set IP_R5 [::IxLoad new ixNetIpV4V6Range]
```



ixNet objects needs to be added in the list before they are configured!

```
$IP_5 rangeList.appendItem -object $IP_R5
```

```
$IP_R5 config \
```

```
-count                1 \  
-name                 "IP-R5" \  
-gatewayAddress       "0.0.0.0" \  
-enabled              true \  
-autoMacGeneration    true \  
-mss                  1460 \  
-incrementBy          "0.0.0.1" \  
-prefix               16 \  
-gatewayIncrement     "0.0.0.0" \  
-gatewayIncrementMode "perSubnet" \  
-generateStatistics   false \  
-ipAddress             "10.10.0.6" \  
-ipType               "IPv4"
```

```
set Impair_R1 [$IP_R5 getExtensionRange $Impair_1]
```

```
set DefaultProfile [::IxLoad new ixNetImpairProfile]
```

```
$DefaultProfile config \
```

```
-addTcpFlagsFilter     false \  
-jitter                0 \  
-reorderPISkip         1 \  
-seed                  0 \  
-typeOfService         "any" \  
-dropSequenceLength    1 \
```

```
-protocol "any" \  
-addFragment false \  
-addBandwidth false \  
-delay 1 \  
-addDelay true \  
-impairOrder  
"Delay;Drop;DropSeq;Reorder;ReorderPI;Duplicate;Fragment;FragmentSeq;Bandwidth" \  
-sendFirstFragmentOnly false \  
-addDrop false \  
-reorderLength 1 \  
-addDuplicate false \  
-reorderPILength 1 \  
-sendOverlappingFragments false \  
-reorderPITimeout 1000 \  
-addReorderPI false \  
-reorder 1 \  
-addFragmentSequence false \  
-expectTcpFlags "SYN" \  
-destinationIp "any" \  
-fragmentSequenceSkip 1 \  
-addBandwidthIn false \  
-selectTcpFlags "SYN;RST;ACK" \  
-gap 1 \  
-destinationPort 0 \  
-fragmentSequenceLength 1 \  
-sourcePort 0 \  
-bandwidthUnitsIn "kbps" \  
-name "DefaultProfile" \  
-mtuSequence 1000 \  

```



```
-dropSequenceSkip          1 \  
-mtu                       1000 \  
-addReorder                false \  
-defaultp                  true \  
-bandwidthUnits            "kbps" \  
-reorderPIInterval        1 \  
-sourceIp                  "any" \  
-sendFragmentsInReverseOrder false \  
-addDropSequence          false  
  
$Impair_R1 config \  
-enabled                   true \  
-name                      "Impair-R1" \  
-profile                   $DefaultProfile  
  
set MAC_R10 [$IP_R5 getLowerRelatedRange "MacRange"]  
  
$MAC_R10 config \  
-count                     1 \  
-name                      "MAC-R10" \  
-enabled                   true \  
-mtu                       1500 \  
-mac                      "00:0A:0A:00:06:00" \  
-incrementBy               "00:00:00:00:00:01"  
  
set VLAN_R1 [$IP_R5 getLowerRelatedRange "VlanIdRange"]  
  
$VLAN_R1 config \  

```

```
-incrementStep          1 \  
-uniqueCount           4094 \  
-name                  "VLAN-R1" \  
-innerIncrement        1 \  
-innerUniqueCount      4094 \  
-enabled               true \  
-innerFirstId          1 \  
-increment             1 \  
-priority              1 \  
-firstId               1 \  
-innerIncrementStep    1 \  
-idIncrMode            2 \  
-innerEnable           false \  
-innerPriority         1
```

SEE ALSO



ImpairProfileList

SYNOPSIS

DESCRIPTION

List of Impair profiles to be used by the ImpairProfile in a ImpairSessionData object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

ImpairProfile

SYNOPSIS

DESCRIPTION

Defines a new impairment profile.

SUBCOMMANDS

OPTIONS

`name`

Name of this profile.

Default value = "None"

`defaulttp`

Set to True True to make this the default profile. When True, this profile is assigned to new impair ranges.

Default value = "False"

`seed`

The seed number for random number generation. Set to zero if the seed is random.

Default value = "0"

Delay parameters

`addDelay`

If `true`, this impairment is applied to the packet stream.

Default value = "True"

`delay`

The *Delay* impairment characteristic allows you to insert latency errors into a packet stream.



Specifies the delay for each packet, in milliseconds.

Default value = "1"

`jitter`

Specifies the jitter value, in milliseconds.

A random value from 0 ms to the Jitter value that you specify is added to or subtracted from your specified *Delay* value. Note that the Jitter value cannot be greater than *delay*.

Default value = "0"

Drop / Drop Sequence parameters

`addDrop`

The *Drop* impairment characteristic allows you emulate random packet loss from a packet stream.

If `true`, this impairment is applied to the packet stream.

Default value = "False"

`addDropSequence`

The *Drop Sequence* impairment characteristic allows you emulate sequential packet loss from a packet stream. In this case, a specified number of packets will be dropped at a specified interval.

If `true`, this impairment is applied to the packet stream.

Default value = "False"

`dropSequenceSkip`

The number of packets that will be transmitted before one or more packets are dropped.

Default value = "1"

`dropSequenceLength`

The number of packets that will be dropped.

The Drop Sequence setting specifies that a sequence of *d* packets is dropped after each transmitted sequence of *s* packets.

For example, if *s* = 2 and *d* = 3, the transmitted packets are:

1, 2, 6, 7, 11, 12, 16, 17, 21, 22.

Default value = "1"

Reorder parameters

`addReorder`

The *Reorder* impairment characteristic allows you emulate packet reordering based on a time delay. In this case, because some packets are delayed during transmission, they arrive out of order in the packet stream. The delay pattern repeats after a specified number of packets (number of packets skipped plus the number of packets delayed) have been sent.

If `true`, this impairment is applied to the packet stream.

Default value = "False"

`gap`

Specifies the number of packets to skip before reordering packet.

Default value = "1"

`reorder`

Specifies how long the packets are to be delayed (number of milliseconds).

Default value = "20"

`reorderLength`

Specifies the number of consecutive packets to reorder.

Default value = "1"

Reorder Sequence parameters

`addReorderPI`

The *Reorder Sequence* impairment characteristic allows you emulate delay caused by packet reordering. In this form of impairment, packets are delayed during transmission by reordering the packet interval.

If `true`, this impairment is applied to the packet stream.

Default value = "False"

`reorderPISkip`

The number of packets (*s*) to transmit prior to delaying the transmission of *d* packets.

Default value = "1"



`reorderPILength`

The number of packets (d) to take out of the stream for delayed transmission.

Default value = "1"

`reorderPIInterval`

The number of packets (m) to transmit before transmitting the d packets that were previously taken out of the stream.

Default value = "1"

`reorderPITimeout`

The maximum time that a packet may be delayed, specified in milliseconds.

Default value = "1000"

Duplicate parameters

`addDuplicate`

The *Duplicate* impairment characteristic allows you emulate the appearance of duplicate packets in a packet stream. In this case, a specified percentage of packets will be duplicated.

If `true`, this impairment is applied to the packet stream.

Default value = "False"

Fragment / Fragment Sequence parameters

`addFragment`

(Fragment only) The *Fragment* impairment characteristic allows you emulate various packet fragmentation scenarios.

If `true`, this impairment is applied to the packet stream.

Default value = "False"

`mtu`

(Fragment only) The maximum transmission unit for packets that will be fragmented.

Default value = "1000"

`addFragmentSequence`

(Fragment Sequence only) The *Fragment Sequence* impairment characteristic allows you to emulate various packet fragmentation scenarios. In this form of impairment, the packets selected for fragmentation are chosen based on a defined packet sequence.

If `true`, this impairment is applied to the packet stream.

Default value = "False"

`fragmentSequenceSkip`

(Fragment Sequence only) The number of packets (*s*) to skip before fragmenting packets.

Default value = "1"

`fragmentSequenceLength`

(Fragment Sequence only) The number of packets (*d*) to select from the stream for fragmentation.

Default value = "1"

`mtuSequence`

(Fragment Sequence only) The maximum transmission unit for the packets that will be fragmented.

MTU defines the packet size after fragmentation. For example, if MTU=220, the Impair plug-in breaks a packet of 661 bytes into 4 fragments.

Default value = "1000"

`sendFragmentsInReverseOrder`

(Fragment / Fragment Sequence) If `true`, transmit fragments in reverse order.

This setting allows testing of worst-case reassembly scenarios.

Default value = "False"

`sendFirstFragmentOnly`

(Fragment / Fragment Sequence) If `true`, transmit only the first fragment of each datagram. All other fragments are discarded.

If you also select *Reverse Fragments*, only the fragment that would have been sent last is sent.

This feature allows you to test reassembly timeout mechanisms.

Default value = "False"

`sendOverlappingFragments`



(Fragment / Fragment Sequence) If `true`, the IP stack creates and sends random, but legitimate, IP fragments whose data offset and length are randomly selected. In this case, that the receiving end detects overlapping data in the fragments it receives.

This setting is useful for testing reassembly mechanisms at the other end.

Default value = "False"

Inbound Rate / Outbound Rate parameters

`addBandwidth`

(Outbound Rate only) Adds an impairment characteristic to the outbound traffic that allows you to limit egress traffic speed, and thereby simulate a lower bandwidth network.

Default value = "False"

`bandwidthUnits`

(Outbound Rate only) The bandwidth unit to use. See `BandwidthUnitsChoices` for the list of choices.

Default value = "kbps"

`addBandwidthIn`

(Inbound Rate only) Adds an impairment characteristic to the inbound traffic that allows you to limit ingress traffic speed, and thereby simulate a lower bandwidth network.

Default value = "False"

`bandwidthUnitsIn`

(Inbound Rate only) The bandwidth unit to use. See `BandwidthUnitsInChoices` for the list of choices.

Default value = "kbps"

Packets to Impair parameters

`destinationIp`

A destination IP address and prefix on which to filter.

Impairment will be applied only on packets targeted to this destination.

You can specify a host address (such as 192.168.85.10/32) or a network address (such as 192.168.85.0/24).

You can also specify IPv6 addresses, both in the long form (such as 2008:0007:0031:0000:0000:0000:0000:0001/64), or in the short form (such as 2008:7:31::1/64).

The default value is *any* address (in which case, all packets are impaired).

Default value = "any"

`sourceIp`

A source IP address and prefix on which to filter.

Impairment will be applied only on packets received from the specified source.

You can specify a host address (such as 192.168.85.10/32) or a network address (such as 192.168.85.0/24).

You can also specify IPv6 addresses, both in the long form (such as 2008:0007:0031:0000:0000:0000:0001/64), or in the short form (such as 2008:7:31::1/64).

The default value is *any* address (in which case, all packets are impaired).

Default value = "any"

`sourcePort`

The source port number on which to filter.

Impairment will be applied to only those packets that have this source port number.

The default value is zero (in which case, all packets are impaired).

Default value = "0"

`destinationPort`

The destination port number on which to filter.

Impairment will be applied to only those packets that have this destination port number.

The default value is zero (in which case, all packets are impaired).

Default value = "0"

`protocol`

The type of protocol to which the impairment will be applied:

- any – all protocols
- ICMP
- TCP
- UDP
- ICMPv6

The default value is *any* protocol (in which case, all packets are impaired).



Default value = "any"

`typeOfService`

Indicates the Type of Service to which the impairment will be applied:

- any – all TOS
- Minimum Cost (0x02)
- Maximum Reliability (0x04)
- Maximum Throughput (0x08)
- Minimum Delay (0x10)
- Class 1 (0x20)
- Class 2 (0x40)
- Class 3 (0x60)
- Class 4 (0x80)
- Express Forwarding (0xA0)
- Control (0xC0)

You can also manually enter any custom TOS value (between 0x00 – 0xFF, or between 0 – 255).

The default value is *any* TOS value (in which case, all packets are impaired).

Default value = "any"

`addTcpFlagsFilter`

If `true`, impairment will be applied to only those TCP packets having specific TCP flags set, as specified in the Select TCP Flags and Expect TCP Flags fields.

The default setting is Unchecked. Selecting this parameter enables the Select TCP Flags and Expect TCP Flags fields.

Default value = "False"

`selectTcpFlags`

A comma-separated list of TCP flags to be examined on the packet. See `SelectTcpFlagsChoices` for the list of choices.

Default value = "SYN;RST;ACK"

`expectTcpFlags`

A comma-separated list of TCP flags that must be set in the packet for that packet to be selected.

Default value = "SYN"

impairOrder

A comma-separated list that defines the order that the impairments will be applied in. See ImpairOrderChoices for the list of choices.

Default value = "Delay;Drop;DropSeq;Reorder;ReorderPI;Duplicate;Fragment;FragmentSeq;Bandwidth"

EXAMPLE

```
set DefaultProfile [::IxLoad new ixNetImpairProfile]
$DefaultProfile config \
  -addTcpFlagsFilter                false \
  -jitter                          0 \
  -reorderPISkip                    1 \
  -seed                             0 \
  -typeOfService                    "any" \
  -dropSequenceLength              1 \
  -protocol                         "any" \
  -addFragment                      false \
  -addBandwidth                     false \
  -delay                            1 \
  -addDelay                         true \
  -impairOrder                      "Delay;Drop;DropSeq;Reorder;ReorderPI;Duplicate;Fragment;FragmentSeq;Bandwidth" \
  -sendFirstFragmentOnly           false \
  -addDrop                          false \
  -reorderLength                    1 \
  -addDuplicate                     false \
  -reorderPILength                  1 \
  -sendOverlappingFragments         false \
  -reorderPITimeout                 1000 \
  -addReorderPI                     false \
  -reorder                          1 \
  -addFragmentSequence              false \
  -expectTcpFlags                   "SYN" \
  -destinationIp                    "any" \
  -fragmentSequenceSkip             1 \
  -addBandwidthIn                   false \
  -selectTcpFlags                   "SYN;RST;ACK" \
  -gap                               1 \
  -destinationPort                  0 \
  -fragmentSequenceLength           1 \
  -sourcePort                       0 \
  -bandwidthUnitsIn                 "kbps" \
  -name                             "DefaultProfile" \
  -mtuSequence                      1000 \
  -dropSequenceSkip                 1 \
  -mtu                              1000 \
  -addReorder                       false \
  -defaultp                          true \
  -bandwidthUnits                   "kbps" \
  -reorderPIInterval                1 \
  -sourceIp                         "any" \
  -sendFragmentsInReverseOrder      false \
  -addDropSequence                  false
```

SEE ALSO



BandwidthUnitsChoices

SYNOPSIS

DESCRIPTION

List of bandwidth units used for an `addBandwidth` (outbound rate) impairment.

SUBCOMMANDS

OPTIONS

Value	Description
kbps	KByte/sec
kbit	KBit/sec
mbps	MByte/sec
mbit	MBit/sec

EXAMPLE

SEE ALSO



BandwidthUnitsInChoices

SYNOPSIS

DESCRIPTION

List of bandwidth units used for an `addBandwidthIn` (inbound rate) impairment.

SUBCOMMANDS

OPTIONS

Value	Description
kbps	KByte/sec
kbit	KBit/sec
mbps	MByte/sec
mbit	MBit/sec

EXAMPLE

SEE ALSO

SelectTcpFlagsChoices

SYNOPSIS

DESCRIPTION

List of TCP flags used for the `selectTcpFlags` option of the `ImpairProfile` object.

SUBCOMMANDS

OPTIONS

Value	Description
SYN	SYN flag
ACK	ACK flag
FIN	FIN flag
RST	RST flag
URG	URG flag
PSH	PSH flag
ECE	ECE flag
CWR	CWR flag
ALL	All flags
NONE	No flags

EXAMPLE

SEE ALSO



ExpectTcpFlagsChoices

SYNOPSIS

DESCRIPTION

List of TCP flags used for the `expectTcpFlags` option of the `ImpairProfile` object.

SUBCOMMANDS

OPTIONS

Value	Description
SYN	SYN flag
ACK	ACK flag
FIN	FIN flag
RST	RST flag
URG	URG flag
PSH	PSH flag
ECE	ECE flag
CWR	CWR flag
ALL	All flags
NONE	No flags

EXAMPLE

SEE ALSO

ImpairOrderChoices

SYNOPSIS

DESCRIPTION

List of choices for the `impairOrder` option of the `ImpairProfile` object.

SUBCOMMANDS

OPTIONS

Value	Description
Delay	Delay impairment
Drop	Drop impairment
DropSeq	Drop Sequence impairment
Reorder	Reorder impairment
ReorderPI	Reorder Sequence impairment
Duplicate	Duplicate impairment
Fragment	Fragment impairment
FragmentSeq	Fragment Sequence impairment
Bandwidth	Inbound / Outbound Rate impairment

EXAMPLE

SEE ALSO



ImpairSessionData

SYNOPSIS

DESCRIPTION

Configures the Impair global settings and session parameters.

SUBCOMMANDS

OPTIONS

`profiles`

Name of the list of Impair profiles to be used. This must be an `ImpairProfileList` object.

Default value = "None"

EXAMPLE

SEE ALSO

Impair Plugin Example

This section shows an example of how to create an Impair plugin in the Tcl API.

Impair Plugin Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.10.0.79
# Impair.tcl made on Aug 14 2008 15:31
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              "Network1" \
    -macMappingMode   0 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear

Begin appending items to global plugin list.

set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
    -enabled          true \
    -name              "GratARP"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.



Impair Plugin Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

Configure the TCP
portion of the stack.

```
$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -name "TCP" \
```

Impair Plugin Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior              0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                "master" \
    -negotiateMasterSlave           true

$Ethernet_1 config \
    -advertise10Full                true \
    -name                            "Ethernet-1" \
    -autoNegotiate                  true \
    -advertise100Half               true \
    -advertise10Half                true \
    -speed                          "k100FD" \
    -advertise1000Full              true \
    -advertise100Full               true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.



Impair Plugin Example

```
set MAC_VLAN_11 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_11

$MAC_VLAN_11 config \
    -name "MAC/VLAN-11"

$MAC_VLAN_11 childrenList.clear

set IP_5 [::IxLoad new ixNetIPv4V6Plugin]
# ixNet objects needs to be added in the
# list before they are configured!
$MAC_VLAN_11 childrenList.appendItem -object $IP_5

$IP_5 config \
    -name "IP-5"

$IP_5 childrenList.clear

$IP_5 extensionList.clear

set Impair_2 [::IxLoad new ixNetImpairPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$IP_5 extensionList.appendItem -object $Impair_2

$Impair_2 config \
    -name "Impair-2"

$MAC_VLAN_11 extensionList.clear
$Ethernet_1 extensionList.clear
```

Configure the MAC addresses and VLAN tags.

Create an IP range and append it to the MAC and VLAN ranges.

Create an Impair plugin and append it to the IP range as an extension protocol

Clear the remaining lists of extension protocols.

Impair Plugin Example

```
#####  
# Setting the ranges starting with the plugin on top of the stack  
#####  
$IP_5 rangeList.clear  
  
set IP_R5 [::IxLoad new ixNetIpV4V6Range]  
# ixNet objects needs to be added in the list before they are  
configured!  
$IP_5 rangeList.appendItem -object $IP_R5  
  
$IP_R5 config \  
-count 1 \  
-name "IP-R5" \  
-gatewayAddress "0.0.0.0" \  
-enabled true \  
-autoMacGeneration true \  
-mss 1460 \  
-incrementBy "0.0.0.1" \  
-prefix 16 \  
-gatewayIncrement "0.0.0.0" \  
-gatewayIncrementMode "perSubnet" \  
-generateStatistics false \  
-ipAddress "10.10.0.6" \  
-ipType "IPv4"  
  
set Impair_R2 [$IP_R5 getExtensionRange $Impair_2]
```

Configure an address range
and append it to the IP plugin.

Impair the address range by
appending it as an extension protocol.



Impair Plugin Example

```
set DefaultProfile [::IxLoad new ixNetImpairProfile]
$DefaultProfile config \
    -addTcpFlagsFilter                false \
    -jitter                          0 \
    -reorderPISkip                    1 \
    -seed                             0 \
    -typeOfService                    "any" \
    -dropSequenceLength              1 \
    -protocol                         "any" \
    -addFragment                     false \
    -addBandwidth                    false \
    -delay                            1 \
    -addDelay                         true \
    -impairOrder
"Delay;Drop;DropSeq;Reorder;ReorderPI;Duplicate;Fragment;FragmentSeq;Bandwidth" \
    -sendFirstFragmentOnly          false \
    -addDrop                        false \
    -reorderLength                  1 \
    -addDuplicate                   false \
    -reorderPILength               1 \
    -sendOverlappingFragments       false \
    -reorderPITimeout              1000 \
    -addReorderPI                  false \
    -reorder                        1 \
    -addFragmentSequence            false \
    -expectTcpFlags                 "SYN" \
    -destinationIp                  "any" \
    -fragmentSequenceSkip           1 \
    -addBandwidthIn                 false \
    -selectTcpFlags                 "SYN;RST;ACK" \
    -gap                             1 \
    -destinationPort                0 \
    -fragmentSequenceLength         1 \
    -sourcePort                     0 \
    -bandwidthUnitsIn               "Kbps" \
    -name                            "DefaultProfile" \
    -mtuSequence                    1000 \
    -dropSequenceSkip              1 \
    -mtu                            1000 \
    -addReorder                    false \
    -defaultp                       true \
    -bandwidthUnits                 "Kbps" \
    -reorderPIInterval              1 \
    -sourceIp                       "any" \
    -sendFragmentsInReverseOrder    false \
    -addDropSequence                false
```

Configure an Impair profile.

Impair Plugin Example

```

$Impair_R2 config \
  -enabled      true \
  -name         "Impair-R2" \
  -profile      $DefaultProfile

set MAC_R10 [$IP_R5 getLowerRelatedRange "MacRange"]

$MAC_R10 config \
  -count        1 \
  -name         "MAC-R10" \
  -enabled      true \
  -mtu          1500 \
  -mac          "00:0A:0A:00:06:00" \
  -incrementBy  "00:00:00:00:00:01"

set VLAN_R1 [$IP_R5 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
  -incrementStep 1 \
  -uniqueCount   4094 \
  -name          "VLAN-R1" \
  -innerIncrement 1 \
  -innerUniqueCount 4094 \
  -enabled       true \
  -innerFirstId  1 \
  -increment     1 \
  -priority      1 \
  -firstId       1 \
  -innerIncrementStep 1 \
  -idIncrMode    2 \
  -innerEnable   false \
  -innerPriority  1

```

Name the Impair profile.

Configure the MAC addresses for the IP range.

Configure the VLAN tags for the IP range.

IPSec Plugin

SYNOPSIS

DESCRIPTION

Configures an IPSec plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`rangeList`

Name of the list of ranges used by this plugin.

Default value = "None"

EXAMPLE

```
set IPsec_1 [::IxLoad new ixNetIPSecPlugin]
# ixNet objects needs to be added in the list before they are configured!
$IP_3 childrenList.appendItem -object $IPSec_1

$IPSec_1 config \
    -name "IPSec-1"

$IPSec_1 childrenList.clear

$IPSec_1 extensionList.clear

$IP_3 extensionList.clear

$Emulated_Router_1 extensionList.clear

$MAC_VLAN_8 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO



IPSecRangeList

SYNOPSIS

DESCRIPTION

List of `IPSecRange` objects for use by a `IPSecPlugin` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

IPSecRange

SYNOPSIS

DESCRIPTION

Creates an IPSec address range for addition to an `IPSecRangeList` object.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

Basic Parameters

`ikeVersion`

The keying protocol to be used for the tunnel negotiation phase:

- IKEv1: Use IKE version 1 to establish security associations between IPsec peers.
- IKEv2: Use IKE version 2 to establish security associations between IPsec peers.
- Manual: Use manual keying to configure the security policy options. In this case, you configure the keys in the Keys grid (refer to IPsec Range Parameters - Keys). Note that manual keying is not supported in Ixia port-to-port configurations.

The default is IKEv2.

Note: All ranges within a must be configured with the same *IKE Version* (IKEv1, IKEv2, or Manual).

Values= "ikev1", "ikev2", "manual", Default value = "ikev1"

`testScenario`

The type of IPsec test scenario for which you are defining this configuration:



- Site to Site: Two sites are connected through a pair of IPsec Secure Gateways. When this option is selected, the fields pertaining to Xauth and ModeCfg are disabled.
- Remote Access: An individual client is connected to a LAN through a secure tunnel. In this scenario, the client is operating as its own Secure Gateway. When this option is selected, the fields pertaining to Emulated Subnet are disabled.

The default is Site to Site.

Default value = "site2site"

Network Configuration Parameters

`emulatedSubnetIpType`

Specifies the IP version to be used for the emulated subnets in the test:

- IPv4
- IPv6

Note that the IPsec plug-in supports mixing IP types within a network stack. For example, you can define IPv6 addresses for the IPsec layer and IPv4 addresses for the IP and Emulated Router layers. The IP and Emulated Router layers must, however, be of the same type. For more information, refer to Support for Mixed IP Types.

The default is IPv4.

Default value = "IPv4"

`numEHCount`

The total number of hosts to be created for each of the emulated subnets.

This parameter is configurable only for site-to-site tests. In a remote-access test, an emulated client is operating as its own Secure Gateway; therefore, the IPsec plug-in sets the count to 1.

The default is 1, the minimum is 1 and the maximum is 65,534.

Default value = "1"

`psk`

Pre-shared key.

Default = "ipsec"

`username`

The User Name field configures EAP in IKEv2, and Xauth in IKEv1. A username may be any unique identifier of the user, such as a login name, an email address, or an X.500 Distinguished Name. These usernames are sent to the DUT for authentication.

During the EAP exchange, the Responder may request the EAP identity of the Initiator; in this case, the configured User Name is sent. If the string is empty, the Initiator ID is sent.

This is a string value, with a maximum of 1024 characters.

Note that user names must be unique. The default value is *ipsec*.

Default = "ipsec"

password

The Password field is used for EAP-MD5 in IKEv2 and Xauth in IKEv1. The password, if specified, is sent to the DUT for authentication. This parameter takes a string value, with a maximum of 1024 characters.

Note that when this field is used for EAP-MD5, a null password is not permitted.

Passwords do not have to be unique; you can use the same password for all user names. The default value is *ipsec*.

Default = "ipsec"

emulatedSubnet

The base address for enumerating all the emulated subnets in the range.

The default IPv4 address is 40.0.0.0, and the default IPv6 address is ::2800:0.

Default value = "40.0.0.0"

protectedSubnet

The base address for enumerating all the protected subnets in the range.

The default IPv4 value is 70.0.0.0, and the default IPv6 value is ::4600:0.

Default value = "70.0.0.0"

emulatedSubnetSuffix

Mask width for `emulatedSubnet`.

Default value = "24"

protectedSubnetSuffix

Mask width for `protectedSubnet`.

Default value = "24"

esnIncrementBy

The increment to be used for enumerating all the emulated subnets in the range.

The default IPv4 value is 0.0.1.0, and the default IPv6 value is ::100.

For each address in the IP range, a subnet will be generated by incrementing the emulated subnet field with the increment value. For example, if you have an IP range with a count of 5, the following subnets will be created on the port:

40.0.0.0/24



40.0.1.0/24

40.0.2.0/24

40.0.3.0/24

40.0.4.0/24

The number of hosts created on each subnet is defined in the *Host Count* field.

Note: The *ESN Increment by* value must be the same on the initiator side and the responder side. If there is a mismatch (0.0.1.1 versus 0.0.1.0, for example), the tunnels will come up but the traffic will fail.

Default value = "0.0.1.0"

`psnIncrementBy`

The increment to be used for enumerating all the protected subnets in the range. The generated subnets will be used as traffic selectors.

The default IPv4 value is 0.0.1.0, and the default IPv6 value is ::100.

Note: The *PSN Increment By* value must be the same on the initiator side and the responder side. If there is a mismatch (0.0.1.1 versus 0.0.1.0, for example), the tunnels will come up but the traffic will fail.

Default value = "0.0.1.0"

`peerPublicIPType`

The IP version to be used for describing the range:

- IPv4
- IPv6

The default is IPv4.

Default value = "IPv4"

`peerPublicIP`

The host name or public IP address of the peer.

The default IPv4 value is 1.1.1.1, and the default IPv6 value is ::3C00:1.

You can specify a host name only for Port-to-DUT tests and only when the *Encapsulation Mode* is set to Tunnel Mode. Host names are resolved at run time.

Default value = "1.1.1.1"

`dpdIdlePeriod`

The interval for sending DPD messages, in seconds. For example, if you set this to 60, the IPsec plug-in sends DPD HELLO messages every 60 seconds to each peer defined for the range. This value must be smaller than the tunnel lifetimes.

Default value = "None"

`dpdTimeout`

Hash key used for ESP-mode traffic originating from the Left Subnet and destined for the Right Subnet. The forward hash key is a variable length value; the key length is determined by the Phase 2 encryption algorithm that you have configured. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

The default value is 10.

`singlePH`

Select if this is a single protected subnet on the responder side. Selecting this field inhibits the generation of PSNs.

Note that this option is valid only for Port-to-DUT tests.

Default value = "False"

`egRange`

The IP range that IPSec will use for emulated gateways.

Default value = "None"

Phase 1 Parameters

`authMethod`

Specifies the authentication method for IKE phase 1. The choices are:

- **Pre-Shared Key:** If you select this method, enter the desired value in the *Pre-Shared Key* column.
- **Certificates:** If you select this method, use the Certificates tab in the Network Plug-in Settings window to configure the certificate parameters. This authentication method requires the CA root certificate, plus a client certificate for each tunnel.
- **EAP:** If you select this method (which is supported only with IKEv2):
 1. Specify the EAP username in the *User Name* column.
 2. Use the EAP-SIM tab or EAP-AKA tab in the Network Plug-in Settings window to configure the EAP parameters.
 3. Make sure that the CA root certificate is available: it is required for EAP authentication.
 4. If the *EAP Method* is TLS, also ensure that you have a client certificate for each tunnel.

The default is Pre-Shared Key.

Default value = "psk"

`psk`

The Pre-Shared Key value. This is a string value, with a maximum of 4096 characters.

The default is *ipsec*.



Note: Make certain that the Pre-Shared Key value does not include a trailing space. IxLoad will treat the trailing space as part of the value. Some DUTs will drop an Authentication Failed notification payload, while others will issue a Payload_Malformed notification payload. In any case, the tunnel will be dropped by the DUT.

Default value = "ipsec"

`enableNatt`

Enable this parameter when running IPsec over NAT devices. When enabled, the IPsec plug-in implements NAT-T for all the traffic in the range.

NAT-T is configurable in IPv4 environments only. The default value is false.

`lifeTimePhase1`

Specifies the Phase 1 Security Association (SA) lifetime, in seconds.

The valid range of values is 0 through 31,557,600. The default value is 3600.

Default value = "3600"

`ikeMode`

Specifies the IKE (Internet Key Exchange) mode of communications for phase 1. The choices are:

- Main Mode - 6 messages exchanged with identity protection.
- Aggressive Mode - 3 messages exchanged without identity protection.

The default is Main Mode.

Default value = "main"

`hashAlgoPhase1`

Specifies the hashing algorithm to use for Phase 1. The choices are:

- HMAC-MD5: Message-Digest Algorithm 5.
- HMAC-SHA1: Secure Hash Algorithm 1.
- AES-XCBC: AUTH_AES_XCBC_96 algorithm, defined in RFC3566. Supported by IKEv2 only.

The default is HMAC-MD5.

API values = "md5", "sha1", "aes-xcbc".

Default value = "md5"

`dhGroup`

Specifies the DH Group. The public-private cryptography used to create the shared secret uses an algorithm called Diffie-Hellman. DH Groups use different bit length selections in this calculation. The choices are:

- DH-1
- DH-2
- DH-5
- DH-14
- DH-15
- DH-16

The default is DH-2.

Default value = "dh2"

`dpdIdlePeriod`

The interval for sending DPD messages, in seconds. For example, if you set this to 60, the IPsec plug-in sends DPD HELLO messages every 60 seconds to each peer defined for the range. This value must be smaller than the tunnel lifetimes. The default value is 1000.

`encAlgoPhase1`

Specifies the encryption algorithm used to protect communications during phase 1 message exchange. The choices are:

- DES
- 3-DES
- AES-128
- AES-192
- AES-256

The default is 3-DES.

Default value = "3des"

`prfAlgo`

Specifies the algorithm used to perform Pseudo-Random Functions (key derivations). The choices are:

- HMAC-MD5: Message-Digest Algorithm 5.
- HMAC-SHA1: Secure Hash Algorithm 1.
- AES-XCBC: AUTH_AES_XCBC_96 algorithm, defined in RFC3566.

This parameter is enabled for IKEv2 only.

The default value is HMAC-MD5.

Default value = "md5"

`xauth`

Enables or disables extended authentication.

The default setting is unchecked.

Default value = "False"



modeCfg

Specifies the Mode Configuration mode. This parameter is valid only when the *Test Scenario* parameter is set to Remote Access.

The choices are:

- Push: The Responder allocates an IP address for the Initiator to use as a traffic endpoint. In this case, the Responder pushes the allocated address to the Initiator. This mode uses the CFG_SET / CFG_ACK transaction sequence.
- Pull: The Responder allocates an IP address for the Initiator to use as a traffic endpoint. In this case, the Initiator requests (pulls) the allocated address from the Responder. This mode uses the CFG_REQUEST / CFG_REPLY transaction sequence.
- None: ModeCfg is not enabled. In this case, the traffic endpoint uses the underlying IP range address; this is the same IP address that is used for IKE control plane negotiations.

The default value is None.

API values = "none", "push", "pull".

Default value = "none"

ipCompression

When enabled, the IPsec plug-in provides support for the IP Payload Compression Protocol (IPComp). IPComp is negotiated during IKE phase 2 negotiations. When enabled, IxLoad compresses the IP packets prior to encryption, using the DEFLATE compression algorithm. The resulting reduction in size of the packets can significantly improve performance on a VPN device.

This setting is disabled by default.

API Default = "false"

modeCfgFirstAddress

Defines the base address to be used for the ModeCfg address pool (the IP addresses that the server port will assign to the clients).

The three ModeCfg "Address" parameters are used only by responder ports. That is, they are used only for a Responder Mode test or for the responder port in a port-to-port test.

The default value is 30.0.0.1.

modeCfgAddressIncrement

Defines the increment value for the ModeCfg address pool.

The default value is 0.0.0.1.

modeCfgAddressSuffix

Defines the IP address suffix for the ModeCfg address pool.

The default value is 24, the minimum value is 1, and the maximum value is 128.

Phase 2 Parameters

`enablePFS`

If checked, enables PFS (perfect forward secrecy).

The default setting is unchecked.

Default value = "False"

`initialContact`

When enabled, the IPsec plug-in will send the INITIAL_CONTACT notification payload as part of IKE SA establishment.

This parameter is disabled by default.

(Note that the IPsec plug-in always ignores the INITIAL_CONTACT notification payload, if it is received.)

API Default = false

`lifeTimePhase2`

Specifies the Security Association (SA) lifetime, in seconds.

The valid range is from 1 to 2,147,483,647. The default value is 28800.

Default value = "28800"

`ahNespMode`

Specifies the AH (Authentication Header) and ESP (Encapsulating Security Payload) options. The choices are:

- AH Only
- ESP Only
- Both AH and ESP

The default is ESP Only.

Default value = "ESPOnly"

`encapMode`

Specifies the IKE phase 2 encapsulation mode. The choices are:

- Tunnel Mode
- Transport Mode

Note that in IxLoad tests using transport mode, the data traffic terminates in the DUT: the data is not forwarded to the protected hosts on the Ixia port.

API values = "tunnel", "transport".



Default value = "tunnel"

hashAlgoPhase2

Specifies the hashing algorithm to use for Phase 2. The choices are:

- HMAC-MD5
- HMAC-SHA1

The default is HMAC-MD5.

Default value = "md5"

pfsGroup

Specifies the PFS Group. The choices are:

- DH-1
- DH-2
- DH-5
- DH-14
- DH-15
- DH-16

The default is DH-2.

Default value = "dh2"

eapMethod

Specifies the EAP method to use, when EAP is selected as the Authentication Method. The choices are:

- MD5: Message-Digest Algorithm 5.
- AKA: EAP for UMTS Authentication and Key Agreement.
- SIM: EAP using Subscriber Identity Module.
- TLS: EAP with Transport Layer Security.

The default is MD5.

API values = "md5", "aka", "sim", "tls".

Default value = "md5"

encAlgoPhase2

Specifies the encryption algorithm used to protect communications during phase 1 and phase 2 message exchange. The choices are:

- Null
- DES
- 3-DES
- AES-128
- AES-192
- AES-256

The default is 3-DES.

Default value = "3des"

`ipsecIDTypeInitiator`

Selects how IxLoad offers the local Emulated Gateway ID type for tunnel negotiations. The setting of this parameter determines the contents of the Identification Type and Identification Data fields in the IPsec packet sent to the DUT. (The Identification Type field describes the type of information contained in the Identification Data field. See RFC 2407 for more information.)

The choices are:

- `ID_IP_ADDR`: IxLoad sets the Identification Type field to 1 and inserts the Emulated Gateway address into the Identification Data field as a single four-octet IPv4 address.
- `ID_IP_ADDR_SUBNET`: IxLoad sets the Identification Type field to 4 and inserts the Emulated Gateway address into the Identification Data field as two four-octet values: an IPv4 address and an IPv4 network mask. (This option is not supported by IKEv2.)
- `ID_FQDN`: IxLoad sets the Identification Type field to 2 and inserts the Emulated Gateway address into the Identification Data field as a fully-qualified domain name string. For example, "foo.bar.com".
- `ID_USER_FQDN`: IxLoad sets the Identification Type field to 3 and inserts the Emulated Gateway address into the Identification Data field as a fully-qualified username string. For example, "piper@foo.bar.com".
- `ID_DER_ASN1_DN`: IxLoad sets the Identification Type field to 9 and inserts the Emulated Gateway address into the Identification Data field as a binary DER encoding of an ASN.1 X.500 Certificate Distinguished Name.
- `ID_KEY_ID`: IxLoad sets the Identification Type field to 11 and inserts the Emulated Gateway address into the Identification Data field as an opaque byte stream that may be used to pass vendor-specific information necessary to identify which pre-shared key should be used to authenticate Aggressive mode negotiations. `ID_KEY_ID` is recommended for Network Access Identifiers (NAIs) that do not include the realm component (reference: draft-eronen-ipsec-ikev2-clarifications). `ID_KEY_ID` is supported by IKEv2 only.

The default is `ID_IP_ADDR`.

API values = "ip-addr-id", "ip-subnet-id", "fqdn-id", "fqdn-user", "der-asn1-dn", "key-id".

Default value = "ip-addr-id"

`fqdnSeedInitiator`

If you set the *Local ID Type* parameter to `ID_FQDN` or `ID_USER_FQDN`, enter the user name that IxLoad inserts into the IPsec packets to identify the emulated gateway.

This is a string value, with a maximum of 1024 characters.



For FQDN_USER, if you enter `user$@foo.bar.com`, IxLoad creates the user names `user1@foo.bar.com`, `user2@foo.bar.com`, `user3@foo.bar.com`, and so on.

Default value = ""

`enableDPD`

When enabled, each IKE peer in the range uses the Dead Peer Detection (DPD) protocol to determine proof of liveness of the other peer. When DPD is enabled, the IPsec plug-in sends DPD HELLO messages according to a configured interval.

Note: If DPD is enabled, IxLoad always sends the DPD messages regardless of the traffic that is being sent over the tunnel.

DPD enables an IPsec endpoint to confirm that its peer is still up. DPD is implemented in IKE through the use of an asynchronous, bidirectional message exchange:

- DPD HELLO
- DPD HELLO ACK

A complete DPD exchange (transmission of DPD HELLO and receipt of the corresponding DPD HELLO ACK) serves as proof of liveness. If a VPN device does not receive a response to a DPD HELLO within a specified time, it assumes that the peer is dead or unreachable, and tears down the tunnel.

The IPsec plug-in implementation of DPD does not use an explicit retry mechanism. For example, if you set the idle period to 5 seconds and the timeout to 14 seconds, the plug-in will send two DPD HELLOs (at 5 and 10 seconds) within the timeout period. If at least one of those hellos receives a DPD HELLO ACK, the timer will be reset and the tunnel will remain up.

Note that regardless of whether DPD is enabled, the IPsec plug-in always responds to DPD messages received from the DUT.

Default value = "false".

`ipsecIDTypeResponder`

Selects how IxLoad offers the Protected Hosts (peer) ID type for tunnel negotiations. The setting of this parameter determines the contents of the Identification Type and Identification Data fields in the IPSec packet sent to the DUT. (The Identification Type field describes the type of information contained in the Identification Data field. See RFC 2407 for more information.)

The choices are:

- **ID_IP_ADDR:** IxLoad sets the Identification Type field to 1 and inserts the Emulated Gateway address into the Identification Data field as a single four-octet IPv4 address.
- **ID_IP_ADDR_SUBNET:** IxLoad sets the Identification Type field to 4 and inserts the Emulated Gateway address into the Identification Data field as two four-octet values: an IPv4 address and an IPv4 network mask. (This option is not supported by IKEv2.)
- **ID_FQDN:** IxLoad sets the Identification Type field to 2 and inserts the Emulated Gateway address into the Identification Data field as a fully-qualified domain name string. For example, "foo.bar.com".
- **ID_USER_FQDN:** IxLoad sets the Identification Type field to 3 and inserts the Emulated Gateway address into the Identification Data field as a fully-qualified username string. For example, "piper@foo.bar.com".
- **ID_DER_ASN1_DN:** IxLoad sets the Identification Type field to 9 and inserts the Emulated Gateway address into the Identification Data field as a binary DER encoding of an ASN.1 X.500 Certificate Distinguished Name.
- **ID_KEY_ID:** IxLoad sets the Identification Type field to 11 and inserts the Emulated Gateway address into the Identification Data field as an opaque byte stream that may be used to pass vendor-specific information necessary to identify which pre-shared key should be used to authenticate Aggressive mode negotiations. **ID_KEY_ID** is recommended for Network Access Identifiers (NAIs) that do not include the realm component (reference: draft-eronen-ipsec-ikev2-clarifications). **ID_KEY_ID** is supported by IKEv2 only.

The default is **ID_IP_ADDR_SUBNET**.

API values = "ip-addr-id", "ip-subnet-id", "fqdn-id", "fqdn-user", "der-asn1-dn", "key-id".

Default value = "ip-addr-id"

`fqdnSeedResponder`

If you set the *Peer ID Type* parameter to **ID_FQDN** or **ID_USER_FQDN**, enter the user name that IxLoad inserts into the IPsec packets to identify the protected hosts.

This is a string value, with a maximum of 1024 characters.

For **FQDN_USER**, if you enter `user$@foo.bar.com`, IxLoad creates the user names `user1@foo.bar.com`, `user2@foo.bar.com`, `user3@foo.bar.com`, and so on.

Default value = ""

`manualKeyingOpts`

If manual keying is enabled, this is the name of the object where the manual keying options are defined.

Default value = ""

`txPreFrag`

When enabled, the IPsec plug-in will—if necessary—pre-fragment IPsec-encapsulated payloads into multiple smaller UDP packets prior to encrypting the payload. This is a transmit-only option; it is not negotiated, and the two ends need not agree on it. The fragment size is determined by the *MTU* setting in the MAC/VLAN network stack element.



Pre-fragmentation is applicable to Tunnel Mode only. In Tunnel Mode there are two IP headers, thus two places where IP-level fragmentation can be done. The default behaviour is to fragment at the outer IP header (post-fragmentation). With pre-fragmentation enabled, fragmentation is performed at the inner IP header.

When disabled, the IPsec plug-in performs post-fragmentation on the IP packets. In this case, the packet is first encapsulated and then fragmented at the outer IP header.

To configure pre-fragmentation:

1. Set the *MTU* value (in the MAC/VLAN stack element) to the desired packet size.
2. Enable the *Pre-fragmentation* parameter.

For example, if you set the *MTU* value to 600, and you have a UDP payload that is 2400 bytes long, the plug-in will fragment it into four IP datagrams prior to encrypting the payload.

API Default value = "false"

XAUTH Parameters

username

The User Name field configures EAP in IKEv2, and Xauth in IKEv1. A username may be any unique identifier of the user, such as a login name, an email address, or an X.500 Distinguished Name. These usernames are sent to the DUT for authentication.

During the EAP exchange, the Responder may request the EAP identity of the Initiator; in this case, the configured User Name is sent. If the string is empty, the Initiator ID is sent.

This is a string value, with a maximum of 1024 characters.

Note that user names must be unique. The default value is *ipsec*.

Default value = "ipsec"

password

The Password field is used for EAP-MD5 in IKEv2 and Xauth in IKEv1. The password, if specified, is sent to the DUT for authentication. This parameter takes a string value, with a maximum of 1024 characters.

Note that when this field is used for EAP-MD5, a null password is not permitted.

Passwords do not have to be unique; you can use the same password for all user names. The default value is *ipsec*.

Default value = "ipsec"

userGroups

A Boolean value that enables or disables User Groups for extended authentication.

The default setting is unchecked.

Default value = "False"

groupName

A comma-separated list of user groups configured on the DUT. To specify more than one user group, separate the group names with commas. For example: *groupA*, *groupB*, *groupC*, and so on.

The default value is `vpngroup`.

Default value = "`vpngroup`"

EXAMPLE

```
$IPsec_R3 config \  
-psnIncrementBy          "0.0.1.0" \  
-singlePH                false \  
-numEHCount             1 \  
-psk                    "ipsec" \  
-enableNatt             false \  
-enabled                true \  
-peerPublicIP           "1.1.1.1" \  
-dpdTimeout             10 \  
-ipsecIDTypeInitiator   "ip-addr-id" \  
-publishStats           false \  
-ikeMode                "main" \  
-encAlgoPhase2          "3des" \  
-encAlgoPhase1          "3des" \  
-userGroups             false \  
-modeCfgAddressIncrement "0.0.0.1" \  
-xauth                  false \  
-modeCfgAddressSuffix   24 \  
-emulatedSubnetIpType   "IPv4" \  
-modeCfgFirstAddress     "30.0.0.1" \  
-ipsecIDTypeResponder   "ip-addr-id" \  
-modeCfg                "none" \  
-ipCompression          false \  
-hashAlgoPhase1         "md5" \  
-protectedSubnet        "70.0.0.0" \  
-peerPublicIPType       "IPv4" \  
-groupName              "vpngroup" \  
-hashAlgoPhase2         "md5" \  
-pfsGroup               "dh2" \  
-eapMethod              "md5" \  
-encapMode              "tunnel" \  
-ahNespMode             "ESPOnly" \  
-username               "ipsec-username" \  
-ikeVersion             "ikev2" \  
-enablePFS              false \  
-initialContact         false \  
-emulatedSubnet         "40.0.0.0" \  
-authMethod             "eap" \  
-testScenario           "site2site" \  
-esnIncrementBy        "0.0.1.0" \  
-lifeTimePhase1        3600 \  
-lifeTimePhase2        28800 \  
-protectedSubnetSuffix  24 \  
-prfAlgo                "md5" \  
-password               "ipsec-pass" \  
-fqdnSeedInitiator     "" \  
-enableDPD             false \  
-emulatedSubnetSuffix  24 \  
-enableMultipleP2perP1 false \  
-dhGroup                "dh2" \  
-dpdIdlePeriod          1000 \  
-fqdnSeedResponder     "" \  
-txPreFrag              false \  
-manualKeyingOpts      $my_ixNetIPSecManualKeyingOpts
```

SEE ALSO



IPSecPortGroupData

SYNOPSIS

DESCRIPTION

Configures the IPSec port parameters that apply to all of the address ranges defined in the port configuration.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`role`

The role that this will play in the test:

- Initiator: The will function as the initiator in the test (will initiate the tunnels).
- Responder: The will function as the responder in the test (will listen for and respond to IPSec connection requests from the peer).

The default value is *Initiator* in a Port-to-DUT test. When configuring a Responder Mode test, set the value to *Responder*.

`teardownRate`

The number of tunnels to tear down per second.

The default is 10, and the minimum is 1. There is no maximum.

Default value = "10"

`overrideGlobalOptions`

If `true`, the values that you specify in this dialog override the corresponding values specified in the Global Settings window.

Default value = "False"

`useMaxInitiationRate`

When checked, the `maxInitiationRate` value that you specify on this tab overrides that specified on the Global Settings window.

Default value = "False"

`maxInitiationRate`

The desired maximum initiation rate (the number of tunnels to initiate, per second).

Default value = "50"

`useMaxPendingTunnels`

If `true`, the `maxPendingTunnels` value that you specify on this tab overrides that specified on the Global Settings window.

Default value = "False"

`maxPendingTunnels`

If `usemaxPendingTunnels` is `true`, this parameter specifies the value that overrides the global `maxPendingTunnels` value.

Default value = "50"

EXAMPLE

```
set my_ixNetIPSecPortGroupData [$Network1 getNetworkSpecificData "IPSecPlugin"]
$my_ixNetIPSecPortGroupData config \
  -useMaxPendingTunnels           false \
  -useMaxInitiationRate          false \
  -maxInitiationRate             5 \
  -maxPendingTunnels            5 \
  -role                          "Initiator" \
  -overrideGlobalOptions         false \
  -teardownRate                  10
```

SEE ALSO

IPSecSessionData

SYNOPSIS

DESCRIPTION

Configures the IPSec global settings and session parameters.

SUBCOMMANDS

OPTIONS

`eapAkaTuples`

Name of the list of EAP AKA tuples.

Default value = "None"

`eapSimTuples`

Name of the list of EAP SIM tuples.

Default value = "None"

`enableExtraPortLogging`

If checked, IxLoad enables protocol-level logging. These logs are created on the Ixia chassis, in following location:

Program Files\Ixia\IxOS\<<IxOS version>\nfs\rw\ <card>\<port>

The log files are collected by the IxDiagnostics application.

Default Value = "False"

`ipsecTunnelSetup`

Name of the `IPSecTunnelSetup` object that defines the tunnel setup parameters.

Default value = "None"

`ipsecCertificates`



Name of the `IPSecCertificates` object that defines the certificate setup parameters.

Default value = "None"

`maxInitiationRate`

If `true`, `IxLoad` attempts to create tunnels at its fastest possible rate.

If `false`, `IxLoad` attempts to create tunnels at the rate that you specify as the *Initiation Rate* parameter.

Default value = "50"

`maxPendingTunnels`

If `true`, `IxLoad` attempts to create the largest possible pool of pending tunnels, and continues to initiate tunnels irrespective of how many tunnels are waiting to be set up.

If `false`, `IxLoad` attempts to create a pool of pending tunnels no larger than the value that you specify as the *Maximum Number of Pending Tunnels* parameter.

Default value = "50"

`negotiationStartDelay`

Additional delay time to wait before starting tunnel negotiation, in seconds.

Default value = "0", Min = 0, Max = 86400.

`teardownRate`

The number of tunnels to tear down per second.

Default value = "10" Min = 1, no maximum.

`enableWildcardTsr`

If checked, proposes a wildcard Responder traffic selector (TSr), regardless of the configured Protected Subnet. (The wildcard values are 0.0.0.0/0 for IPv4 and ::/0 for IPv6). This parameter is disabled by default.

API default = false

`enablePlutoModeCfgWildcardTsr`

If checked, proposes a wildcard Responder traffic selector ("Proxy ID") when ModeCFG is enabled. (The wildcard values are 0.0.0.0/0 for IPv4 and ::/0 for IPv6). This parameter is enabled by default.

API default = true

EXAMPLE

```
$my_ixNetIPSecSessionData config \  
-negotiationStartDelay           0 \  
-enableExtraPortLogging         false \  
-teardownRate                   10 \  
-maxInitiationRate              5 \  
-maxPendingTunnels              5 \  
-ipsecCertificates              $my_ixNetIPSecCertificates \  
-ipsecTunnelSetup               $my_ixNetIPSecTunnelSetup
```

SEE ALSO



IPSecTunnelSetup

SYNOPSIS

DESCRIPTION

Configures the IPSec tunnel setup options. Global settings apply to all network groups and all ranges defined for a test.

SUBCOMMANDS

OPTIONS

`testType`

The type of test that this IPSec encapsulation will support:

- Port to DUT
- Port to Port

If you are setting up a back-to-back test, select the Port to Port option.

Default value = "P2D"

`tunnelSetupTimeout`

The number of seconds to wait for a response from the DUT before declaring that a tunnel setup attempt has failed.

The default is 30, the minimum is 1, the maximum is 600.

Default value = "30"

`numRetries`

The number of attempts that the IPSec plug-in makes to renegotiate the Phase 1 and 2 SAs. If all attempts at renegotiation fail, the plug-in drops the tunnel.

The default is 0, the minimum is 0, the maximum is 100.

Default value = "0"

`retryInterval`

The number of seconds to wait before retrying the tunnel creation.

The default is 10, the minimum is 1, the maximum is 60.

Default value = "10"

`retryDelay`

Specifies the desired delay between subsequent attempts, specified in seconds.

The default is 10, the minimum is 1, the maximum is 60.

Default value = "10"

`sendCiscoVid`

If checked, IxLoad sends the Cisco-Unity Vendor ID payload type. Valid for IKEv1 only.

Default value = "False"

`useMaxInitiationRate`

If `true`, IxLoad attempts to create tunnels at its fastest possible rate.

If `false`, IxLoad attempts to create tunnels at the rate that you specify as the *Initiation Rate* parameter.

Default value = "False"

`useMaxPendingTunnels`

If `true`, IxLoad attempts to create the largest possible pool of pending tunnels, and continues to initiate tunnels irrespective of how many tunnels are waiting to be set up.

If `false`, IxLoad attempts to create a pool of pending tunnels no larger than the value that you specify as the *Maximum Number of Pending Tunnels* parameter.

Default value = "False"

`enableRekey`

Enables or disables renegotiation of Phase 1 and Phase 2 SAs on expiry of tunnel lifetimes:

- When disabled, tunnels are torn down when their lifetimes expire.
- When enabled, the tunnels' Phase 1 and Phase 2 options are renegotiated before their lifetimes expire, and the tunnels stay up.

The rekey parameters control the renegotiation process.

Default value = "False"



rekeyRetries

The total number of rekey retries permitted.

This defines the number of attempts that the IPSec plug-in makes to renegotiate the Phase 1 and 2 SAs. If all attempts at renegotiation fail, the IPSec plug-in drops the tunnel.

The valid range of values is from 0 through 10,000. The default value is 0.

Default value = "0"

rekeyFuzzPercentage

The maximum rekey fuzz percentage.

The fuzz percentage is used to randomize rekeying intervals. It is randomly applied to the Rekey Margin to either shrink (for values under 100) or enlarge (for values over 100) the window of time during which the IPSec plug-in performs rekeying for the tunnels. It prevents all the rekey attempts from occurring at the same time and overloading the DUT.

The valid range of values is from 0 through 100. The default value is 0.

Default value = "0"

rekeyMargin

The rekey margin, in seconds.

This is the number of seconds that are subtracted from the connection expiration time, to ensure that creation of new IPsec SAs begins before the current IPsec SAs expire.

The valid range of values is from 0 through 10,000. The default value is 0.

Default value = "0"

EXAMPLE

```
set my_ixNetIPSecTunnelSetup [::IxLoad new ixNetIPSecTunnelSetup]
$my_ixNetIPSecTunnelSetup config \
  -retryInterval          10 \
  -useMaxPendingTunnels  false \
  -enableRekey           false \
  -useMaxInitiationRate  false \
  -sendCiscoVid         false \
  -testType              "P2D" \
  -rekeyRetries          0 \
  -tunnelSetupTimeout    30 \
  -retryDelay            10 \
  -rekeyMargin           10 \
  -rekeyFuzzPercentage   0 \
  -numRetries            0
```

SEE ALSO

LOD_IPSecEapTuple

SYNOPSIS

DESCRIPTION

Configures the common portion of the EAP SIM and AKA tuple.

SUBCOMMANDS

OPTIONS

`imsi`

A string value that represents the International Mobile Subscriber Identity.

Default value="" (null)

`rand`

A hexadecimal number that represents the 128-bit random challenge generated by the DUT.

Default value="" (null)

EXAMPLE

SEE ALSO

LOD_IPSecEapAkaTupleList

SYNOPSIS

DESCRIPTION

List of `IPSecEapAkaTuple` objects.



SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

LOD_IPSecEapAkaTuple

SYNOPSIS

DESCRIPTION

Configures the AKA portion of an EAP AKA tuple

SUBCOMMANDS

OPTIONS

ck

A 128-bit hexadecimal value representing the Cipher Key. The CK is used for encryption.

(On EAP-AKA full authentication, keying material (a Master Key) is generated from the Integrity Key (IK), the Cipher Key (CK), and the peer identity.)

Default value="" (null)

ik

A 128-bit hexadecimal value representing the Integrity Key. The IK is a session key used for integrity checks.

(On EAP-AKA full authentication, keying material (a Master Key) is generated from the Integrity Key (IK), the Cipher Key (CK), and the peer identity.)

Default value="" (null)

SEE ALSO

LOD_IPSecEapSimTuple

SYNOPSIS

DESCRIPTION

Configures the SIM portion of an EAP AKA tuple

SUBCOMMANDS

OPTIONS

`kc`

A hexadecimal number that represents the 64-bit ciphering key used as a session key for encryption of the over-the-air channel.

The Kc key was originally intended to be used as an encryption key over the air interface, but in the EAP-SIM protocol, it is used for deriving keying material and is not directly used. (Note that the secrecy of Kc is critical to the security of this protocol.)

Default value = "" (null)

`sres`

A hexadecimal number that represents the 32-bit signed response generated by the SIM.

Default value = "" (null)

EXAMPLE

```
$my_ixNetIPSecSessionData eapSimTuples.clear

set my_ixNetIPSecEapSimTuple [::IxLoad new ixNetIPSecEapSimTuple]
# ixNet objects needs to be added in the list before they are configured!
$my_ixNetIPSecSessionData eapSimTuples.appendItem -object $my_ixNetIPSecEapSimTuple

$my_ixNetIPSecEapSimTuple config \
  -kc          "0xa0a1a2a3a4a5a6a7" \
  -rand        "0x101112131415161718191a1b1c1d1e1f" \
  -sres        "0xd1d2d3d4" \
  -imsi        ""
```

SEE ALSO

LOD_IPSecManualKeyingOpts

SYNOPSIS

DESCRIPTION

If manual keying is enabled, this object defines the keying options.

SUBCOMMANDS

OPTIONS

`forwardEncryptKey`

Encryption key used for traffic originating from the Left Subnet and destined for the Right Subnet. The forward encryption key is a variable length value; the key length is determined by the Phase 2 encryption algorithm that you have configured. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

API default value="" (null)

`forwardEncryptKeyIncrement`

The increment value for the Forward Encryption Key. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.



The default value is 0x00.

API default value="0x00"

`forwardHashKeyAH`

Hash key used for AH-mode traffic originating from the Left Subnet and destined for the Right Subnet. The forward hash key is a variable length value; the key length is determined by the Phase 2 encryption algorithm that you have configured. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

API default value="" (null)

`forwardHashKeyAHincrement`

The increment value for the Forward Hash Key/AH, for each tunnel. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

The default value is 0x00.

API default value="0x00" (null)

`forwardHashKeyESP`

Hash key used for ESP-mode traffic originating from the Left Subnet and destined for the Right Subnet. The forward hash key is a variable length value; the key length is determined by the Phase 2 encryption algorithm that you have configured. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

API default value="" (null)

`forwardHashKeyESPincrement`

The increment value for the Forward Hash Key/ESP, for each tunnel. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

The default value is 0x00.

API default value="0x00"

`forwardSPI`

The Security Parameter Index for IPsec traffic originating from the Left Subnet and destined for the Right Subnet. The SPI is a 32-bit value.

You can enter the Forward SPI using either decimal or hexadecimal notation (enter hexadecimal values with a leading 0x). If you enter the value in decimal, IxLoad automatically converts your entry to a hexadecimal number.

API default value="0" (null)

```
forwardSPIincrement
```

The incrementor for the Forward SPI.

You can enter the increment value in either decimal or hexadecimal notation (enter hexadecimal values with a leading 0x). If you enter it in decimal, IxLoad automatically converts your entry to a hexadecimal number.

API default value="0" (null)

```
reverseEncryptKey
```

Encryption key used for traffic originating from the Right Subnet and destined for the Left Subnet. The reverse encryption key is a variable length value; the key length is determined by the Phase 2 encryption algorithm that you have configured. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

API default value="" (null)

```
reverseEncryptKeyIncrement
```

Value for incrementing the Reverse Encryption Key, for each tunnel. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

The default value is 0x00.

API default value="0x00" (null)

```
reverseHashKeyAH
```

Hash key used for AH-mode traffic originating from the Right Subnet and destined for the Left Subnet. The reverse hash key is a variable length value; the key length is determined by the Phase 2 encryption algorithm that you have configured. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

API default value="" (null)

```
reverseHashKeyAHincrement
```

Value for incrementing the Reverse Hash Key/AH, for each tunnel. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

The default value is 0x00.

```
defaultValue="0x00" />
```



`reverseHashKeyESP`

Hash key used for ESP-mode traffic originating from the Right Subnet and destined for the Left Subnet. The reverse hash key is a variable length value; the key length is determined by the Phase 2 encryption algorithm that you have configured. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

API default value="" (null)

`reverseHashKeyESPincrement`

Value for incrementing the Reverse Hash Key/ESP, for each tunnel. You can enter the value as a string or as a hexadecimal number (use a leading 0x for hexadecimal). If you enter the value as a string, IxLoad automatically converts it to an ASCII value.

The default value is 0x00.

API default value="0x00"

`reverseSPI`

The Security Parameter Index for IPsec traffic originating from the Right Subnet and destined for the Left Subnet. The SPI is a 32-bit value.

You can enter the Reverse SPI using either decimal or hexadecimal notation (enter hexadecimal values with a leading 0x). If you enter the value in decimal, IxLoad automatically converts your entry to a hexadecimal number.

API default value="0"

`reverseSPIincrement`

The incrementor for the the Reverse SPI.

You can enter the increment value in either decimal or hexadecimal notation (enter hexadecimal values with a leading 0x). If you enter it in decimal, IxLoad automatically converts your entry to a hexadecimal number.

API default value="0"

EXAMPLE

```
set my_ixNetIPSecManualKeyingOpts [::IxLoad new ixNetIPSecManualKeyingOpts]
$my_ixNetIPSecManualKeyingOpts config \
  -forwardHashKeyESPincrement      "0x00" \
  -reverseEncryptKey               "abcdefabcdefabcdefabcdef" \
  -reverseSPI                      0 \
  -reverseHashKeyAH                "" \
  -reverseHashKeyESP              "abcdefabcdef9876" \
  -forwardEncryptKeyIncrement      "0x00" \
  -forwardSPI                      0 \
  -reverseHashKeyESPincrement      "0x00" \
  -forwardHashKeyAHincrement       "0x00" \
  -reverseSPIincrement            0 \
  -forwardHashKeyESP              "abcdefabcdef1223" \
  -reverseEncryptKeyIncrement      "0x00" \
  -forwardSPIincrement            0 \
  -forwardEncryptKey              "abcdefabcdefabcdefabcdef" \
  -forwardHashKeyAH               "" \
  -reverseHashKeyAHincrement       "0x00"
```

SEE ALSO

LOD_IPSecAssociate

SYNOPSIS

DESCRIPTION

Associates an IPSec plugin with a port group.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

IPSecCertificates

SYNOPSIS

DESCRIPTION

Configures the certificate parameters when the chosen authentication method is *Certificates*.

SUBCOMMANDS

OPTIONS

`uniqueCert`

If true, IxLoad uses the same certificate to negotiate every tunnel. This can significantly speed up the negotiation process, but it does not stress the DUT's ability to cache certificates or to negotiate tunnels using multiple certificates, as would happen in an actual VPN.

If you select a cache as the Certificate source and the cache contains more than one certificate, IxLoad selects the certificate file with the oldest timestamp.

Default value = "False"

`certSource`

If enabled, IxLoad gets the certificates from the Certificate Authority (CA). If you select this option, IxLoad deletes any cached certificates from the chassis (from the folder specified for 'Certificates Folder').

Default value = "kNewCert"

`certSubjectAltDN`

A comma-separated list of subject alternative names. The subject alternative name is an X.509 v3 extension that permits various literal values to be included in the configuration file.

defaultValue="" (null)

`caURL`

Certificate Authority URL. Check this option to use a certificate authority (CA) server for authentication. Enter the CA server's URL in the field. IxLoad uses Simple Certificate Enrollment Protocol (SCEP) to obtain signed certificates from the CA.



This option is not supported in a port-to-port test.

Default value = ""

caDN

Issuing CA Distinguished Name. Name of the Certificate Authority (CA) that issued the DUT's certificate.

This field includes the Distinguished Name fields and values that IxLoad sends to the DUT's CA. These can include the following fields:

- CN: Common name
- E: Email address
- OU: Organizational unit
- O: Organization name
- L: Locality
- S: State or province
- C: 2-letter country or region name

For example,

CN=Liesl Benjamin, E=liesl@ixia.com, OU=Security, O=Ixia, L=Los Angeles S=California, C=US

Default value = ""

certSubjectDN

Subject Distinguished Name. A name designating the owner of the certificate.

This field includes the Distinguished Name fields and values that IxLoad sends to the DUT's CA. These can include the same fields as described for the *Issuing CA Distinguished Name* parameter.

Default value = ""

remoteIkeId

Attribute that identifies the DUT in its certificate. You can enter the following in this field:

- A fully-qualified domain name (FQDN).
Syntax: @<domain>
For example, @ixiacom.com
- An email address.
Syntax: user@domain
For example, liesl@ixiacom.com
- An IP address.
Syntax: IP=<address>
For example, IP=192.168.0.1

- A context string.

Syntax: attribute=value

For example, CN=liesl benjamin, O=ixia, C=us

This is a string value, with a maximum length of 2048 characters.

Default value = ""

bitSize

Bit Size for the Keys. The choices are: 512, 1024, 2048.

Default value = "k512"

saveCert

If `true`, IxLoad stores certificates in the specified folder on the chassis.

The default folder is specified in `cacheCertFolder`.

Default value = "False"

cacheCertFolder

Folder where certificates are stored.

Default value = "C:\Program Files\Ixia\CachedCerts"

certParentFolder

Root path of certificate folder.

Default value = "C:\Program Files\Ixia\CachedCerts"

certNumber

Number of certificates cached.

If the number of tunnels exceeds the number of certificates, IxLoad reuses certificates as necessary.

Default value = ""

earlyExpDate

Earliest expiry date and time of cached certificates.

Default value = ""



lateExpDate

Latest expiry date and time of cached certificates.

Default value = ""

usePerRangeCertNameExp

If enabled, IxLoad expands the \$ (if present) in the *Subject Distinguished Name* field on a per-range basis.

If disabled (the default), IxLoad expands the \$ globally.

Default value = "False"

EXAMPLE

```
set my_ixNetIPSecSessionData [$Test1 getSessionSpecificData "IPSecPlugin"]
set my_ixNetIPSecCertificates [::IxLoad new ixNetIPSecCertificates]
$my_ixNetIPSecCertificates config \
    -uniqueCert                false \
    -usePerRangeCertNameExp    false \
    -caURL                     "" \
    -bitSize                   "k512" \
    -remoteIkeId               "" \
    -lateExpDate               "" \
    -cacheCertFolder           "C:\\Program Files\\Ixia\\CachedCerts" \
    -saveCert                   false \
    -certSubjectAltDN          "" \
    -certSubjectDN             "" \
    -certParentFolder          "C:\\Program Files\\Ixia\\CachedCerts" \
    -earlyExpDate              "" \
    -certSource                 "kNewCert" \
    -caDN                       "" \
    -certNumber                 ""
```

SEE ALSO

IPSec Example

This section shows an example of how to create an IPSec plugin in the Tcl API.

IPSec Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.10.0.79
# IPsec.tcl made on Aug 14 2008 15:25
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              "Network1" \
    -macMappingMode   0 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear

Begin appending items to global plugin list.

set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
    -enabled          true \
    -name              "GratARP"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.



IPSec Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

Configure the TCP
portion of the stack.

```
$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
    -name "TCP" \
```

IPSec Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior              0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                 "master" \
    -negotiateMasterSlave            true

$Ethernet_1 config \
    -advertise10Full                 true \
    -name                             "Ethernet-1" \
    -autoNegotiate                    true \
    -advertise100Half                 true \
    -advertise10Half                  true \
    -speed                             "k100FD" \
    -advertise1000Full                true \
    -advertise100Full                 true \
    -cardElm

$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.



IPSec Example

```
set MAC_VLAN_8 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_8

$MAC_VLAN_8 config \
    -name "MAC/VLAN-8"

$MAC_VLAN_8 childrenList.clear

set Emulated_Router_1 [::IxLoad new ixNetEmulatedRouterPlugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_8 childrenList.appendItem -object $Emulated_Router_1

$Emulated_Router_1 config \
    -name "Emulated Router-1"

$Emulated_Router_1 childrenList.clear

set IP_3 [::IxLoad new ixNetIPv4V6Plugin]
# ixNet objects needs to be added in the list before they are configured!
$Emulated_Router_1 childrenList.appendItem -object $IP_3

$IP_3 config \
    -name "IP-3"

$IP_3 childrenList.clear

set IPSec_1 [::IxLoad new ixNetIPSecPlugin]
# ixNet objects needs to be added in the list before they are configured!
$IP_3 childrenList.appendItem -object $IPSec_1

$IPSec_1 config \
    -name "IPSec-1"

$IPSec_1 childrenList.clear
$IPSec_1 extensionList.clear
$IP_3 extensionList.clear
$Emulated_Router_1 extensionList.clear
$MAC_VLAN_8 extensionList.clear
$Ethernet_1 extensionList.clear
```

Configure the MAC addresses and VLAN tags.

Configure an Emulated Router.

Configure an address range for the Emulated Router.

Add an IPSec plugin.

Clear the lists of extension protocols.

IPSec Example

```
#####
# Setting the ranges starting with the plugin on top of the stack
#####
$IPSec_1 rangeList.clear

set IPSec_R1 [::IxLoad new ixNetIPSecRange]
# ixNet objects needs to be added in the list
# before they are configured!
$IPSec_1 rangeList.appendItem -object $IPSec_R1

$IPSec_R1 config \
  -psnIncrementBy "0.0.1.0" \
  -singlePH false \
  -numEHCount 1 \
  -psk "ipsec" \
  -ipsecIDTypeInitiator "ip-subnet-id" \
  -ikeMode "main" \
  -encAlgoPhase2 "3des" \
  -dhGroup "dh2" \
  -userGroups false \
  -groupName "vpngroup" \
  -xauth false \
  -emulatedSubnet IpType "IPv4" \
  -ipsecIDTypeResponder "ip-subnet-id" \
  -modeCfg "none" \
  -hashAlgoPhase1 "md5" \
  -fqdnSeedInitiator "" \
  -peerPublicIP "1.1.1.1" \
  -hashAlgoPhase2 "md5" \
  -pfsGroup "dh2" \
  -encapMode "tunnel" \
  -ahNespMode "ESPOnly" \
  -username "ipsec" \
  -ikeVersion "ikev1" \
  -enablePFS false \
  -emulatedSubnet "40.0.0.0" \
  -authMethod "psk" \
  -testScenario "site2site" \
  -esnIncrementBy "0.0.1.0" \
  -lifeTimePhase1 3600 \
```

Append the IPSec extension to the IP address range.

Configure the IPSec settings.

IPSec Example

```
-protectedSubnet          "70.0.0.0" \  
-protectedSubnetSuffix   24 \  
-prfAlgo                  "md5" \  
-password                 "ipsec" \  
-peerPublicIPType        "IPv4" \  
-name                     "IPSec-R1" \  
-emulatedSubnetSuffix    24 \  
-enabled                  true \  
-encAlgoPhase1            "3des" \  
-lifeTimePhase2          28800 \  
-fqdnSeedResponder       ""  
  
set IP_R3 [$IPSec_R1 getLowerRelatedRange "IPv4V6Range"]  
  
$IP_R3 config \  
-count                    1 \  
-name                     "IP-R3" \  
-gatewayAddress           "0.0.0.0" \  
-enabled                  true \  
-autoMacGeneration        true \  
-mss                      1460 \  
-incrementBy              "0.0.0.1" \  
-prefix                   16 \  
-gatewayIncrement         "0.0.0.0" \  
-gatewayIncrementMode     "perSubnet" \  
-generateStatistics        false \  
-ipAddress                 "10.10.0.4" \  
-ipType                   "IPv4"  
  
$Emulated_Router_1 rangeList.clear
```

Configure an IP range for the IPSec tunnels.

IPSec Example

```
set ER_R1 [::IxLoad new ixNetEmulatedRouterRange]
# ixNet objects needs to be added in the list
# before they are configured!
$Emulated_Router_1 rangeList.appendItem -object $ER_R1

$ER_R1 config \
    -count 1 \
    -name "ER-R1" \
    -gatewayAddress "0.0.0.0" \
    -enabled true \
    -autoMacGeneration true \
    -mss 1460 \
    -incrementBy "0.0.0.1" \
    -prefix 16 \
    -gatewayIncrement "0.0.0.0" \
    -gatewayIncrementMode "perSubnet" \
    -generateStatistics false \
    -ipAddress "10.10.0.3" \
    -ipType "IPv4"

set MAC_R7 [$ER_R1 getLowerRelatedRange "MacRange"]

$MAC_R7 config \
    -count 1 \
    -name "MAC-R7" \
    -enabled true \
    -mtu 1500 \
    -mac "10:EF:3C:1E:00:00" \
    -incrementBy "00:00:00:00:00:01"
```

Configure an address range for the Emulated Router.

Configure the MAC addresses for the IP range.

PPPoX Plugin

SYNOPSIS

DESCRIPTION

Configures a PPPoX plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`rangeList`

Name of the list of IP address ranges used by this plugin. The list must be a `PppoxRangeList` object.

This option is read-only.

Default value = "None"

EXAMPLE

```
set PPPoX_1 [::IxLoad new ixNetPppoxPlugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_10 childrenList.appendItem -object $PPPoX_1

$PPPoX_1 config \
    -name "PPPoX-1"

$PPPoX_1 childrenList.clear

$PPPoX_1 extensionList.clear

$MAC_VLAN_10 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO



PppoxPortGroupData

SYNOPSIS

DESCRIPTION

Configures the PPPoX network group settings.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`associates`

Name of the list of associates. This list must an `AssociateList` object.

This option is read only.

Default value = "None"

`overrideGlobalRateControls`

If `false`, the global setup and teardown rate values will be equally divided among the ports.

If `true`, The setup and teardown parameters defined at the port level will override those defined at the global level.

For example, if you have set the initial setup rate to 150 on the global level, and you have defined two ports, these 150 session setups will be evenly distributed across the ports (75 for each). If you then enable Override Global Rate Controls, you can modify the number of session setups for each of the ports (such as changing the distribution from 75-75 to 120-30).

Default value = "False"

`setupRateInitial`

The number of PPP sessions to set up, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingRequests`

The maximum number of PPP sessions that can be outstanding at any given time. The minimum is 1, the maximum is 1000.

Default value="300"

`teardownRateInitial`

The number of PPP sessions to tear down, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingReleases`

The maximum number of PPP sessions that can be released at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

`useWaitForCompletionTimeout`

Enables the application to wait for a specified amount of time for the sessions to negotiate before declaring a negotiation timeout.

Default value = "False"

`waitForCompletionTimeout`

If `useWaitForCompletionTimeout` is true, specify the number of seconds that the application will wait for the sessions to negotiate.

The default is 120, the minimum is 1, and the maximum is 65535.

Default value = "120"

`enablePerSessionStatGeneration`

Enables or disables per-session statistics generation. When enabled, PPP protocol statistics are generated during the session negotiation phase of an L2TP or PPP test and written to a CSV file. The CSV file is generated at the end of the session negotiation phase. The concatenated results for each port are returned as a single file.



Statistics are generated only for client ports because server ports do not establish any sessions during the negotiation phase.

Default value = "False"

`perSessionStatFilePrefix`

If `enablePerSessionStatGeneration` is true, specify the prefix to use for the name of the per-session PPP protocol statistics file.

The per-session PPP protocol statistics file names are of the form:

```
StatsFilePrefix_chassis_card_port_TimeStamp_.csv
```

The CSV files are saved in this folder:

```
C:\Program Files\Ixia\IxNetwork\data\result\[UserName]\Per Session Stats  
PPP_L2TP\
```

Default value = "MY_PREFIX"

`role`

The role that the PPPoX network group plays in the test configuration. Must be one of the choices in the `RoleChoices` object.

Note: A client and a server cannot both be set at the same time on the same network group.

Default value = "client"

`filterDataPlaneBeforeL7`

Default value = "False"

EXAMPLE

SEE ALSO

RoleChoices

SYNOPSIS

DESCRIPTION

Role that the PPPoX network group plays in the test configuration.

SUBCOMMANDS

OPTIONS

Value	Description
client	The network group functions as a PPP client.
server	The network group functions as a PPP server.

EXAMPLE

SEE ALSO



PLSessionDataBase

SYNOPSIS

DESCRIPTION

Configures the PPPoX and L2TP network group settings.

SUBCOMMANDS

OPTIONS

`setupRateInitial`

The number of PPP sessions to set up, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingRequests`

The maximum number of PPP sessions that can be outstanding at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

`teardownRateInitial`

The number of PPP sessions to tear down, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingReleases`

The maximum number of PPP sessions that can be released at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

EXAMPLE

SEE ALSO



PppoxSessionData

SYNOPSIS

DESCRIPTION

Global settings for PPPoX.

SUBCOMMANDS

OPTIONS

`setupRateInitial`

The number of PPP sessions to set up, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingRequests`

The maximum number of PPP sessions that can be outstanding at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

`teardownRateInitial`

The number of PPP sessions to tear down, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingReleases`

The maximum number of PPP sessions that can be released at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

EXAMPLE

SEE ALSO



PppoxRangeList

SYNOPSIS

DESCRIPTION

List of PPPoX ranges. This list must be a list of `PppoxRange` objects.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

PppoxRange

SYNOPSIS

DESCRIPTION

Properties of a range of IP addresses within PPPoX plugin.

SUBCOMMANDS

OPTIONS

Same as PLRangeBase plus the following:

`pppoeOptions`

Name of the list of PPPoE options.

Default value = "PPPoE Options"

`padiTimeout`

Timeout value for a PADI (PPPoE Active Discovery Initiation) no-response, in seconds. The minimum value is 1, the default value is 10, and the maximum is 65535.

Default value = "10"

`padiRetries`

Number of PADI retries. The minimum value is 1, the default value is 5, and the maximum is 65535.

Default value = "5"

`padrTimeout`

Timeout for a PADR (PPPoE Active Discovery Request) no-response, in seconds. The minimum value is 1, the default value is 10, and the maximum is 65535.

Default value = "10"

`padrRetries`

Number of PADR retries. The minimum value is 1, the default value is 5, and the maximum is 65535.



Default value = "5"

`enableMruNegotiation`

The maximum-receive-unit (MRU) checks the size of the received frames on the PPP line. When MRU Negotiation is disabled, it must not be negotiated to a size larger than 1,492 to guarantee compatibility with Ethernet segments limited to 1,500 octets. By default, this feature is disabled.

If this feature is enabled, MRU Negotiation is available for the client and server mode, and can negotiate a size larger than 1,492. The upper limit can be up to 13,312 octets.

If this feature is disabled, and the desired MRUs for the client and server are not equal, the session will use the default value of 1,500.

Default value = "False"

`serviceOptions`

Provides access to the Service Options dialog from which you select the desired PPPoE service retrieval mode. See `ServiceOptionsChoices` for the list of valid values for this option.

This parameter is used only for PPP clients.

Default value = "anyService"

`serviceName`

The Access Concentrator Service Name. This parameter is used only for PPP servers.

Default value = ""

`acOptions`

Provides access to the AC Options dialog from which you select the desired PPPoE AC retrieval mode. See `AcOptionsChoices` for the list of valid values for this option.

This parameter is used only for PPP clients.

Default value = "useFirstResponder"

`acNameTable`

List of AC names.

Default value = "None"

`acName`

The Access Concentrator Name. This parameter is used only for PPP servers.

Default value = ""

`acMacTable`

List of MAC addresses.

Default value = "None"

`enableRedial`

If `true`, PPPoE redial is enabled.

Default value = "True"

`redialTimeout`

PPPoE redial timeout, in seconds. The minimum value is 1, the default value is 10, and the maximum is 65535.

Default value = "10"

`redialMax`

Maximum number of PPPoE redials. The minimum value is 1, the default value is 20, and the maximum is 65535.

Default value = "20"

Intermediate Agent (IA) Parameters

`serverSignalLoopId`

This parameter enables or disables the insertion of sub-options 0x01 and 0x02 (Remote ID and Circuit ID) into the DSLF tag in PADO and PADS packets. It inserts the exact values received from the client (in the PADI / PADR packets).

This parameter is applicable only when the *Role* parameter (in Port Settings) is set to *server*.

Default value = "False"

`clientSignalLoopId`

This parameter enables or disables the insertion of sub-options 0x01 and 0x02 (Remote ID and Circuit ID) into the DSLF tag in PADI and PADR packets.

This parameter is applicable only when the *Role* parameter (in Port Settings) is set to *client*.

Default value = "False"



`agentCircuitId`

The value to be inserted into the Agent Circuit ID field of the PPPoE tag.

The Agent Circuit ID uniquely identifies the Access Node and the access loop logical port on the Access Node on which the discovery stage PPPoE packet was received.

The string is from 0 to 32 characters in length.

Default value = ""

`agentRemoteId`

The value to be inserted into the Agent Remote ID field of the PPPoE tag.

The Agent Remote ID uniquely identifies the user on the associated access loop logical port on the Access Node on which the PPPoE discovery packet was received.

The string is from 0 to 32 characters in length.

Default value = ""

`serverSignalLoopChar`

This parameter enables or disables the insertion of sub-options 0x81 and 0x82 into the DSLF tag in PADO and PADS packets. It inserts the exact values received from the client (in the PADI / PADR packets).

This parameter is applicable only when the *Role* parameter (in Port Settings) is set to *server*.

Default value = "False"

`clientSignalLoopChar`

This parameter enables or disables the insertion of sub-options 0x81 and 0x82 into the DSLF tag in PADI and PADR packets.

This parameter is applicable only when the *Role* parameter (in Port Settings) is set to *client*.

Default value = "False"

`actualRateUpstream`

This parameter specifies the value to be included in the vendor specific PPPoE tag. It is the actual upstream data rate (sub-option 0x82), in kbps. This value can be edited only if Client Signal Loop Characteristics is `true`.

The default is 10, the minimum is 1, and the maximum is 65535.

Default value = "10"

`actualRateDownstream`

This parameter specifies the value to be included in the vendor specific PPPoE tag. It is the actual downstream data rate (sub-option 0x81), in kbps. This value can be edited only if *Client Signal Loop Characteristics* is `true`.

The default is 10, the minimum is 1, and the maximum is 65535.

Default value = "10"

`serverSignalIWF`

This parameter enables or disables the insertion of sub-option 0xFE (signaling of interworked sessions) into the DSLF tag in PADO and PADS packets. The server inserts the option in replies only if the client also sent it.

Note that the length field (in the PPPoE tag) is set to 0x00.

This parameter is applicable only when the *Role* parameter (in Port Settings) is set to *server*.

Default value = "False"

`clientSignalIWF`

This parameter enables or disables the insertion of sub-option 0xFE (signaling of interworked sessions) into the DSLF tag in PADI and PADR packets.

Note that the length field (in the PPPoE tag) is set to 0x00.

This parameter is applicable only when the *Role* parameter (in Port Settings) is set to *client*.

Default value = "False"

`serverSignalLoopEncapsulation`

This parameter enables or disables the insertion of sub-option 0x90 into the DSLF tag in PADO and PADS packets. It inserts the exact values received from the client (in the PADI / PADR packets).

This parameter is applicable only when the *Role* parameter (in Port Settings) is set to *server*.

Default value = "False"

`clientSignalLoopEncapsulation`

This parameter enables or disables the insertion of sub-option 0x90 into the DSLF tag in PADI and PADR packets.

This parameter is applicable only when the *Role* parameter (in Port Settings) is set to *client*.

Default value = "False"



dataLink

Data Link. A one-byte field included with sub-option 0x90. See `DataLinkOptions` for the list of valid choices.

Default value = "ethernet"

encaps1

Encapsulation 1. A one-byte field included with sub-option 0x90. See `Encaps1Options` for the list of valid choices.

Default value = "untaggedEthernet"

encaps2

Encapsulation 2. A one-byte field included with sub-option 0x90. See `Encaps1Options` for the list of valid choices.

Default value = "na"

EXAMPLE

```

$PPPoX_1 rangeList.clear

set PPPoX_R1 [::IxLoad new ixNetPppoxRange]
# ixNet objects needs to be added in the list before they are configured!
$PPPoX_1 rangeList.appendItem -object $PPPoX_R1

$PPPoX_R1 config \
  -serverSignalLoopEncapsulation      false \
  -padrTimeout                        10 \
  -serverBaseIID                      "00:11:22:11:00:00:00:01" \
  -enableEchoReq                      false \
  -authTimeout                        10 \
  -actualRateUpstream                10 \
  -ncpTimeout                          10 \
  -clientIpIncr                       "0.0.0.1" \
  -lcpOptions                          "LCP Options" \
  -serverNetmaskOptions               "disableExtension" \
  -serverSignalLoopId                 false \
  -ipv6PoolPrefixLen                  48 \
  -clientSignalLoopChar                false \
  -clientIIDIncr                      1 \
  -authRetries                        20 \
  -enableMrNegotiation                false \
  -authType                            "none" \
  -clientNetmask                       "255.0.0.0" \
  -ncpType                             "IPv4" \
  -echoReqInterval                    10 \
  -domainList                          "Domain Groups" \
  -clientBaseIID                      "00:11:11:11:00:00:00:01" \
  -clientNetmaskOptions               "disableExtension" \
  -clientSignalLoopId                 false \
  -authOptions                         "Authentication Options" \
  -lcpTimeout                          10 \
  -pppoeOptions                       "PPPoE Options" \
  -ncpRetries                          3 \
  -serviceName                         "" \
  -enableRedial                        true \
  -serverPrimaryDnsAddress             "10.10.10.10" \
  -lcpRetries                          3 \
  -redialTimeout                       10 \
  -clientSignalLoopEncapsulation      false \
  -acName                              "" \
  -enableDomainGroups                 false \
  -padiTimeout                         10 \
  -ipv6PoolPrefix                      "1:1:1::" \
  -enableEchoRsp                       true \
  -agentRemoteId                       "" \
  -serverIpIncr                       "0.0.0.0" \
  -clientPrimaryDnsAddress             "8.8.8.8" \
  -papPassword                         "password" \
  -encaps1                             "untaggedEthernet" \
  -agentCircuitId                      "" \
  -serverSignalIWF                     false \
  -chapName                            "user" \
  -clientDnsOptions                    "disableExtension" \
  -serverSecondaryDnsAddress           "11.11.11.11" \
  -encaps2                             "na" \
  -acOptions                           "useFirstResponder" \
  -actualRateDownstream                10 \
  -redialMax                           20 \
  -lcpTermTimeout                      15 \
  -padrRetries                          5 \
  -ipv6AddrPrefixLen                   64 \

```



```
-serverSignalLoopChar      false \  
-clientBaseIp              "1.1.1.1" \  
-papUser                   "user" \  
-dataLink                  "ethernet" \  
-name                      "PPPoX-R1" \  
-serverNetmask             "255.255.255.0" \  
-lcpTermRetries           3 \  
-clientSecondaryDnsAddress "9.9.9.9" \  
-enabled                   true \  
-serverIIDIncr            1 \  
-mtu                       1492 \  
-serverDnsOptions         "disableExtension" \  
-serverBaseIp             "2.2.2.2" \  
-serviceOptions           "anyService" \  
-numSessions              1 \  
-padiRetries              5 \  
-useMagic                 true \  
-chapSecret               "secret" \  
-clientSignalIWF          false  
  
$PPPoX_R1 domainGroupList.clear  
  
$PPPoX_R1 acNameTable.clear  
  
$PPPoX_R1 acMacTable.clear  
  
set MAC_R9 [$PPPoX_R1 getLowerRelatedRange "MacRange"]  
  
$MAC_R9 config \  
-count                    1 \  
-name                    "MAC-R9" \  
-enabled                 true \  
-mtu                     1500 \  
-mac                    "6A:40:7E:B6:00:00" \  
-incrementBy             "00:00:00:00:00:01"  
  
set VLAN_R1 [$PPPoX_R1 getLowerRelatedRange "VlanIdRange"]  
  
$VLAN_R1 config \  
-incrementStep           1 \  
-uniqueCount            4094 \  
-name                   "VLAN-R1" \  
-innerIncrement         1 \  
-innerUniqueCount       4094 \  
-enabled                true \  
-innerFirstId           1 \  
-increment              1 \  
-priority               1 \  
-firstId               1 \  
-innerIncrementStep     1 \  
-idIncrMode             2 \  
-innerEnable            false \  
-innerPriority           1
```

SEE ALSO



ServiceOptionsChoices

SYNOPSIS

DESCRIPTION

List of values for the `serviceOptions` option for the `PppoxRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
<code>anyService</code>	A service name is not needed; a null service is sent during Discovery indicating to the access concentrator that the service is not specified.
<code>serviceName</code>	Allows a matching service name to be configured.

EXAMPLE

SEE ALSO

AcOptionsChoices

SYNOPSIS

DESCRIPTION

List of values for the `acOptions` option for the `PppoxRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
<code>useFirstResponder</code>	The first access concentrator that responds to the PADI message will be selected for use by the access client.
<code>matchServiceName</code>	The Access Concentrator will be named the same as the service name used during Discovery.
<code>matchAcName</code>	The Access Concentrator name will match a name that you configure. The <code>PppAcNameList</code> contains the list of AC names.
<code>matchAcMac</code>	The Access Concentrator MAC address will match a MAC address that you configure. The <code>PppAcMacList</code> contains the list of AC names.

EXAMPLE

SEE ALSO



Encaps1Choices

SYNOPSIS

DESCRIPTION

List of valid choices for the `encaps1` option of the `PppoxRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
na	NA
untaggedEthernet	Untagged Ethernet
singleTaggedEthernet	Single-tagged Ethernet

EXAMPLE

SEE ALSO

Encaps2Choices

SYNOPSIS

DESCRIPTION

List of valid choices for the `encaps2` option of the `PppoxRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
na	NA
pppoaLlc	PPPoA LLC
pppoaNull	PPPoA NULL
ipoaLlc	IPoA LLC
ipoaNull	IPoA NULL
ethernetOverAal5LlcwFcs	Ethernet over AAL5 LLC w FCS
ethernetOverAal5LlcwoFcs	Ethernet over AAL5 LLC w/o FCS
ethernetOverAal5NullwFcs	Ethernet over AAL5 Null w FCS
ethernetOverAal5NullwoFcs	Ethernet over AAL5 Null w/o FCS

EXAMPLE

SEE ALSO



Encaps2Choices

SYNOPSIS

DESCRIPTION

List of valid choices for the `encaps2` option of the `pppoxRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
na	NA
pppoaLlc	PPPoA LLC
pppoaNull	PPPoA NULL
ipoaLlc	IPoA LLC
ipoaNull	IPoA NULL
ethernetOverAal5LlcwFcs	Ethernet over AAL5 LLC w FCS
ethernetOverAal5LlcwoFcs	Ethernet over AAL5 LLC w/o FCS
ethernetOverAal5NullwFcs	Ethernet over AAL5 Null w FCS
ethernetOverAal5NullwoFcs	Ethernet over AAL5 Null w/o FCS

EXAMPLE

SEE ALSO

DataLinkChoices

SYNOPSIS

DESCRIPTION

List of valid choices for the `dataLink` option of the `PppoxRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
<code>atmAal5</code>	ATM AAL5
<code>ethernet</code>	Ethernet

EXAMPLE

SEE ALSO



PppoxAcNameList

SYNOPSIS

DESCRIPTION

List of access concentrator names. This list must be a list of `PppoxAcName` objects.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

PppoxAcName

SYNOPSIS

DESCRIPTION

Configures an access concentrator name for use in a `PppoxAcNameList`.

SUBCOMMANDS

OPTIONS

`select`

If `true`, this AC name will be used.

Default value = "False"

`percentage`

Percentage of sessions to be directed to this access concentrator. The total of the percentages of all the AC names in a list must add up to 100%.

defaultValue="0"

`acName`

Name of the access concentrator.

defaultValue=""

EXAMPLE

SEE ALSO



PppoxAcMacList

SYNOPSIS

DESCRIPTION

List of access concentrator MAC addresses. This list must be a list of PppoxAcMac objects.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



PppoxAcMac

SYNOPSIS

DESCRIPTION

Configures an access concentrator MAC address for use in a `PppoxMacList`.

SUBCOMMANDS

OPTIONS

`select`

If `true`, this AC MAC address will be used.

Default value = "False"

`percentage`

Percentage of sessions to be directed to this AC MAC address. The total of the percentages of all the AC MACs in a list must add up to 100%.

Default value="0"

`acMac`

MAC address of the matching Access Concentrator name. Enter the MAC address as six octets separated by colons (for example: 00:00:00:11:22:33).

Default value = ""

EXAMPLE

SEE ALSO

PppoX Plugin Example

This section shows an example of how to create a PPPoX plugin in the Tcl API.

PppoX Plugin Example

```
#####  
# IxLoad ScriptGen created TCL script  
# Network1 serialized using version 4.10.0.79  
# PPPoX.tcl made on Aug 14 2008 15:27  
#####  
  
set Network1 [::IxLoad new ixNetworkGroup $chassisChain]  
$Network1 config \  
    -comment          "" \  
    -name              "Network1" \  
    -macMappingMode   0 \  
    -linkLayerOptions 0  
  
$Network1 globalPlugins.clear  
  
Begin appending items to global plugin list.  
  
set GratARP [::IxLoad new ixNetGratArpPlugin]   
# ixNet objects needs to be added in the list   
# before they are configured!  
$Network1 globalPlugins.appendItem -object $GratARP  
  
$GratARP config \  
    -enabled          true \  
    -name              "GratARP"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.

PppoX Plugin Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP

$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
```

Configure the TCP
portion of the stack.



PppoX Plugin Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior              0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                "master" \
    -negotiateMasterSlave           true

$Ethernet_1 config \
    -advertise10Full                true \
    -name                            "Ethernet-1" \
    -autoNegotiate                   true \
    -advertise100Half               true \
    -advertise10Half                true \
    -speed                           "k100FD" \
    -advertise1000Full              true \
    -advertise100Full               true \
    -cardElm                         \
$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.

PppoX Plugin Example

```
set MAC_VLAN_10 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_10

$MAC_VLAN_10 config \
    -name "MAC/VLAN-10"

$MAC_VLAN_10 childrenList.clear

set PPPoX_1 [::IxLoad new ixNetPppoxPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$MAC_VLAN_10 childrenList.appendItem -object $PPPoX_1

$PPPoX_1 config \
    -name "PPPoX-1"

$PPPoX_1 childrenList.clear
$PPPoX_1 extensionList.clear
$MAC_VLAN_10 extensionList.clear
$Ethernet_1 extensionList.clear
```

Configure the MAC addresses and VLAN tags.

Create the PPPoX plugin.

Clear the lists of extension protocols.



PppoX Plugin Example

```
#####  
# Setting the ranges starting with the plugin on top of the stack  
#####  
$PPPoX_1 rangeList.clear  
  
set PPPoX_R1 [::IxLoad new ixNetPppoxRange] Configure the  
PPPoX plugin.  
# ixNet objects needs to be added in the list  
# before they are configured!  
$PPPoX_1 rangeList.appendItem -object $PPPoX_R1  
  
$PPPoX_R1 config \  
-serverSignalLoopEncapsulation      false \  
-padrTimeout                        10 \  
-serverBaseIID                      "00:11:22:11:00:00:00:01" \  
-enableEchoReq                      false \  
-authTimeout                        10 \  
-actualRateUpstream                10 \  
-ncpTimeout                         10 \  
-clientIpIncr                       "0.0.0.1" \  
-lcpOptions                         "LCP Options" \  
-serverNetmaskOptions               "disableExtension" \  
-serverSignalLoopId                 false \  
-ipv6PoolPrefixLen                  48 \  
-clientSignalLoopChar               false \  
-clientIIDIncr                      1 \  
-authRetries                        20 \  
-enableMruNegotiation               false \  
-authType                           "none" \  
-clientNetmask                      "255.0.0.0" \  
-ncpType                             "IPv4" \  
-echoReqInterval                    10 \  
-domainList                         "Domain Groups" \  
-clientBaseIID                      "00:11:11:11:00:00:00:01" \  
-clientNetmaskOptions               "disableExtension" \  
-clientSignalLoopId                 false \  
-authOptions                        "Authentication Options" \  
-lcpTimeout                         10 \  
-pppoeOptions                       "PPPoE Options" \  
-ncpRetries                         3 \  
-serviceName                       "" \  
-enableRedial                       true \  

```

PppoX Plugin Example

```

$PPPoX_R1 domainGroupList.clear
$PPPoX_R1 acNameTable.clear
$PPPoX_R1 acMacTable.clear

set MAC_R9 [$PPPoX_R1 getLowerRelatedRange "MacRange"]

$MAC_R9 config \
  -count 1 \
  -name "MAC-R9" \
  -enabled true \
  -mtu 1500 \
  -mac "6A:40:7E:B6:00:00" \
  -incrementBy "00:00:00:00:00:01"

set VLAN_R1 [$PPPoX_R1 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
  -incrementStep 1 \
  -uniqueCount 4094 \
  -name "VLAN-R1" \
  -innerIncrement 1 \
  -innerUniqueCount 4094 \
  -enabled true \
  -innerFirstId 1 \
  -increment 1 \
  -priority 1 \
  -firstId 1 \
  -innerIncrementStep 1 \
  -idIncrMode 2 \
  -innerEnable false \
  -innerPriority 1

```

Clear the lists of PPPoX plugin child objects, if not used.

Configure the MAC addresses for the PPPoX IP range.

Configure the VLAN tags for the DHCP IP ranges.

L2TP Plugin

SYNOPSIS

DESCRIPTION

Configures an L2TP plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`l2tpRanges`

Name of the L2tpRangeList containing the address ranges used by this plugin.

Default value = "None"

EXAMPLE

```
set L2TP_1 [::IxLoad new ixNetL2tpPlugin]
# ixNet objects needs to be added in the list before they are configured!
$IP_4 childrenList.appendItem -object $L2TP_1

$L2TP_1 config \
    -name "L2TP-1"

$L2TP_1 childrenList.clear

$L2TP_1 extensionList.clear

$IP_4 extensionList.clear

$MAC_VLAN_9 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO



L2tpPortGroupData

SYNOPSIS

DESCRIPTION

Configures the L2TP EAPoUDP Rate Control and Protocol Parameters.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`associates`

Name of the list of associates. This list must an `AssociateList` object.

This option is read only.

Default value = "None"

`overrideGlobalRateControls`

If `false`, the global setup and teardown rate values will be equally divided among the ports.

If `true`, The setup and teardown parameters defined at the port level will override those defined at the global level.

For example, if you have set the initial setup rate to 150 on the global level, and you have defined two ports, these 150 session setups will be evenly distributed across the ports (75 for each). If you then enable Override Global Rate Controls, you can modify the number of session setups for each of the ports (such as changing the distribution from 75-75 to 120-30).

Default value = "False"

`setupRateInitial`

The number of PPP sessions to set up, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingRequests`

The maximum number of PPP sessions that can be outstanding at any given time. The minimum is 1, the maximum is 1000.

Default value="300"

`teardownRateInitial`

The number of PPP sessions to tear down, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingReleases`

The maximum number of PPP sessions that can be released at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

`useWaitForCompletionTimeout`

Enables the application to wait for a specified amount of time for the sessions to negotiate before declaring a negotiation timeout.

Default value = "False"

`waitForCompletionTimeout`

If `useWaitForCompletionTimeout` is true, specify the number of seconds that the application will wait for the sessions to negotiate.

The default is 120, the minimum is 1, and the maximum is 65535.

Default value = "120"

`enablePerSessionStatGeneration`

Enables or disables per-session statistics generation. When enabled, PPP protocol statistics are generated during the session negotiation phase of an L2TP or PPP test and written to a CSV file. The CSV file is generated at the end of the session negotiation phase. The concatenated results for each port are returned as a single file.



Statistics are generated only for client ports because server ports do not establish any sessions during the negotiation phase.

Default value = "False"

`perSessionStatFilePrefix`

If `enablePerSessionStatGeneration` is true, specify the prefix to use for the name of the per-session PPP protocol statistics file.

The per-session PPP protocol statistics file names are of the form:

```
StatsFilePrefix_chassis_card_port_TimeStamp_.csv
```

The CSV files are saved in this folder:

```
C:\Program Files\Ixia\IxNetwork\data\result\[UserName]\Per Session Stats  
PPP_L2TP\
```

Default value = "MY_PREFIX"

`role`

The role that the L2TP network group plays in the test configuration:

- lac - L2TP Access Concentrator (LAC)
- lns - L2TP Network Server (LNS)

Default value = "lac"

EXAMPLE

SEE ALSO

PLSessionDataBase

SYNOPSIS

DESCRIPTION

Configures the PPPoX and L2TP network group settings.

SUBCOMMANDS

OPTIONS

`setupRateInitial`

The number of PPP sessions to set up, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingRequests`

The maximum number of PPP sessions that can be outstanding at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

`teardownRateInitial`

The number of PPP sessions to tear down, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingReleases`

The maximum number of PPP sessions that can be released at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

EXAMPLE

SEE ALSO

L2tpSessionData

SYNOPSIS

DESCRIPTION

Global L2TP settings.

SUBCOMMANDS

OPTIONS

`setupRateInitial`

The number of PPP sessions to set up, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingRequests`

The maximum number of PPP sessions that can be outstanding at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

`teardownRateInitial`

The number of PPP sessions to tear down, per second. The default is 300, the minimum is 1, the maximum is 1000.

Default value = "300"

`maxOutstandingReleases`

The maximum number of PPP sessions that can be released at any given time. The minimum is 1, the maximum is 1000.

Default value = "300"

EXAMPLE

SEE ALSO

L2tpRangeList

SYNOPSIS

DESCRIPTION

List of L2tpRange objects.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



L2tpRange

SYNOPSIS

DESCRIPTION

Configures the L2TP Range Parameters.

SUBCOMMANDS

OPTIONS

Same as PLRangeBase plus the following:

`basicOptions`

Default value = "L2PT Options"

`tunnelSourceIp`

Default value = "None"

`tunnelDestinationIp`

Defines the base address to be used for the L2TP tunnel destinations.

Default value = "10.10.10.1"

`tunnelIncrementBy`

Defines the increment to be used for enumerating all the addresses in the destination range.

Default value = "0.0.0.1"

`controlPlaneOptions`

Default value = "Control Plane Options"

`sessionsPerTunnel`

The number of PPP sessions that each L2TP tunnel may carry.

The default is 1, the minimum is 1, and the maximum is 32000.

Default value = "1"

`noCallTimeout`

The amount of time, in seconds to wait to receive an L2TP request for connection. If it does not receive a call within this time, the tunnel is closed. The default is 5, the minimum is 1, and the maximum is 180.

Default value = "5"

`enableHelloRequest`

If checked, Hello messages are sent to confirm that a tunnel is still up.

Default value = "False"

`helloRequestInterval`

If Hello Requests are enabled, this value determines the amount of time that can elapse between the time the most recent control or data message are sent, and the time a Hello message is sent. The default is 60, the minimum is 1, and the maximum is 180.

Default value = "60"

`bearerCapability`

Indicates to the DUT the bearer device types from which incoming calls will be accepted. You can set this parameter to the following values:

- Analog: advertises its bearer device type as analog only.
- Digital: advertises its bearer device type as digital only.
- Both: advertises its bearer device types as analog and digital.

The default value is Both.

See `BearerCapabilityChoices` for the list of choices.

Default value = "3"

`bearerType`

Device type requested for outgoing calls. You can set this parameter to the following values:

- Analog: requests analog device capability.
- Digital: requests digital device capability.

The default value is Digital.

See `BearerTypeChoices` for the list of choices.

Default value = "2"

`controlMsgsRetryCounter`



Number of times a control message for which an acknowledgment has not been received will be retransmitted. The default is 30, the minimum is 1, and the maximum is 100.

Default value = "30"

`initRetransmitInterval`

Initial amount of time that can elapse before an unacknowledged control message is retransmitted.

If a control message is retransmitted and still does not receive a reply from the DUT, the control message will be retransmitted at increasingly longer intervals until it receives a reply. The Max Retransmit Interval parameter establishes the upper limit on retransmit interval.

The default is 2, the minimum is 1, and the maximum is 65535.

Default value = "2"

`maxRetransmitInterval`

Unacknowledged control messages are retransmitted.

If a control message is transmitted at the Maximum Retransmit Interval and still does not receive a reply, the associated tunnel is torn down along with the PPP sessions within it.

The default is 8, the minimum is 1, and the maximum is 65535.

Default value = "8"

`receiveWindowSize`

Configures the size of the sliding window used for managing control message transmission. The values for this parameter are expressed in units of unacknowledged control messages. For example, if you set this parameter to 4, the DUT can send control messages until it has four messages for which it is waiting for acknowledgments. At that point, it must wait for one or more of the messages to be acknowledged before it can send any new control messages.

The default is 10, the minimum is 1, and the maximum is 2048.

Default value = "10"

`enableRedial`

If the L2TP link goes down and this parameter is enabled, the DUT will be redialed to reestablish the link.

Default value = "False"

`redialInterval`

Number of seconds that can elapse between attempts to redial the DUT to re-establish a downed L2TP link. The default is 10, the minimum is 1, and the maximum is 65535.

Default value = "10"

`maxRedialAttempts`

The maximum number attempts IxLoad will make to redial the DUT to re-establish a downed L2TP link. The default is 20, the minimum is 1, and the maximum is 65535.

Default value = "20"

`dataPlaneOptions`

Default value = "Data Plane Options"

`enableControlChecksum`

Enables the use of UDP checksums on control messages.

The L2TP RFC (RFC 2661) recommends that UDP checksums always be enabled on control packets.

Default value = "True"

`enableDataChecksum`

Enables the use of UDP checksums on data messages.

Default value = "False"

`udpSourcePort`

The UDP port used to send requests to the DUT for L2TP connections. The well-known port number for L2TP is 1701.

Default value = "1701"

`udpDestinationPort`

The UDP port that the DUT uses to listen for L2TP connection requests. The well-known port number for L2TP is 1701.

Default value = "1701"

`useLengthBitInPayload`

If `true`, sets the Length bit in data messages, adding the Length field to the header and indicating that it is present.

Default value = "False"

`useOffsetBitInPayload`

If `true`, sets the Offset bit in data messages, adding the Offset Length field to the header and indicating that it is present.



Default value = "False"

`offsetByte`

If `useOffsetBitInPayload` is `true`, the `Offset Byte` field specifies the byte value used to pad the header from the end of the `Offset Length` field to the beginning of the payload. The default is 0, the minimum is 0, and the maximum is 255.

Default value = "0"

`offsetLength`

The `Offset Length` field specifies the number of octets past the L2TP header at which the payload data starts. The default is 0, the minimum is 0, and the maximum is 255.

Default value = "0"

`useSequenceNoInPayload`

If `true`, sets the `Sequence` bit in data messages, adding the `Sequence Number` fields to the header and indicating that they are present.

Default value = "False"

`l2tpAuthOptions`

Default value = "L2PT Authentication Options"

`useHiddenAVPs`

If `true`, `Attribute Value Pair` hiding is enabled. This enables the use of hidden AVPs, `Attribute-Value Pairs` (parameters and values) within control messages that are protected by encryption.

Hiding AVPs is done to hide sensitive control message data such as user passwords or user IDs.

To use Hidden AVPs, you must also configure the `Host` and `Secret` fields, which define the shared secret for a host.

Default value = "False"

`tunnelAuthentication`

Enables a LAC or LNS to authenticate the identity of a peer it is contacting or being contacted by during control connection establishment.

If `Tunnel Authentication` is enabled, the hosts exchange control messages that include the host names and a shared secret. If the expected response and response received do not match, the tunnel will not be established.

To use Tunnel Authentication, you must also configure the `lacHostName` and `lacSecret` fields, which define the shared secret for a host.

See `Tunnel AuthenticationChoices` for the list of choices.

Default value = "none"

`lacHostName`

A text string identifying IxLoad to the DUT for the purposes of Hidden AVPs and Tunnel Authentication. The default value is *ixia*. The text string can have a maximum of 32 characters.

Default value = "ixia"

`lacSecret`

A text string used as shared secret between the test port and the DUT for Hidden AVPs and Tunnel Authentication. The default value is *ixia*. The text string can have a maximum of 32 characters.

Default value = "ixia"

`lacToLNSMapping`

This parameter defines how the LAC (DUT) accesses the LNS (Ixia port). See `lacToLNSMappingChoices` for the list of choices.

Default value = "gateway"

`lnsIpNumber`

The number of IP addresses that will be created to simulate an LNS. The default is 1, the minimum is 1, and the maximum is 65535.

Default value = "1"

`baseLnsIp`

The first IP address that will be used to simulate an LNS.

Default value = "0.0.0.0"

`incrementBy`

The amount of increase between each incremented IP address.

Default value = "1"

`ipIncrementOctet`



The octet in the Base LNS IP address that is incremented to create additional IP addresses. Octets are numbered 1 to 4 from left (most-significant) to right (least-significant).

Default value = "4"

`lnsIpList`

Name of the list of LNS IP addresses. Must be an `L2tpLnsIpAddressList` object.

Default value = "LNS IPs"

`lnsIpAddresses`

Default value = "None"

`domainToIpList`

Default value = "Domain To LNS"

`sessionAllocMethod`

Method for allocating sessions among tunnels. See `sessionAllocMethod` for the choices.

Default value = "nextTunnel"

`peerHostName`

On LAC ports, this is a text string identifying `IxLoad` to the DUT for the purposes of Hidden AVPs and Tunnel Authentication. This text string is also used for the hostname AVP. The default value is `ixia`. The text string can have a maximum of 32 characters.

On LNS ports, this is the hostname expected by the LNS in authentication.

Default value = "ixia"

`tunnelStartId`

Default value = "1"

`framingCapability`

Framing capability. See `framingCapabilityChoices` for the choices.

Default value = "1"

`sessionStartId`

Default value = "1"

txConnectSpeed

Default value = "268435456"

rxConnectSpeed

Default value = "268435456"

enableProxy

Default value = "True"

EXAMPLE

```
$L2TP_1 l2tpRanges.clear
```

```
set L2TP_R1 [::IxLoad new ixNetL2tpRange]
```

```
# ixNet objects needs to be added in the list before they are configured!
```

```
$L2TP_1 l2tpRanges.appendItem -object $L2TP_R1
```

```
$L2TP_R1 config \
```

```
-authTimeout                10 \  
-lacToLNSMapping            "gateway" \  
-authRetries                 20 \  
-authType                   "none" \  
-sessionsPerTunnel          1 \  
-echoReqInterval            10 \  
-domainList                  "Domain Groups" \  
-peerHostName                "ixia" \  
-useHiddenAVPs               false \  
-incrementBy                 1 \  
-ncpRetries                  3 \  
-serverPrimaryDnsAddress     "10.10.10.10" \  
-clientDnsOptions            "disableExtension" \  

```



```
-enableHelloRequest          false \  
-lcpTermTimeout             15 \  
-baseLnsIp                  "0.0.0.0" \  
-name                        "L2TP-R1" \  
-lcpTermRetries             3 \  
-serverIIDIncr              1 \  
-rxConnectSpeed             268435456 \  
-clientBaseIID              "00:11:11:11:00:00:00:01" \  
-numSessions                 1 \  
-tunnelAuthentication       "none" \  
-serverBaseIID              "00:11:22:11:00:00:00:01" \  
-ncpTimeout                 10 \  
-tunnelDestinationIp        "10.10.10.1" \  
-ipv6PoolPrefixLen          48 \  
-l2tpAuthOptions            "L2PT Authentication Options" \  
-clientIIDIncr              1 \  
-udpDestinationPort         1701 \  
-lacSecret                   "ixia" \  
-ipIncrementOctet           4 \  
-ncpType                     "IPv4" \  
-lnsIpList                   "LNS IPs" \  
-authOptions                 "Authentication Options" \  
-offsetByte                  0 \  
-enableRedial                false \  
-lcpRetries                  3 \  
-maxRetransmitInterval      8 \  
-chapName                    "user" \  
-useSequenceNoInPayload     false \  

```

```
-serverSecondaryDnsAddress      "11.11.11.11" \  
-basicOptions                   "L2PT Options" \  
-lacHostName                    "ixia" \  
-serverNetmask                  "255.255.255.0" \  
-bearerCapability               "3" \  
-receiveWindowSize              10 \  
-serverDnsOptions               "disableExtension" \  
-clientPrimaryDnsAddress        "8.8.8.8" \  
-lnsIpNumber                    1 \  
-tunnelIncrementBy              "0.0.0.1" \  
-chapSecret                     "secret" \  
-enableEchoReq                  false \  
-lcpOptions                      "LCP Options" \  
-serverNetmaskOptions           "disableExtension" \  
-helloRequestInterval           60 \  
-clientNetmask                  "255.0.0.0" \  
-initRetransmitInterval         2 \  
-clientNetmaskOptions           "disableExtension" \  
-sessionAllocMethod             "nextTunnel" \  
-enableControlChecksum          true \  
-framingCapability              "1" \  
-useLengthBitInPayload          false \  
-ipv6PoolPrefix                 "1:1:1::" \  
-enableEchoRsp                  true \  
-serverIpIncr                   "0.0.0.0" \  
-papPassword                     "password" \  
-txConnectSpeed                 268435456 \  
-ipv6AddrPrefixLen              64 \  

```



```
-redialInterval          10 \  
-clientBaseIp           "1.1.1.1" \  
-domainToIpList         "Domain To LNS" \  
-controlMsgsRetryCounter 30 \  
-clientSecondaryDnsAddress "9.9.9.9" \  
-enabled                 true \  
-mtu                     1492 \  
-serverBaseIp           "2.2.2.2" \  
-noCallTimeout          5 \  
-clientIpIncr           "0.0.0.1" \  
-dataPlaneOptions       "Data Plane Options" \  
-enableDataChecksum     false \  
-enableProxy            true \  
-lcpTimeout             10 \  
-enableDomainGroups     false \  
-bearerType             "2" \  
-offsetLength           0 \  
-udpSourcePort          1701 \  
-maxRedialAttempts      20 \  
-sessionStartId         1 \  
-papUser                "user" \  
-controlPlaneOptions    "Control Plane Options" \  
-useOffsetBitInPayload  false \  
-tunnelStartId          1 \  
-useMagic                true
```

```
$L2TP_R1 domainGroupList.clear
```

```
$L2TP_R1 lnsIpAddresses.clear
```

```
set IP_R4 [$L2TP_R1 getLowerRelatedRange "IPv4V6Range"]
```

```
$IP_R4 config \
```

```
-count 1 \  
-name "IP-R4" \  
-gatewayAddress "0.0.0.0" \  
-enabled true \  
-autoMacGeneration true \  
-mss 1460 \  
-incrementBy "0.0.0.1" \  
-prefix 16 \  
-gatewayIncrement "0.0.0.0" \  
-gatewayIncrementMode "perSubnet" \  
-generateStatistics false \  
-ipAddress "10.10.0.5" \  
-ipType "IPv4"
```

```
set MAC_R8 [$IP_R4 getLowerRelatedRange "MacRange"]
```

```
$MAC_R8 config \
```

```
-count 1 \  
-name "MAC-R8" \  
-enabled true \  
-mtu 1500 \  
-mac "00:0A:0A:00:05:00" \  
-incrementBy "00:00:00:00:00:01"
```



```
set VLAN_R1 [$IP_R4 getLowerRelatedRange "VlanIdRange"]
```

```
$VLAN_R1 config \
```

```
-incrementStep          1 \
-uniqueCount            4094 \
-name                   "VLAN-R1" \
-innerIncrement         1 \
-innerUniqueCount      4094 \
-enabled                true \
-innerFirstId           1 \
-increment              1 \
-priority               1 \
-firstId                1 \
-innerIncrementStep    1 \
-idIncrMode             2 \
-innerEnable            false \
-innerPriority          1
```

SEE ALSO

BearerCapabilityChoices

SYNOPSIS

List of choices for the BearerCapability option of the L2tpRange object.

SUBCOMMANDS

OPTIONS

Value	Description
1	Digital: advertises its bearer device type as digital only.
2	Analog: advertises its bearer device type as analog only.
3	Both: advertises its bearer device types as analog and digital.

EXAMPLE

SEE ALSO



BearerTypeChoices

SYNOPSIS

DESCRIPTION

List of choices for the `BearerCapability` option of the `L2tpRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
1	Digital: requests digital device capability.
2	Analog: requests analog device capability.

EXAMPLE

SEE ALSO

TunnelAuthenticationChoices

SYNOPSIS

DESCRIPTION

List of choices for the `tunnelAuthentication` option of the `L2tpRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
none	Tunnel Authentication Disabled
hostname	Authenticate Hostname"

EXAMPLE

SEE ALSO



LacToLNSMappingChoices

SYNOPSIS

DESCRIPTION

List of choices for the `LacToLNSMappingChoices` option of the `L2tpRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
domain	Through Domain: The LAC maps to the LNS through one of the Domain-to-LNS mappings defined on the Domain to LNS dialog (access to this dialog is through the <i>Domain to LNS</i> column).
gateway	Through Gateway: The LAC maps to the LNS through the gateway specified in the IP tab.

EXAMPLE

SEE ALSO

SessionAllocMethodChoices

SYNOPSIS

DESCRIPTION

List of session allocation method choices for the `sessionAllocMethod` option of the `L2tpRange` object.

SUBCOMMANDS

OPTIONS

Value	Description
<code>nextTunnel</code>	Distribute sessions among tunnels
<code>fillTunnel</code>	Fill tunnels in order

EXAMPLE

SEE ALSO



FramingCapabilityChoices

SYNOPSIS

DESCRIPTION

Values for framingCapability option of L2TPRange object.

SUBCOMMANDS

OPTIONS

Value	Description
1	Synchronous
2	Asynchronous

EXAMPLE

SEE ALSO

L2tpLnsIpAddressList

SYNOPSIS

DESCRIPTION

List of L2tpLnsIpAddress objects.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



L2tpLnsIpAddress

SYNOPSIS

DESCRIPTION

LNS-toIP address mapping for use in an `L2tpLnsIpAddressList` object.

SUBCOMMANDS

OPTIONS

`selected`

If true, this address will be used.

Default value = "False"

`address`

IP address.

Default Value = ""

EXAMPLE

SEE ALSO

PLAssociate

SYNOPSIS

DESCRIPTION

Associates a PPOX or L2TP client with a PPPoX or L2TP server.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

L2tp Plugin Example

This section shows an example of how to create an L2TP plugin in the Tcl API.

L2tp Plugin Example

```
#####  
# IxLoad ScriptGen created TCL script  
# Network1 serialized using version 4.10.0.79  
# L2TP.tcl made on Aug 14 2008 15:26  
#####  
  
set Network1 [::IxLoad new ixNetworkGroup $chassisChain]  
$Network1 config \  
    -comment          "" \  
    -name              "Network1" \  
    -macMappingMode   0 \  
    -linkLayerOptions 0  
  
$Network1 globalPlugins.clear  
  
Begin appending items to global plugin list.  
  
set GratARP [::IxLoad new ixNetGratArpPlugin]  
# ixNet objects needs to be added in the list  
# before they are configured!  
$Network1 globalPlugins.appendItem -object $GratARP  
  
$GratARP config \  
    -enabled          true \  
    -name             "GratARP"
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, enable Gratuitous ARP.

L2tp Plugin Example

```
set TCP [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $TCP
```

Configure the TCP
portion of the stack.

```
$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 7200 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -ip_no_pmtu_disc false \
    -tcp_window_scaling false \
```



L2tp Plugin Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
    -teardownInterfaceWithUser      false \
    -name                            "Settings" \
    -interfaceBehavior              0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType                "master" \
    -negotiateMasterSlave           true

$Ethernet_1 config \
    -advertise10Full                true \
    -name                            "Ethernet-1" \
    -autoNegotiate                   true \
    -advertise100Half               true \
    -advertise10Half                true \
    -speed                           "k100FD" \
    -advertise1000Full              true \
    -advertise100Full               true \
    -cardElm                         \
$my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear
```

Configure the Dynamic
Control plane settings.

Configure the physical
layer properties.

L2tp Plugin Example

```
set MAC_VLAN_9 [::IxLoad new ixNetL2EthernetPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_9

$MAC_VLAN_9 config \
    -name "MAC/VLAN-9"

$MAC_VLAN_9 childrenList.clear

set IP_4 [::IxLoad new ixNetIpV4V6Plugin]
# ixNet objects needs to be added in the list
# before they are configured!
$MAC_VLAN_9 childrenList.appendItem -object $IP_4

$IP_4 config \
    -name "IP-4"

$IP_4 childrenList.clear

set L2TP_1 [::IxLoad new ixNetL2tpPlugin]
# ixNet objects needs to be added in the list
# before they are configured!
$IP_4 childrenList.appendItem -object $L2TP_1

$L2TP_1 config \
    -name "L2TP-1"

$L2TP_1 childrenList.clear
$L2TP_1 extensionList.clear

$IP_4 extensionList.clear

$MAC_VLAN_9 extensionList.clear

$Ethernet_1 extensionList.clear
```

Configure the MAC addresses and VLAN tags.

Configure an IP address range.

Configure an L2TP plugin and append it to the IP range.

Name the L2TP plugin.

Clear the lists of extension protocols.



L2tp Plugin Example

```
#####
# Setting the ranges starting with the plugin on top of the stack
#####
$L2TP_1 l2tpRanges.clear

set L2TP_R1 [::IxLoad new ixNetL2tpRange]
# ixNet objects needs to be added in the list
# before they are configured!
$L2TP_1 l2tpRanges.appendItem -object $L2TP_R1

$L2TP_R1 config \
  -authTimeout 10 \
  -lacToLNSMapping "gateway" \
  -authRetries 20 \
  -authType "none" \
  -sessionsPerTunnel 1 \
  -echoReqInterval 10 \
  -domainList "Domain Groups" \
  -peerHostName "ixia" \
  -useHiddenAVPs false \
  -incrementBy 1 \
  -ncpRetries 3 \
  -serverPrimaryDnsAddress "10.10.10.10" \
  -clientDnsOptions "disableExtension" \
  -enableHelloRequest false \
  -lcpTermTimeout 15 \
  -baseLnsIp "0.0.0.0" \
  -name "L2TP-R1" \
  -lcpTermRetries 3 \
  -serverIIDIncr 1 \
  -rxConnectSpeed 268435456 \
  -clientBaseIID "00:11:11:11:00:00:00:01" \
  -numSessions 1 \
  -tunnelAuthentication "none" \
  -serverBaseIID "00:11:22:11:00:00:00:01" \
  -ncpTimeout 10 \
  -tunnelDestinationIp "10.10.10.1" \
  -ipv6PoolPrefixLen 48 \
  -l2tpAuthOptions "L2TP Authentication Options" \
  -clientIIDIncr 1 \
  -udpDestinationPort 1701 \
```

Configure the address ranges for the L2TP plugin.

Configure the L2TP plugin.

L2tp Plugin Example

```
set IP_R4 [$L2TP_R1 getLowerRelatedRange "IpV4V6Range"]
```

```
$IP_R4 config \  
  -count 1 \  
  -name "IP-R4" \  
  -gatewayAddress "0.0.0.0" \  
  -enabled true \  
  -autoMacGeneration true \  
  -mss 1460 \  
  -incrementBy "0.0.0.1" \  
  -prefix 16 \  
  -gatewayIncrement "0.0.0.0" \  
  -gatewayIncrementMode "perSubnet" \  
  -generateStatistics false \  
  -ipAddress "10.10.0.5" \  
  -ipType "IPv4"
```

Configure an IP range for the L2TP plugin.

```
set MAC_R8 [$IP_R4 getLowerRelatedRange "MacRange"]
```

```
$MAC_R8 config \  
  -count 1 \  
  -name "MAC-R8" \  
  -enabled true \  
  -mtu 1500 \  
  -mac "00:0A:0A:00:05:00" \  
  -incrementBy "00:00:00:00:00:01"
```

Configure the MAC addresses for the IP range.



L2tp Plugin Example

```
set VLAN_R1 [$IP_R4 getLowerRelatedRange "VlanIdRange"]

$VLAN_R1 config \
  -incrementStep 1 \
  -uniqueCount 4094 \
  -name "VLAN-R1" \
  -innerIncrement 1 \
  -innerUniqueCount 4094 \
  -enabled true \
  -innerFirstId 1 \
  -increment 1 \
  -priority 1 \
  -firstId 1 \
  -innerIncrementStep 1 \
  -idIncrMode 2 \
  -innerEnable false \
  -innerPriority 1
```

Configure the VLAN tags for the IP range.

GTP Plugin



GTPSPlugin

SYNOPSIS

DESCRIPTION

Configures a GTP SGSN plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`sgsnRangeList`

List of emulated SGSNs. This must be a `GTPSRange` object.

Default value = "None"

`ueRangeList`

List of emulated UEs. This must be a `GTPSUERange` object.

Default value = "None"

ixGTPVersion

GTP version.

Default value = "3.20.1.51"

EXAMPLE

```
set GTP_1 [::IxLoad new ixNetGTPSPlugin]
# ixNet objects needs to be added in the list before they are configured!
$MAC_VLAN_5 childrenList.appendItem -object $GTP_1

$GTP_1 config \
  -ixGTPVersion          "3.20.1.79" \
  -name                  "GTP-1"

$GTP_1 childrenList.clear

$GTP_1 extensionList.clear

$MAC_VLAN_5 extensionList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO

GTPS Plugin Example



GTPSPluginSessionData

SYNOPSIS

DESCRIPTION

Configures the GTP SGSN global parameters including the QoS to TOS global settings that establish a mapping of DiffServ values to UMTS traffic classes. This mapping applies to all network groups and all ranges defined for the test.

SUBCOMMANDS

OPTIONS

`qoSToTOSMapping_pHBConversational`

IP Diffserv value associated to UMTS Streaming traffic class.

Default value = "2"

`qoSToTOSMapping_pHBStreaming`

IP Diffserv value associated to UMTS Streaming traffic class.

Default value = "2"

`qoSToTOSMapping_pHBInteractivePlus`

IP Diffserv value associated to UMTS InteractivePlus traffic class.

Default value = "2"

`qoSToTOSMapping_pHBBackground`

IP Diffserv value associated to UMTS Background traffic class.

Default value = "0"

`qoSToTOSMapping_pHBInteractive`

IP Diffserv value associated to UMTS Interactive traffic class.

Default value = "3"

`qoStoTOSMapping_pHBSignalisation`

IP Diffserv value associated to UMTS signalling.

Default value = "2"

`qoStoTOSMapping_pHBAF1`

IP TOS field value associated to diffserv AF1.

Default value = "10"

`qoStoTOSMapping_pHBAF2`

IP TOS field value associated to diffserv AF2.

Default value = "18"

`qoStoTOSMapping_pHBAF3`

IP TOS field value associated to diffserv AF3.

Default value = "26"

`qoStoTOSMapping_pHBAF4`

IP TOS field value associated to diffserv AF4.

Default value = "34"

`setupRateInitial`

The number of PDP contexts (multimedia sessions) to set up within the first second.

The default is 300, the minimum is 1, and the maximum is 144,000.

Default value = "300"

`maxOutstandingRequests`

The threshold at which the GTP plug-in begins throttling back the number of new PDP contexts being set up.

The default is 300, the minimum is 1, and the maximum is 144,000.

Default value = "300"

`teardownRateInitial`



The number of PDP contexts to release in the first second after the stop command is received.

The default is 300, the minimum is 1, and the maximum is 144,000.

Default value = "300"

`maxOutstandingReleases`

The maximum number of PDP contexts waiting for termination at any time by the GTP network stack element.

The default is 300, the minimum is 1, and the maximum is 144,000.

Default value = "300"

`pduFilterList`

Name of the list of PDU filters. This must be a `GTPFilterList` object.

Default value = "None"

EXAMPLE

SEE ALSO

GTPSPluginPortGroupData

SYNOPSIS

DESCRIPTION

Configures the GTP SGSN network group session parameters.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`associates`

Name of the list of associates. This list must an `AssociateList` object.

This option is read only.

Default value = "None"

`currentActivity`

Name of the `GTPSActivity` object used by this port group.

Default value = "None"

`pduFilterList`

Name of the `GTPSPDUFilterList` object used by this object. This option is read only.

Default value = "None"



`overrideGlobalRateControls`

If selected, all the rate settings defined at the global level will be overridden by the rate settings defined on this port.

For example, if you have set the initial setup rate to 100 on the global level, and you have defined two ports, these 100 session setups will be evenly distributed across the ports (50 for each). If you then enable Override Global Setup Rate, you can modify the number of session setups for each of the ports (such as changing the distribution from 50-50 to 60-40).

Default value = "False"

`setupRateInitial`

The number of multimedia sessions to set up within the first second.

Default value = "300"

`maxOutstandingRequests`

The threshold at which the GTP network stack element begins throttling back the number of new client sessions being set up.

Default value = "300"

`teardownRateInitial`

The number of multimedia sessions to terminate in the first second after the stop command is received.

Default value = "300"

`maxOutstandingReleases`

The maximum number of sessions waiting for termination at any time by the GTP network stack element.

Default value = "300"

EXAMPLE

SEE ALSO



GTPSActivity

SYNOPSIS

DESCRIPTION

Configures a GTP traffic activity.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID

Default value = "0"

`tearDownIndicator`

Default value = "False"

`createSecondaryContext`

If false, the GTP plug-in will create a secondary PDP contexts only when the QoS requirements differ from those of the primary PDP Context.

If true, the GTP plug-in will create a secondary context for this activity even if the QoS requirements are identical to those of the primary PDP Context.

Default value = "True"

`sAPN`

The Access Point Name.

Note: All of the IxLoad activities defined for a GTP range must be configured with the same APN.

Default value = "web.ixiacom.com"

`sPCO_Protocol`

Default value = ""

sPCO_Container

Default value = ""

iTFTOp

The TFT Information Element operation code:

- Spare
- Create new TFT
- Delete existing TFT
- Add packet filters to existing TFT
- Replace packet filters in existing TFT
- Delete packet filters from existing TFT
- No TFT operation (use this code if a parameters list is included but no packet filter list is included in the traffic flow template information)
- Reserved

The TFT IE operation code occupies bits 6, 7, and 8 in octet 3 of the TFT information element.

Default value = "1"

iTFTNumFilter

The number of packet filters in the TFT Information Element packet filter list. (The Number of packet filters field occupies bits 1, 2, 3, and 4 in octet 3 of the TFT information element.)

If the *Operation* value is either "Delete existing TFT" or "No TFT operation," set the Filter Count to zero. For all other operations, the Filter Count must be greater than zero and less than or equal to 8.

Default value = "1"

sTFTFilters

The TFT (Traffic Flow Template) filter type.

Default value = "None"

sTFTFiltersCustom

Default value = "0X0X03500050"

QoS Parameters

delayClass



The Delay Class specifies how much delay is acceptable for the traffic.

Default value = "0"

reliabilityClass

The Reliability Class specifies how reliable the traffic delivery must be. A value of 1 specifies the highest level of reliability, and a value of 5 specifies the lowest level of reliability.

Default value = "2"

precedenceClass

The Precedence Class specifies the priority of the traffic. The precedence settings are High, Normal, and Low.

Default value = "2"

peakThroughput

Peak Throughput specifies the required throughput during periods of peak traffic activity.

The default value is 32k (32,000 octets per second).

Default value = "6"

meanThroughput

Mean throughput specifies the required mean throughput, expressed as either:

- The mean number of octets per hour, or
- The best available throughput.

The default value is Best.

Default value = "31"

deliveryOfErroneousSDU

Specifies whether or not erroneous SDUs are delivered:

- Yes: erroneous SDUs are delivered.
- No: erroneous SDUs are not delivered.

The default value is No.

Default value = "1"

deliveryOrder

Default value = "2"

`trafficClass`

UMTS QoS defines four traffic classes:

- Conversational (such as voice and video conferencing)
- Steaming (used for video streaming)
- Interactive (such as voice and online gaming)
- Background (such as email and file transfers)

The default value is Background.

Default value = "4"

`maxSDUSize`

Specifies the maximum Service Data Unit (SDU) size, in octets.

Default value = "151"

`maxBitRateUL`

Maximum BitRate for Uplink. Specifies the maximum upstream bit rate, in kbps.

Default value = "8640"

`maxBitRateDL`

Maximum BitRate for Downlink. Specifies the maximum downstream bit rate, in kbps.

Default value = "8640"

`residualBER`

Default value = "7"

`sDUErrorRatio`

Specifies the acceptable ratio of SDU errors to SDUs transmitted.

Default value = "6"

`transferDelay`

Specifies the maximum acceptable transfer delay, in milliseconds.

Default value = "4000"



trafficHandlingPriority

Specifies the desired traffic handling priority. There are three priority levels:

- Priority Level 1:
- Priority Level 2:
- Priority Level 3:

Default value = "1"

guaranteedBitRateUL

Specifies the guaranteed upstream bit rate, in kbps.

Default value = "64"

guaranteedBitRateDL

Specifies the guaranteed downstream bit rate, in kbps.

Default value = "64"

signalingIndication

Default value = "0"

sourceStatisticsDescriptor

Default value = "0"

maxBitRateDLEx

Default value = "0"

guaranteedBitRateDLEx

Default value = "0"

EXAMPLE

SEE ALSO



GTPSPDUFilterList

SYNOPSIS

DESCRIPTION

List of `GTPSPDUFilter` objects for use by a `GTPSPuginSessionData` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

GTPSPDUFilter

SYNOPSIS

DESCRIPTION

Creates a GTP PDU filter `GTPSPDUFilterList` object.

SUBCOMMANDS

OPTIONS

`enabled`

If `true`, this PDU filter will be used.

Default value = "False"

`filter`

The PDU filter definition.

Default value = "TX g3_gtp_v1_pdu Message_Type=
G3_GTP_v1_Message_Type_Delete_PDP_Context_Request replace
3614000d000c26ee000500010101ffff0013fe1499"

EXAMPLE

SEE ALSO



GTPSPDUFilterList2

SYNOPSIS

DESCRIPTION

List of `GTPSPDUFilter` objects.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

GTPSRangeList

SYNOPSIS

DESCRIPTION

List of `GTPSRange` objects for use by a `GTPSPlugin` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



GTPSRange

SYNOPSIS

DESCRIPTION

Creates an address range for use in a `GTPSRangeList` object.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`macRange`

Name of the list of MAC addresses used by this IP address range.

Default value = "None"

`vlanRange`

Name of the list of VLAN tags used by this IP address range.

Default value = "None"

SGSN Configuration

`dtiMode`

When enabled, the DTI bit of the Direct Tunnel Flags IE is set to 1, indicating to the GGSN that the SGSN is invoking a direct tunnel for this PDP Context.

When disabled, the DTI bit of the Direct Tunnel Flags IE is set to 0 (or the Direct Tunnel Flags IE is absent), indicating to the GGSN that the SGSN is not invoking a direct tunnel for this PDP Context. In this case, all other fields of the Direct Tunnel Flags IE are ignored.

The default is false.

`rAIMNC2`

Mobile network code digit 2.

The default is 1.

`userPlanesStartIp`

The first IP Address for this range of user planes.

Default value = "192.168.0.100"

`cAMEL`

The value to be placed in the CAMEL Charging Information Container IE (reference: 3GPP TS 29.060, 7.7.54).

The CAMEL (Customised Application for Mobile network Enhanced Logic) Charging container may be used for session and cost control and for other operator-specific services.

Default value = ""

`ggsnIp`

The IP address of the GGSN (the DUT). This is the GGSN interface to which this emulated SGSN will connect.

Default value = "192.168.1.0"

`userPlanesPrefix`

The netmask for this range of user plane addresses. This parameter is enabled only if the *Use "Control Plane" IP for "User Plane"* parameter is not selected.

The default value is 16.

`mCC`

Mobile Country Code. The MCC digits to be placed in the Geographic Location field in the User Location Information IE (reference: 3GPP TS 29.060, 7.7.51),

The Mobile Country Code (MCC) consists of three digits. It uniquely identifies the country of domicile of the mobile subscriber.



Default value = "226"

lAC

Location Area Code. The value of the LAC to be placed in the Geographic Location field in the User Location Information IE (reference 3GPP TS 29.060, 7.7.51).

The Location Area Code (LAC) is a two-octet hexadecimal code identifying a location area within a GSM PLMN.

Default value = "FFFE"

daylight

The value of the Daylight Saving Time field in the MS Time Zone IE (reference: 3GPP TS 29.060, 7.7.52).

The Daylight Saving Time adjustment. The choices are:

0 = No adjustment for daylight saving time.

1 = Add a one hour adjustment.

2 = Add a two hour adjustment.

3 = Reserved.

Default value = "1"

rncPrefix

The netmask for this range of RNC user plane addresses.

The default value is 16.

echoInterval

The Echo request sending period (in seconds).

Default value = "60"

timezone

The value of the Time Zone field in the MS Time Zone IE (reference: 3GPP TS 29.060, 7.7.52).

The MS Time Zone IE is used to indicate the offset between universal time and local time in steps of 15 minutes. This field uses the same format as the Time Zone IE in 3GPP TS 24.008.

The default is 1, the minimum is 0, and the maximum is 255.

Default value = "1"

rATType

The value to be placed in the RAT Type IE (reference: 3GPP TS 29.060, 7.7.50).

The RAT type information element indicates which Radio Access Technology is currently serving the UE. The choices are:

- UTRAN
- GERAN
- WLAN

The default is UTRAN.

Default value = "1"

rAIMNC3

Mobile network code digit 3.

The default is 1.

controlPlanesPrefix

The netmask for this range of control plane addresses.

The default value is 16.

Default value = "1"

cI

Cell ID. The value of the CI to be placed in the Geographic Location field in the User Location Information IE (reference 3GPP TS 29.060, 7.7.51).

The BSS and cell within the BSS is identified within a location or routing area by adding a Cell Identity (CI) to the location or routing area identification. The CI is a two-octet hexadecimal value.

Default value = "'1111'"

mNC

Mobile Network Code. The MNC digits to be placed in the Geographic Location field in the User Location Information IE (reference: 3GPP TS 29.060, 7.7.51),

The Mobile Network Code (MNC) consists of two or three digits for GSM applications. The MNC identifies the home GSM PLMN of the mobile subscriber.

Default value = "'10F'"



rncUserPlaneIp

The Radio Network Controller (RNC) user plane IP address.

When Direct Tunnel mode is enabled, the SGSN initiates the Update PDP Context procedure immediately following completion of the Create PDP Context procedure. The Update PDP Context procedure indicates (via the Direct Tunnel Flags IE) that the tunnel will be changed from the SGSN to the RNC. In the message, the RNC IP address and port number are communicated to the GGSN. The GTP-u tunnel is then changed such that the communication is directly between the RNC and the GGSN.

The default value is 192.168.0.200.

rAIRAC

The default is "56".

controlPlanesStartIp

The first IP Address for this range of control planes.

Default value = "'192.168.0.2'"

rAILAC

The default is "1234".

userPlanesPerPort

The number of IP addresses to use for the user plane.

Default value = "1"

rAC

Routing Area Code. The RAC used by the EUs. The Routing Area Code (RAC) is a one-octet code that is unique within a Location Area.

This is a hexadecimal value.

Default value = "FF"

controlPlanesGW

The gateway that this range of control plane addresses will use to access the GGSN. This parameter is disabled if the SGSN is in the same subnet as the GGSN.

The default value is 0.0.0.0.

sAC

Service Area Code. The value of the SAC to be placed in the Geographic Location field in the User Location Information IE (reference: 3GPP TS 29.060, 7.7.51).

The Service Area Code (SAC), together with the PLMN-ID and the LAC, constitute the Service Area Identifier (SAI).

Default value = "1011"

echoRequest

Enabling this parameter specifies that echo request messages are required. An Echo Request is sent on a path to another GSN to find out if the peer GSN is alive.

Default value = "True"

rAIMNC1

Mobile network code digit 1.

The default is 1.

ggsnCount

Indicates the number of GGSN IP addresses in use for the test. When the GGSN IP Count is greater than 1, the GGSN IP addresses will be assigned for the emulated SGSNs in a round-robin fashion. For example, if:

You are using three Ixia ports to emulate the SGSNs, one SGSN per port, with IP addresses 192.168.1.2, 192.168.1.3, and 192.168.1.4.

And the GGSN IP address is 192.168.1.100, and the GGSN IP Count parameter is set to 2.

Then IxLoad associates the SGSN and GGSN addresses as follows:

192.168.1.2 connects to 192.168.1.100

192.168.1.3 connects to 192.168.1.101

192.168.1.4 connects to 192.168.1.100

In contrast, if the GGSN IP Count parameter is set to 1, then all three emulated SGSNs will connect that that single GGSN interface.

The default value is 1, the minimum is 1, and the maximum is 1000.

rncGW



The gateway that this range of RNC user plane addresses will use to access the GGSN. This parameter is disabled if the SGSN is in the same subnet as the GGSN.

The default value is 0.0.0.0.

`aPNRestriction`

The value to be placed in the APN Restriction IE (reference: 3GPP TS 29.060, 7.7.49).

An APN Restriction value may be configured for each APN in the GGSN. It is used to determine, on a per MS basis, whether the MS is allowed to establish PDP Contexts to other APNs.

The APN Restriction information element, when used in messages from the GGSN to the SGSN, contains an unsigned integer that indicates the level of restriction imposed on primary PDP Contexts created to the associated APN. When used in messages from the SGSN to GGSN, it contains an unsigned integer that indicates the highest level of restriction type for all currently active PDP Contexts associated with the subscriber.

In both cases, the meaning of the value contained within the IE is as defined in 3GPP TS 23.060 [4], sub-clause 15.4.

The choices are:

- Disabled
- No restrictions
- Public 1
- Public 2
- Private 1
- Private 2

The default value is Disabled.

Default value = "-1"

`controlPlanesPerPort`

The number of IP addresses to use for control plane traffic.

Default value = "1"

`userPlanesGW`

The gateway that this range of user plane addresses will use to access the GGSN. This parameter is enabled only if the *Use "Control Plane" IP for "User Plane"* parameter is not selected. In addition, it is disabled if the SGSN is in the same subnet as the GGSN.

The default value is 0.0.0.0.

`locationType`

The value of the Geographic Location Type field in the User Location Information IE (see 3GPP TS 29.060, 7.7.51).

The choices are:

- Cell ID
- Service Area Code

The default is Cell ID.

Default value = "0"

enabled

If true, this SGSN configuration will be active.

rAIMCC2

Mobile country code digit 2. The default is 1.

rAIMCC3

Mobile country code digit 3. The default is 1.

rAIMCC1

Mobile country code digit 1. The default is 1.

useControlPlaneIpForUserPlane

Set this option to `true` if you require a single IP address for the control plane and the user plane. When selected, IxLoad uses the Control Planes Start IP address for both the control plane and the user plane.

Default value = "False"

EXAMPLE

```
$GTP_1 sgsnRangeList.clear

set SGSN_R1 [::IxLoad new ixNetGTPSRange]
# ixNet objects needs to be added in the list before they are configured!
$GTP_1 sgsnRangeList.appendItem -object $SGSN_R1
$GTP_1 sgsnRangeList.clear
```



```
set SGSN_R1 [::IxLoad new ixNetGTPSRange]

# ixNet objects needs to be added in the list before they are configured!

$GTP_1 sgsnRangeList.appendItem -object $SGSN_R1

$SGSN_R1 config \

    -dtiMode                false \
    -rAIMNC2                 1 \
    -userPlanesStartIp      "192.168.0.100" \
    -cAMEL                   "" \
    -ggsnIp                  "192.168.1.0" \
    -userPlanesPrefix       16 \
    -mCC                      226 \
    -lAC                      "FFFE" \
    -daylight                1 \
    -rncPrefix               16 \
    -echoInterval            60 \
    -timezone                 1 \
    -rATType                  "1" \
    -rAIMNC3                  1 \
    -controlPlanesPrefix     16 \
    -cI                       "1111" \
    -mNC                       "10F" \
    -rncUserPlaneIp          "192.168.0.200" \
    -rAIRAC                    "56" \
    -controlPlanesStartIp    "192.168.0.2" \
    -rAILAC                    "1234" \
    -userPlanesPerPort        1 \
    -rAC                       "FF" \
```

```
-controlPlanesGW          "0.0.0.0" \  
-sAC                      "1011" \  
-echoRequest             true \  
-rAIMNC1                 1 \  
-ggsnCount               1 \  
-rncGW                   "0.0.0.0" \  
-aPNRestriction          -1 \  
-controlPlanesPerPort    1 \  
-userPlanesGW            "0.0.0.0" \  
-locationType            0 \  
-enabled                 true \  
-rAIMCC2                 1 \  
-rAIMCC3                 1 \  
-rAIMCC1                 1 \  
-useControlPlaneIpForUserPlane true  
  
set MAC_R4 [$SGSN_R1 getLowerRelatedRange "MacRange"]  
  
$MAC_R4 config \  
-count                   1 \  
-name                    "MAC-R4" \  
-enabled                 true \  
-mtu                     1500 \  
-mac                     "50:14:C3:99:00:00" \  
-incrementBy             "00:00:00:00:00:01"  
  
set VLAN_R1 [$SGSN_R1 getLowerRelatedRange "VlanIdRange"]  
  
$VLAN_R1 config \  

```



```
-incrementStep          1 \  
-uniqueCount           4094 \  
-name                  "VLAN-R1" \  
-innerIncrement        1 \  
-innerUniqueCount      4094 \  
-enabled               true \  
-innerFirstId          1 \  
-increment             1 \  
-priority              1 \  
-firstId               1 \  
-innerIncrementStep    1 \  
-idIncrMode            2 \  
-innerEnable           false \  
-innerPriority         1
```

```
$GTP_1 ueRangeList.clear
```

SEE ALSO

GTPSUERangeList

SYNOPSIS

DESCRIPTION

List of `GTPSUERange` objects for use by a `GTPSPlugin` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO



GTPSUERange

SYNOPSIS

DESCRIPTION

Configures the GTP protocol User Equipment options for use in a `GTPSUERangeList` object.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`rangeUniqueId`

Unique name of the GTP UE range.

Default value = "UE range 0"

`sgsnId`

The emulated SGSN with which this UE range is associated.

Default value = ""

`count`

The total number of simulated users.

Default value = "10"

`incrementBy`

The value to use for incrementing the IMSI MSIN value for each emulated subscriber.

Default value = "1"

`ipType`

The PDP type used by the UEs in this range. The choices are:

0 = IPv4

1 = IPv6

Default value = "0"

`iMSI_MCC`

IMSI MCC. The first IMSI Mobile Country Code (MCC) in this range. The MCC is a three-digit code that identifies a subscriber's home country.

The minimum value is 200, the default is 226, and the maximum is 999.

Default value = "226"

`iMSI_MNC`

IMSI MNC. The first IMSI Mobile Network Code (MNC) in this range. The MNC is a two- or three-digit code that identifies the home PLMN.

The Mobile Network Code is either two digits (European standard) or three digits (North American standard).

The minimum value is 1, the default is 01, and the maximum is 999.

Default value = "1"

`iMSI_MSIN`

IMSI MSIN. The first IMSI Mobile Subscriber Identification Number (MSIN) in this range. The MSIN is a unique 10-digit number that a wireless operator uses to identify the mobile phone.

The minimum value is 0, the default is 1, and the maximum is 9,999,999,999.

Default value = "1"

`aPNIncrement`

This parameter allows you to use a unique APN for each emulated UE in the range. For example:



- You assign `www.local1.com` as the *Access Point Name* (as described in GTP Traffic Activity Parameter Definitions).
- You enable the *Increment APN* parameter.
- You create three emulated UEs.

In this case, IxLoad will assign the following APNs to the UEs: `www.local1-1.com`, `www.local1-2.com`, `www.local1-3.com`.

API Default = false

iMEI

IMEI. The first IMEI value in this range. An IMEI is constructed of three parts:

- An eight-digit Type Allocation Code (TAC)
- A six-digit Serial Number (SRN)
- A one-digit Spare code.

The IxLoad GTP plug-in creates a unique IMEI value for each IMSI value that it creates, based on the *Increment By* and *Count* values (the *Increment By* and *Count* parameters are on the Basic tab).

The minimum is 10,000,000,000,000 and the maximum is 99,999,999,999,999. the default value is 99,990,000,000,001.

Note: Although the IMEI must be 15 characters in length, you must enter only the first 14 characters. The last number(15th character) of the IMEI is a check digit calculated using the Luhn algorithm.

Default value = "99990000000001"

sV

The software version (SV) number that will be appended to the IMEI to generate the IMEISV value.

The SV is a two-digit number. All values from 00 through 99 are valid. The default value is 11.

mSISDN

The first Mobile Station ISDN (MSISDN) value in this range. The MSISDN is the number that a subscriber dials to call another subscriber.

An MSISDN number is constructed of three parts:

- Country Code (CC) of the country in which the EU is registered.
- National Destination Code (NDC) of the network.
- Subscriber Number.

Start value for MSISDN (increment By and Count To will be the same as for IMSI).

The minimum is 10,000 and the maximum is 999,999,999,999,999. The default value is 40,700,000,000.

Default value = "40700000001"

aPNIncrement

This parameter allows you to use a unique APN for each emulated UE in the range. For example:

- You assign `web.ixiacom.com` as the *Access Point Name*.
- You enable the *Increment APN* parameter.
- You create three emulated UEs.

In this case, IxLoad will assign the following APNs to the UEs: `web.ixiacom.com`, `web.ixiacom.com1`, `web.ixiacom.com2`.

Default value = "False"

authType

PCU protocol.

Value	Description
0	(default) None
1	PAP
2	CHAP

authUIId

The user name for PAP or CHAP authentication.

Every emulated mobile subscriber in the range uses the same User ID.

API Default = ""(none)

authPwd

The password for PAP or CHAP authentication.

Every emulated mobile subscriber in the range uses the same Password.

API Default = ""(none)

aPNRollover

The maximum APN value. When this value is reached, IxLoad resets the APN value to 1.

Enter a value of zero to disable APN rollover.

The default value is zero.

API Default = 0

Mobility parameters

`enableMobility`

Enable this parameter to perform a mobility test.

Default value = "False"

`mobilePath`

The sequence of SGSNs that the mobile stations will traverse.

Default value = ""

`uePathList`

Name of the list of UE paths.

Default value = "None"

`mobilityInterval`

The total time (in seconds) it will take the mobile station to return to the starting SGSN. The interval between each mobility event will be equal to this interval divided by the number of hops in the path.

Mini = "60", Default = "60"

`maxIntervalVariation`

Set this value to randomize the *Mobility Interval* value.

Default value = "0"

`startDelay`

The number of seconds to wait before starting the UEs mobility operation.

Default value = "0"

`maxDelayVariation`

Set this value to randomize the *Start Delay* value.

Min= "0", Max= "50", Default = "0"

`startFromRandomSGSN`

If true, each UE will start from a random SGSN on the path.

Default = "False"

EXAMPLE

```
$UE_R2 config \  
-count 10 \  
-iMSI_MCC "226" \  
-authType "1" \  
-enableMobility false \  
-name "UE-R2" \  
-mSISDN "40700100001" \  
-startFromRandomSGSN false \  
-enabled true \  
-sV "11" \  
-publishStats false \  
-incrementBy 1 \  
-mobilityInterval 60 \  
-startDelay 0 \  
-authPwd "Pass" \  
-aPNIncrement false \  
-maxDelayVariation 0 \  
-iMEI "99990000100001" \  
-maxIntervalVariation 0 \  
-authUID "this_is_me" \  
-ipType "0" \  
-parentSGSN $SGSN_R2  
  
$UE_R2 config \  
-iMSI_MNC "IxL_String:11"\  
-iMSI_MSIN "IxL_String:1000100001"
```

SEE ALSO



GTPSUEMobilePathEntryList

SYNOPSIS

DESCRIPTION

List of `GTPSUEMobilePathEntry` objects.

Use this object to create a mobility path to emulate users traversing a path among a group of SGSNs.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

GTPSUEMobilePathEntry

SYNOPSIS

DESCRIPTION

Configures a GTP User Entry for use in a `GTPSUEMobilePath` object.

SUBCOMMANDS

OPTIONS

`nextSGSN`

Name of the next SGSN to be traversed.

Default value = ""

EXAMPLE

SEE ALSO



GTPGPlugin

SYNOPSIS

DESCRIPTION

Configures a GTP GGSN plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`ggsnRangeList`

List of emulated GGSNs. This must be a `GTPGRange` object.

Default value = "None"

`ixGTPVersion`

GTP version.

Default value = "3.20.1.51"

EXAMPLE

SEE ALSO

GTPG Example



GTPGRangeList

SYNOPSIS

DESCRIPTION

List of `GTPGRange` objects for use by a `GTPGPlugin` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

GTPGRange

SYNOPSIS

DESCRIPTION

Range of addresses for a GTP GGSN plugin.

SUBCOMMANDS

OPTIONS

From IpDynamicProviderRange:

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`macRange`

Name of the list of MAC addresses used by this IP address range.

Default value = "None"

`vlanRange`

Name of the list of VLAN tags used by this IP address range.

Default value = "None"

`ipType`

Default value = "0"

`count`



Number of IP addresses to create.

Default value = "1"

`iPPoolStartIPv4`

Starting IP address for IPv4 address ranges.

Default value = "172.16.0.1"

`iPPoolSizeIPv4`

Number of IPv4 addresses to create

Default value = "100"

`iPPoolStartIPv6`

Starting IP address for IPv6 address ranges.

Default value = "0000:0000:0000:0000:0000:0000:0A0A:0A02"

`iPPoolSizeIPv6`

Number of IPv6 addresses to create

Default value = "100"

EXAMPLE

SEE ALSO

GTPGPluginSessionData

SYNOPSIS

DESCRIPTION

Configures the GTP GGSN global parameters, including the PDP context and UMTS QoS settings for each tunnel.

This mapping applies to all network groups and all ranges defined for the test.

SUBCOMMANDS

OPTIONS

`generalOptions_n3BufferSize`

Default value = "8192"

`generalOptions_processPriority`

Default value = "10"

`generalOptions_errorLevel`

Default value = "30"

`qoSToTOSMapping_pHBConversational`

IP Diffserv value associated to UMTS Streaming traffic class.

Default value = "2"

`qoSToTOSMapping_pHBStreaming`

IP Diffserv value associated to UMTS Streaming traffic class.

Default value = "2"

`qoSToTOSMapping_pHBInteractivePlus`

IP Diffserv value associated to UMTS InteractivePlus traffic class.



Default value = "2"

qoSToTOSMapping_pHBBackground

IP Diffserv value associated to UMTS Background traffic class.

Default value = "0"

qoSToTOSMapping_pHBInteractive

IP Diffserv value associated to UMTS Interactive traffic class.

Default value = "3"

qoSToTOSMapping_pHBSignalisation

IP Diffserv value associated to UMTS signalling.

Default value = "2"

qoSToTOSMapping_pHBFAF1

IP TOS field value associated to diffserv AF1.

Default value = "10"

qoSToTOSMapping_pHBFAF2

IP TOS field value associated to diffserv AF2.

Default value = "18"

qoSToTOSMapping_pHBFAF3

IP TOS field value associated to diffserv AF3.

Default value = "26"

qoSToTOSMapping_pHBFAF4

IP TOS field value associated to diffserv AF4.

Default value = "34"

EXAMPLE

SEE ALSO



GTPGPluginPortGroupData

SYNOPSIS

DESCRIPTION

Configures the GTP SGSN network group session parameters.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`iP`

Default value = "192.168.1.0"

`ggsnUserPlaneIPv4`

The first IPv4 Address for this range of user planes for the GGSN.

Default value = "22.22.22.22"

`ggsnUserPlaneIPv6`

The first IPv6 Address for this range of user planes for the GGSN.

Default value = "fec1::1"

`gateway`

The gateway that this GGSN will use.

Default value = "192.168.1.1"

`netMask`

The IP netmask of this emulated GGSN.

Default value = "16"

`pduFilterList`

Name of the `GTPGPDUFilterList` object that will be used by this port group.

Default value = "None"

EXAMPLE

SEE ALSO



GTPGPDUFilterList

SYNOPSIS

DESCRIPTION

List of `GTPGPDUFilter` objects for use by a `GTPGPluginSessionData` object.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

GTPGPDUFilter

SYNOPSIS

DESCRIPTION

Creates a GTP PDU filter `GTPGPDUFilterList` object.

SUBCOMMANDS

OPTIONS

`enabled`

If `true`, this filter will be used.

Default value = "False"

`filter`

Filter definition.

Default value = "TX g3_gtp_v1_pdu
Message_Type=G3_GTP_v1_Message_Type_Delete_PDP_Context_Request replace
3614000d000c26ee000500010101ffff0013fe1499"

EXAMPLE

SEE ALSO



Global Plugins

This section describes the global plugins.

FilterPlugin

SYNOPSIS

DESCRIPTION

Configures a filter to filter traffic on an Ixia port. Filters are applied on all ports in the network group.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`auto`

If `true`, filters will be configured automatically to allow traffic for protocols defined in the current stack.

Default value = "True"

`all`

If `true`, all traffic is allowed through; no traffic is filtered out.

Default value = "False"

`pppoecontrol`

Default value = "False"

`pppoenetwork`

Default value = "False"



isis

If true, the ISIS traffic is allowed to pass through.

Default value = "False"

ip

A list of IP protocol names or numbers to enable. Numbers are expressed in decimal or 0x<hex digits> format. Ranges may be separated by a dash (-). A blank entry signifies no IP protocol filtering.

Default value = "" (null)

tcp

A list of TCP source or destination well-known port names or port numbers to enable. Ranges may be separated by a dash (-). A blank entry signifies no UDP port filtering.

Default value = "" (null)

udp

A list of UDP source or destination well-known port names or port numbers to enable. Ranges may be separated by a dash (-). A blank entry signifies no UDP port filtering.

Default value = "" (null)

mac

A list of MAC type names or numbers to enable. Numbers are expressed in decimal or 0x<hex digits> format. Ranges may be separated by a dash (-). A blank entry signifies no MAC type filtering.

Default value = "" (null)

icmp

A list of ICMP type names or numbers to enable. Numbers are expressed in decimal or 0x<hex digits> format. Ranges may be separated by a dash (-). A blank entry signifies no ICMP type filtering.

Default value = "" (null)

EXAMPLE

```
set Filter [::IxLoad new ixNetFilterPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $Filter

$Filter config \
    -all                false \
    -pppoecontrol      false \
    -isis              false \
    -name              "Filter" \
    -auto              true \
    -udp               "" \
    -tcp               "" \
    -mac               "" \
    -pppoenetwork      false \
    -ip                "" \
    -icmp              ""
```

SEE ALSO



Filter Plugin Example

```
set Filter [::IxLoad new ixNetFilterPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $Filter
```

```
$Filter config \
  -all false \
  -pppoecontrol false \
  -isis false \
  -name "Filter" \
  -auto false \
  -udp "" \
  -tcp "" \
  -mac "IP; ARP; VLAN; IPv6" \
  -pppoenetwork false \
  -ip "ICMP; IGMP; IP" \
  -icmp ""
```

GratArpPlugin

SYNOPSIS

DESCRIPTION

Enables the Grat ARP plugin.

The *Grat ARP* global service allows you to configure a test to broadcast a gratuitous ARP request packet to all connected interfaces before starting the test. In this way, the emulated network nodes advertise their own addresses, ensuring that the DUT has valid ARP cache entries.

In a gratuitous ARP packet, the ARP Sender Protocol Address and ARP Target Protocol Address are both set to the IP address of the source host, and the ARP Sender Hardware Address is set to the link-layer address of the source host.

SUBCOMMANDS

OPTIONS

name

Name of the instance of the plugin.

Default value = "None"

enabled

If `true`, the Grat ARP service is enabled.

Default value = "True"

EXAMPLE

```
set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
    -enabled           true \
    -name              "GratARP"
```

SEE ALSO



GratArp Plugin Example

```
set GratARP [::IxLoad new ixNetGratArpPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $GratARP

$GratARP config \
  -enabled          true \
  -name             "GratARP"
```



DnsPlugin

SYNOPSIS

DESCRIPTION

Configures the DNS global servers.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`domain`

The DNS domain for the host.

Default value = "" (null)

`timeout`

The amount of time an entry should remain in cache memory before being flushed.

Default value = "5"

`nameServerList`

Name of the list of DNS servers to be used.

Default value = "None"

`searchList`

Name of the list of DNS servers to be searched.

Default value = "None"

hostList

Name of the list of DNS hosts to be used.

Default value = "None"

EXAMPLE

```
set DNS [::IxLoad new ixNetDnsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $DNS

$DNS config \
  -domain           " " \
  -name             "DNS" \
  -timeout          5

$DNS hostList.clear

$DNS searchList.clear

$DNS nameServerList.clear
```

SEE ALSO



DnsNameServerList

SYNOPSIS

DESCRIPTION

A list of IP addresses for the name servers available to the Ixia ports. These will be queried in turn for any symbolic host reference. The list contains items that are of `DnsNameServer` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

DnsNameServer

SYNOPSIS

DESCRIPTION

Identifies the IP address of a DNS name server available to the Ixia ports for use in a `DnsNameServerList`.

SUBCOMMANDS

OPTIONS

`nameServer`

IP address of the DNS name server.

Default value="127.0.0.1"

EXAMPLE

SEE ALSO



DnsSearchList

SYNOPSIS

DESCRIPTION

A list of suffixes to be appended to any symbolic reference in an attempt to resolve the reference. The list contains items that are of `DnsSearch` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

DnsSearch

SYNOPSIS

DESCRIPTION

Identifies a suffix to be used in a `DnsSearchList`.

SUBCOMMANDS

OPTIONS

`search`

Suffix to be added to a `DnsSearchList`.

Default value = "Untitled"

EXAMPLE

SEE ALSO



DnsHostList

SYNOPSIS

DESCRIPTION

A list of host names and static IP address. The list contains items that are of `DnsHost` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

DnsHost

SYNOPSIS

DESCRIPTION

Identifies a host name and static IP address for use in a `DnsHostList`.

SUBCOMMANDS

OPTIONS

`hostIP`

Host IP address.

Default value = "127.0.0.1"

`hostName`

The name of the host that corresponds to `hostIP`.

Default value = "localhost"

`alias1`

An alias for `hostName`. Aliases provide for name changes, alternate spellings, shorter hostnames, or generic hostnames (for example, `localhost`).

Default value = "" (null)

`alias2`

A second alias for `hostName`.

Default value = "" (null)

EXAMPLE

SEE ALSO

Dns Plugin Example

```
set DNS [::IxLoad new ixNetDnsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $DNS

$DNS config \
  -domain                "ixiacom.com" \
  -name                  "DNS" \
  -timeout                5

$DNS hostList.clear

set my_ixNetDnsHost [::IxLoad new ixNetDnsHost]
# ixNet objects needs to be added in the list before they are configured!
$DNS hostList.appendItem -object $my_ixNetDnsHost

$my_ixNetDnsHost config \
  -alias2                "localhost-alias2" \
  -hostName              "localhost" \
  -alias1                "localhost-alias1" \
  -hostIP                "127.0.0.1"

$DNS searchList.clear

set my_ixNetDnsSearch [::IxLoad new ixNetDnsSearch]
# ixNet objects needs to be added in the list before they are configured!
$DNS searchList.appendItem -object $my_ixNetDnsSearch

$my_ixNetDnsSearch config \
  -search                ".com"

$DNS nameServerList.clear

set my_ixNetDnsNameServer [::IxLoad new ixNetDnsNameServer]
# ixNet objects needs to be added in the list before they are configured!
$DNS nameServerList.appendItem -object $my_ixNetDnsNameServer

$my_ixNetDnsNameServer config \
  -nameServer            "127.0.0.1"
```



TCPPlugin

SYNOPSIS

DESCRIPTION

Configures the global settings for a TCP plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`tcp_abort_on_overflow`

Reserved.

Default value = "False"

`tcp_adv_win_scale`

Reserved.

Default value = "2"

`tcp_app_win`

Reserved.

Default value = "31"

`tcp_bic`

Reserved.

Default value = "0"

tcp_bic_fast_convergence

Reserved.

Default value = "1"

tcp_bic_low_window

Default value = "14"

tcp_dsack

Reserved.

Default value = "True"

tcp_ecn

If true, Explicit Congestion Notification is enabled.

Default value = "False"

tcp_fack

Reserved.

Default value = "True"

tcp_fin_timeout

FIN Timeout. The number of seconds the client or server waits to receive a final FIN before closing a socket. A FIN Timeout is usually used to prevent denial-of-service attacks.

Default value = "60"

tcp_frto

Fragment Reassembly Timer. ReThe number of seconds the TCP should keep IP fragments before discarding them.

Default value = "0"

tcp_keepalive_intvl

The number of seconds between repeated keep-alive probes.

Default value = "75"



`tcp_keepalive_probes`

Number of keep-alive probes sent out before determining that a link is down.

Default value = "9"

`tcp_keepalive_time`

If a link has no activity on it for the time specified, keep-alive probes are sent to determine if the link is still up. The Keep-alive Time value is expressed in seconds.

Default value = "7200"

`tcp_low_latency`

Reserved.

Default value = "0"

`tcp_max_orphans`

Reserved.

Default value = "8192"

`tcp_max_syn_backlog`

Reserved.

Default value = "1024"

`tcp_max_tw_buckets`

Reserved.

Default value = "180000"

`tcp_mem_low`

Reserved.

Default value = "24576"

`tcp_mem_pressure`

Reserved.

Default value = "32768"

tcp_mem_high

Reserved.

Default value = "49152"

tcp_moderate_rcvbuf

Reserved.

Default value = "0"

tcp_no_metrics_save

Reserved.

Default value = "False"

tcp_orphan_retries

Reserved.

Default value = "0"

tcp_reordering

Reserved.

Default value = "3"

tcp_retrans_collapse

Default value = "True"

tcp_retries1

Retransmit Retries 1. The number of times TCP will attempt to retransmit a segment on an established connection. If the number of retransmit attempts exceeds this value, TCP requests that the network layer update the route. The default is the RFC 1122 specified minimum of 3 retransmissions.

Default value = "3"

tcp_retries2

Retransmit Retries 2. If the number of retransmissions of the same segment reaches this threshold, TCP closes the connection. The default value is 15, which corresponds to a duration of approximately between 13 to 30 minutes, depending on the retransmission timeout.

Default value = "15"



tcp_rfc1337

Reserved.

Default value = "False"

tcp_rmem_min

Reserved.

Default value = "4096"

tcp_rmem_default

Reserved.

Default value = "262144"

tcp_rmem_max

Reserved.

Default value = "262144"

tcp_sack

If true, RFC 2018 TCP Selective Acknowledgements are enabled.

Default value = "True"

tcp_stdurg

Reserved.

Default value = "False"

tcp_synack_retries

Number of times an un-acknowledged SYN-ACK for a passive TCP connection will be re-transmitted.

Default value = "5"

tcp_syn_retries

Number of times an un-acknowledged SYN for an active TCP connection will be re-transmitted.

Default value = "5"

`tcp_timestamps`

If `true`, the client or server inserts a timestamp into each packet.

Note: Enabling the TCP Timestamp option adds 12 bytes to the TCP header. This has the effect of reducing the effective MSS configured.

Default value = "True"

`tcp_tw_recycle`

If `true`, fast recycling of TIME-WAIT sockets is enabled. Enabling this option is not recommended when working with NAT (Network Address Translation).

Default value = "False"

`tcp_tw_reuse`

If `true`, allows the reuse of TIME-WAIT sockets for new connections. Enable this option only if you are certain that it is safe from a protocol viewpoint.

Default value = "False"

`tcp_vegas_alpha`

Reserved.

Default value = "2"

`tcp_vegas_beta`

Reserved.

Default value = "6"

`tcp_vegas_cong_avoid`

Reserved.

Default value = "0"

`tcp_vegas_gamma`

Reserved.

Default value = "2"

`tcp_westwood`



Reserved.

Default value = "0"

`tcp_window_scaling`

If `true`, RFC 1323 TCP window scaling is enabled. The TCP Window Scaling feature allows the use of a large window (greater than 64K) on a TCP connection, if the other end supports it.

Default value = "True"

`ip_no_pmtu_disc`

Reserved.

Default value = "False"

`tcp_wmem_min`

Reserved.

Default value = "4096"

`tcp_wmem_default`

Reserved.

Default value = "262144"

`tcp_wmem_max`

Reserved.

Default value = "262144"

`tcp_ipfrag_time`

Fragment Reassembly Timer. The number of seconds the TCP should keep IP fragments before discarding them.

Default value = "30"

`tcp_port_min`

Minimum TCP source port value. The source port specifies which ports to use for client connections. The Min value specifies the lower bound (the lowest permissible port number).

Default value = "1024"

tcp_port_max

Maximum TCP source port value. The Maximum source port value specifies the upper bound (the highest permissible port number).

Default value = "65535"

tcp_rto_min

Minimum Retransmission Timeout value.

Default value = "200"

tcp_rto_max

Maximum Retransmission Timeout value.

Default value = "120000"

EXAMPLE

```
set TCP [::IxLoad new ixNetTCPPlugin]

# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $TCP

$TCP config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -tcp_retries1 3 \
    -tcp_keepalive_time 75 \
    -tcp_moderate_rcvbuf 0 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
```



```
-tcp_rto_min          1000 \  
-tcp_reordering      3 \  
-tcp_vegas_cong_avoid 0 \  
-tcp_keepalive_intvl 7200 \  
-tcp_rmem_max        262144 \  
-tcp_orphan_retries  0 \  
-tcp_max_tw_buckets  180000 \  
-tcp_wmem_default    4096 \  
-tcp_low_latency     0 \  
-tcp_rmem_min        4096 \  
-tcp_adv_win_scale   2 \  
-tcp_wmem_min        4096 \  
-tcp_port_min        1024 \  
-tcp_stdurg          false \  
-tcp_port_max        65535 \  
-tcp_fin_timeout     60 \  
-tcp_no_metrics_save false \  
-tcp_dsack           true \  
-tcp_mem_high        49152 \  
-tcp_frto            0 \  
-tcp_app_win         31 \  
-ip_no_pmtu_disc     false \  
-tcp_window_scaling  false \  
-tcp_max_orphans     8192 \  
-tcp_mem_pressure    32768 \  
-tcp_syn_retries     5 \  
-name                "TCP" \  
-tcp_max_syn_backlog 1024 \  

```

```
-tcp_mem_low          24576 \  
-tcp_fack             true \  
-tcp_retrans_collapse true \  
-tcp_rmem_default     4096 \  
-tcp_keepalive_probes 9 \  
-tcp_abort_on_overflow false \  
-tcp_tw_reuse         false \  
-tcp_wmem_max         262144 \  
-tcp_vegas_gamma      2 \  
-tcp_synack_retries   5 \  
-tcp_timestamps      true \  
-tcp_vegas_beta       6 \  
-tcp_sack             true \  
-tcp_bic_fast_convergence 1 \  
-tcp_bic_low_window   14
```

SEE ALSO



TCP Plugin Example

```
set TCP [::IxLoad new ixNetTCPPlugin]  
# ixNet objects needs to be added in the list before they are configured!  
$Network1 globalPlugins.appendItem -object $TCP
```

```
$TCP config \  
-tcp_bic 0 \  
-tcp_tw_recycle true \  
-tcp_retries2 5 \  
-tcp_retries1 3 \  
-tcp_keepalive_time 75 \  
-tcp_moderate_rcvbuf 0 \  
-tcp_rfc1337 false \  
-tcp_ipfrag_time 30 \  
-tcp_rto_max 60000 \  
-tcp_vegas_alpha 2 \  
-tcp_ecn false \  
-tcp_westwood 0 \  
-tcp_rto_min 1000 \  
-tcp_reordering 3 \  
-tcp_vegas_cong_avoid 0 \  
-tcp_keepalive_intvl 7200 \  
-tcp_rmem_max 262144 \  
-tcp_orphan_retries 0 \  
-tcp_max_tw_buckets 180000 \  
-tcp_wmem_default 4096 \  
-tcp_low_latency 0 \  
-tcp_rmem_min 4096 \  
-tcp_adv_win_scale 2 \  
-tcp_wmem_min 4096 \  
-tcp_port_min 1024 \  
-tcp_stdurg false \  
-tcp_port_max 65535 \  
-tcp_fin_timeout 60 \  
-tcp_no_metrics_save false \  
-tcp_dsack true \  
-tcp_mem_high 49152 \  
-tcp_frto 0 \  
-tcp_app_win 31 \  
-ip_no_pmtu_disc false \  
-tcp_window_scaling false \  
-tcp_max_orphans 8192 \  
-tcp_mem_pressure 32768 \  
-tcp_syn_retries 5 \  
-name "TCP" \  

```

RoutesPlugin

SYNOPSIS

DESCRIPTION

Configures *Routes* global service, which allows the network group to be associated with a set of IP routes.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`routes`

List of routes. This list must be a `RouteList` object.

Default value = "None"

EXAMPLE

```
set Routes_1 [::IxLoad new ixNetRoutesPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $Routes_1

$Routes_1 config \
    -name                "Routes-1"

$Routes_1 routes.clear
```

SEE ALSO



RouteList

SYNOPSIS

DESCRIPTION

A list of IP routes for use by the `RoutesPlugin`. The list contains items that are of `Route` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

Route

SYNOPSIS

DESCRIPTION

Configures a an IP route for use in a `RouteList`.

SUBCOMMANDS

OPTIONS

`protocol`

The IP protocol to use for this route. See `ProtocolChoices` for the list of valid choices.

Values = "IPv4", "IPv6". Default value = "IPv4"

`targetType`

Indicates the type of route target is to be used. See `TargetTypeChoices` for the list of valid choices

Values = "net", "host". Default value = "net"

`destinationIp`

Depending on the value of `targetType`, either a network address or host address.

Default value = "10.10.0.0"

`gateway`

The gateway address for the route entry.

Default value = "10.10.10.1"

`prefixLength`

If `targetType` is set to `net`, this is the subnet mask associated with the network.

Default value = "24"

EXAMPLE

SEE ALSO

ProtocolChoices

SYNOPSIS

DESCRIPTION

List of valid choices for the `protocol` option of the `Route` object.

SUBCOMMANDS

OPTIONS

Value	Description
IPv4	IPv4 protocol
IPv6	IPv6 protocol

EXAMPLE

SEE ALSO



TargetTypeChoices

SYNOPSIS

DESCRIPTION

List of valid choices for the `targetType` option of the `Route` object.

SUBCOMMANDS

OPTIONS

Value	Description
net	An entire network or subnet.
host	A single host.

EXAMPLE

SEE ALSO

Routes Plugin Example

```
set Routes_1 [::IxLoad new ixNetRoutesPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $Routes_1

$Routes_1 config \
    -name "Routes-1"

$Routes_1 routes.clear
```



IxLoadSettingsPlugin

SYNOPSIS

DESCRIPTION

Configures the IxLoad Global Service (Dynamic Control Plane) settings.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`teardownInterfaceWithUser`

If `true`, the interfaces will come up with the users and will go down when users go down.

This option is enabled only if the `interfaceBehavior` is `true`.

Default value = "False"

`interfaceBehavior`

If `true`, enables dynamic control plane. The interfaces are created on demand but are not destroyed until the test ends.

Default value = "0"

EXAMPLE

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
  -teardownInterfaceWithUser      false \
  -name                          "Settings" \
  -interfaceBehavior              0

set Ethernet_1 [$Network1 getL1Plugin]

set my_ixNetEthernetELMPlugin [::IxLoad new ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
  -negotiationType                "master" \
  -negotiateMasterSlave          true

$Ethernet_1 config \
  -advertise10Full                true \
  -name                          "Ethernet-1" \
  -autoNegotiate                  true \
  -advertise100Half               true \
  -advertise10Half                true \
  -speed                          "k100FD" \
  -advertise1000Full              true \
  -advertise100Full               true \
  -cardElm                       $my_ixNetEthernetELMPlugin

$Ethernet_1 childrenList.clear

$Ethernet_1 extensionList.clear
```

SEE ALSO



IxLoad Settings Plugin Example

```
set Settings [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Network1 globalPlugins.appendItem -object $Settings

$Settings config \
  -teardownInterfaceWithUser      true \
  -name                            "Settings" \
  -interfaceBehavior              1
```

Mobile Subscribers Plugins

This section describes the Mobile Subscriber plugin.

MobileSubscribersPlugin

SYNOPSIS

DESCRIPTION

Creates a network stack element representing 3G mobile subscribers.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`childrenList`

Name of the list of next-lower layer plugins.

Default value = "None"

`extensionList`

Name of the list of protocol extensions.

Default value = "None"

`rangeList`

List of MobileSubscriber ranges. New elements can be added to the using `appendItem`. The elements of the list can be modified, but the list cannot be replaced.

Default value="None".

EXAMPLE

```
set Mobile_Subscribers_1 [::IxLoad new ixNetMobileSubscribersPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Emulated_Router_1 childrenList.appendItem -object $Mobile_Subscribers_1

$Mobile_Subscribers_1 childrenList.clear

$Mobile_Subscribers_1 extensionList.clear
```

SEE ALSO

MobileSubscribersRangeList

SYNOPSIS

DESCRIPTION

List of `MobileSubscribersRange` objects for use by a `MobileSubscribersPlugin` object.

SUBCOMMANDS

OPTIONS

None.

EXAMPLE

SEE ALSO



MobileSubscribersRange

SYNOPSIS

DESCRIPTION

Creates an address range for use in a `GTPSRangeList` object.

SUBCOMMANDS

OPTIONS

`authType`

Authentication type. Values = PAP, CHAP.

API default value = PAP

`calledStationId`

Identifier for the target network (the APN).

The default is *apn.ixiacom.com*.

`chapName`

The user name for CHAP authentication. This is the user name of the peer that will be authenticated. The default value is *user*. The maximum length of the string is 256 characters. Refer to *Configuring PPP Authentication Strings* for detailed information.

API default value = "" (none)

`chapSecret`

The secret for CHAP authentication. Both authenticating devices must use the same secret. The default value is *secret*. The maximum length of the string is 256 characters. Refer to *Configuring PPP Authentication Strings* for detailed information.

API default value="" (none)

`delayClass`

The Delay Class specifies how much delay is acceptable for the traffic. Following are the available values:

- 0 0 0: Subscribed delay class
- 0 0 1: Delay class 1
- 0 1 0: Delay class 2
- 0 1 1: Delay class 3
- 1 0 0: Delay class 4 (best effort)
- 1 1 1: Reserved

The default value is Subscribed.

API values:

Value	Description
0 (default)	Subscribed
1	Delay class 1
2	Delay class 2
3	Delay class 3
4	Delay class 4
7	Reserved

`deliveryOfErroneousSDU`

Specifies whether or not erroneous SDUs are delivered. Following are the available values:

- Subscribed: Subscribed delivery of erroneous SDUs.
- No Detect: No detect ('-')
- Yes: Erroneous SDUs are delivered ('yes')
- No: Erroneous SDUs are not delivered ('no')
- Reserved

The default value is No.

API values:

Value	Description
0	Subscribed
1 (default)	No Detect
2	Yes
3	No
7	Reserved

`deliveryOrder`

Specifies delivery ordering. Following are the available values:

- Subscribed: Subscribed delivery order.
- Yes: With delivery order ('yes')



- No: Without delivery order ('no')
- Reserved

The default value is No.

API values:

Value	Description
0	Subscribed
1	Yes
2 (default)	No
3	Reserved

`enabled`

If `true`, this range is used in the test. If `false`, the range is not used in the test. Default= `true`.

`guaranteedBitRateDL`

Specifies the guaranteed downstream bit rate, in kbps.

The Guaranteed bit rate for downlink value is ignored if either of the following is true:

- Traffic Class is Interactive class or Background class,
- Maximum bit rate for uplink is set to 0 kbps.

The default value is 64 and the maximum value is 256000.

API default value = 64

`guaranteedBitRateUL`

Specifies the guaranteed upstream bit rate, in kbps.

The Guaranteed bit rate for uplink value is ignored if either of the following is true:

- Traffic Class is Interactive class or Background class,
- Maximum bit rate for uplink is set to 0 kbps.

The default value is 64 and the maximum value is 256000.

API default value= 64

`iMEI`

The first IMEI value in this range. An IMEI is constructed of three parts:

- An eight-digit Type Allocation Code (TAC)
- A six-digit Serial Number (SRN)
- A one-digit Spare code.

The default value for the first range is 99990000000001. The IxLoad Mobile Subscribers plug-in creates a unique IMEI value for each range that you add by incrementing the prior IMEI value by the Subscriber Count value. For example, if the IMEI value for range 1 is 99990000000001 and you specify a Subscriber Count of 20, then the IMEI value for range 2 will be 999900000000021. Note that, once created, an IMEI value will not change based on modifications to the Subscriber Count values.

Note: Although the IMEI is 15 characters in length, you must enter only the first 14 characters. The last number (15th character) of the IMEI is a check digit calculated using the Luhn algorithm.

API defaultValue = 99990000000001

imsiMCC

The first IMSI Mobile Country Code (MCC) in this range. The MCC is a three-digit code that identifies a subscriber's home country.

The default is 226.

API values: min = 1, max = 999, default = 226.

imsiMNC

The first IMSI Mobile Network Code (MNC) in this range. The MNC is a two- or three-digit code that identifies the home PLMN.

The Mobile Network Code is either two digits (European standard) or three digits (North American standard).

The default is 11.

API values: min = 10, max = 999, default = 11.

imsiMSIN

The first IMSI Mobile Subscriber Identification Number (MSIN) in this range. The MSIN is a unique 10-digit number that a wireless operator uses to identify the mobile phone. It is appended to MCC and MNC to form the IMSI.

The default value for the first range is 1000000001. The IxLoad Mobile Subscribers plug-in creates a unique IMSI MSIN value for each range that you add by incrementing the prior IMSI MSIN value by the Subscriber Count value. For example, if the IMSI MSIN value for range 1 is 1000000001 and you specify a Subscriber Count of 20, then the IMSI MSIN value for range 2 will be 1000000021. Note that, once created, an IMSI MSIN value will not change based on modifications to the Subscriber Count values.

API values: min = 1000000000, max = 9999999999, default = 1000000001

incrementBy

IP increment value for this range. This parameter is available only when *Static Mode* is enabled.

The default value is 0.0.0.1.



`interimUpdateInterval`

The interval (in minutes) for sending interim accounting updates.

The default is 1, the minimum is 1, and the maximum is 7200.

API values: min = 1, max = 7200, default = 1.

`ipAddress`

The first static IP address for this range of mobile subscribers. This parameter is available only when *Static Mode* is enabled.

The default value is 50.0.0.1.

`ipType`

IP version used on this range.

API values: IPv4, IPv6, Default = "IPv4"

`maxBitRateDL`

Specifies the maximum downstream bit rate, in kbps.

The default value is 8640 and the maximum value is 256000.

API values: min =0, max = 256000, default = 8640.

`maxBitRateUL`

Specifies the maximum upstream bit rate, in kbps.

The default value is 8640 and the maximum value is 256000.

API values: min =0, max = 256000, default = 8640.

`maxSDUSize`

Specifies the maximum Service Data Unit (SDU) size, in octets.

The default value is 151 and the maximum value is 153.

API values: min = 0, max = 151, default = 151.

`meanThroughput`

Mean throughput specifies the required mean throughput, expressed as either the mean number of octets per hour, or the best available throughput. Following are the available values:

- 2M (2,000,000,000 octets per hour).
- 5M (5,000,000,000 octets per hour).
- 10M (10,000,000,000 octets per hour).
- 20M (20,000,000,000 octets per hour).
- 50M (50,000,000,000 octets per hour).
- Reserved.
- Best effort (Throughput is made available to the MS on a per-need and availability basis.)

The default value is Best effort.

API values:

Value	Description
0	Subscribed mean throughput
1	100 octets / hour
2	200 octets / hour
3	500 octets / hour
4	1 000 octets / hour
5	2 000 octets / hour
6	5 000 octets / hour
7	10 000 octets / hour
8	20 000 octets / hour
9	50 000 octets / hour
10	100 000 octets / hour
11	200 000 octets / hour
12	500 000 octets / hour
13	1 000 000 octets / hour
14	2 000 000 octets / hour
15	5 000 000 octets / hour
16	10 000 000 octets / hour
17	20 000 000 octets / hour
18	50 000 000 octets / hour
30	Reserved
31 (default)	Best effort

mSISDN

The first Mobile Station ISDN (MSISDN) value in this range. The MSISDN is the number that a subscriber dials to call another subscriber.

An MSISDN number is constructed of three parts:



- Country Code (CC) of the country in which the UE is registered.
- National Destination Code (NDC) of the network.
- Subscriber Number.

The default value for the first range is 40700000001. The IxLoad Mobile Subscribers plug-in creates a unique MSISDN value for each range that you add by incrementing the prior MSISDN value by the Subscriber Count value. For example, if the MSISDN value for range 1 is 40700000001 and you specify a Subscriber Count of 20, then the MSISDN value for range 2 will be 407000000021. Note that, once created, an MSISDN value will not change based on modifications to the Subscriber Count values.

API default = 40700000001

`papPassword`

The password for PAP authentication. Both authenticating devices must use the same password. The default value is `password`. The maximum length of the string is 256 characters. Refer to Configuring PPP Authentication Strings for detailed information.

API default value = "" (none)

`papUser`

The user name for PAP authentication. This is the user name of the peer that will be authenticated. The default value is `user`. The maximum length of the string is 256 characters. Refer to Configuring PPP Authentication Strings for detailed information.

API default value = "" (none)

`peakThroughput`

Peak Throughput specifies the required throughput during periods of peak traffic activity. Following are the available values:

- Subscribed peak throughput.
- Up to 1,000 octets per second.
- Up to 2,000 octets per second.
- Up to 4,000 octets per second.
- Up to 8,000 octets per second.
- Up to 16,000 octets per second.
- Up to 32,000 octets per second.

The default value is 32k (32,000 octets per second).

API values:

Value	Description
0	Subscribed peak throughput
1	Up to 1 000 octets / second
2	Up to 2 000 octets / second

3	Up to 4 000 octets / second
4	Up to 8 000 octets / second
5	Up to 16 000 octets / second
6 (default)	Up to 32 000 octets / second
7	Up to 64 000 octets / second
8	Up to 128 000 octets / second
9	Up to 256 000 octets / second
15	Reserved

`precedenceClass`

The Precedence Class specifies the priority of the traffic. Following are the available values:

- 0 0 0: Subscribed precedence
- 0 0 1: High priority
- 0 1 0: Normal priority
- 0 1 1: Low priority
- 1 1 1: Reserved

The default value is Normal.

API values:

Value	Description
0	Subscribed precedence
1	High priority
2 (default)	Normal priority
3	Low priority
7	Reserved

`prefix`

The IP address prefix used for creating the range of static IP addresses. This parameter is available only when *Static Mode* is enabled.

The default value is 16.

`publishStats`

Determines whether or not the per-session statistics for this range will be published. Default = `false`.

`release`

The 3GPP network architecture release level that is supported by the Mobile Subscriber plug-in.

The default is R99.



API values:

Value	Description
98	R98
99 (default)	R99
05	R05
07	R07

reliabilityClass

The Reliability Class specifies how reliable the traffic delivery must be. Following are the available values:

- 0 0 0: Subscribed reliability class.
- 0 0 1: Unused. (If received, it is interpreted as 0 1 0.)
- 0 1 0: Unacknowledged GTP; Acknowledged LLC and RLC, Protected data.
- 0 1 1: Unacknowledged GTP and LLC; Acknowledged RLC, Protected data.
- 1 0 0: Unacknowledged GTP, LLC, and RLC, Protected data.
- 1 0 1: Unacknowledged GTP, LLC, and RLC, Unprotected data.
- 1 1 1: Reserved.

The default value is 0 1 0.

API values:

Value	Description
0	Subscribed reliability class
1	Unused.If received, it shall be interpreted as '010'
2 (default)	Unacknowledged GTP; Acknowledged LLC and RLC, Protected data
3	Unacknowledged GTP and LLC; Acknowledged RLC, Protected data
4	Unacknowledged GTP, LLC, and RLC, Protected data
5	Unacknowledged GTP, LLC, and RLC, Unprotected data
7	Reserved

residualBER

Specifies the acceptable ratio of bit errors to SDUs transmitted. Residual Bit Error Rate (BER) indicates the undetected bit error ratio in the delivered SDUs. Following are the available values:

API values:

Value	Description
0	Subscribed residual BER
1	5/100

2	1/100
3	5/1,000
4	4/1,000
5	1/1,000
6	1/10,000
7 (default)	1/100,000
8	1/1,000,000
9	6/100,000,000
15	Reserved

`sduErrorRatio`

Specifies the acceptable ratio of SDU errors to SDUs transmitted. The SDU error ratio indicates the fraction of SDUs lost or detected as erroneous. Following are the available values:

- Subscribed SDU error ratio
- 1/100
- 7/1,000
- 1/1,000
- 1/10,000
- 1/100,000
- 1/1,000,000
- 1/10
- Reserved.

The default value is 1/1,000,000.

API values:

Value	Description
0	Subscribed SDU error ratio
1	1/100
2	7/1,000
3	1/1,000
4	1/10,000
5	1/100,000
6 (default)	1/1,000,000
7	1/10
15	Reserved

`signalingIndication`



Indicates whether or not the QoS of the PDP context is optimized for traffic signalling. The available values are:

- Not optimized for signalling traffic
- Optimized for signalling traffic

The Signalling Indication value is ignored if the Traffic Class is one of the following:

- Conversational class
- Streaming class
- Background class

The default value is Not Optimized.

API values:

Value	Description
0 (default)	Not optimized for signaling
1	Optimized for signaling

`sourceStatisticsDescriptor`

Specifies characteristics of the source of submitted SDUs. Following are the available values:

- Unknown.
- Speech.

The Source Statistics Descriptor parameter is available only if the *Traffic Class* is either Subscribed, Conversational, or Streaming.

The default value is Unknown.

API values:

Value	Description
0 (default)	Unknown
1	Speech

`staticMode`

When enabled, the plug-in uses static mode IP addressing. In this case, the values provided in the *IP Type*, *IP Address*, *IP Increment*, and *Mask* parameters are used to create the IP addresses for the mobile subscribers. In addition, when this option is enabled:

- Subscriber authentication is disabled, and
- RADIUS accounting must also be enabled.

When static mode disabled, the plug-in uses dynamic IP addressing. In this case, the RADIUS client requests allocation of IP addresses to the subscribers at the beginning of the user sessions. The AAA server allocates the IP address when the subscribers are successfully authenticated for network access.

This parameter is disabled by default.

subscriberCount

The total number of emulated subscribers.

API values: min = 1, max = 32000, default = 1.

subscriberIncrement

The value to use for incrementing the IMSI MSIN value for each emulated subscriber.

The default value is 1.

API values: min = 1, default = 1

trafficClass

The UMTS traffic class to use for this PDP context. Following are the available values:

- Subscribed traffic class.
- Conversational (such as voice and video conferencing).
- Steaming (used for video streaming).
- Interactive (such as voice and online gaming).
- Background (such as email and file transfers).
- Reserved.

The default value is Background.

API values:

Value	Description
0	Subscribed traffic class
1	Conversational class
2	Streaming class
3	Interactive class
4(default)	Background class
7	Reserved

trafficHandlingPriority

Specifies the desired traffic handling priority. Following are the available values:

- Subscribed traffic handling priority.
- Priority Level 1.
- Priority Level 2.
- Priority Level 3.

This parameter is available only when the Traffic Class is Interactive.

The Traffic handling priority value is ignored if the Traffic Class is one of the following:



- Conversational class
- Streaming class
- Background class

The default value is Level 1.

API values:

Value	Description
0	Subscribed traffic handling priority
1 (default)	Priority level 1
2	Priority level 2
3	Priority level 3

`transferDelay`

Specifies the maximum acceptable transfer delay, in milliseconds. The valid range of values is from 10 to 4000 ms.

The Transfer Delay value is ignored if Traffic Class is one of the following:

- Interactive class
- Background class.

API values: min = 0, max = 4000, default = 4000.

EXAMPLE

```
set Subscriber_R1 [::IxLoad new ixNetMobileSubscribersRange]
# ixNet objects needs to be added in the list before they are configured!
$Mobile_Subscribers_1 rangeList.appendItem -object $Subscriber_R1

$Subscriber_R1 config \
    -reliabilityClass          2 \
    -interimUpdateInterval    1 \
    -delayClass                0 \
    -sduErrorRatio            6 \
    -publishStats              false \
    -prefix                    16 \
    -staticMode                true \
    -guaranteedBitRateUL      64 \
    -authType                  "PAP" \
    -maxSDUSize                151 \
    -guaranteedBitRateDL      64 \
    -incrementBy              "0.0.0.1" \
    -signalingIndication       0 \
    -imsiMSIN                  1000000001 \
    -iMEI                      "999900000000001" \
    -imsiMCC                   226 \
    -trafficHandlingPriority   1 \
    -deliveryOrder             2 \
    -calledStationId           "apn.ixiacom.com" \
    -papPassword               "password" \
    -chapName                  "user" \
    -peakThroughput            6 \
    -subscriberIncrement       1 \
    -precedenceClass           2 \
    -maxBitRateDL              8640 \
    -papUser                   "user" \
    -subscriberCount           1 \
    -ipType                    "IPv4" \
    -meanThroughput            31 \
    -mSISDN                    "40700000001" \
    -chapSecret                "secret" \
    -maxBitRateUL              8640 \
    -transferDelay             4000 \
    -enabled                   true \
    -trafficClass              4 \
    -imsiMNC                   11 \
    -release                   "99" \
    -deliveryOfErroneousSDU    1 \
    -sourceStatisticsDescriptor 0 \
    -ipAddress                 "50.0.0.1" \
    -residualBER               7
```

SEE ALSO



MobileSubscribersSessionData

SYNOPSIS

DESCRIPTION

Configures the Mobile Subscribers global parameters.

SUBCOMMANDS

OPTIONS

`maxOutstanding`

The number of Mobile Subscriber control connections that can be in the initiation state or the terminating state at any time. This setting must be equal to or greater than the number of assigned ports.

API values: Min = 1, max = 255, default = 20.

`setupRate`

The number of Mobile Subscriber control connections to initiate each second. This setting must be equal to or greater than the number of assigned ports.

API values: Min = 1, max = 255, default = 5.

`teardownRate`

The number of Mobile Subscriber control connections to tear down each second. This setting must be equal to or greater than the number of assigned ports.

API values: Min = 1, max = 255, default = 5.

EXAMPLE

SEE ALSO



MobileSubscribersPortGroupData

SYNOPSIS

DESCRIPTION

Configures the Mobile Subscribers network group session parameters.

SUBCOMMANDS

OPTIONS

`activityID`

Activity ID.

Default value = "0"

`activities`

List of activities.

Default Value = "None"

`associates`

Name of the list of associates. This list must an `AssociateList` object.

This option is read only.

Default value = "None"

`ggsnMccMnc`

The IMSI Mobile Country Code (MCC) and Mobile Network Code (MNC) associated with the network to which the GGSN belongs.

The default value is 22611.

(Note that you also define MCC and MNC values at the range level. In this case, the values are associated with the subscribers in that range.)

API default values: 22611

`globalTimeout`



The maximum duration for establishment of a control-session, a test-session, and the start-sessions command.

The default is 240, the minimum is 0, and the maximum is 65535.

If the value is set to zero, there is no timeout.

API values: min = 0, max = 65535, default = 240.

`maxOutstanding`

The number of Mobile Subscriber control connections that can be in the initiation state or the terminating state at any time. This setting must be equal to or greater than the number of assigned ports.

API values: min = 1, max = 255, default = 20.

`overrideGlobalRateOptions`

If selected, all the rate settings defined at the global level (the Network Plug-in Settings) will be overridden by the rate settings defined on this port.

For example, if you have set the Setup Rate to 10 on the global level, and you have defined two ports, these 10 session setups will be evenly distributed across the ports (5 for each). If you then enable Override Global Setup Rate, you can modify the number of session setups for each of the ports (such as changing the distribution such that 7 sessions will be set up on one port, while 3 session will be set up on the other port).

API values: true, false (default).

`setupRate`

The number of Mobile Subscriber control connections to initiate each second. This setting must be equal to or greater than the number of assigned ports.

API values: min = 1, max = 255, default = 5.

`sV`

The software version (SV) number that will be appended to the IMEI to generate the IMEISV value.

The SV is a two-digit number. All values from 00 through 99 are valid. The default value is 11.

API values: min = 2 (digits), max = 2 (digits), default = 11.

`teardownRate`

The number of Mobile Subscriber control connections to tear down each second. This setting must be equal to or greater than the number of assigned ports.

API values: min = 1, max = 5, default = 5.

EXAMPLE

SEE ALSO



Radius Plugin

SYNOPSIS

DESCRIPTION

Defines a Radius plugin. A Radius plugin is an extension to a Mobile Subscribers plugin.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`rangeList`

List or Radius ranges.

Default value = "None"

EXAMPLE

```
set RADIUS_1 [::IxLoad new ixNetRadiusPlugin]
# ixNet objects needs to be added in the list before they are configured!
$Mobile_Subscribers_1 extensionList.appendItem -object $RADIUS_1

$Emulated_Router_1 extensionList.clear

$MAC_VLAN_2 extensionList.clear

$Ethernet_1 extensionList.clear

$MAC_VLAN_11 extensionList.clear

$Ethernet_1 extensionList.clear

#####
# Setting the ranges starting with the plugin on top of the stack
#####
$Mobile_Subscribers_1 rangeList.clear
```

SEE ALSO



RadiusRangeList

SYNOPSIS

DESCRIPTION

List of Radius ranges. The list contains items that are of `RadiusRange` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

RadiusRange

SYNOPSIS

DESCRIPTION

Configures a range of Radius addresses.

SUBCOMMANDS

OPTIONS

`name`

Name of the instance of the plugin.

Default value = "None"

`enabled`

If `True`, the range base is enabled.

Default value="True".

`accountingPort`

The Accounting UDP port.

The default is port 1813 (the officially assigned port number for RADIUS accounting).

API values: Min = 1, max = 65535, default = 1813.

`accountingServer`

The IP address of the RADIUS Accounting Server.

The default is 1.1.1.1.

API values: Default = 1.1.1.1

`authenticationPort`

The Authentication UDP port.

The default port is 1812 (the officially assigned port number for RADIUS authentication).



API values: Min = 1, max = 65535, default = 1812.

`authenticationServer`

The IP address of the RADIUS Authentication Server.

The default is 1.1.1.1.

API values: Default = 1.1.1.1

`enableAccounting`

Enables support for RADIUS accounting.

If you enable the Mobile Subscriber *Static Mode* parameter, IxLoad automatically enables the *Enable Accounting* parameter (because, in this case, the Accounting-Request exchange is mandatory.)

If you disable the Mobile Subscriber *Static Mode* parameter, you can enable or disable RADIUS accounting as required by your test.

API values: Default = true

`retries`

The RADIUS retry value. If a message response times out, the GGSN will resend the message the number of times specified by this parameter. (This applies to all the messages sent by the client: Access-Request, Accounting-Request Start, Accounting-Request Interim-Update, and Accounting-Request Stop.)

The default is 3, the minimum is 1, and the maximum is 10.

API values: Min = 1, max = 10, default = 3.

`secret`

The shared secret used by Ixia RADIUS clients and RADIUS servers.

The default value is the text string *secret*.

API values: Default = "secret"

`timeout`

The RADIUS timeout value. This is the amount of time that the GGSN waits for a reply to the message that it sent to the RADIUS server. (This applies to all the messages sent by the client: Access-Request, Accounting-Request Start, Accounting-Request Interim-Update, and Accounting-Request Stop.)

The Timeout value doubles for each retry.

The default is 10, the minimum is 1, and the maximum is 255.

API values = Min = 1, max = 255, default = 10.

`tunnelAttributeSet`

Name of the set of RADIUS tunnel attributes. The tunnel attributes are a set of RadiusOptionSet objects.

API values: Default = "None".

EXAMPLE

```
$RADIUS_R1 config \  
-retries 3 \  
-accountingServer "1.1.1.1" \  
-enabled true \  
-secret "secret" \  
-accountingPort 1813 \  
-enableAccounting true \  
-timeout 10 \  
-authenticationPort 1812 \  
-authenticationServer "1.1.1.1" \  
-tunnelAttributeSet $DefaultTunnelAttributes
```

SEE ALSO



RadiusOptionSetList

SYNOPSIS

DESCRIPTION

List of objects of `DHCPOptionSet` type.

SUBCOMMANDS

OPTIONS

EXAMPLE

SEE ALSO

RadiusOptionSet

SYNOPSIS

DESCRIPTION

Configures a Radius option set for use by an IP/Radius range.

SUBCOMMANDS

OPTIONS

`defaulttp`

If true, this option set is assigned to new ranges.

API values: Default = "False".

`ipType`

IP type used on this option set.

API values: IPv4, IPv6, Default = "IPv4".

`name`

Name of the option set.

API values: Default = "OptionSetName".

EXAMPLE

```
set DefaultTunnelAttributes [::IxLoad new ixNetRadiusOptionSet]
$DefaultTunnelAttributes config \
  -defaulttp true \
  -name "DefaultTunnelAttributes" \
  -ipType "IPv4"
$DefaultTunnelAttributes messages.clear
```

SEE ALSO



RadiusMessageList

SYNOPSIS

DESCRIPTION

List of objects of RadiusMessage type.

SUBCOMMANDS

OPTIONS

EXAMPLE

```
set RADIUS_R1 [$Subscriber_R1 getExtensionRange $RADIUS_1]

set DefaultTunnelAttributes [::IxLoad new ixNetRadiusOptionSet]
$DefaultTunnelAttributes config \
    -defaulttp true \
    -name "DefaultTunnelAttributes" \
    -ipType "IPv4"

$DefaultTunnelAttributes messages.clear
```

SEE ALSO



RadiusMessage

SYNOPSIS

DESCRIPTION

Configures a Radius message.

SUBCOMMANDS

OPTIONS

`defaulttp`

If true, this option set is applied to new ranges.

API values: Default = "False".

`ipType`

IP version used with this option type.

API values: IPv4, IPv6, default = "IPv4"

`messageType`

The RADIUS message type. The plug-in supports the following message types:

- Access-Request
- Access-Accept
- Accounting-Request-START
- Accounting-Request-Interim-Update

Accounting-Request-STOP

API values:

Value	Description
ACCESS_REQUEST (default)	Access-Request"
ACCESS_ACCEPT	Access-Accept
ACCOUNTING_REQUEST_START	Accounting-Request-START
ACCOUNTING_REQUEST_STOP	Accounting-Request-STOP

ACCOUNTING_REQUEST_INTERI M	Accounting-Request-Interim-Update
--------------------------------	-----------------------------------

name

Name of the option set.

API values: Default = "OptionSetName".

optionTlvs

Name of the list of RadiusOptionTLV objects.

API values: Default = "None"

EXAMPLE

```
set AccessRequest [::IxLoad new ixNetRadiusMessage]
# ixNet objects needs to be added in the list before they are configured!
$DefaultTunnelAttributes messages.appendItem -object $AccessRequest

$AccessRequest config \
    -name                "AccessRequest" \
    -defaulttp           true \
    -messageType         "ACCESS_REQUEST" \
    -ipType              "IPv4"
```

SEE ALSO



RadiusOptionTLVList

SYNOPSIS

DESCRIPTION

List of Radius type-length-value entities for Radius options. The list contains items that are of RadiusOptionTLV type.

SUBCOMMANDS

OPTIONS

EXAMPLE

```
set AccessRequest [::IxLoad new ixNetRadiusMessage]
# ixNet objects needs to be added in the list before they are configured!
$DefaultTunnelAttributes messages.appendItem -object $AccessRequest

$AccessRequest config \
    -name "AccessRequest" \
    -defaultp true \
    -messageType "ACCESS_REQUEST" \
    -ipType "IPv4"

$AccessRequest optionTlvs.clear
```

SEE ALSO

RadiusOptionTLV

SYNOPSIS

DESCRIPTION

Configures a RADIUS Profile parameter (type-length-value field) for use by a `DHCPOptionTLVList`.

SUBCOMMANDS

OPTIONS

`code`

The attribute number or sub-attribute number, as defined in technical specification *3GPP TS 29.061 V7.7.0*.

Each attribute has a predefined numeric value. You need to specify the code only when you add an attribute that is not on the predefined list.

API values: Min = 0, max = 65535, default = 1.

`name`

The RADIUS attribute name.

This field provides a combo box with a predefined list of RADIUS attribute names. When you select one of these options, `IxLoad` enters appropriate values into the Code, Type, and Value columns.

The attributes present on the drop-down list are defined in technical specification *3GPP TS 29.061 V7.7.0*. Refer to topic "Sub-attributes of the 3GPP Vendor-Specific attribute" in that document for detailed information.

API values: Default = "OptionValueName"

`rfc`

True if defined in RFC documents.

API values: Default = false.

`value`

The value to assign to the attribute. You will define the value in one of the following ways:



- *AUTO*: The Ixia port automatically determines the value, based on configuration of the RADIUS Client and Mobile subscribers plug-ins.
- *VALUE*: You must manually enter a value. For example, you must enter a valid value for the Called-Station-Id attribute: IxLoad does not provide a default value.
- Drop-down list: Some attributes provide a drop-down list from which you select a value. For example, the 3GPP-RAT-Type attribute provides a drop-list of all the valid values.
- Constant: In some cases, an attribute will have a single, non-modifiable value.

For any attribute that permits user selection or entry, IxLoad provides a default value. For example, IxLoad provides the default value of *UTRAN* for the 3GPP-RAT-Type attribute.

API default: Default =""VALUE"".

type

The attribute format type:

- String
- Hexadecimal
- Domain Name
- Zero Length
- Boolean
- 8 bit Integer
- 8 bit Integer List
- 16 bit Integer
- 16 bit Integer List
- 32 bit Integer
- 32 bit Integer List
- IPv4 Prefix
- IPv4 Address
- IPv4 Address List

For each attribute that you select, IxLoad provides a default type. Each predefined attribute has a predefined type. You need to specify the type only when you add an attribute that is not on the predefined list.

API values:

Value	Description
10	String
20	Hexadecimal value
30	Domain name
40	Zero length
50	Boolean
60	8-bit integer
61	8-bit integer list
70	16-bit integer

71	16-bit integer list
80	32-bit integer
81	32-bit integer list
90	IPv4 prefix
100	IPv6 prefix
110	IPv4 address
111	IPv4 address list
120	IPv6 address
121	IPv6 address list

EXAMPLE

```
set 3GPP_IMSI [::IxLoad new ixNetRadiusOptionTLV]
# ixNet objects needs to be added in the list before they are configured!
$AccessRequest optionTlvs.appendItem -object $3GPP_IMSI

$3GPP_IMSI config \
  -rfc true \
  -code "26/10415/1" \
  -type 10 \
  -name "3GPP-IMSI" \
  -value "AUTO"
```

SEE ALSO



RadiusSessionData

SYNOPSIS

DESCRIPTION

Configures the Radius general parameters.

SUBCOMMANDS

OPTIONS

`optionSets`

Name of the list of options used by this plugin. The list must be a `RadiusOptionSetList` object.

Default value = "None"

EXAMPLE

```
set DefaultTunnelAttributes [::IxLoad new ixNetRadiusOptionSet]
$DefaultTunnelAttributes config \
  -defaulttp true \
  -name "DefaultTunnelAttributes" \
  -ipType "IPv4"

$DefaultTunnelAttributes messages.clear
```

SEE ALSO

Mobile Subscribers Example

This section shows an example of how to create a Mobile Subscribers plugin in the Tcl API.

Mobile Subscribers Example

```
#####
# IxLoad ScriptGen created TCL script
# Network1 serialized using version 4.30.0.131
# 3GPP_radius_mobile_subs.tcl made on Aug 17 2009 13:33
#####

set Network1 [::IxLoad new ixNetworkGroup $chassisChain]
$Network1 config \
    -comment          "" \
    -name              "Network1" \
    -macMappingMode   0 \
    -linkLayerOptions 0

$Network1 globalPlugins.clear
$Network1 globalPlugins.appendItem -object $Filter_1

$Filter_1 config \
    -all                false \
    -pppoecontrol       false \
    -isis               false \
    -auto                true \
    -udp                 "" \
    -tcp                 "" \
    -mac                 "" \
    -pppoenetwork       false \
    -ip                  "" \
    -icmp                ""

set GratARP_1 [::IxLoad new ixNetGratArpPlugin]
$Network1 globalPlugins.appendItem -object $GratARP_1

$GratARP_1 config \
    -forwardGratArp    false \
    -enabled            true
```

Create a network group.

Clear the global plugins list.

Begin appending items to global plugin list.

Optionally, create a filter to filter traffic.

Optionally, enable Gratuitous ARP.



Mobile Subscribers Example

```
set TCP_1 [::IxLoad new ixNetTCPPlugin]
# ixNet objects needs to be added in the list before they
are configured!
$Network1 globalPlugins.appendItem -object $TCP_1

$TCP_1 config \
    -tcp_bic 0 \
    -tcp_tw_recycle true \
    -tcp_retries2 5 \
    -disable_min_max_buffer_size false \
    -tcp_retries1 3 \
    -tcp_keepalive_time 7200 \
    -tcp_rfc1337 false \
    -tcp_ipfrag_time 30 \
    -tcp_rto_max 60000 \
    -tcp_vegas_alpha 2 \
    -tcp_ecn false \
    -tcp_westwood 0 \
    -tcp_rto_min 1000 \
    -tcp_reordering 3 \
    -tcp_vegas_cong_avoid 0 \
    -tcp_keepalive_intvl 75 \
    -tcp_rmem_max 262144 \
    -tcp_orphan_retries 0 \
    -tcp_max_tw_buckets 180000 \
    -tcp_wmem_default 4096 \
    -tcp_low_latency 0 \
    -tcp_rmem_min 4096 \
    -tcp_adv_win_scale 2 \
    -tcp_wmem_min 4096 \
    -tcp_port_min 1024 \
    -tcp_stdurg false \
    -tcp_port_max 65535 \
    -tcp_fin_timeout 60 \
    -tcp_no_metrics_save false \
    -tcp_dsack true \
    -tcp_mem_high 49152 \
    -tcp_frto 0 \
    -tcp_app_win 31 \
    -tcp_vegas_beta 6 \
    -tcp_window_scaling false \
    -tcp_max_orphans 8192 \
    -tcp_mem_pressure 32768 \
    -tcp_syn_retries 5 \
```

Configure the TCP
portion of the stack.

Mobile Subscribers Example

```
set DNS_1 [::IxLoad new ixNetDnsPlugin]
# ixNet objects needs to be added in the list before they are
configured!
$Network1 globalPlugins.appendItem -object $DNS_1

$DNS_1 config \
    -domain          "" \
    -timeout         30

$DNS_1 hostList.clear

$DNS_1 searchList.clear

$DNS_1 nameServerList.clear

set Settings_1 [::IxLoad new ixNetIxLoadSettingsPlugin]
# ixNet objects needs to be added in the list before they are
configured!
$Network1 globalPlugins.appendItem -object $Settings_1

$Settings_1 config \
    -teardownInterfaceWithUser  false \
    -interfaceBehavior          0

set Ethernet_1 [$Network1 getLiPlugin]

set my_ixNetEthernetELMPlugin [::IxLoad new
ixNetEthernetELMPlugin]
$my_ixNetEthernetELMPlugin config \
    -negotiationType      "master" \
    -negotiateMasterSlave true

set my_ixNetDualPhyPlugin [::IxLoad new ixNetDualPhyPlugin]
$my_ixNetDualPhyPlugin config \
    -medium
```

Optionally, configure the DNS settings.

Configure the Dynamic Control plane settings.

Configure the physical layer properties.



Mobile Subscribers Example

```
$Ethernet_1 config \  
-advertise10Full true \  
-directedAddress "01:80:C2:00:00:01" \  
-autoNegotiate true \  
-advertise100Half true \  
-advertise10Half true \  
-enableFlowControl false \  
-speed "k100FD" \  
-advertise1000Full true \  
-advertise100Full true \  
-cardElm $my_ixNetEthernetELMPugin \  
-cardDualPhy $my_ixNetDualPhyPlugin  
  
$Ethernet_1 childrenList.clear  
  
set MAC_VLAN_2 [::IxLoad new ixNetL2EthernetPlugin]  
# ixNet objects needs to be added in the list before they are  
configured!  
$Ethernet_1 childrenList.appendItem -object $MAC_VLAN_2  
  
$MAC_VLAN_2 childrenList.clear  
  
set Emulated_Router_1 [::IxLoad new ixNetEmulatedRouterPlugin]  
# ixNet objects needs to be added in the list before they are  
configured!  
$MAC_VLAN_2 childrenList.appendItem -object $Emulated_Router_1  
  
$Emulated_Router_1 childrenList.clear
```

Configure the Ethernet parameters.

Configure the MAC addresses and VLAN tags.

Configure the Emulated Router.

Mobile Subscribers Example

```
set Mobile_Subscribers_1 [::IxLoad new ixNetMobileSubscribersPlugin]
# ixNet objects needs to be added in the list before they are
configured!
$Emulated_Router_1 childrenList.appendItem -object
$Mobile_Subscribers_1

$Mobile_Subscribers_1 childrenList.clear

$Mobile_Subscribers_1 extensionList.clear

set RADIUS_1 [::IxLoad new ixNetRadiusPlugin]
# ixNet objects needs to be added in the list before they are
configured!
$Mobile_Subscribers_1 extensionList.appendItem -object $RADIUS_1

$Emulated_Router_1 extensionList.clear

$MAC_VLAN_2 extensionList.clear

$Ethernet_1 extensionList.clear
```

Configure a Mobile Subscribers plugin.

Configure a Radius plugin .

Clear the extension lists.



Mobile Subscribers Example

```
#####  
# Setting the ranges starting with the plugin on top of the stack  
#####  
$Mobile_Subscribers_1 rangeList.clear  
set Subscriber_R1 [::IxLoad new ixNetMobileSubscribersRange]  
# ixNet objects needs to be added in the list before they are  
configured!  
$Mobile_Subscribers_1 rangeList.appendItem -object $Subscriber_R1  
  
$Subscriber_R1 config \  
-reliabilityClass                2 \  
-interimUpdateInterval           1 \  
-sduErrorRatio                   6 \  
-publishStats                    false \  
-guaranteedBitRateUL             64 \  
-authType                        "PAP" \  
-maxSDUSize                       151 \  
-guaranteedBitRateDL             64 \  
-signalingIndication             0 \  
-imsiMSIN                        1000000001 \  
-iMEI                            "999900000000001" \  
-imsiMCC                          226 \  
-trafficHandlingPriority          1 \  
-deliveryOrder                   2 \  
-delayClass                       0 \  
-papPassword                      "password" \  
-chapName                         "user" \  
-meanThroughput                  31 \  
-peakThroughput                   6 \  
-subscriberIncrement              1 \  
-precedenceClass                  2 \  
-maxBitRateDL                     8640 \  
-papUser                          "user" \  
-subscriberCount                  1 \  
-ipType                           "IPv4" \  
.  
.  
-residualBER                       7  
  
set RADIUS_R1 [$Subscriber_R1 getExtensionRange $RADIUS_1]
```

Configure an address range for the Mobile Subscribers plugin.

Mobile Subscribers Example

```
set DefaultTunnelAttributes [::IxLoad new ixNetRadiusOptionSet]
$DefaultTunnelAttributes config \
    -defaulttp true \
    -name "DefaultTunnelAttributes" \
    -ipType "IPv4"
$DefaultTunnelAttributes messages.clear
```

Configure the tunnel attributes option set.

```
set AccessRequest [::IxLoad new ixNetRadiusMessage]
# ixNet objects needs to be added in the list before they are
configured!
$DefaultTunnelAttributes messages.appendItem -object
$AccessRequest
```

Configure a Radius message for the option set.

```
$AccessRequest config \
    -name "AccessRequest" \
    -defaulttp true \
    -messageType "ACCESS_REQUEST" \
    -ipType "IPv4"
```

```
$AccessRequest optionTlvs.clear
```

Configure a TLV for the message.

```
set 3GPP_IMSI [::IxLoad new ixNetRadiusOptionTLV]
# ixNet objects needs to be added in the list before they are
configured!
$AccessRequest optionTlvs.appendItem -object $3GPP_IMSI
```

```
$3GPP_IMSI config \
    -rfc true \
    -code "26/10415/1" \
    -type 10 \
    -name "3GPP-IMSI" \
    -value "AUTO"
```



Mobile Subscribers Example

```
$RADIUS_R1 config \  
  -retries 3 \  
  -accountingServer "1.1.1.1" \  
  -enabled true \  
  -secret "secret" \  
  -accountingPort 1813 \  
  -enableAccounting true \  
  -timeout 10 \  
  -authenticationPort 1812 \  
  -authenticationServer "1.1.1.1" \  
  -tunnelAttributeSet $DefaultTunnelAttributes
```

Configure the Radius range properties.

```
$Emulated_Router_1 rangeList.clear
```

Add the range list to the emulated router.

```
set ER_R1 [::IxLoad new ixNetEmulatedRouterRange]  
# ixNet objects needs to be added in the list before they are  
configured!  
$Emulated_Router_1 rangeList.appendItem -object $ER_R1
```

```
$ER_R1 config \  
  -count 100 \  
  -enableGatewayArp false \  
  -generateStatistics false \  
  -autoCountEnabled false \  
  -enabled true \  
  -autoMacGeneration true \  
  -incrementBy "0.0.0.1" \  
  -prefix 16 \  
  -gatewayIncrement "0.0.0.0" \  
  -gatewayIncrementMode "perSubnet" \  
  -mss 1460 \  
  -gatewayAddress "0.0.0.0" \  
  -ipAddress "10.10.0.101" \  
  -ipType "IPv4"
```

Mobile Subscribers Example

```
set MAC_R2 [$ER_R1 getLowerRelatedRange "MacRange"]

$MAC_R2 config \
  -count 100 \
  -mac "00:0A:0A:00:65:00" \
  -mtu 1500 \
  -enabled true \
  -incrementBy "00:00:00:00:00:01"

set VLAN_R2 [$ER_R1 getLowerRelatedRange "VlanIdRange"]

$VLAN_R2 config \
  -incrementStep 1 \
  -innerIncrement 1 \
  -firstId 1 \
  -uniqueCount 4094 \
  -idIncrMode 2 \
  -enabled false \
  -innerFirstId 1 \
  -innerIncrementStep 1 \
  -priority 1 \
  -increment 1 \
  -innerUniqueCount 4094 \
  -innerEnable false \
  -innerPriority 1
```

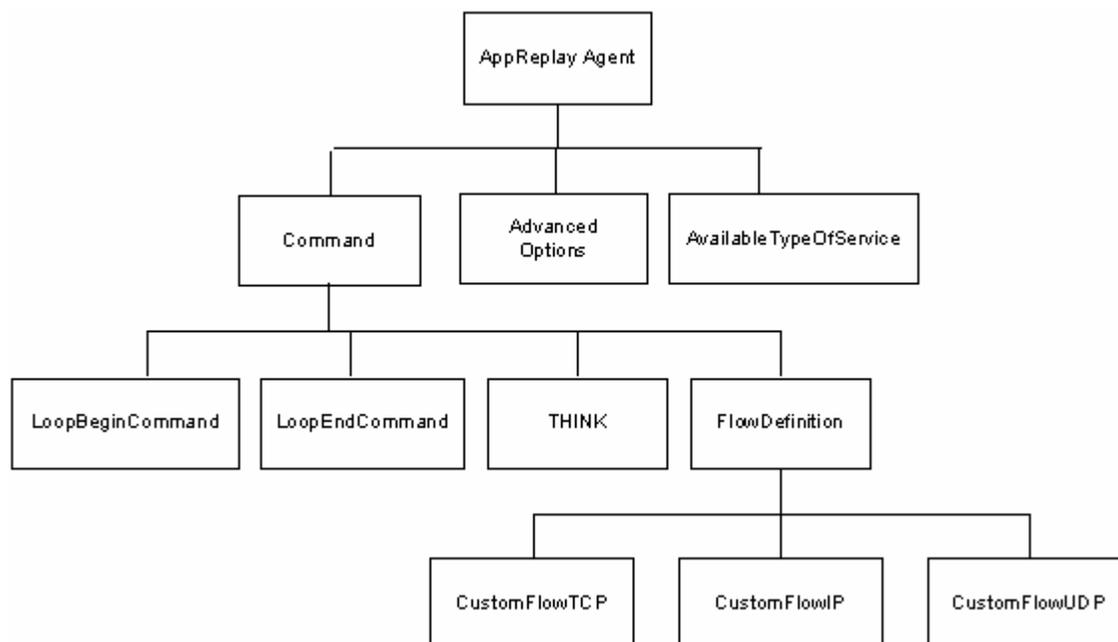
Configure the MAC addresses for the range.

Configure the VLAN tags for the range.

7

AppReplay

The IxLoad Application Replay API consists of the Application Replay Peer Agent and its commands.



In this chapter:

Objectives	781
Application Replay Peer Agent	782
Global Statistics	801

Objectives

The objectives (userObjective) you can set for Application Replay are listed below. Test objectives are set in the ixTimeline object.

- simulatedUsers
- peerCount (displays as Initiator Peer Count in the GUI)
- connectionRate
- concurrentConnections
- throughputMBps
- throughputKBps
- transactionRate



Application Replay Peer Agent

Application Replay Peer Agent - create an Application Replay agent

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set AppReplayPeer1 [$Traffic1_Network1 activityList.appendItem options...]
$AppReplayPeer1 agent.config
```

DESCRIPTION

An ApplicationReplay peer agent is added to the `activityList` object. The `activi` object is added to the `ixNetTraffic` object using the `appendItem` subcomfrom the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time .

`concurrentObjectiveBehaviour`

An optional parameter that is used to achieve the concurrent connections number to the configured value.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity AppReplayPeer1 of NetTraffic Traffic1@Network1
#####
```

```
set Activity_AppReplayPeer1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "AppReplay Peer" ]  
$Activity_AppReplayPeer1 agent.config \  
    -concurrentObjectiveBehaviour    1 \  
    -enable                          true \  
    -name                            "AppReplayPeer1"
```

SEE ALSO

`ixNetTraffic`



Flow Definition

FlowDefinition—Defines a remote peer activity and port.

SYNOPSIS

```
set ServerTraffic1_ServerNetwork1 [::IxLoad new ixNetTraffic]
set Activity_AppReplayPeer1 [$Traffic1_Network1 activityList.appendItem \
    -protocolAndType                "AppReplay Peer" ]
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the list of protocol flows using the appendItem subcommand from the ixConfigSequenceContainer command.

SUBCOMMANDS

None

OPTIONS

None.

EXAMPLE

```
Activity_AppReplayPeer1 agent.pm.protocolFlows.clear
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem \
    -id                "CustomFlowTCP" \
    -captureFile       "C:/Captures/httpuser.cap" \
    -sessionSelectionLogic 0 \
    -remotePeer        "Traffic1_AppReplayPeer1" \
    -responderPort     80
    -filt_InitiatorIP  "198.18.0.1" \
    -filt_ResponderPort "80" \
    -filt_InitiatorPort "6140" \
    -filt_ResponderIP  "198.18.0.101" \
    -overrideResponderPort false \
```

SEE ALSO

CustomFlowTCP



CustomFlowTCP

Custom FlowTCP —Specifies a custom TCP session, defined by a capture file, that is replayed between the initiator and the responder host.

SYNOPSIS

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the `ProtocolFlows` list of using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None

OPTIONS

`captureFile`

Capture file, in pcap format, that the TCP session is replayed from.

`sessionSelectionLogic`

The session logic determining the point in the capture file where replay of the TCP session starts from.

Value	Description
0	Follow First SYN. Replay starts from the first SYN packet found in the capture file. The initiator and responder addresses and port numbers are taken from the source IP, destination IP, source port, and destination port (respectively) in the first TCP SYN packet.
1	User-defined Filter. Replay starts from the first SYN packet found in the capture file that matches the filter criteria. To configure the filter, the initiator and responder addresses and port numbers have to be defined.

`filt_InitiatorIP`

The initiator IP in case a user-defined filter has been chosen.

`filt_ResponderIP`

The responder IP in case a user-defined filter has been chosen.

`filt_InitiatorPort`

The initiator port in case a user-defined filter has been chosen.

```
filt_ResponderPort
```

The responder port in case a user-defined filter has been chosen.

```
overrideResponderPort
```

If `true`, enables you to override the responder port number defined in the `FlowDefinition` object. If `false`, the responder port is set according to the filter configuration.

EXAMPLE

```
Activity_AppReplayPeer1 agent.pm.protocolFlows.clear

$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem \
    -id                "CustomFlowTCP" \
    -captureFile       "C:/Captures/httpuser.cap" \
    -sessionSelectionLogic 0 \
    -remotePeer        "Traffic1_AppReplayPeer1" \
    -responderPort     80
    -filt_InitiatorIP  "198.18.0.1" \
    -filt_ResponderPort "80" \
    -filt_InitiatorPort "6140" \
    -filt_ResponderIP   "198.18.0.101" \
    -overrideResponderPort false
```

SEE ALSO

`FlowDefinition`



CustomFlowUDP

Custom FlowUDP — Specifies a custom UDP session, defined by a capture file, that is replayed between the initiator and the responder host.

SYNOPSIS

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the `ProtocolFlows` list of using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None

OPTIONS

`id`

Name of the command. Default = "CustomFlowUDP"

`flow_name`

Name of the flow. Default = "" (null)

`captureFile`

Capture file, in pcap format, that the UDP session is replayed from. Default = ""(none)

`destination`

Destination of the traffic from the initiator:

- If the destination is a DUT, this is the IP address of the DUT.
- If the destination is another AppReplay peer, this is the name of the AppReplay activity.

Default = "None"

`sessionSelectionLogic`

The session logic determining the point in the capture file where replay of the TCP session starts from. Default = false.

Value	Description
0	Follow first UDP Packet. Replay starts from the first UDP packet found in the capture file.
1	User-defined Filter. Replay starts from the first UDP packet found in the capture file that matches the filter criteria. To configure the filter, define the initiator and responder IP addresses and port numbers (<code>filt_InitiatorIP</code> , <code>filt_ResponderIP</code> , <code>filt_InitiatorPort</code> , and <code>filt_ResponderPort</code>).

`filt_InitiatorIP`

If `sessionSelectionLogic=1`, this parameter defines the initiator IP of the packet to begin playback from. Default=""(null).

`filt_ResponderIP`

If `sessionSelectionLogic=1`, this parameter defines the responder IP of the packet to begin playback from. Default = ""(none)

`filt_InitiatorPort`

If `sessionSelectionLogic=1`, this parameter defines the initiator port of the packet to begin playback from. Specify "[ANY]" for any port. Default = ""(none)

`filt_ResponderPort`

If `sessionSelectionLogic=1`, this parameter defines the responder port of the packet to begin playback from. Specify "[ANY]" for any port. Default = ""(none)

`overrideResponderPort`

If `true`, you can override the responder port number defined in the `FlowDefinition` object. If `false`, the responder port is set according to the filter configuration. Default = `false`.

`responderPort`

Port number that responding peer listens on. Default = 10000.

By default, this parameter is read-only. If `overrideResponderPort` is `true`, you can change the port number. If you change the port number, the responding peer automatically updates itself with the new port number.

EXAMPLE

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.clear
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem \
  -id "CustomFlowUDP" \
  -flow_name "CustomFlow2" \
  -sessionSelectionLogic 0 \
  -filt_InitiatorIP "" \
  -destination "None" \
  -overrideResponderPort false \
  -filt_ResponderPort "" \
  -captureFile "" \
  -filt_ResponderIP "" \
  -responderPort 10000 \
  -filt_InitiatorPort ""
```

SEE ALSO

FlowDefinition

CustomFlowIP

Custom FlowIP — Specifies a custom IP session, defined by a capture file, that is replayed between the initiator and the responder host.

SYNOPSIS

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the `ProtocolFlows` list of using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None

OPTIONS

`id`

Name of the command. Default = "CustomFlowIP"

`flow_name`

Name of the flow. Default = "" (null)

`cycleThroughInitiatorPortUDP`

If `true` and this Custom Flow-IP command is run more than once during a test, different UDP port numbers are used each time the command runs. Specify the UDP port range in `udpPortRange`. Default = "false"

`cycleThroughInitiatorPortTCP`

If `true` and this Custom Flow-IP command is run more than once during a test, different TCP port numbers are used each time the command runs. Specify the TCP port range in `tcpPortRange`. Default = "false"

`captureFile`

Capture file, in pcap format, that the TCP session is replayed from. Default = "" (null)

`destination`

Destination of the traffic from the initiator:



- If the destination is a DUT, this is the IP address of the DUT.
- If the destination is another AppReplay peer, this is the name of the AppReplay activity.

Default = "None"

packetSelectionLogic

The packet selection logic determining the point in the capture file where replay of the TCP session starts from.

Value	Description
0 (Default)	Follow first IP Packet. Replay starts from the first IP packet found in the capture file.
1	User-defined Filter. Replay starts from the first IP packet found in the capture file that matches the filter criteria. To configure the filter, define the initiator and responder IP addresses and port numbers (<code>filt_InitiatorIP</code> , <code>filt_ResponderIP</code> , <code>filt_InitiatorPort</code> , and <code>filt_ResponderPort</code>).

`filt_InitiatorIP`

If `sessionSelectionLogic=1`, this parameter defines the initiator IP of the packet to begin playback from. Default=""(null).

`filt_ResponderIP`

If `sessionSelectionLogic=1`, this parameter defines the responder IP of the packet to begin playback from. Default=""(null).

`filt_InitiatorPort`

If `sessionSelectionLogic=1`, this parameter defines the initiator port of the packet to begin playback from. Specify "[ANY]" for any port. Default=""(null).

`filt_ResponderPort`

If `sessionSelectionLogic=1`, this parameter defines the responder port of the packet to begin playback from. Specify "[ANY]" for any port. Default=""(null).

`maintainInterPacketTime`

If `true`, AppReplay attempts to maintain the same timing between consecutive packets in the replayed traffic as in the original flow. If `false`, AppReplay does not attempt to reproduce the timing between packets in the flow. Default=`false`.

`overrideResponderPortTCP`

If `true`, you can override the responder port number defined in the `FlowDefinition` object. If `false`, the responder port is set according to the filter configuration. Default=`false`.

`responderPortTCP`

TCP port number that responding peer listens on. Default=`10000`.

By default, this parameter is read-only. If `overrideResponderPortTCP` is `true`, you can change the port number. If you change the port number, the responding peer automatically updates itself with the new port number.

`overrideResponderPortUDP`

If `true`, you can override the responder port number defined in the `FlowDefinition` object. If `false`, the responder port is set according to the filter configuration. Default=`false`.

`responderPortUDP`

UDP port number that responding peer listens on. Default=`10000`

By default, this parameter is read-only. If `overrideResponderPortUDP` is `true`, you can change the port number. If you change the port number, the responding peer automatically updates itself with the new port number.

`tcpPortRange`

Range of TCP port numbers used for traffic from this peer. Default="" (null)

`udpPortRange`

Range of UDP port numbers used for traffic from this peer. Default="" (null)

`useIPAddressFromCaptureFile`

If `true`, the replayed traffic uses the same IP addresses as the original flow in the capture file. Default=`false`

EXAMPLE

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.clear
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem \
  -id "CustomFlowIP" \
  -cycleThroughInitiatorPortUDP false \
  -flow_name "CustomFlow3" \
  -filt_InitiatorIP "" \
  -udpPortRange "" \
  -packetSelectionLogic 0 \
  -cycleThroughInitiatorPortTCP false \
  -destination "None" \
  -overrideResponderPortTCP false \
  -tcpPortRange "" \
  -maintainInterPacketTime false \
  -overrideResponderPortUDP false \
  -responderPortUDP 10000 \
  -responderPortTCP 10000 \
  -filt_ResponderPort "" \
  -captureFile "" \
  -filt_ResponderIP "" \
  -useIPAddressFromCaptureFile false \
  -filt_InitiatorPort ""
```

SEE ALSO

FlowDefinition

LoopBeginCommand

LoopBeginCommand — Specifies the beginning of a command loop.

SYNOPSIS

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the ProtocolFlows list of using the appendItem subcommand from the ixConfigSequenceContainer command.

SUBCOMMANDS

None

OPTIONS

id

Name of the command. Default = "LoopBeginCommand"

LoopCount

Number of times the loop is executed. Default = 5. Min=0, Max = 2147483647.

EXAMPLE

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem \  
-id "LoopBeginCommand" \  
-LoopCount 5
```

SEE ALSO

FlowDefinition



LoopEndCommand

LoopEndCommand — Specifies the end of a command loop.

SYNOPSIS

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the ProtocolFlows list of using the appendItem subcommand from the ixConfigSequenceContainer command.

SUBCOMMANDS

None

OPTIONS

id

Name of the command. Default = "LoopEndCommand"

EXAMPLE

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem \  
-id "LoopEndCommand"
```

SEE ALSO

FlowDefinition

Think

THINK — Pauses execution of a command loop.

SYNOPSIS

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the `ProtocolFlows` list of using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None

OPTIONS

`id`

Name of the command. Default = "THINK".

`minimumInterval`

Minimum length of time to pause, in ms. Default = 1000. Min = 1, Max = 2147483647.

`maximumInterval`

Maximum length of time to pause, in ms. Default = 1000. Min = 1, Max = 2147483647.

EXAMPLE

```
$Activity_AppReplayPeer1 agent.pm.protocolFlows.appendItem \  
-id "LoopEndCommand"
```

SEE ALSO

`FlowDefinition`



availableTosList

availableTosList — Create a list of available TOS choices.

SYNOPSIS

```
$Activity_AppReplayPeer1 agent.pm.availableTosList.appendItem
```

DESCRIPTION

An option is added to the `availableTosList` using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None

OPTIONS

`id`

Name of the list. Default = "AvailableTypeOfService".

`tos_value`

TOS value to be added to the list. Default = "Best Effort 0x0".

If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

Value	Description
"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control

EXAMPLE

```
$Activity_AppReplayPeer1 agent.pm.availableTosList.clear

$Activity_AppReplayPeer1 agent.pm.availableTosList.appendItem \
  -id "AvailableTypeOfService" \
  -tos_value "Class 1 (0x20)"
```

SEE ALSO



Advanced Options

AdvancedOptions—Defines the App Replay client's global options.

SYNOPSIS

```
$Activity_AppReplayPeer1 agent.pm.advOptions.config \  
-max_concurrent_flows      1 \  
-payloadVerification        0 \  
-typeOfService              "Best Effort (0x0)" \  
-enableTOS                  false
```

DESCRIPTION

Defines the App Replay client's global options.

SUBCOMMANDS

None

OPTIONS

enableTOS

Enables the setting of the TOS (Type of Service) bits in the IP header of the packets.

Value	Description
0	(default) TOS bits not enabled.
1	TOS bits enabled.

enableOOSforUDP

Enables out-of-sequence packet handling for UDP packets.

Value	Description
0	(default) Disabled.
1	Enabled.

max_concurrent_flows

Maximum number of flows that each simulated user can send at one time. Default = 1.

payloadVerification

Packet payload is being verified based on the following two options:

- `Verify Content (0)`: The content of the payload is verified byte-by-byte against the expected payload. This verification option is set by default.

- Verify Length (1): The length of the payload is verified against the expected value. Choosing this option yields better throughput performance.
-

EXAMPLE

```
$Activity_AppReplayPeer1 agent.pm.advOptions.config \
  -max_concurrent_flows      1 \
  -enableTOS                  false \
  -payloadVerification        1 \
  -enableOOSforUDP            false \
  -enableAdvanceStats         false \
  -typeOfService              "Best Effort (0x0)" \
  -SeqNumInPayload            01
```

SEE ALSO

Global Statistics

The following table describes the global statistics for the Application Replay peer. AppReplay statistics are available on both a global basis and per-flow.



Note: The segment latency statistics are only displayed if you enable Advanced Statistics on the Advanced Options tab.

Statistic	Description
Test Objective Statistics	
AppReplay Application Initiator Peer Count	Number of Application Replay initiator peers created.
AppReplay Application Responder Peer Count	Number of Application Replay responder peers created.
AppReplay Connection Rate	Rate (in connections per second) at which Application Replay peers connected to each other.
AppReplay Concurrent Sessions	Number of concurrent sessions established between peers.
AppReplay Transaction Rate	Rate (in transactions per second) at which Application Replay peers completed transactions. For Application Replay peers, one transaction consists of a Layer-7 protocol's request packet, and the responses to that packet.
AppReplay Initiator Total Bytes Sent/sec	Rate at which the initiators sent data.
AppReplay Application Initiator Total Bytes Received/sec	Rate at which the initiators received data.



AppReplay Initiator Total Throughput	Combined rate at which the initiators sent and received data.
AppReplay Responder Total Bytes Sent/sec	Rate at which the responders sent data.
AppReplay Responder Total Bytes Received/sec	Rate at which the responders received data.
AppReplay Responder Total Throughput	Combined rate at which the responders sent and received data.
Total Connection Statistics	
AppReplay Connection Requests Sent	Number of connection requests sent by the initiators to the responders.
AppReplay Connection Requests Successful	Number of connection attempts that succeeded.
AppReplay Connection Requests Failed	Number of connection attempts that failed.
AppReplay Connection Requests Received	Number of connection requests received by the responders.
AppReplay Connections Accepted	Number of connections accepted by the responders. This statistic measures the number of successful connections from the point of view of the responder.
AppReplay Connections Failed	Number of connections that were established but then closed because they would have exceeded the maximum number of connections that the responder could support. The maximum number of connections that the responder can accept is calculated based on the test configuration and depends on the resources available on the load module, such as memory.
AppReplay Active Connections	Number of connections currently active.
Total Transaction Statistics	
AppReplay Total Transactions Initiated	Total number of TCP or UDP transactions initiated.
AppReplay Total Transactions Successful	Total number of TCP or UDP transactions that succeeded.
Total Flow Replay Statistics	
AppReplay Total Flow Replays Initiated	Total number of TCP, IP, or UDP flow replays initiated. A TCP flow consists of a SYN, SYN+ACK, FIN, and FIN+ACK packets, all for the same session. To be considered valid, a flow must begin with a SYN packet, and end with packets from both the initiator and the responder with the FIN flag set, or a RESET.

AppReplay Total Active Flow Replays	Total number of flows being replayed.
AppReplay Total Flow Replays Succeeded	Total number of flows replayed successfully.
AppReplay Total Flow Replays Failed	Total number of flow replays that failed for any reason.
AppReplay Total Flow Replays Failed Error	Total number of flow replays that failed due to a network error.
AppReplay Total Flow Replays Failed Timeout	Total number of flow replays that failed due to a timeout.
AppReplay Total Flow Replays Failed Mismatch	Total number of flow replays that failed because the replayed session did not match the session in the pcap file.
AppReplay Total Flow Replays Aborted	Total number of flow replays aborted for any reason. Aborted flows are flows in which the session is terminated abnormally. Flows can be aborted if a Reset is received from the far end, or the test is forcefully stopped while sessions are in progress, or for other reasons.
Initiator Total Bytes Statistics	
AppReplay Initiator Total Bytes Sent	Total number of bytes sent by the initiators.
AppReplay Initiator Total Bytes Received	Total number of bytes received by the initiators.
AppReplay Initiator Total Bytes Sent and Received	Combined total of bytes sent and received by the initiators.
Responder Total Bytes Statistics	
AppReplay Responder Total Bytes Sent	Total number of bytes sent by the responders.
AppReplay Responder Total Bytes Received	Rate at which the responders received data.
AppReplay Responder Total Bytes Sent and Received	Combined total number of bytes sent and received by the responders.
Control Tx/Rx Statistics	
AppReplay Segment Transmission Initiated	Number of segments for which transmission has begun. Segments are counted based on how they are formed in the capture file. For example, if one segment in the capture file becomes split between two segments while being replayed, it is still counted as only one segment when it is received. Conversely, if two segments in the capture file are packed into a single segment during replay, they are counted as two segments.
AppReplay Segment Transmission Succeeded	Number of segments successfully transmitted (Initiator side).
AppReplay Segment Transmission Failed	Total number of segments that failed transmission (Initiator side).
AppReplay Segment Transmission Failed (Error)	Number of segments that failed transmission due to a network error (Initiator side).

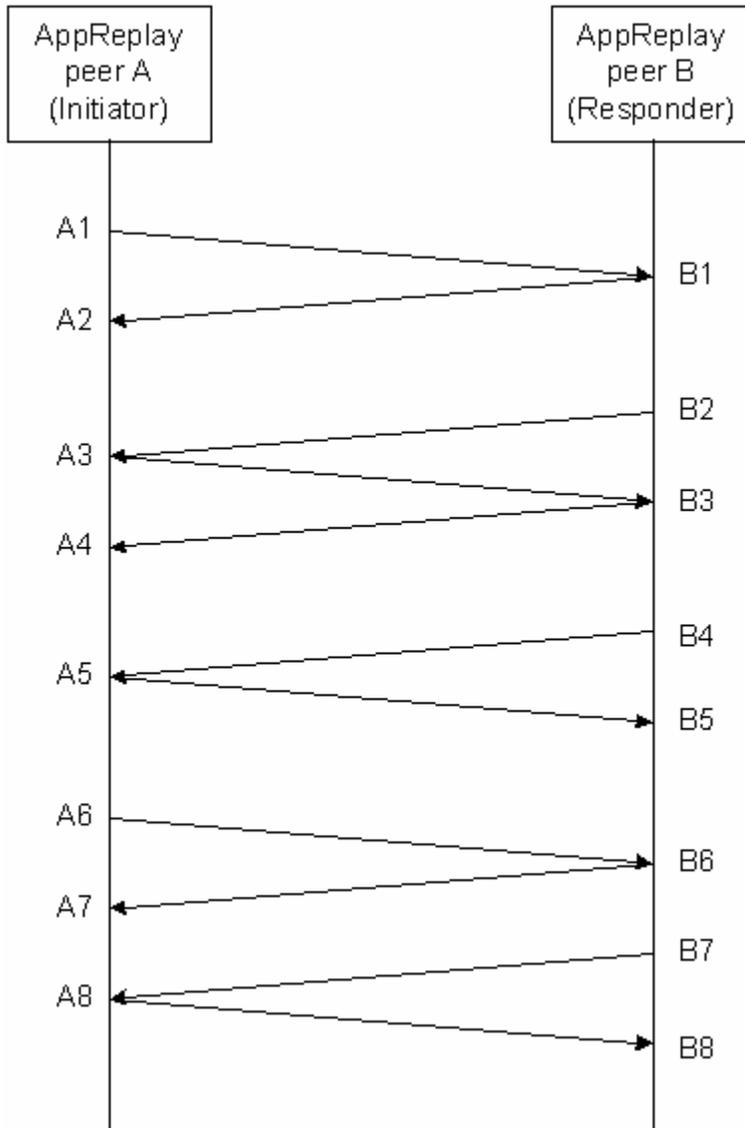


AppReplay Segment Transmission Failed (Timeout)	Number of segments that failed transmission due to a timeout (Initiator side).
AppReplay Segment Reception Initiated	Number of segments that the responders are receiving.
AppReplay Segment Reception Succeeded	Number of segments successfully received (Responder side).
AppReplay Segment Reception Failed	Total number of segments that were not received (Responder side).
AppReplay Segment Reception Failed (Error)	Number of segments that were not received due to a network error (Responder side).
AppReplay Segment Reception Failed (Timeout)	Number of segments that were not received due to a timeout (Responder side).
AppReplay Segment Reception Failed (Mismatch)	Number of segments received that did not match the segments in the pcap file (Responder side).
AppReplay UDP Lost Packets	Number of UDP packets that were transmitted but not received.
AppReplay UDP Out Of Sequence Packets	Number of UDP packets received out of sequence. This statistic only displays if <i>Enable Out of Sequence Packet Handling for UDP</i> is enabled on the Advanced Options tab.
Packet Latency Statistics	
Inter Segment First Response Latency (for Initiated Flows)	<p>The average delay between the time the initiator receives the first segment from the responder, after the initiator has sent a segment to the responder.</p> <p>In the diagram below, this statistic calculates latency by measuring time at the following points:</p> <p>A2 - A1 A4 - A3 A7 - A6</p> <p>This statistic displays a value only if the initiator has sent a segment prior to receiving a segment.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>

<p>Inter Segment Last Response Latency (for Responded Flows)</p>	<p>The average delay between the time the responder receives the first segment from the initiator, after the responder has sent a segment to the initiator.</p> <p>In the diagram below, this statistic calculates latency by measuring time at the following points:</p> <p>B3 - B2</p> <p>B5 - B4</p> <p>B8 - B7</p> <p>This statistic displays a value only if the responder has sent a segment prior to receiving a segment.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
<p>Inter Segment Last Response Latency (for Initiated Flows)</p>	<p>The average delay between the time the initiator receives the final segment from the responder, after the initiator has sent a segment to the responder.</p> <p>The received segment is considered the final segment if the flow ends after this segment, or if the initiator sends a segment after this segment.</p> <p>In the diagram below, this statistic calculates latency by measuring time at the following points:</p> <p>A3 - A1</p> <p>A5 - A3</p> <p>A8 - A6</p> <p>This statistic displays a value only if the initiator has sent a segment prior to receiving the final segment.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
<p>Inter Segment Last Response Latency (for Responded Flows)</p>	<p>The average delay between the final segment received by the responder from the initiator after the responder sent a segment to the initiator.</p> <p>The received segment is considered the final segment if the flow ends after this, or if the initiator sends a segment after this.</p> <p>In the diagram below, this statistic calculates latency by measuring time at the following points:</p> <p>B3 - B2</p> <p>B6 - B4</p> <p>B8 - B7</p> <p>This statistic displays a value only if the responder has sent a segment prior to receiving the final segment.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>

Session Life Time (for Initiated Flows)	<p>The average duration between the time the initiator sends or receives the first segment and sends or receives the final segment over a TCP session.</p> <p>In the diagram below, this statistic calculates latency by measuring time at the following points:</p> <p style="text-align: center;">A8 - A1</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
Session Life Time (for Responded Flows)	<p>The average duration between the time the responder sends or receives the first segment and sends or receives the final segment over a TCP session.</p> <p>In the diagram below, this statistic calculates latency by measuring time at the following points:</p> <p style="text-align: center;">B8 - B1</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>

The following diagram shows an example of segment exchanges in an AppReplay session and identifies the points at which the Latency statistics are measured. The diagram assumes peer A is running as an Initiator only and peer B is a Responder only.





Application Test

This section describes the Application Test Tcl API objects.

In this chapter:

Objectives	808
IxVerify VAAT in IxLoad	809
Initial Overhead	809
Adding an IxVerify Activity	810
Statistics	810

Objectives

The objectives (userObjective) you can set for Application Test are listed below. Test objectives are set in the ixTimeline object.

- simulatedUsers

IxVerify VAAT in IxLoad

The main steps in configuring IxVerify test in IxLoad are:

- Follow the standard IxLoad configuration flow. (Define Client Network and so on....)
- Add IxVerify activity in the client traffic.
- Select agenda to run or record a new one.
- Edit agenda options, if needed.
- Continue standard IxLoad configuration flow.

For running a test through Tcl API, refer to [Running an IxLoad Tcl Program](#) on page 3-27.

Initial Overhead

This section performs the necessary overhead steps to get a test started. As shown in the example, you need to load verify in the IxAppPluginManager.

```
#-----  
  
# Connect  
  
#-----  
  
# IxLoad onnect should always be called, even for local scripts  
::IxLoad connect $::IxLoadPrivate::SimpleSettings::remoteServer  
  
# once we've connected, make sure we disconnect, even if there's a problem  
if [catch {  
  
# Loads plugins for specific protocols configured in this test  
  
#  
  
global ixAppPluginManager  
  
$ixAppPluginManager load "verify"  
  
# setup logger  
  
set logtag "IxLoad-api"  
  
set logName "simpleapptestclient"  
  
set logger [::IxLoad new ixLogger $logtag 1]  
  
set logEngine [$logger getEngine]  
  
$logEngine setLevels $::ixLogger(kLevelDebug) $::ixLogger(kLevelDebug)  
  
$logEngine setFile $logName 2 256 1
```

Adding an IxVerify Activity

There are two parameters that needs to be defined here: the name of the apptest PFT file and the path.

PFT files are test scripts that are created in IxLoad Client UI. These files are saved in the same directory as IxLoad repositories in the <Repository format. After creating the Application Test script from IxLoad UI, set the pft_filename parameter to the pft file and pft_path to the directory in which the repository or pft file reside.

```
#-----  
# Construct Client Traffic  
#-----  
set expected "clnt_traffic"  
set clnt_traffic [::IxLoad new ixClientTraffic -name $expected]  
$clnt_traffic agentList.appendItem \  
    -name          "my_verify_client" \  
    -protocol      "verify" \  
    -type          "Client"
```

The PFT Agenda file name:

```
$clnt_traffic agentList(0).pm.verifyplugin.config -pft_filename "test.pft"
```

Path:

```
$clnt_traffic agentList(0).pm.verifyplugin.config -pft_path "c:\\\"
```

The rest of the configuration process is the same as for other IxLoad Tcl API tests.

Statistics

The table below describes the statistics that appear in the Application Test client.

Statistic	Description
Simulated Users	Number of virtual users simulated during the test.
Successful Rounds	Total number of times the simulated users completed one iteration of the Agenda.
Failed Rounds	Total number of times the simulated users started but did not complete an iteration of this Agenda.

Pages	Total number of page (GET) requests issued by the simulated users.
Successful Connections	<p>The total number of times the simulated users were able to connect to the application under test.</p> <p>This number is always less than or equal to the number of successful hits, because several hits might use the same HTTP connection if the Persistent Connection option is enabled (on the Browser Emulation tab).</p>
Failed Connections	<p>Total number of times the simulated users tried to connect to the application under test but were unable to.</p> <p>This number is always less than or equal to the number of failed hits, because hits can fail for reasons other than a failed connection.</p>
Successful Hits	The total number of times the simulated users made an HTTP request and received the correct HTTP response from the application under test. (Each request for a gif, jpeg, html file, etc., counts as a single hit.)
Failed Hits	The total number of times the simulated users made an HTTP request, but did not receive the correct HTTP response from the application under test. (Each request for a gif, jpeg, html file, etc., counts as a single hit.)
Response Data Size	<p>The size, in bytes, of all the HTTP responses sent by the application under test.</p> <p>Application Test uses this value to calculate the Throughput.</p>
HTTP Response Status 200	<p>Number of HTTP 200-series (OK) responses received by the simulated users.</p> <p>200-series responses indicate that the client's request was successfully received, understood, and accepted.</p>
HTTP Response Status 300	<p>Number of HTTP 300-series (Redirection) responses received by the simulated users.</p> <p>300-series responses indicate that further action needs to be taken by the user agent in order to fulfill the request.</p>
HTTP Response Status 301	<p>Number of HTTP 301 (Moved Permanently) responses received by the simulated users.</p> <p>A 301 response indicates that the requested resource has been assigned a new permanent URI.</p>
HTTP Response Status 305	<p>Number of HTTP 305 (Use Proxy) responses received by the simulated users.</p> <p>A 305 response indicates that the requested resource must be accessed through the proxy given by the Location field.</p>
HTTP Response Status 400	<p>Number of HTTP 400-series (Bad Request) responses received by the simulated users.</p> <p>400-series responses indicate that the request could not be understood by the server due to malformed syntax.</p>

HTTP Response Status 401	<p>Number of HTTP 401 (Unauthorized) responses received by the simulated users.</p> <p>A 401 response indicates that the request requires user authentication.</p>
HTTP Response Status 403	<p>Number of HTTP 403 (Forbidden) responses received by the simulated users.</p> <p>A 403 response indicates that the server understood the request, but is refusing to fulfill it.</p>
HTTP Response Status 404	<p>Number of HTTP 404 (Not Found) responses received by the simulated users.</p> <p>A 404 response indicates that the server has not found anything matching the Request-URI.</p>
HTTP Response Status 405	<p>Number of HTTP 405 (Method Not Allowed) responses received by the simulated users.</p> <p>A 405 response indicates that the method specified in the Request-Line is not allowed for the resource identified by the Request-URI.</p>
HTTP Response Status 407	<p>Number of HTTP 407 (Proxy Authentication Required) responses received by the simulated users.</p> <p>A 407 response is similar to 401 (Unauthorized), but indicates that the client must first authenticate itself with the proxy.</p>
HTTP Response Status 408	<p>Number of HTTP 408 (Request Timeout) responses received by the simulated users.</p> <p>A 408 response indicates that the client did not produce a request within the time that the server was prepared to wait.</p>
HTTP Response Status 413	<p>Number of HTTP 413 (Request Entity Too Large) responses received by the simulated users.</p> <p>A 413 response indicates that the server is refusing to process a request because the request entity is larger than the server is willing or able to process.</p>
HTTP Response Status 414	<p>Number of HTTP 414 (Request-URI Too Long) responses received by the simulated users.</p> <p>A 414 response indicates that the server is refusing to service the request because the Request-URI is longer than the server is willing to interpret.</p>
HTTP Response Status 500	<p>Number of HTTP 500-series responses received by the simulated users.</p> <p>500-series responses indicate cases in which the server is aware that it has erred or is incapable of performing the request.</p>
HTTP Response Status 501	<p>Number of HTTP 501 (Not Implemented) responses received by the simulated users.</p> <p>A 501 response indicates that the server does not support the functionality required to fulfill the request.</p>

HTTP Response Status 502	<p>Number of HTTP 502 (Bad Gateway) responses received by the simulated users.</p> <p>A 502 response indicates that the server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.</p>
HTTP Response Status 503	<p>Number of HTTP 503 (Service Unavailable) responses received by the simulated users.</p> <p>A 503 response indicates that the server is currently unable to handle the request due to a temporary overloading or maintenance of the server.</p>
HTTP Response Status 504	<p>Number of HTTP 504 (Gateway Timeout) responses received by the simulated users.</p> <p>A 504 response indicates that the server, while acting as a gateway or proxy, did not receive a timely response from the upstream server specified by the URI (e.g. HTTP, FTP, LDAP) or some other auxiliary server (e.g. DNS) that it needed to access in attempting to complete the request.</p>
HTTP Response Status 505	<p>Number of HTTP 505 (HTTP Version Not Supported) responses received by the simulated users.</p> <p>A 505 response indicates that the server does not support, or refuses to support, the HTTP protocol version that was used in the request message.</p>
Page Time	Time required, in seconds, for a simulated user to make an upper-level request to the application being tested.
Time To First Byte	Time elapsed between the moment when a request is sent and the time the simulated user receives the first data byte.
DNS Lookup Time	Time required to resolve a host name to an IP address.
Connect Time	<p>Time required for a simulated user to connect to the application being tested (the application under test). Effectively, this is the time elapsed from the beginning of the HTTP request to the establishment of the TCP/IP connection.</p> <p>If the Persistent Connection option is enabled (on the Browser Emulation tab), there may not be a value for Connect Time because the HTTP connection remains open between successive HTTP requests.</p>
Send Time	Time required for the simulated users to write an HTTP request to the application under test, in seconds.
Response Time	<p>Time required for the application under test to send the object of an HTTP request back to a simulated user, in seconds.</p> <p>Effectively, this is the time elapsed from the end of the HTTP request until the simulated user has received the complete item it had requested (Time to first byte + Receive Time - Process Time).</p> <p>(Process Time is the time required Application Test to parse an HTTP response from the application under test and populate the document-object model (the DOM)).</p>



Hit Time	<p>Time required to complete a successful HTTP request, in seconds,. Each request for a gif, jpeg, html file, etc., counts as a single hit.</p> <p>Hit Time is the sum of the Connect Time, Send Time, Response Time, and Process Time.</p> <p>(Process Time is the time required Application Test to parse an HTTP response from the application under test and populate the document-object model (the DOM)).</p>
MemFree	Amount of remaining memory available on the Ixia port.

9

Bulk MGCP

This section describes the MGCP Tcl API objects.

In this chapter:

API Overview	816
MGCP Client API	817
MGCP Server API	823
Parameters	826
MGCP Client Agent	828
MGCP Server Agent	856
Bulk MGCP Statistics	869

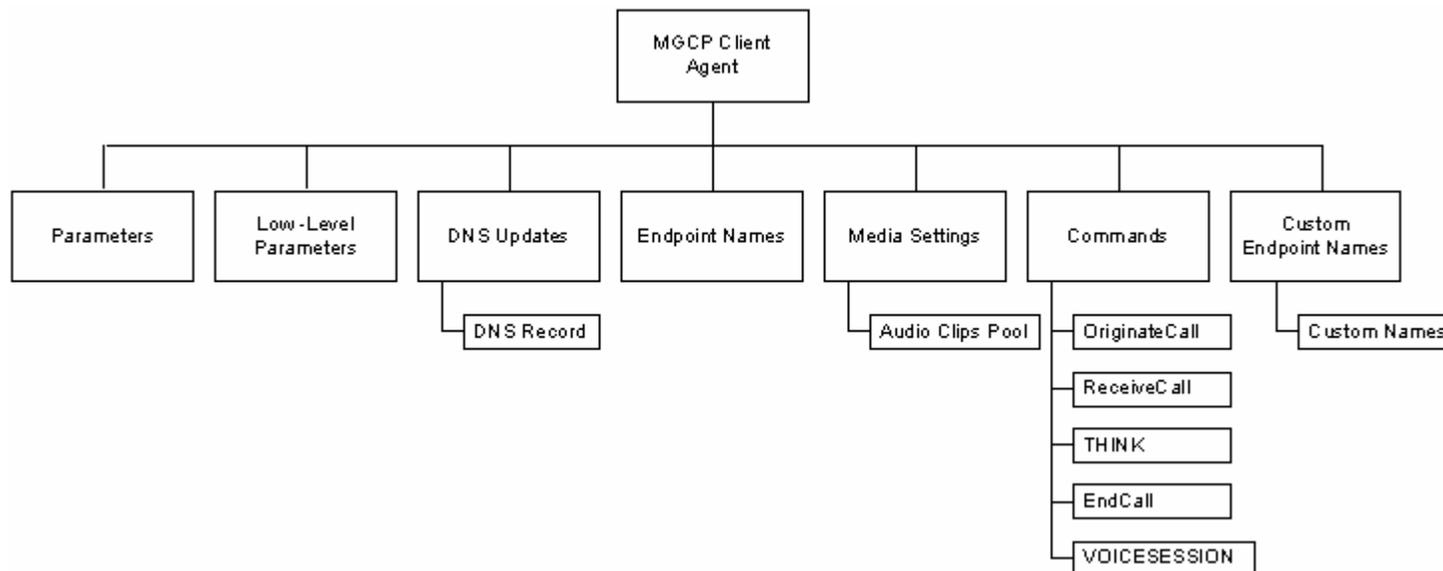


API Overview

The IxLoad MGCP API consists of MGCP client and server agents, with separate APIs for configuring each major aspect of the agents' functionality.

MGCP Client API

The IxLoad MGCP Client API commands are organized as shown in the figure below.



Objectives

The objectives (userObjective) you can set for MGCP are listed below. Test objectives are set in the ixTimeline object.

- simulatedUsers
- calls (displays as “Endpoints” in the GUI)
- transactionRate
- bhca
- callsPerSec (displays as “Calls Initiated Per Second” in the GUI)

MGCP Client Agent

The MGCP Client Agent creates an IxLoad agent that simulates an MGCP gateRefer to `MGCP Client Agent` on page 23-12 for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time .
protocol	Protocol used by the client agent.



Option	Description
type	Defines the agent as either a client or server.

Parameters

Sets an MGCP client's basic parameters. Refer to `Parameters` on page 23-21 for a full description of this command. The most significant options of this command are listed below.

Option	Description
GatewaySourcePort	Source port for MGCP commands.
CallAgent_port	Call agent that controls this Gateway.
enableTosMGCP	Enable TOS for MGCP traffic.
type_of_service_for_mgcp	IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for MGCP.
enableTosRTP	Enables the setting of the TOS (Type of Service) bits in the header of the RTP data packets.
type_of_service_for_rtp	IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for RTP data packets.

Low-Level Parameters

Sets an MGCP client's low-level parameters. Refer to `Low Level Parameters` on page 23-23 for a full description of this command. The most significant options of this command are listed below.

Option	Description
CommandTimeout	Command timeout.
LocalMediaProperties	String containing the encoding of endpoint media parameters.
AcknowledgeResponses	Specifies if the sent command will include K parameter with the ID of last received response.

DNS Record

Adds a DNS record to the list to be sent to a DNS server. Refer to `DNS Record` on page 23-24 for a full description of this command. The most significant options of this command are listed below.

Option	Description
dns_record_time_to_live	Used for DNS update query to specify time of validity of the updated DNS record.
dns_ip_port	Hostname:port number or IP address:port number of DNS server to which DNS records will be sent.

Endpoint Names

Adds a DNS record to the list to be sent to a DNS server. Refer to `Endpoint Names` on page 23-25 for a full description of this command. The most significant options of this command are listed below.

Option	Description
GatewayName	Gateway IP address:port or domain name:port.
NumberOfEndpoints	Number of endpoints hosted by the gateway.
UseCustomNames	Specifies whether to use custom names or not.
EndpointNamePrefix	Prefix applied to endpoint name.
EndpointNameSuffix	Suffix applied to endpoint name.
EndpointNameStartAt	Initial value of variable portion of endpoint name.
EndpointNameExpan	Width of variable used to create endpoint names that are unique within a gateway.
EndpointNameStep	Amount of increase in the variable (the Endpoint Name Expand On parameter) used to create unique base endpoint names.
EndpointPhonePrefix	String containing digits to be used at beginning of phone num
EndpointPhoneSuffix	String containing digits to be used at the end of the phone number.
EndpointPhoneStartAt	Initial value of variable portion of phone number.
EndpointPhoneStep	Amount of increase in variable to create additional phone numbers.
NumGateways	Number of gateways.
GatewayNamePrefix	String prefixed to gateway name.
GatewayNameSuffix	String suffixed to gateway name.
GatewayNameStartAt	Initial value of variable portion of gateway name.
GatewayNameExpan	Width of variable used to create unique gateway names.
GatewayNameStep	Amount of increase in variable used for gateway name.

Media Settings

Selects and configure the streaming audio files for the multimedia session that the client will play over RTP. Refer to `Media Settings` on page 23-27 for a full description of this command. The most significant options of this command are:

Option	Description
szCodecName	Codec to be used to encode waveform audio files listed in the Audio Clips Pool.



Option	Description
szCodecDetails	Displays the properties of the codec such as the number of bytes per frame of compressed audio, and the rate at which packets are sent over the connection.
szCodecDescr	Codec description.
bModifyPowerLevel	If <code>true</code> , <code>IxLoad</code> modifies the volume of the compressed audio.
szPowerLevel	If <code>bModifyPowerLevel</code> is <code>true</code> , this parameter specifies the amount of gain (volume) added to compressed audio.
bUseJitter	Enables or disables use of the jitter buffer.
bJitMs	Defines the method used to set the jitter buffer size.
nJitterBuffer	Jitter Buffer size, in packets.
nJitterMs	Jitter Buffer size, in milliseconds.
bUseCompensation	Enables or disables use of the compensation jitter buffer.
bCompMs	Defines the method used to set the compensation jitter buffer size.
nCompJitterBuffer	Compensation jitter buffer maximum size, in packets.
nCompJitterMs	Compensation jitter buffer maximum size, in milliseconds.
nCompMaxDropped	Maximum dropped consecutive packets.
bUseMOS	Enables or disables use of MOS.
bMosOnMax	Defines whether MOS is calculated for a subset of streams or for all streams.
nMosMaxStreams	Maximum number of concurrent streams used in MOS calcu
nMosInterval	Frequency at which <code>IxLoad</code> samples the RTP streams to genthe MOS scores.
nDtmfDuration	Length of time allowed to play the DTMF sequence.
nDtmfInterdigits	Duration (in milliseconds) of the DTMF interdigit signal.
bLimitDtmf	Enable or disable limitation on the number of DTMF streams to be processed.
nDtmfStreams	Number of streams to which path confirmation will be applied.
nPcInterval	If Synthetic path confirmation is selected, this is the interval at which <code>IxLoad</code> add the synthetic RTP packets to the stream.
nSessionType	Type of voice session.
szDtmfSeq	DTMF sequence used for path confirmation.
szPeerCodecName	Name of codec used by peer.
szPeerCodecDetails	Details of codec used by peer.
szPeerDtmfSeq	DTMF sequence used by peer.
nPeerDtmfDuration	DTMF duration used by peer.
nPeerDtmfInterdigits	Inter-digits interval used by peer.
audioClipsTable	This list contains the waveform audio files that the MGCP cli will play.

Commands

Creates the list of MGCP commands that the client will send. Refer to [Commands](#) on page 23-31 for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	MGCP command to be executed.

Audio Clips Pool

Defines an audio file to be included in the list that the MGCP client will play. Refer to [Audio Clips Pool](#) on page 23-33 for a full description of this command. The most significant options of this command are listed below.

Option	Description
szWaveName	Waveform audio (.wav) file.
szDataFormat	Encoding format of waveform audio file.
nSampleRate	Number of samples taken per second from the recording source.
nResolution	Number of bits per sample.
nChannels	Number of audio channels.
nDuration	Playing time of audio file.
nSize	Size of audio file, in bytes.
szRawWaveName	Name and path of wave file to be added to the list.

Custom Endpoint Names

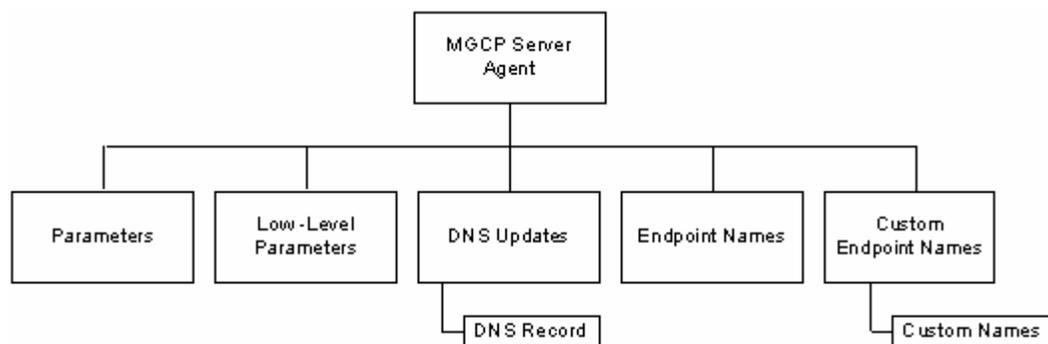
Retrieves the list of custom endpoint names generated by [Endpoint Names](#). Refer to [Custom Endpoint Names](#) on page 23-34 for a full description of this command. The most significant options of this command are:

Option	Description
endpoint_names	List of custom endpoint names to be used.



MGCP Server API

The figure below shows the MGCP Server API structure.



MGCP Server Agent

The MGCP Server Agent command simulates an MGCP Call Agent. Refer to *MGCP Server Agent* on page 23-35 for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time .
protocol	Protocol used by the client agent.
type	Defines the agent as either a client or server.

Low-Level Parameters

Sets the MGCP Server Agent's low-level commands. Refer to *Low Level Parameters* on page 23-43 for a full description of this command. The most sigoptions of this command are listed below.

Option	Description
CommandTimeout	If no response to a command is received within this number of seconds, a error is declared.
AcknowledgeResponse s	Specifies if the sent command will include the K parameter with the ID of last received response.



DNS Updates

Configures the list of DNS records that will be sent to a DNS server. Refer to `DNS Updates` on page 23-44 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>dns_record_time_to_live</code>	Used for DNS update query to specify time of validity of the updated DNS record.
<code>dns_source_ip</code>	IP address indicated as the source of the DNS records.

DNS Record

Configures a DNS record that will be added to the list to be sent to a DNS server. Refer to `DNS Record` on page 23-45 for a full description of this command. The most significant options of this command are:

Option	Description
<code>dns_record_name</code>	Name to be added to DNS database.
<code>dns_record_addresses</code>	IP address to be added to DNS database.

Endpoint Names

Configures the names used for MGCP endpoints. Refer to `Endpoint Names` on page 23-46 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>NumberOfEndpoints</code>	Number of endpoints hosted by the gateway.
<code>UseCustomNames</code>	Specifies whether to use custom names or not.
<code>EndpointNamePrefix</code>	Prefix applied to endpoint name.
<code>EndpointNameSuffix</code>	Suffix applied to endpoint name.
<code>EndpointNameStartAt</code>	Initial value of variable portion of endpoint name.
<code>EndpointNameExpandOn</code>	Width of variable used to create endpoint names that are unique within a gateway.
<code>EndpointNameStep</code>	Amount of increase in the variable (the <code>Endpoint Name Expand On</code> parameter) used to create unique base endpoint names.
<code>EndpointPhonePrefix</code>	String containing digits to be used at beginning of phone number.
<code>EndpointPhoneSuffix</code>	String containing digits to be used at the end of the phone number.
<code>EndpointPhoneStartAt</code>	Initial value of variable portion of phone number.
<code>EndpointPhoneStep</code>	Amount of increase in variable to create additional phone numbers.

Option	Description
NumGateways	Number of gateways.
GatewayNamePrefix	String prefixed to gateway name.
GatewayNameSuffix	String suffixed to gateway name.
GatewayNameStartAt	Initial value of variable portion of gateway name.
GatewayNameExpandO n	Width of variable used to create unique gateway names.
GatewayNameStep	Amount of increase in variable used for gateway name.

Custom Endpoint Names

Retrieves the list of custom endpoint names generated by `Endpoint Names`. Refer to `Custom Endpoint Names` on page 23-48 for a full description of this command. The most significant options of this command are listed below.

Option	Description
endpoint_names	List of custom endpoint names to be used.

Parameters

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.pm.parameters.config
```

DESCRIPTION

An MGCP server's basic parameters are set by modifying the options of the `pm.parameters` option of the MGCP Server Agent object using `appendItem`. Note the use of the `'pm.'` component in the name.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`call_agent_name`

Call agent FQDN name that controls this gateway. (Default = "prica.ixi").

`listen_port_start`

Initial port that the agent listens on for new MGCP connections. Minimum = "1" maximum = "65,535." (Default = "2,727").

`listen_port_stop`

Number of ports that the agent listens on for new MGCP connections. Minimum = "1." (Default = "1").

`listen_port_step`

Increment value applied initially to `listen_port_start` and to each subsequent value to create the list of listening ports. Minimum = "1." (Default = "1").

`enableTosMGCP`

Enable TOS for MGCP traffic.

Value	Description
0	(default) TOS bits disabled.
1	TOS bits enabled.

`type_of_service_for_mgcp`

If `enableTosMGCP` is `true`, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

Value	Description
"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control



MGCP Client Agent

MGCP Client Agent - configure an MGCP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.config
```

DESCRIPTION

An MGCP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity MGCPClient1 of NetTraffic Traffic1@Network1
#####
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem \
```

```
-protocolAndType           "MGCP Client" ]  
$Activity_MGCPClient1 agent.config \  
-enable                    true \  
-name                      "MGCPClient1"
```

SEE ALSO

`ixNetTraffic`



Parameters

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.pm.parameters.config
```

DESCRIPTION

An MGCP client's basic parameters are set by modifying the options of the `pm.parameters` option of the `MGCP Client Agent` object.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`GatewaySourcePort`

Source port for MGCP commands. This may be changed by Call Agent by using `NotifiedEntity` parameter. Minimum = "1," maximum = "65535." (Default = "2,427").

`CallAgent_port`

Call agent that controls this Gateway. Example: `"192.168.8.9:2427"` or `prica.ixialab.com`. (Default = "None"). The following suboptions exist for this option:

Value	Description
<code>enableDns</code>	Enable (1) or disable (0) DNS for symbolic destinations (IxLoad agents) for this call agent. Default = 0.
<code>Protocol id</code>	ID of the call agent protocol issuing the DNS request. Default = "mgcp."

`enableTosMGCP`

Enable TOS for MGCP traffic.

Value	Description
0	(default) TOS bits not enabled.
1	TOS bits enabled.

type_of_service_for_mgcp

If `enableTosMGCP` is `true`, then this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

Value	Description
"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control

enableTosRTP

Enables the setting of the TOS (Type of Service) bits in the header of the RTP data packets.

Value	Description
0	(default) TOS bits not enabled.
1	TOS bits enabled.

type_of_service_for_rtp

If `enableTosRTP` is `true`, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for RTP data packets. See `type_of_service_for_mgcp` for the list of choices (Default = "Best Effort (0x0)").

EXAMPLE

```
$Activity_MGCPClient1 agent.pm.parameters.config \
    -type_of_service_for_mgcp           "Best Effort (0x0)" \
    -CallAgent_port                     "Traffic2_MGCPServer1:2727" \
    -type_of_service_for_rtp           "Best Effort (0x0)" \
    -enableTosRTP                       true \
    -CallAgent                          "" \
    -GatewaySourcePort                  2427 \
    -enableTosMGCP                       true \
```



`-implicitLoopCheck`

`true`

SEE ALSO

`ixNetTraffic`

Low Level Parameters

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.pm.ll_parameters.config
```

DESCRIPTION

An MGCP client's low-level parameters are set by modifying the options of the `pm.ll_parameters` option of the `MGCP Client Agent` object.

SUBCOMMANDS

None.

OPTIONS

`CommandTimeout`

If no response to a command is received within this number of seconds, a error is declared. Minimum = "1," Maximum = "120." (Default = "30").

`LocalMediaProperties`

String containing the encoding of endpoint media parameters. Default = "v:on, e:off," `maxLength` = "2,048."

`AcknowledgeResponses`

Specifies if the sent command will include the `K` parameter with the ID of last received response. (Default = "0").

EXAMPLE

```
$Activity_MGCPClient1 agent.pm.ll_parameters.config \  
-LocalMediaProperties          "v:on, e:off" \  
-CommandTimeout                30 \  
-AcknowledgeResponses          true \  
-RingCount                     2
```

SEE ALSO

`MGCP Client Agent`



DNS Record

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.pm.dns_update_parameters.config
```

DESCRIPTION

The `DNS Update Parameters` command is used to add DNS records to the list of records that will be sent to the DNS server to update it with changes to the gatename.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOption` subcommands defined in the `ixConfig` command.

OPTIONS

`dns_record_time_to_live`

Used for DNS update query to specify time of validity of the updated DNS record. Default = "43,200."

`dns_ip_port`

Hostname:port number or IP address:port number of DNS server to which DNS records will be sent. Default = "192.168.1.1:53."

EXAMPLE

```
$Activity_MGCPClient1 agent.pm.dns_update_parameters.config \  
-dns_record_name           "" \  
-dns_record_address        "" \  
-dns_record_time_to_live   43200 \  
-dns_source_ip             "" \  
-enable_dns_updates        false \  
-dns_ip_port                "192.168.1.1:53"
```



Endpoint Names

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.pm.endpoint_parameters.config
```

DESCRIPTION

Configures the names used for MGCP endpoints. An MGCP client's endpoint update parameters are set by modifying the options of the `pm.endpoint_parameters` option of the MGCP Client Agent object.

SUBCOMMANDS

None.

OPTIONS

GatewayName

Gateway IP address:port or domain name:port. (Default = "ixloadmgw.ixia-lab.com").

NumberOfEndpoints

Number of endpoints hosted by the gateway. Minimum = "1," maximum = "15,000." (Default = "2").

UseCustomNames

Specifies whether to use custom names or not. (Default = "0").

EndpointNamePrefix

Prefix applied to endpoint name. (Default = "aaln/").

EndpointNameSuffix

Suffix applied to endpoint name. (Default={}).

EndpointNameStartAt

Initial value of variable portion of endpoint name. Minimum = "0," maximum = "4,294,967,295." (Default = "0").

EndpointNameExpand



On

Width of variable used to create endpoint names that are unique within a gate way. Minimum = "1," maximum = "5." (Default = "1").

EndpointNameStep

Amount of increase in the variable (the Endpoint Name Expand On parameter) used to create unique base endpoint names. Minimum = "1," maximum = "3,000." (Default = "1").

EndpointPhonePrefix

String containing digits to be used at beginning of phone number. (Default = {}).

EndpointPhoneSuffix

String containing digits to be used at the end of the phone number. (Default = {}).

EndpointPhoneStartAt

Initial value of variable portion of phone number. Minimum = "0" Maximum = "4,294,967,295." (Default = "1,000").

EndpointPhoneStep

Amount of increase in variable to create additional phone numbers. Minimum = "1," maximum = "3,000." (Default = "1").

NumGateways

Number of gateways. Minimum = "1," maximum = "3,000." (Default = "2").

GatewayNamePrefix

String prefixed to gateway name. (Default = "ix").

GatewayNameSuffix

String suffixed to gateway name. (Default = ".ixia-lab.com").

GatewayNameStartAt

Initial value of variable portion of gateway name. Minimum = "0," maximum = "4,294,967,295." (Default = "3,000").

GatewayNameExpandOn

Width of variable used to create unique gateway names. Minimum = "1," maxi= "5." (Default = "1").

GatewayNameStep

Amount of increase in variable used for gateway name. Minimum = "1," maxi= "3,000." (Default = "1").

EXAMPLE

```
$Activity_MGCPClient1 agent.pm.endpoint_parameters.config \  
-GatewayName                "ixloadmgw.ixia-lab.com" \  
-NumGateways                2 \  
-EndpointPhonePrefix        "" \  
-EndpointNameSuffix         "" \  
-EndpointPhoneStartAt      1000 \  
-EndpointNameExpandOn      1 \  
-GatewayNamePrefix         "ix" \  
-NumberOfEndpoints         2 \  
-GatewayNameStep           1 \  
-EndpointNameStartAt       0 \  
-EndpointNameStep          1 \  
-EndpointPhoneStep         1 \  
-GatewayNameStartAt       3000 \  
-UseCustomNames            false \  
-EndpointPhoneSuffix       "" \  
-EndpointNamePrefix        "aaln/" \  
-GatewayNameSuffix         ".ixia-lab.com" \  
-GatewayNameExpandOn      1
```

SEE ALSO

MGCP Client Agent



Media Settings

Media Settings—Selects and configures the streaming audio files for the multisession that the client will play over RTP.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.pm.mediaSettings.config
```

DESCRIPTION

An MGCP client's media settings are set by modifying the options of the `pm.mediaSettings` option of the `MGCP Client Agent` object.

SUBCOMMANDS

None.

OPTIONS

`szCodecName`

Codec to be used to encode waveform audio files listed in the Audio Clips Pool. The choices are:

Value	Description
"G711ALaw"	(default) G.711 A-law
"G711ULaw"	G.711 mu-law
"G729A"	G.729A
"G729B"	G.729B
"G726"	G.726
"G723_1"	G.723.1

`szCodecDetails`

Displays the properties of the codec such as the number of bytes per frame of compressed audio, and the rate at which packets are sent over the connection. (Default = {}).

`szCodecDescr`

Codec description. (Default = {}).



bModifyPowerLevel

If true, IxLoad modifies the volume of the compressed audio. (Default = "0").

szPowerLevel

If bModifyPowerLevel is true, this parameter specifies the amount of gain (voladded to compressed audio. The choices are :

Value	Description
"PL0"	(default) 0 dB
"PL_10"	-10 dB
"PL_20"	-20 dB
"PL_30"	-30 dB

bUseJitter

Enables or disables use of the jitter buffer. (Default = "0").

bJitMs

Defines the method used to set the jitter buffer size.

Value	Description
0	(Default). Jitter buffer size is set by nJitterBuffer.
1	Jitter buffer size is set by nJitterMs.

nJitterBuffer

Jitter Buffer size, in packets. Minimum = "1," maximum = "300." (Default = "1").

nJitterMs

Jitter Buffer size, in milliseconds. Minimum = "1," maximum = "3,000." (Default = "20").

bUseCompensation

Enables or disables use of the compensation jitter buffer. (Default = "0").

bCompMs

Defines the method used to set the compensation jitter buffer size.

Value	Description
0	(Default). Compensation jitter buffer size is set by <code>nCompJitterBuffer</code> .
1	Compensation jitter buffer size is set by <code>nCompJitterMs</code> .

`nCompJitterBuffer`

Compensation jitter buffer maximum size, in packets. Minimum = "0," maxi= "300." (Default = "50").

`nCompJitterMs`

Compensation jitter buffer maximum size, in milliseconds. Minimum = "0," maximum = "3,000." (Default = "1,000").

`nCompMaxDropped`

Maximum dropped consecutive packets. Minimum = "1," maximum = "100." (Default = "7").

`bUseMOS`

Enables or disables use of MOS. (Default = "0").

`bMosOnMax`

Defines whether MOS is calculated for a subset of streams or for all streams.

Value	Description
0	(Default). MOS calculation is applied to all streams.
1	MOS calculation is applied to the number of streams specified by <code>nMosMax</code> .

`nMosMaxStreams`

Maximum number of concurrent streams used in MOS calculation. Minimum = "1." (Default = "1").

`nMosInterval`

Frequency at which `IxLoad` samples the RTP streams to generate the MOS scores. Minimum = "2," maximum = "30." (Default = "3").



nDtmfDuration

Length of time allowed to play the DTMF sequence. Minimum = "60," maxi= "999." (Default = "100").

nDtmfInterdigits

Duration (in milliseconds) of the DTMF interdigit signal. Minimum = "30," maximum = "9999." (Default = "40").

bLimitDtmf

Enable or disable limitation on the number of DTMF streams to be processed. (Default = "1").

Value	Description
0	DTMF applied to all streams.
1	(Default) DTMF limited to number of streams specified by nDtmfStreams.

nDtmfStreams

Number of streams to which path confirmation will be applied. Minimum = "1" maximum = "900." (Default = "10").

nPcInterval

If Synthetic path confirmation is selected, this is the interval at which IxLoad add the synthetic RTP packets to the stream. Minimum = "1." (Default = "500").

nSessionType

Type of voice session. The choices are:

Value	Description
"0"	(default) Plays audio file specified by szAudioFile.
"1"	Perform DTMF path confirmation.
"2"	Perform synthetic DTMF path confirmation.

szDtmfSeq

DTMF sequence used for path confirmation. (Default = "12,345").

szPeerCodecName

Name of codec used by peer. (Default = {}).

szPeerCodecDetails

Details of codec used by peer. (Default = {}).

szPeerDtmfSeq

DTMF sequence used by peer. (Default = {}).

nPeerDtmfDuration

DTMF duration used by peer. (Default = "0").

nPeerDtmfInterdigits

Inter-digits interval used by peer. (Default = "0").

audioClipsTable

This is a list of type `Audio Clips Pool`. This list contains the waveform audio files that the MGCP client will play. (Default = {}).

EXAMPLE

```
$Activity_MGCPClient1 agent.pm.mediaSettings.config \  
-nPcInterval 500 \  
-nJitterBuffer 1 \  
-nDtmfInterdigits 40 \  
-nCompMaxDropped 7 \  
-nPeerDtmfDuration 0 \  
-nJitterMs 20 \  
-nAudioPoolTime 1181544691 \  
-nDtmfDuration 100 \  
-szPeerCodecName "" \  
-groupBox_MOS1 false \  
-szPeerCodecDetails "" \  

```



```
-bMosOnMax                0 \  
-groupBox_JB1             false \  
-nMosInterval             3 \  
-nCompJitterBuffer        50 \  
-bUseJitter               false \  
-szCodecName              "G711ALaw" \  
-szPeerDtmfSeq            "" \  
-bLimitDtmf              true \  
-bUseMOS                  false \  
-bJitMs                   0 \  
-szCodecDescr             "ITU-T G.711 is a standard to  
represent 8 bit compressed pulse code modulation (PCM) samples for signals of  
voice frequencies, sampled at the rate of 8000 samples/second. G.711 encoder  
will create a 64 Kbps bitstream. A-Law G.711 PCM encoder converts 13 bit  
linear PCM samples into 8 bit compressed PCM (logarithmic form) samples, and  
the decoder does the conversion vice versa." \  
-bCompMs                  0 \  
-nDtmfStreams             10 \  
-szPowerLevel             "PL_20" \  
-szDtmfSeq                "12345" \  
-nCompJitterMs           1000 \  
-nPeerDtmfInterdigits     0 \  
-nMosMaxStreams           1 \  
-szCodecDetails           "BF160PT20" \  
-nSessionType             0 \  
-bModifyPowerLevel        false \  
-bUseCompensation         false
```

SEE ALSO

MGCP Client Agent



Audio Clips Pool

Audio Clips Pool - define an audio file to be included in the list that the MGCP client will play.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
set Activity_MGCPClient1.mediaSettings.AudioClipsPool.config
```

DESCRIPTION

An Audio Clips Pool object is a part of a mediaSettings object, which is a memof a MGCP Client Agent object. Its options are configured as per the ixConfig subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard config, cget, and getOptions subcommands defined in the ixConfig command.

OPTIONS

szWaveName

Waveform audio (.wav) file. (Default = "...").

szDataFormat

Encoding format of waveform audio file. (Default = "...").

nSampleRate

Number of samples taken per second from the recording source. (Default = "0").

nResolution

Number of bits per sample. (Default = "0").

nChannels

Number of audio channels: "0" = mono, "1" = stereo. (Default = "0").

nDuration

Playing time of audio file. (Default = "0").

nSize

Size of audio file, in bytes. (Default = "0").

szRawWaveName

Name and path of wave file to be added to the list. (Default = "{}").

EXAMPLE

```
set clientTraffic [::IxLoad new ixClientTraffic options]
$clientTraffic agentList.appendItem ...
$clientTraffic agentList(0).mediaSettings.AudioClipsPool.appendItem\
    -szWaveName "meeting.wav" \
    -szDataFormat "PCM" \
    -nSampleRate "8000" \
    -nResolution "16" \
    -nChannels "1" \
    -nDuration "30" \
    -nSize "65748" \
    -szRawWaveName "/files/audio/wav/meeting.wav"
```

SEE ALSO

Media Settings



Commands

Commands—Creates the list of MGCP commands that the client will send.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.pm.scenarios.appendItem
```

DESCRIPTION

A command is added to the Scenarios object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

MGCP command to be executed. One of the following:

Command	Description
OriginateCall	Sets up a multimedia session with the specified destination.
THINK	Pause during command list processing. You should include a {Think} command whenever necessary to allow the destination to process the preceding commands. You can configure a pause of fixed length or of random length.
EndCall	Terminates the MGCP session.
ReceiveCall	Accepts a call from another endpoint.
VOICESESSION	Plays one of the waveform audio files listed in the Audio Clips Pool on the Media Settings tab. The MGCP client sends the file to the destination configured for the previous Originate Call command in the command list.

Arguments for `id = OriginateCall`

`Destination`

Destination of the call, which is usually another endpoint. If the destination is an IxLoad MGCP server agent, specify the name of the agent. (Default = "99,312,345").

Arguments for id = THINK

MinDuration

Minimum length of the pause, in milliseconds. To configure a fixed-length pause, enter the same value in this field and MaxDuration. (Default = "1").

MaxDuration

Maximum length of the pause, in milliseconds. To configure a fixed-length pause, enter the same value in this field and MinDuration. (Default = "1").

Arguments for id = ReceiveCall

RSpeakSequenceFile

File containing media description. (Default = "mgcp_speak_config").

Arguments for id = VOICESESSION

szAudioFile

Waveform audio file that will be played during the session. This must be an `szWaveName` object contained within the `Audio Clips Pool` object. (Default = "<None>").

nPlayMode

If `true`, the audio file plays for a fixed number of times. If `false`, the audio file plays continuously. (Default = "0").

nRepeatCount

If `nPlayMode` is `true`, this parameter sets the number of times that the audio file will play. (Default = "1").

nPlayTime

Length of time to play the audio file. Specify the units of time in the `nTimeUnit`.

nTimeUnit

Units of time used to set the audio file play time (`nPlayTime`). The choices are:



Value	Description
"0"	(default) Seconds
"1"	Minutes
"2"	Hours
"3"	Days

nTotalTime

(Read-only). Total length of time that the audio file will be played.

nSessionType

Type of voice session. The choices are:

Value	Description
"0"	(default) Plays audio file specified by szAudioFile.
"1"	Perform DTMF path confirmation.
"2"	Perform synthetic DTMF path confirmation.

nWavDuration

(Read-only). Length of selected audio (.wav) file.

szDtmfSeq

For a path confirmation Voice Session, (nSessionType = 1 or 2), this is the DTMF sequence. (Default = "12345").

EXAMPLE

```
$Activity_MGCPClient1 agent.pm.scenarios.appendItem \  
-id "OriginateCall" \  
-Destination "99312345" \  
$Activity_MGCPClient1 agent.pm.scenarios.appendItem \  
-id "ReceiveCall" \  
-RSpeakSequenceFile "mgcp_speak_config" \  
$Activity_MGCPClient1 agent.pm.scenarios.appendItem \  

```

```
-id "EndCall" \  
-Dummy 1
```

SEE ALSO

MGCP Client Agent



Custom Endpoint Names

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_MGCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_MGCPClient1 agent.pm.customNamesList.endpoint_names.config
```

DESCRIPTION

Retrieves the list of custom endpoint names generated by `Endpoint Names`.

SUBCOMMANDS

None.

SUB-OBJECTS

`endpoint_names`

List of custom endpoint names to be used. This is a list of `Custom Name` objects, which have the following format:

Value	Description
<code>endpoint_name</code>	Endpoint name. Default={ }.
<code>destination_number</code>	Phone number of endpoint. Default={ }.

EXAMPLE

```
$Activity_MGCPClient1 agent.pm.custom\  
    endpoint_names
```

SEE ALSO

`Endpoint Names`



MGCP Server Agent

MGCP Server Agent - create an MGCP server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_MGCPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_MGCPServer1 agent.config
```

DESCRIPTION

An MGCP server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
#####
# Activity MGCPServer1 of NetTraffic Traffic2@Network2
#####
set Activity_MGCPServer1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType                "MGCP Server" ]
$Activity_MGCPServer1 agent.config \
```

```
-enable true \  
-name "MGCPServer1"
```

SEE ALSO

`ixNetTraffic`



Parameters

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_MGCPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_MGCPServer1 agent.pm.parameters.config
```

DESCRIPTION

An MGCP server's basic parameters are set by modifying the options of the `pm.parameters` option of the `MGCP Server Agent` object.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`call_agent_name`

Call agent FQDN name that controls this gateway. (Default = "prica.ixi")

`listen_port_start`

Initial port that the agent listens on for new MGCP connections. Minimum = "1" maximum = "65,535." (Default = "2,727").

`listen_port_step`

Increment value applied initially to `listen_port_start` and to each subsequent value to create the list of listening ports. Minimum = "1." (Default = "1").

`enableTosMGCP`

Enable TOS for MGCP traffic.

Value	Description
0	(default) TOS bits not enabled.
1	TOS bits enabled.

`type_of_service_for_mgcp`

If `enableTosMGCP` is `true`, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

Value	Description
"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control

EXAMPLE

```
$Activity_MGCPServer1 agent.pm.parameters.config \  
-listen_port_stop 1 \  
-type_of_service_for_mgcp "Best Effort (0x0)" \  
-listen_port_start 2727 \  
-call_agent_name "prica.ixia-lab.com" \  
-enableTosMGCP true \  
-listen_port_step 1
```

SEE ALSO

MGCP Server Agent



Low Level Parameters

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_MGCPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_MGCPServer1 agent.pm.ll_parameters.config
```

DESCRIPTION

An MGCP server's low-level parameters are set by modifying the options of the `pm.ll_parameters` option of the MGCP Server Agent object.

SUBCOMMANDS

None.

OPTIONS

CommandTimeout

If no response to a command is received within this number of seconds, a error is declared. Minimum = "1," maximum = "120." (Default = "30").

AcknowledgeResponses

Specifies if the sent command will include the K parameter with the ID of last received response. (Default = "0").

EXAMPLE

```
$Activity_MGCPServer1 agent.pm.ll_parameters.config \  
-CommandTimeout                30 \  
-AcknowledgeResponses          true
```

SEE ALSO

MGCP Server Agent

DNS Updates

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_MGCPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_MGCPServer1 agent.pm.dns_update_parameters.config
```

DESCRIPTION

An MGCP server's DNS update parameters are set by modifying the options of the `pm.dns_update_parameters` option of the MGCP Server Agent object.

SUBCOMMANDS

None.

OPTIONS

`enable_dns_updates`

Updates a DNS server with updates to the gateway names. The DNS server must be configured to accept Update Queries from the IxLoad IP address. The first IP in range will be used to source the DNS Query packets. (Default = "0").

`dns_records`

List of DNS records to be sent to the DNS servers. This is a list of `DNS Record` objects.

`dns_record_name`

Name to be added to DNS database. (Default = {}).

`dns_record_address`

IP address to be added to DNS database. (Default = {}).

`dns_record_time_to_live`

Used for DNS update query to specify time of validity of the updated DNS record. (Default = "43,200").

`dns_source_ip`

IP address indicated as the source of the DNS records. (Default = {}).



dns_ip_port

Hostname:port number or IP address:port number of DNS server to which DNS records will be sent.
(Default = "192.168.1.1:53").

EXAMPLE

```
$Activity_MGCPServer1 agent.pm.dns_update_parameters.config \  
-dns_record_name           "" \  
-dns_record_address        "" \  
-dns_record_time_to_live   43200 \  
-dns_source_ip             "" \  
-enable_dns_updates        false \  
-dns_ip_port                "192.168.1.1:53"
```

SEE ALSO

MGCP Server Agent, DNS Record

DNS Record

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_MGCPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_MGCPServer1 agent.pm.dnsrecord.config
```

DESCRIPTION

The `DnsRecord` command is used to add DNS records to the list of records that will be sent to the DNS server to update it with changes to the gateway name.

The complete list of records is contained in the `dns_records` option of the `DNS Updates` object.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`dns_record_name`

Name to be added to DNS database. (Default = {}).

`dns_record_address`

IP address to be added to DNS database. (Default = {}).

EXAMPLE

```
$Activity_MGCPServer1 agent.pm.dns_update_parameters.config \  
-dns_record_name           " " \  
-dns_record_address        " " \  
-dns_record_time_to_live   43200 \  
-dns_source_ip             " " \  
-enable_dns_updates        false \  
-dns_ip_port                "192.168.1.1:53"
```

SEE ALSO

DNS Updates



Endpoint Names

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_MGCPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_MGCPServer1 agent.pm.endpoint_parameters.config
```

DESCRIPTION

Configures the names used for MGCP endpoints. An MGCP client's endpoint parameters are set by modifying the options of the `pm.endpoint_parameters` option of the `MGCP Server Agent` object.

SUBCOMMANDS

None.

OPTIONS

`NumberOfEndpoints`

Number of endpoints hosted by the gateway. Minimum = "1," maximum = "15,000." (Default = "2").

`UseCustomNames`

Specifies whether to use custom names or not. (Default = "0").

`EndpointNamePrefix`

Prefix applied to endpoint name. (Default = "aaln/").

`EndpointNameSuffix`

Suffix applied to endpoint name. (Default = {}).

`EndpointNameStartAt`

Initial value of variable portion of endpoint name. Minimum = "0," maximum = "4,294,967,295." (Default = "0").

`EndpointNameExpand`

`On`

Width of variable used to create endpoint names that are unique within a gate way. Minimum = "1," maximum = "5." (Default = "1").



EndpointNameStep

Amount of increase in the variable (the Endpoint Name Expand On parameter) used to create unique base endpoint names. Minimum = "1," maximum = "3,000." (Default = "1").

EndpointPhonePrefix

String containing digits to be used at beginning of phone number. (Default = {}).

EndpointPhoneSuffix

String containing digits to be used at the end of the phone number. (Default = {}).

EndpointPhoneStartAt

Initial value of variable portion of phone number. Minimum = "0," maximum = "4,294,967,295." (Default = "1,000").

EndpointPhoneStep

Amount of increase in variable to create additional phone numbers. Minimum = "1," maximum = "3000." (Default = "1").

NumGateways

Number of gateways. Minimum = "1," maximum = "3,000." (Default = "2").

GatewayNamePrefix

String prefixed to gateway name. (Default = "ix").

GatewayNameSuffix

String suffixed to gateway name. (Default = ".ixia-lab.com").

GatewayNameStartAt

Initial value of variable portion of gateway name. Minimum = "0," maximum = "4,294,967,295." (Default = "3,000").

GatewayNameExpand On

Width of variable used to create unique gateway names. Minimum = "1," maximum = "5," (Default = "1").

GatewayNameStep

Amount of increase in variable used for gateway name. Minimum = "1," maxi= "3,000." (Default = "1").

EXAMPLE

```
$Activity_MGCPServer1 agent.pm.endpoint_parameters.config \  
-NumGateways 4 \  
-EndpointPhonePrefix "" \  
-EndpointNameSuffix "" \  
-EndpointPhoneStartAt 1000 \  
-EndpointNameExpandOn 1 \  
-GatewayNamePrefix "ix" \  
-NumberOfEndpoints 2 \  
-GatewayNameStep 1 \  
-EndpointNameStartAt 0 \  
-EndpointNameStep 1 \  
-EndpointPhoneStep 1 \  
-GatewayNameStartAt 3000 \  
-UseCustomNames false \  
-EndpointPhoneSuffix "" \  
-EndpointNamePrefix "aaln/" \  
-GatewayNameSuffix ".ixia-lab.com" \  
-GatewayNameExpandOn 1
```

SEE ALSO

MGCP Server Agent



Custom Endpoint Names

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_MGCPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$set Activity_MGCPServer1 agentList(0).pm.customNamesList \
    endpoint_names
```

DESCRIPTION

Retrieves the list of custom endpoint names generated by `Endpoint Names`.

SUBCOMMANDS

None.

OPTIONS

`endpoint_names`

List of custom endpoint names to be used. This is a list of `Custom Name` objects, which have the following format:

Value	Description
<code>endpoint_name</code>	Endpoint name. Default={ }.
<code>destination_number</code>	Phone number of endpoint. Default={ }.

EXAMPLE

```
$set Activity_MGCPServer1 agentList(0).pm.customNamesList\
    endpoint_names
```

SEE ALSO

`Endpoint Names`

Bulk MGCP Statistics

For Bulk MGCP statistics, see the following:

Bulk MGCP Client Statistics (on page 869)

Bulk MGCP Server Statistics (on page 874)

Bulk MGCP Client Statistics

The following table describes the Bulk MGCP Client statistics.

Statistic	Description
Objectives Statistics	
MGCP Simulated Users	Number of MGCP users simulated during the test.
MGCP connections initiated	Number of MGCP connections initiated during the test.
MGCP connections completed	Number of MGCP connections successfully completed during the test.
MGCP connections active	Number of MGCP connections active.
MGCP connections failed busy	Number of MGCP connections that failed because they received a Busy tone.
MGCP connections failed reorder	Number of initiated MGCP connections that failed because they received a Reorder tone.
Total Message Statistics	
Total MGCP Messages Sent	Total number of MGCP messages sent by the client.
Total MGCP Messages Received	Total number of MGCP messages received by the client.
Total MGCP Messages Malformed	Total number of malformed MGCP messages received by the client.
NTFY messages	
NTFY sent	Number of NTFY (Notify) messages sent by the client.
NTFY recv	Number of NTFY (Notify) messages received by the client.
NTFY success	Number of NTFY (Notify) messages sent by the client that succeeded.
NTFY failed	Number of NTFY (Notify) messages sent by the client that failed.
NTFY dial sent	Number of NTFY (Notify) dialing messages sent by the client.
NTFY dial recv	Number of NTFY (Notify) dialing messages received by the client.
NTFY dial success	Number of NTFY (Notify) dialing messages sent by the client that succeeded.



NTFY dial failed	Number of NTFY (Notify) dialing messages sent by the client that failed.
CRCX messages	
CRCX sent	Number of CRCX (Create Connection) messages sent by the client.
CRCX rcv	Number of CRCX (Create Connection) messages received by the client.
CRCX success	Number of CRCX (Create Connection) messages sent by the client that succeeded.
CRCX failed	Number of CRCX (Create Connection) messages sent by the client that failed.
MDCX messages	
MDCX sent	Number of MDCX (Modify Connection) messages sent by the client.
MDCX rcv	Number of MDCX (Modify Connection) messages received by the client.
MDCX success	Number of MDCX (Modify Connection) messages sent by the client that succeeded.
MDCX failed	Number of MDCX (Modify Connection) messages sent by the client that failed.
DLCX messages	
DLCX sent	Number of DLCX (Delete Connection) messages sent by the client.
DLCX rcv	Number of DLCX (Delete Connection) messages received by the client.
DLCX success	Number of DLCX (Delete Connection) messages sent by the client that succeeded.
DLCX failed	Number of DLCX (Delete Connection) messages sent by the client that failed.
RQNT messages	
RQNT sent	Number of RQNT (Notification Request) messages sent by the client.
RQNT rcv	Number of RQNT (Notification Request) messages received by the client.
RQNT success	Number of RQNT (Notification Request) messages sent by the client that succeeded.
RQNT failed	Number of RQNT (Notification Request) messages sent by the client that failed.
AUEP messages	
AUEP sent	Number of AUEP (Audit Endpoint) messages sent by the client.
AUEP rcv	Number of AUEP (Audit Endpoint) messages received by the client.

AUEP success	Number of AUEP (Audit Endpoint) messages sent by the client that succeeded.
AUEP failed	Number of AUEP (Audit Endpoint) messages sent by the client that failed.
AUCX messages	
AUCX sent	Number of AUCX (Audit Connection) messages sent by the client.
AUCX rcv	Number of AUCX (Audit Connection) messages received by the client.
AUCX success	Number of AUCX (Audit Connection) messages sent by the client that succeeded.
AUCX failed	Number of AUCX (Audit Connection) messages sent by the client that failed.
EPCF messages	
EPCF sent	Number of EPCF (Endpoint Configuration) messages sent by the client.
EPCF rcv	Number of EPCF (Endpoint Configuration) messages received by the client.
EPCF success	Number of EPCF (Endpoint Configuration) messages sent by the client that succeeded.
EPCF failed	Number of EPCF (Endpoint Configuration) messages sent by the client that failed.
RSIP messages	
RSIP sent	Number of RSIP (Restart in Progress) messages sent by the client.
RSIP rcv	Number of RSIP (Restart in Progress) messages received by the client.
RSIP success	Number of RSIP (Restart in Progress) messages sent by the client that succeeded.
RSIP failed	Number of RSIP (Restart in Progress) messages sent by the client that failed.
1XX Responses	
Responses_1XX sent	Number of 100-series responses sent by the client. 100-series responses indicate provisional responses.
Responses_1XX rcv	Number of 100-series responses received by the client.
2XX Responses	
Responses_2XX sent	Number of 200-series responses sent by the client. 200-series responses indicate successful completion.
Responses_2XX rcv	Number of 200-series responses received by the client.
XX Responses	
Responses_3XX sent	Number of 300-series responses sent by the client.
Responses_3XX rcv	Number of 300-series responses received by the client.



4XX Responses	
Responses_4XX sent	Number of 400-series responses sent by the client. 400-series responses indicate a transient error.
Responses_4XX rcv	Number of 400-series responses received by the client.
5XX Responses	
Responses_5XX sent	Number of 500-series responses sent by the client. 500-series responses indicate a permanent error.
Responses_5XX rcv	Number of 500-series responses received by the client.
RTP: Global Stream Transmit Statistics	
RTP Bytes Sent	Total number of bytes sent, including header and payload.
RTP Packets Sent	Total number of packets sent.
RTP Tx Jitter (ns)	Average amount of transmit jitter, in nanoseconds.
RTP Tx Packets Dropped	Number of packets transmitted by the client that were dropped.
RTP: Global Stream Statistics	
RTP Dropped Packets	Number of RTP packets dropped.
RTP Bytes Received	Number of RTP bytes received.
RTP Packets Received	Number of RTP packets received.
RTP Payload Bytes Received	Number bytes received in RTP payloads.
RTP Bad Packets Received	Number of defective RTP packets received.
RTP Lost Packets	Number of packets lost.
RTP Misordered Packets Received	Number of packets received out of order.
RTP Duplicate Packets Received	Number of duplicate packets received.
RTP Jitter Min	Smallest amount of jitter detected.
RTP Jitter Max	Largest amount of jitter detected.
RTP Packets With Jitter Up To 1ms	Packets received with jitter of up to 1ms.
RTP Packets With Jitter Up To 3ms	Packets received with jitter of 1-3ms.
RTP Packets With Jitter Up To 5ms	Packets received with jitter of 3-5ms.
RTP Packets With Jitter Up To 10ms	Packets received with jitter of 5-10ms.
RTP Packets With Jitter Up To 20ms	Packets received with jitter of 10-20ms.
RTP Packets With Jitter Up To 40ms	Packets received with jitter of 20-40ms.
RTP Packets With Jitter More Than 40ms	Packets received with jitter of more than 40ms.

RTP DTMFs Detected	Total number of path confirmation DTMF tone sequences sent.
RTP Good DTMF Sequences Detected	Total number of correct path confirmation DTMF tone sequences received.
RTP Bad DTMF Sequences Detected	Total number of incorrect path confirmation DTMF tone sequences received.
RTP Packets Dropped By Jitter Buffer	Number of packet dropped from the jitter buffer because they arrived later than expected.
Note: In the CSV files, global MOS scores are represented as whole numbers (for example, "345"); in StatViewer (they are represented as floating-point numbers (for example, "3.45").	
RTP MOS Average Instant	Average MOS score at the time of the sampling interval.
RTP MOS Worst Instant	Lowest MOS score at the time of the sampling interval.
RTP MOS Best Instant	Highest MOS score at the time of the sampling interval.
RTP MOS Worst	Lowest MOS score recorded during the test.
RTP MOS Best	Highest MOS score recorded during the test.
RTP MOS Average Per Call	Average MOS score per call.
RTP MOS Worst Per Call	Lowest MOS score per call.
RTP MOS Best Per Call	Highest MOS score per call.
RTP Calls With Continuous Path Confirmation	Number of calls on which path confirmation continued throughtout the call.
RTP Calls With Interrupted Path Confirmation	Number of calls on which path confirmation was interrupted during the call.
RTP Calls Without Path Confirmation	Number of calls on which there was no path confirmation.
Transport Statistics	
MGCP Bytes Transmitted	Number of MGCP bytes transmitted.
MGCP Bytes Received	Number of MGCP bytes received.
MGCP Signaling UDP Packets Transmitted	Number of UDP packets containing MGCP signaling bytes transmitted.
MGCP Signaling UDP Packets Received	Number of UDP packets containing MGCP signaling bytes received.
Per-Stream Statistics	
RTP Path Confirmation Status	Status of path confirmation on the stream. Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
RTP MOS	Average MOS score recorded on the stream.
RTP Worst MOS	Lowest MOS score recorded on the stream.
RTP Best MOS	Highest MOS score recorded on the stream.
RTP Bytes	Number of bytes transmitted on the stream.
RTP Packets	Number of packets transmitted on the stream.



RTP Bad Packets	Number of bad packets transmitted on the stream.
RTP Lost Packets	Number of packets lost on the stream.
RTP Missorder Packets	Number of packets received out of order on the stream.
RTP Duplicate Packets	Number of duplicate packets received on the stream.
RTP Packets With Jitter Up To 1ms	Number of packets received on the stream with jitter up to 1 millisecond.
RTP Packets With Jitter Up To 3ms	Number of packets received on the stream with jitter up to 3 milliseconds.
RTP Packets With Jitter Up To 5ms	Number of packets received on the stream with jitter up to 5 milliseconds.
RTP Packets With Jitter Up To 10ms	Number of packets received on the stream with jitter up to 10 milliseconds.
RTP Packets With Jitter Up To 20ms	Number of packets received on the stream with jitter up to 20 milliseconds.
RTP Packets With Jitter Up To 40ms	Number of packets received on the stream with jitter up to 40 milliseconds.
RTP Packets With Jitter More Than 40ms	Number of packets received on the stream with jitter over 40 milliseconds.
RTP Average Jitter (ns)	Average jitter, in nanoseconds.
RTP Min Jitter (ns)	Lowest jitter recorded, in nanoseconds.
RTP Max Jitter (ns)	Largest jitter recorded, in nanoseconds.
RTP DTMFs Detected	Total number of path confirmation DTMF tone sequences sent.
RTP Good DTMF Sequences Detected	Total number of correct path confirmation DTMF tone sequences received.
RTP Bad DTMF Sequences Detected	Total number of incorrect path confirmation DTMF tone sequences received.
RTP Packets Dropped By Jitter Buffer	Total number of packets dropped from the jitter buffer because they were received late.

Bulk MGCP Server Statistics

The following table describes the Bulk MGCP Server statistics.

Statistic	Description
Total MGCP Commands Received	Total number of MGCP commands received by the server

Total MGCP Responses Received	Total number of MGCP responses received by the server.
Total MGCP Commands Sent	Total number of MGCP commands sent by the server.
Total MGCP Responses Sent	Total number of MGCP responses sent by the server.
Total MGCP Received Malformed Messages	Total number of malformed MGCP messages received by the server.
NTFY messages	
NTFY sent	Number of NTFY (Notify) messages sent by the server.
NTFY rcv	Number of NTFY (Notify) messages received by the server.
NTFY success	Number of NTFY (Notify) messages sent by the server that succeeded.
NTFY failed	Number of NTFY (Notify) messages sent by the server that failed.
NTFY dial sent	Number of NTFY (Notify) dialing messages sent by the server.
NTFY dial rcv	Number of NTFY (Notify) dialing messages received by the server.
NTFY dial success	Number of NTFY (Notify) dialing messages sent by the server that succeeded.
NTFY dial failed	Number of NTFY (Notify) dialing messages sent by the server that failed.
CRCX messages	
CRCX sent	Number of CRCX (Create Connection) messages sent by the server.
CRCX rcv	Number of CRCX (Create Connection) messages received by the server.
CRCX success	Number of CRCX (Create Connection) messages sent by the server that succeeded.
CRCX failed	Number of CRCX (Create Connection) messages sent by the server that failed.
MDCX messages	
MDCX sent	Number of MDCX (Modify Connection) messages sent by the server.
MDCX rcv	Number of MDCX (Modify Connection) messages received by the server.
MDCX success	Number of MDCX (Modify Connection) messages sent by the server that succeeded.
MDCX failed	Number of MDCX (Modify Connection) messages sent by the server that failed.
DLCX messages	



DLCX sent	Number of DLCX (Delete Connection) messages sent by the server.
DLCX recv	Number of DLCX (Delete Connection) messages received by the server.
DLCX success	Number of DLCX (Delete Connection) messages sent by the server that succeeded.
DLCX failed	Number of DLCX (Delete Connection) messages sent by the server that failed.
RQNT messages	
RQNT sent	Number of RQNT (Notification Request) messages sent by the server.
RQNT recv	Number of RQNT (Notification Request) messages received by the server.
RQNT success	Number of RQNT (Notification Request) messages sent by the server that succeeded.
RQNT failed	Number of RQNT (Notification Request) messages sent by the server that failed.
AUEP messages	
AUEP sent	Number of AUEP (Audit Endpoint) messages sent by the server.
AUEP recv	Number of AUEP (Audit Endpoint) messages received by the server.
AUEP success	Number of AUEP (Audit Endpoint) messages sent by the server that succeeded.
AUEP failed	Number of AUEP (Audit Endpoint) messages sent by the server that failed.
AUCX messages	
AUCX sent	Number of AUCX (Audit Connection) messages sent by the server.
AUCX recv	Number of AUCX (Audit Connection) messages received by the server.
AUCX success	Number of AUCX (Audit Connection) messages sent by the server that succeeded.
AUCX failed	Number of AUCX (Audit Connection) messages sent by the server that failed.
EPCF messages	
EPCF sent	Number of EPCF (Endpoint Configuration) messages sent by the server.
EPCF recv	Number of EPCF (Endpoint Configuration) messages received by the server.
EPCF success	Number of EPCF (Endpoint Configuration) messages sent by the server that succeeded.

EPCF failed	Number of EPCF (Endpoint Configuration) messages sent by the server that failed.
RSIP messages	
RSIP sent	Number of RSIP (Restart in Progress) messages sent by the server.
RSIP rcv	Number of RSIP (Restart in Progress) messages received by the server.
RSIP success	Number of RSIP (Restart in Progress) messages sent by the server that succeeded.
RSIP failed	Number of RSIP (Restart in Progress) messages sent by the server that failed.
1XX Responses	
Responses_1XX sent	Number of 100-series responses sent by the server. 100-series responses indicate provisional responses.
Responses_1XX rcv	Number of 100-series responses received by the server.
2XX Responses	
Responses_2XX sent	Number of 200-series responses sent by the server. 200-series responses indicate successful completion.
Responses_2XX rcv	Number of 200-series responses received by the server.
3XX Responses	
Responses_3XX sent	Number of 300-series responses sent by the server.
Responses_3XX rcv	Number of 300-series responses received by the server.
4XX Responses	
Responses_4XX sent	Number of 400-series responses sent by the server. 400-series responses indicate a transient error.
Responses_4XX rcv	Number of 400-series responses received by the server.
5XX Responses	
Responses_5XX sent	Number of 500-series responses sent by the server. 500-series responses indicate a permanent error.
Responses_5XX rcv	Number of 500-series responses received by the server.
Transport Statistics	
MGCP Signaling Bytes Transmitted	Number of MGCP signaling bytes transmitted.
MGCP Signaling Bytes Received	Number of MGCP signaling bytes received.
MGCP Signaling UDP Packets Transmitted	Number of UDP packets containing MGCP signaling bytes transmitted.



MGCP Signaling UDP Packets Received	Number of UDP packets containing MGCP signaling bytes received.
Objectives Statistics	
MGCP Simulated Users	Number of MGCP users simulated during the test.
MGCP connections initiated	Number of MGCP connections initiated during the test.
MGCP connections completed	Number of MGCP connections successfully completed during the test.
MGCP connections active	Number of MGCP connections active.
MGCP connections failed busy	Number of MGCP connections that failed because they received a Busy tone.
MGCP connections failed reorder	Number of initiated MGCP connections that failed because they received a Reorder tone.

10

Bulk SIP

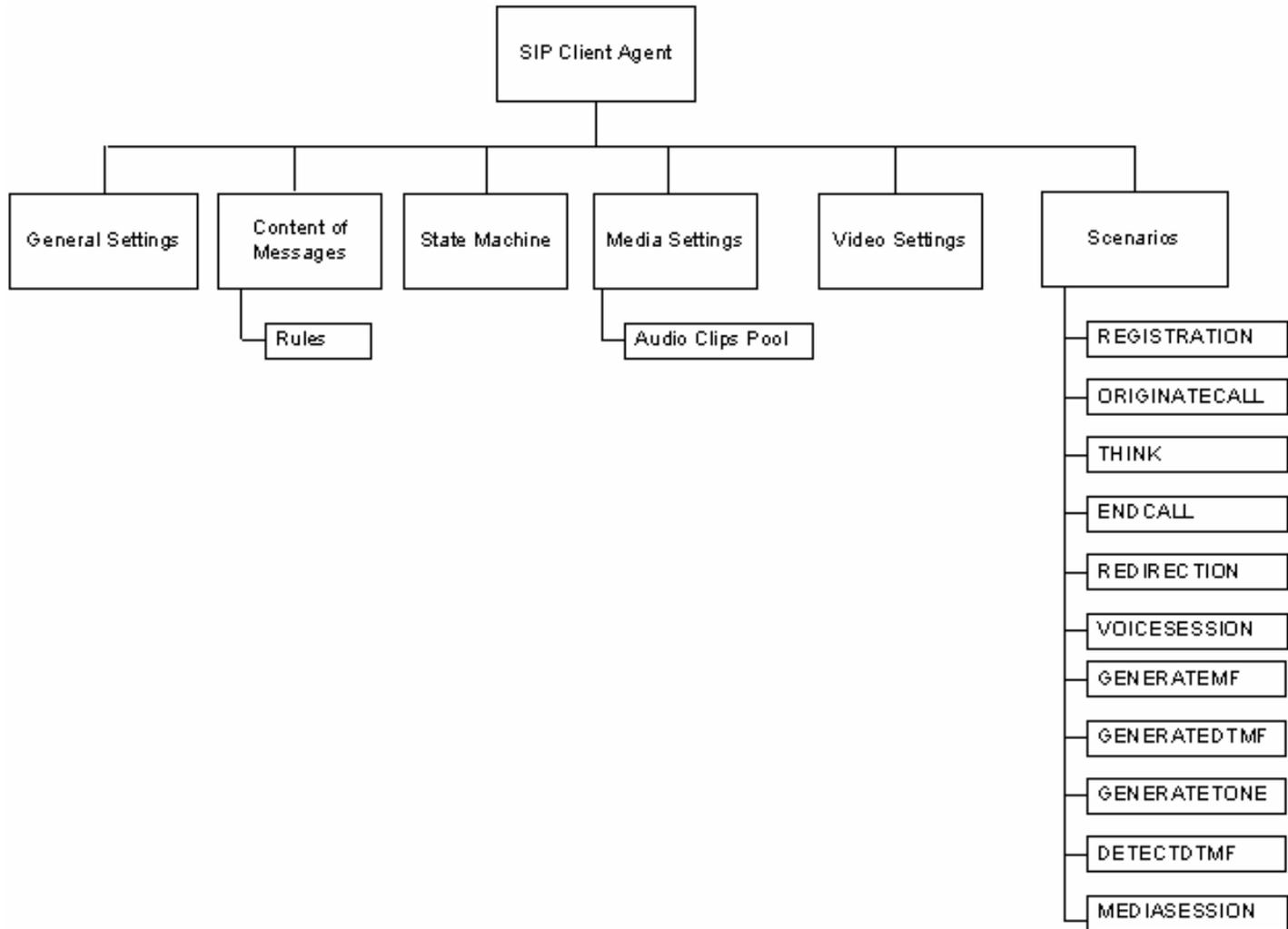
This section describes the SIP Tcl API objects.

In this chapter:

Overview	880
Objectives	880
SIP Client Commands	881
SIP Server Commands	886
SIP Client Agent	889
SIP Server Agent	932
Using Variables in SIP Fields	948
Bulk SIP Statistics	950

Overview

The IxLoad SIP API consists of a client agent, a server agent, and their com



Objectives

The objectives (userObjective) you can set for SIP are listed below. Test objecare set in the ixTimeline object.

- simulatedUsers
- useragents
- transactionRate
- bhca
- callsPerSec (displays as “Calls Initiated Per Second” in the GUI)
- registrationsinitiated (displays as “Registrations Initiated Per Second” in the GUI)
- redirectionsinitiated (displays as “Redirections Initiated Per Second” in the GUI)

SIP Client Commands

This section describes the SIP client Tcl API objects.

SIP Client Agent

The SIP Client Agent command defines a simulated user using SIP to establish and terminate sessions SIP. Refer to `SIP Client Agent` for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time .
protocol	Protocol used by the client agent.
type	Defines the agent as either a client or server.

General Settings

The SIP Client Agent General Settings command sets the SIP client agent's genconfiguration options. Refer to `General Settings` for a full description of this command. The most significant options of this command are listed below.

Option	Description
szAuthUsername	User name to be registered with registrar.
szAuthPassword	Password to be registered with registrar.
szAuthDomain	Domain to be registered with registrar.
szTransport	Type of transport to be used.
nUdpPort	Port number to be used for sending and receiving SIP mesover UDP.
nTcpPort	Port number to be used for sending and receiving SIP mesover TCP.
nUdpMaxSize	Maximum size, in Kb, of a SIP message that will be sent.



Option	Description
szRegistrar	Host name or IP address and port number of registrar.
bRegBefore	If <code>true</code> , before starting the Originate Call/EnCall --> Receive call process, the IxLoad SIP client registers with the proxy server.
enableTosSIP	Enables the setting of the TOS (Type of Service) bits in the header of the SIP packets.
enableTosRtp	Enables the setting of the TOS (Type of Service) bits in the header of the RTP data packets.
type_of_service_for_sip	IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for the SIP packets.
type_of_service_for_rtp	IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for RTP data packets.

Content of Messages

The SIP Client Agent Content of Messages command specifies the content of the SIP messages sent by the client. Refer to `Content of Messages` for a full description of this command. The most significant options of this command are listed below.

Option	Description
bRoute	If <code>true</code> , IxLoad inserts a Route header field into the SIP message.
szRoute	If <code>bRoute</code> is <code>true</code> , this parameter specifies the Route header field used to force the request to follow a fixed route through a listed set of proxies.
bCompact	If <code>true</code> , IxLoad uses the compact forms of the SIP header field notations.
bFolding	If <code>true</code> , the VIA field spans two lines. Some SIP devices may not be able to handle this.
bScattered	If <code>true</code> , IxLoad moves the header fields around in the message order to make it more difficult for the DUT to decode the message.
bAdvisable	If <code>true</code> , the SIP request includes the header fields that are defined as 'mandatory' by the SIP RFC (RFC 3261), plus those that are recommended as 'advisable.'
bOptional	If <code>true</code> , the SIP request includes the header fields that are defined as 'mandatory' by the SIP RFC (RFC 3261), plus those that are listed as 'optional.'
bBestPerformance	If <code>true</code> , IxLoad inserts the headers into the message so that the message can be processed as quickly as possible by the receiving system.
szREQUESTURI	User or service to which the SIP request is being addressed.
szFROM	Initiator of the SIP request.

Option	Description
szTO	Logical recipient of the request.
szCONTACT	Contact header field.
rulesTable	Rules defining how this message will be handled.

Rules

The SIP Client Agent Rules command defines a rule for handling a SIP message. Refer to Rules for a full description of this command. The most significant options of this command are listed below.

Option	Description
szMessage	Type of message the rule will apply to.
szAction	Action that rule performs.
szValue	Numerical value for the szAction.

State Machine

The SIP Client Agent State Machine command configures the SIP client agent's internal timers and other parameters of its state machine. Refer to State Machine for a full description of this command. The most significant options of this comare listed below.

Option	Description
nTimersT1	Estimate of the round-trip time (RTT).
nTimersT2	Maximum retransmit interval, in milliseconds (ms), for non-INVITE requests and INVITE responses.
nTimersT4	Maximum length of time, in milliseconds (ms), that a message will remain in the network.
nTimersTC	Proxy INVITE transaction timeout.
nTimersTD	Wait time for response retransmits.
bUseTimer	If <code>true</code> , IxLoad enforces a timeout limit on transactions.
nTimeout	Transaction timeout interval.
bRecv5xx	If <code>true</code> and IxLoad receives a 5xx series response to a transaction, IxLoad marks it as a failed transaction, and increments the transfailure statistics.
nReRegDuration	In the event that IxLoad fails to register with a registrar, this field defines the amount of time allowed to re-registration.
bNextOnFail	If <code>true</code> and IxLoad encounters a transaction failure, it continues processing SIP requests.



Media Settings

The SIP Client Agent Media Settings command selects and configures the streaming audio files for the multimedia session that the client will play over RTP. Refer to `Media Settings` for a full description of this command. The most significant options of this command are listed below.

Option	Description
szCodecName	Codec to be used to encode waveform audio files listed in the Audio Clips Pool .
szCodecDetails	Displays the properties of the codec such as the number of bytes per frame of compressed audio, and the rate at which packets are sent over the connection.
szCodecDescr	Codec description.
bModifyPowerLevel	If <code>true</code> , <code>lXLoad</code> modifies the volume of the compressed audio.
szPowerLevel	If <code>bModifyPowerLevel</code> is <code>true</code> , this parameter specifies the amount of gain (volume) added to compressed audio.
bUseJitter	Enables or disables use of the jitter buffer.
bJitMs	Defines the method used to set the jitter buffer size.
nJitterBuffer	Jitter Buffer size, in packets.
nJitterMs	Jitter Buffer size, in milliseconds.
bUseCompensation	Enables or disables use of the compensation jitter buffer.
bCompMs	Defines the method used to set the compensation jitter buffer size.
nCompJitterBuffer	Compensation jitter buffer maximum size, in packets.
nCompJitterMs	Compensation jitter buffer maximum size, in milliseconds.
nCompMaxDropped	Maximum dropped consecutive packets.
bUseMOS	Enables or disables use of MOS.
bMosOnMax	Defines whether MOS is calculated for a subset of streams or for all streams.
nMosMaxStreams	Maximum number of concurrent streams used in MOS calcu
nMosInterval	Frequency at which <code>lXLoad</code> samples the RTP streams to genthe MOS scores.
nDtmfDuration	Length of time allowed to play the DTMF sequence.
nDtmfInterdigits	Duration (in milliseconds) of the DTMF interdigit signal.
bLimitDtmf	Enable or disable limitation on the number of DTMF streams to be processed.
nDtmfStreams	Number of streams to which path confirmation will be applied.
nPcInterval	If Synthetic path confirmation is selected, this is the interval at which <code>lXLoad</code> add the synthetic RTP packets to the stream.
nSessionType	Type of voice session.

Option	Description
szDtmfSeq	DTMF sequence used for path confirmation.
szPeerCodecName	Name of codec used by peer.
szPeerCodecDetails	Details of codec used by peer.
szPeerDtmfSeq	DTMF sequence used by peer.
nPeerDtmfDuration	DTMF duration used by peer.
nPeerDtmfInterdigit s	Inter-digits interval used by peer.
audioClipsTable	This list contains the waveform audio files that the SIP client will play.

Audio Clips Pool

The SIP Client Agent Audio Clips Pool defines an audio file to be included in the list that the SIP client will play. Refer to `Audio Clips Pool` for a full description of this command. The most significant options of this command are listed below.

Option	Description
szWaveName	Waveform audio (.wav) file.
szDataFormat	Encoding format of waveform audio file.
nSampleRate	Number of samples taken per second from the recording source.
nResolution	Number of bits per sample.
nChannels	Number of audio channels.
nDuration	Playing time of audio file.
nSize	Size of audio file, in bytes.
szRawWaveName	Name and path of wave file to be added to the list.

Video Settings

The Video Settings tab defines the controls that you can use to define the paramof the synthetic video the SIP server generates for a `MEDIASESSION sceRefer to Video Settings for a full description of this command. The most significant options of this command are listed below.`

Option	Description
videoBitrate	Bit rate of generated (synthetic) video data.
videoBitrateLimit	The videoBitrate limit in Kbps.

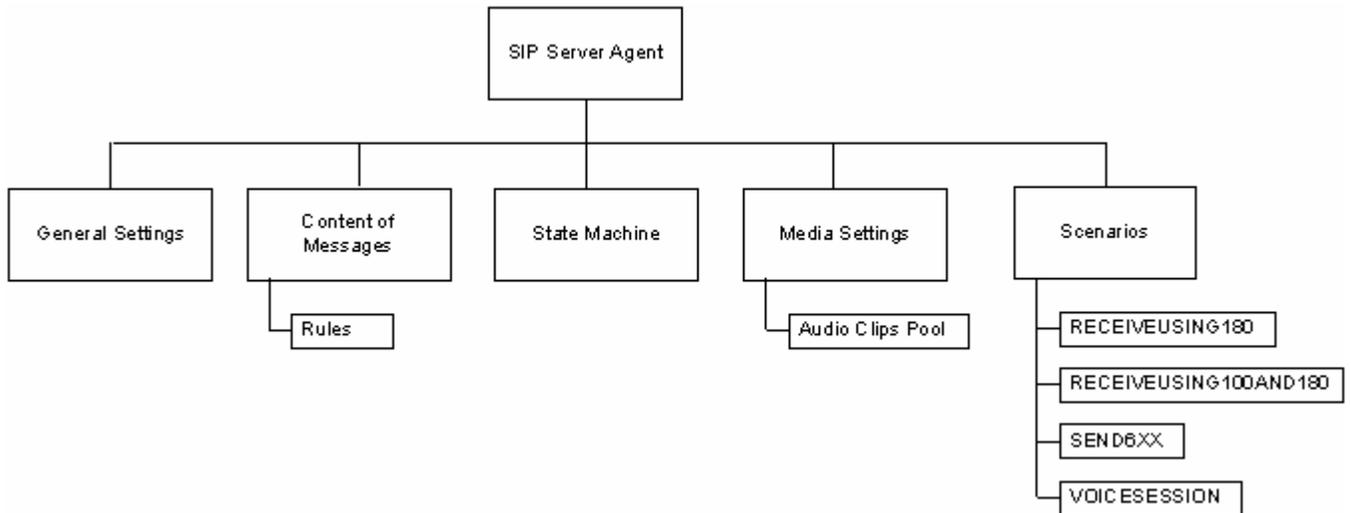
Scenarios

The SIP Client Agent Scenarios is the list of SIP commands that the client will send to a SIP server. Refer to `Scenarios` for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	SIP command to be executed.

SIP Server Commands

The structure of the SIP server API is shown below.



SIP Server Agent

The SIP Server Agent command defines a simulated user using SIP to establish and terminate sessions SIP. Refer to `SIP Server Agent` for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time.
protocol	Protocol used by the client agent.

Option	Description
type	Defines the agent as either a client or server.

General Settings

The SIP Server Agent General Settings command sets the SIP server agent's genconfiguration options. The options for this command are similar to those for the SIP client agent.

Content of Messages

The SIP Server Agent Content of Messages command specifies the content of the SIP messages sent by the server. The options for this command are similar to those for the SIP client agent.

Rules

The SIP Server Agent Rules command defines a rule for handling a SIP message. The options for this command are similar to those for the SIP client agent.

State Machine

The SIP Server Agent State Machine command configures the SIP server agent's internal timers and other parameters of its state machine. The options for this command are similar to those for the SIP client agent.

Media Settings

The SIP Server Agent Media Settings command selects and configures the streaming audio files for the multimedia session that the server will play over RTP. The options for this command are similar to those for the SIP client agent.

Audio Clips Pool

The SIP Server Agent Audio Clips Pool defines an audio file to be included in the list that the SIP server will play. The options for this command are similar to those for the SIP client agent.



Scenarios

The SIP Server Agent Scenarios is the list of SIP commands that the server will send to a SIP client. The options for this command are similar to those for the SIP client agent.

SIP Client Agent

SIP Client Agent - create a SIP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.config
```

DESCRIPTION

An SIP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity SIPClient1 of NetTraffic Traffic1@Network1
#####
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem \
    -protocolAndType                "SIP Client" ]
```



```
#####  
# Timeline1 for activities SIPClient1  
#####  
set Timeline1 [::IxLoad new ixTimeline]  
$Timeline1 config \  
    -rampUpValue          1 \  
    -rampUpType           0 \  
    -offlineTime          0 \  
    -rampDownTime         20 \  
    -standbyTime          0 \  
    -iterations           1 \  
    -rampUpInterval       1 \  
    -sustainTime          20 \  
    -timelineType         0 \  
    -name                  "Timeline1"  
$Activity_SIPClient1 config \  
    -enable                true \  
    -name                  "SIPClient1" \  
    -enableConstraint      false \  
    -userObjectiveValue    100 \  
    -constraintValue       100 \  
    -userObjectiveType     "useragents" \  
    -timeline              $Timeline1  
$Activity_SIPClient1 agent.config \  
    -enable                true \  
    -name                  "SIPClient1"  
$Activity_SIPClient1 agent.pm.generalSettings.config \  
    -dhcpServerPort       5060 \  

```

```
-ipv6Form 0 \  
-bRemoveCredent false \  
-bRegBefore false \  
-type_of_service_for_rtp "Best Effort (0x0)" \  
-_gbDhcpServerPort false \  
-nUdpMaxSize 1024 \  
-nUdpPort 5060 \  
-szAuthDomain "domain\[0000-\  
-vlan_priority_sip 0 \  
-useDhcp false \  
-enableTosSIP false \  
-implicitLoopCheck true \  
-ipPreference 0 \  
-_gbIpPreference false \  
-nPrefQop 0 \  
-szRegistrar "127.0.0.1:5060" \  
-enableVlanPriority_for_sip false \  
-nTcpPort 5060 \  
-szTransport "UDP" \  
-szAuthPassword "password\[0000-\  
-type_of_service_for_sip "Best Effort (0x0)" \  
-szAuthUsername "user\[0000-\  
-enableTosRTP false \  
-compressZeros false  
$Activity_SIPClient1 agent.pm.mediaSettings.config \  
-nPcInterval 500 \  
-nJitterBuffer 1 \  
-nDtmfInterdigits 40 \  

```



```
-nCompMaxDropped          7 \  
-nPeerDtmfDuration        0 \  
-nJitterMs                20 \  
-bSilenceMode             1 \  
-nAudioPoolTime           1178615586 \  
-szBitRate                 "64 kbps" \  
-nDtmfDuration            100 \  
-szPeerCodecName          "" \  
-szSilenceFile            "" \  
-bytesPerFrameBuffer      "" \  
-groupBox_MOS1            false \  
-szPeerCodecDetails       "" \  
-bMosOnMax                 0 \  
-groupBox_JB1             false \  
-nMosInterval             3 \  
-nCompJitterBuffer        50 \  
-bUseJitter               false \  
-szCodecName               "G711ALaw" \  
-szPeerDtmfSeq            "" \  
-bLimitDtmf               true \  
-bUseMOS                  false \  
-bJitMs                   0 \  
-szCodecDescr              "ITU-T G.711 is a standard to  
represent 8 bit compressed pulse code modulation (PCM) samples for signals of  
voice frequencies, sampled at the rate of 8000 samples/second. G.711 encoder  
will create a 64 Kbps bitstream. A-Law G.711 PCM encoder converts 13 bit  
linear PCM samples into 8 bit compressed PCM (logarithmic form) samples, and  
the decoder does the conversion vice versa." \  
-bCompMs                   0 \  
-nDtmfStreams              10 \  
-packetTimeBuffer         "" \  

```

```
-szPowerLevel                "PL_20" \  
-szDtmfSeq                   "12345" \  
-nCompJitterMs               1000 \  
-nPeerDtmfInterdigits        0 \  
-bRtpStartCollector           false \  
-nMosMaxStreams               1 \  
-szCodecDetails               "BF160PT20" \  
-nSessionType                 0 \  
-bUseSilence                  false \  
-bModifyPowerLevel            false \  
-bUseCompensation             false  
  
$Activity_SIPClient1 agent.pm.contentOfMessages.config \  
-bFolding                     false \  
-bBestPerformance             1 \  
-szRoute                      "Route:  
<sip:p1.example.com;lr>,<sip:p2.domain.com;lr>" \  
-bOptional                    false \  
-szCONTACT                    "<sip:id\[00000-\]@IP>" \  
-bAdvisable                   false \  
-bCompact                    false \  
-bRoute                       false \  
-szFROM                       "<sip:id\[00000-\]@IP>" \  
-szTO                         "<sip:id\[50000-\]@IP>" \  
-szREQUESTURI                 "sip:id\[50000-\]@IP" \  
-bScattered                   false  
  
$Activity_SIPClient1 agent.pm.contentOfMessages.rulesTable.clear  
  
$Activity_SIPClient1 agent.pm.stateMachine.config \  
-bNextOnFail                  true \  
-nTimersT4                    5000 \  

```



```
-nReRegDuration          0 \  
-nTimersT1              500 \  
-nTimersT2             4000 \  
-bUseTimer              false \  
-nTimeout               30000 \  
-nTimersTD              32000 \  
-nTimersTC             180000 \  
-bRecv5xx               false  
  
$Activity_SIPClient1 agent.pm.videoSettings.config \  
-videoBitrate           128.0 \  
-videoBitrateLimit      0  
  
$Activity_SIPClient1 agent.pm.scenarios.clear  
  
$Activity_SIPClient1 agent.pm.scenarios.appendItem \  
-id                     "ORIGINATECALL" \  
-symDestination         "Traffic2_SIPServer1:5060" \  
-bNextCommandIsDetect   false \  
-isLastCmd              false \  
-useDhcpForOriginate    false \  
-hasVideo               false \  
-_gbDhcpServerPortForOriginate false \  
-dhcpServerPortForOriginate 5060  
  
$Activity_SIPClient1 agent.pm.scenarios.appendItem \  
-id                     "ENDCALL" \  
-isLastCmd              false \  
-szDummy03              ""  
  
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.clear  
  
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.appendItem \  
-id                     "TypeOfServiceForRTP" \  

```

```
-tos_val_for_rtp                "Best Effort (0x0)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.appendItem \
-id                              "TypeOfServiceForRTP" \
-tos_val_for_rtp                "Class 1 (0x20)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.appendItem \
-id                              "TypeOfServiceForRTP" \
-tos_val_for_rtp                "Class 2 (0x40)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.appendItem \
-id                              "TypeOfServiceForRTP" \
-tos_val_for_rtp                "Class 3 (0x60)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.appendItem \
-id                              "TypeOfServiceForRTP" \
-tos_val_for_rtp                "Class 4 (0x80)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.appendItem \
-id                              "TypeOfServiceForRTP" \
-tos_val_for_rtp                "Express Forwarding (0xA0)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.appendItem \
-id                              "TypeOfServiceForRTP" \
-tos_val_for_rtp                "Control (0xC0)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_rtp.appendItem \
-id                              "TypeOfServiceForRTP" \
-tos_val_for_rtp                "Best Effort (0x0)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.clear
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.appendItem \
-id                              "TypeOfServiceForSIP" \
-tos_val_for_sip                "Best Effort (0x0)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.appendItem \
-id                              "TypeOfServiceForSIP" \
```



```
-tos_val_for_sip                "Class 1 (0x20)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.appendItem \
-id                              "TypeOfServiceForSIP" \
-tos_val_for_sip                "Class 2 (0x40)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.appendItem \
-id                              "TypeOfServiceForSIP" \
-tos_val_for_sip                "Class 3 (0x60)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.appendItem \
-id                              "TypeOfServiceForSIP" \
-tos_val_for_sip                "Class 4 (0x80)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.appendItem \
-id                              "TypeOfServiceForSIP" \
-tos_val_for_sip                "Express Forwarding (0xA0)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.appendItem \
-id                              "TypeOfServiceForSIP" \
-tos_val_for_sip                "Control (0xC0)"
$Activity_SIPClient1 agent.pm.predefined_tos_for_sip.appendItem \
-id                              "TypeOfServiceForSIP" \
-tos_val_for_sip                "Best Effort (0x0)"
```

SEE ALSO

ixNetTraffic

General Settings

General Settings—Sets the SIP client agent's general configuration options.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.pm.generalSettings.config
```

DESCRIPTION

A SIP client's advanced configuration options are set by modifying the options of the `pm.generalSettings` option of the `SIP Client Agent` object.

SUBCOMMANDS

None.

OPTIONS

`ipv6Form`

Specifies `0` (ipv4) or `1` (ipv6) to determine the types of networks (in the `ixNetTraffic`) that the SIP client and server use.

All the fields that support IPv4 addresses also support IPv6 addresses. There are two methods for entering IPv6 addresses in SIP fields: in square brackets (`[]`) or in vertical bar (pipe) symbols (`|`).

For the following options, enclose the address in square brackets (`[]`). For example `[::C212:1003]:5060`

- `szRegistrar;`
- `ORIGINATECALL` command
- `REGISTRATION` command
- `REDIRECTION` command

For information on these options, see `Scenarios`.

In the `Content of Messages` object, the following four options accept IPV6 addresses. Enclose the address for these options in vertical bar (pipe) symbols (`|`). (square brackets are used to enclose sequence generators). The options are:

- `szREQUESTURI`
- `szFROM`
- `szTO`
- `szCONTACT`

`szAuthUsername`



User name to be registered with registrar. You can include variables in this field to automatically generate large numbers of unique user names. See *Using Variables in SIP Fields* on page 20-80. Maximum length = 128 characters. (Default = "user[0000-]").

szAuthPassword

Password to be registered with registrar. You can include variables in this field to automatically generate large numbers of unique passwords. See *Using Variables in SIP Fields* on page 20-80. Maximum length = 128 characters. (Default = "password[0000-]").

szAuthDomain

Domain to be registered with registrar. You can include variables in this field to automatically generate large numbers of unique domains. See *Using Variables in SIP Fields* on page 20-80. Maximum length = 128 characters. (Default = "domain[0000-]").

szTransport

Type of transport to be used. The choices are:

Value	Description
TCP	IxLoad initially uses TCP as the transport. If the remote party answers using UDP, IxLoad accepts the response and switches to UDP as the trans
UDP	IxLoad initially uses UDP as the transport. If the remote party answers using TCP, IxLoad accepts the response and switches to TCP as the trans
Only TCP	IxLoad uses only TCP as the transport. If the remote party answers using UDP, IxLoad discards the response and continues using TCP.
Only UDP	IxLoad uses only UDP as the transport. If the remote party answers using TCP, IxLoad discards the response and continues using UDP.

nUdpPort

Port number to be used for sending and receiving SIP messages over UDP. Mini= "1," maximum = "65,535." (Default = "5,060").

nTcpPort

Port number to be used for sending and receiving SIP messages over TCP. Mini= "1," maximum = "65,535." (Default = "5,060").

nUdpMaxSize

Maximum size, in Kb, of a SIP message that will be sent. If a message exceeds this size, IxLoad ignores it.

szRegistrar

Host name or IP address and port number of registrar. This option also accepts IPV6 addresses that are enclosed in square brackets. (Default = "127.0.0.1:5060").

bRegBefore

If true, before starting the Originate Call/EnCall --> Receive call process, the IxLoad SIP client registers with the proxy server. Registration occurs only once at the beginning of the test. (Default = "0").

enableTosSIP

Enables the setting of the TOS (Type of Service) bits in the header of the SIP packets.

Value	Description
0	(default) TOS bits disabled.
1	TOS bits enabled.

enableTosRtp

Enables the setting of the TOS (Type of Service) bits in the header of the RTP data packets.

Value	Description
0	(default) TOS bits disabled.
1	TOS bits enabled.

type_of_service_for_sip

If enableTosSIP is true, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

Value	Description
"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control

type_of_service_for_rtp



If `enableTosRtp` is true, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for RTP data packets. See `type_of_service_for_sip` for the list of choices. (Default = "Best Effort (0x0)").

`enableVlanPriority_for_sip`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If true, `IxLoad` sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = false).

`vlanPriority`

When `enableVlanPriority` is true, this option sets the vlan priority value.

EXAMPLE

```
$Activity_SIPClient1 agent.pm.generalSettings.config \  
-dhcpServerPort          5060 \  
-ipv6Form                0 \  
-bRemoveCredent         false \  
-bRegBefore              false \  
-type_of_service_for_rtp "Best Effort (0x0)" \  
-_gbDhcpServerPort       false \  
-nUdpMaxSize             1024 \  
-nUdpPort                5060 \  
-szAuthDomain            "domain\[0000-\]" \  
-vlan_priority_sip       0 \  
-useDhcp                 false \  
-enableTosSIP            false \  
-implicitLoopCheck       true \  
-ipPreference            0 \  
-_gbIpPreference         false \  
-nPrefQop                0 \  
-szRegistrar             "127.0.0.1:5060" \  
-enableVlanPriority_for_sip false \  

```

-nTcpPort	5060 \
-szTransport	"UDP" \
-szAuthPassword	"password\[0000-\]" \
-type_of_service_for_sip	"Best Effort (0x0)" \
-szAuthUsername	"user\[0000-\]" \
-enableTosRTP	false \
-compressZeros	false

SEE ALSO

SIP Client Agent



Content of Messages

Content of Messages—Specifies the content of the SIP messages sent by the cli

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.pm.contentOfMessages.config
```

DESCRIPTION

A SIP client's advanced configuration options are set by modifying the options of the `pm.contentOfMessages` option of the `SIP Client Agent` object.

SUBCOMMANDS

None.

OPTIONS

`bRoute`

If `true`, IxLoad inserts a Route header field into the SIP message. The route should contain a list of specified proxies. Use the `szRoute` parameter to specify the route. (Default = "0").

`szRoute`

If `bRoute` is `true`, this parameter specifies the Route header field used to force the request to follow a fixed route through a listed set of proxies. (Default = "Route:< sip:p1.example.com;lr>, < sip:p2.domain.com;lr>").

`bCompact`

If `true`, IxLoad uses the compact forms of the SIP header field notations. The compact form is intended for instances in which messages would otherwise become too large to be carried on the transport available to it (exceeding the maxtransmission unit [MTU] when using UDP, for example). (Default = "0").

`bFolding`

If `true`, the VIA field spans two lines. Some SIP devices may not be able to handle this. (Default = "0").

`bScattered`

If `true`, IxLoad moves the header fields around in the message in order to make it more difficult for the DUT to decode the message. (Default = "0").

`bAdvisable`

If `true`, the SIP request includes the header fields that are defined as 'mandatory' by the SIP RFC (RFC 3261), plus those that are recommended as 'advisable.' (Default = "0").

`bOptional`

If `true`, the SIP request includes the header fields that are defined as 'mandatory' by the SIP RFC (RFC 3261), plus those that are listed as 'optional.' (Default = "0").

`bBestPerformance`

If `true`, IxLoad inserts the headers into the message so that the message can be processed as quickly as possible by the receiving system. If `false`, IxLoad inserts the headers into the message so that it requires maximum processing by the receiving system. (Default = "1").

`szREQUESTURI`

User or service to which the SIP request is being addressed. You can include variin this field to automatically generate large numbers of unique domains. See *Using Variables in SIP Fields* on page 20-80. Maximum length = 128 charThis option also accepts IPV6 addresses that are enclosed in square brack(Default = "sip:id[50000-]@IP").

`szFROM`

Initiator of the SIP request. You can include variables in this field to automatigenerate large numbers of unique domains. See *Using Variables in SIP Fields* on page 20-80. Maximum length = 128 characters. This option also accepts IPV6 addresses that are enclosed in square brackets. (Default = "sip:id[50000-]@IP").

`szTO`

Logical recipient of the request. You can include variables in this field to autogenerate large numbers of unique domains. See *Using Variables in SIP Fields* on page 20-80. Maximum length = 128 characters. This option also accepts IPV6 addresses that are enclosed in square brackets. (Default = "sip:id[50000-]@IP").

`szCONTACT`

The Contact header field value provides a URI whose meaning depends on the type of request or response it is in.The Contact header field has a role similar to the Location header field in HTTP. You can include variables in this field to autogenerate large numbers of unique domains. See *Using Variables in SIP Fields* on page 20-80. Maximum length = 128 characters. This option also accepts IPV6 addresses that are enclosed in square brackets. (Default = "sip:id[50000-]@IP").



rulesTable

This is a list of type Rules. The rules in this list define how this message will be handled. (Default = {}).

EXAMPLE

```
$Activity_SIPClient1 agent.pm.contentOfMessages.config \  
  
-bFolding                false \  
  
-bBestPerformance       1 \  
  
-szRoute                 "Route:  
<sip:p1.example.com;lr>,<sip:p2.domain.com;lr>" \  
  
-bOptional              false \  
  
-szCONTACT               "<sip:id\[00000-\]@IP>" \  
  
-bAdvisable              false \  
  
-bCompact                false \  
  
-bRoute                  false \  
  
-szFROM                  "<sip:id\[00000-\]@IP>" \  
  
-szTO                    "<sip:id\[50000-\]@IP>" \  
  
-szREQUESTURI            "sip:id\[50000-\]@IP" \  
  
-bScattered              false
```

SEE ALSO

SIP Client Agent

Rules

Rules—Defines a rule for handling a SIP message.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.pm.contentOfMessages.rulesTable.appendItem
```

DESCRIPTION

A Rule object is a part of a `rulesList` object, which is part of a `Content of Messages`, which is a member of a `SIP Client Agent` object. Its options are configas per the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`szMessage`

Type of message the rule will apply to. (Default = "REGISTER"). See `szAction` for the list of messages that you can configure rules for.

`szAction`

Action the rule performs. (Default = "{}").

Message	Actions
"REGISTER"	<p>Contact Expires Param(s)</p> <p>Specifies the length of time that the Contact header is valid for. When the Contact header expires, the registration performed on the proxy server is lost.</p> <p>Expires Headers</p> <p>Specifies the relative length of time after which the message or its content expires.</p>

`szValue`

Numerical value for the `szAction`. (Default = "0").

EXAMPLE

```
$Activity_SIPClient1 agent.pm.contentOfMessages.rulesTable.appendItem \  
-id "Rules" \  
-szMessage "REGISTER" \  
-szValue "3600" \  
-szAction "ContactExpiresParam"
```

SEE ALSO

Content of Messages

State Machine

State Machine—Configures the SIP client agent's internal timers and other parameters of its state machine.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.pm.stateMachine.config
```

DESCRIPTION

A SIP client's state machine parameters are set by modifying the options of the `pm.StateMachine` option of the `SIP Client Agent` object.

SUBCOMMANDS

None.

OPTIONS

`nTimersT1`

Estimate of the round-trip time (RTT), in milliseconds (ms). (Default = "500").

`nTimersT2`

Maximum retransmit interval, in milliseconds (ms), for non-INVITE requests and INVITE responses. (Default = "4,000").

`nTimersT4`

Maximum length of time, in milliseconds (ms), that a message will remain in the network. (Default="5,000").

`nTimersTC`

Proxy INVITE transaction timeout. Minimum = 180,000. (Default = "180,000").

`nTimersTD`

Wait time for response retransmits. For UDP, this must be greater than 32 sec (Default = "32,000").

`bUseTimer`



If `true`, `IxLoad` enforces a timeout limit on transactions. If a transaction exceeds the timeout value, `IxLoad` marks it as a failed transaction, and increments the transaction failure statistics. (Default = "0").

`nTimeout`

If `bUseTimer` is `true`, this parameter specifies the transaction timeout interval, in milliseconds (ms). (Default = "30,000").

`bRecv5xx`

If `true` and `IxLoad` receives a 5xx series response to a transaction, `IxLoad` marks it as a failed transaction, and increments the transaction failure statistics. (Default = "0").

`nReRegDuration`

In the event that `IxLoad` fails to register with a registrar, this field defines the amount of time allowed to re-registration. Minimum = "0," maximum = "60,000." (Default = "0").

`bNextOnFail`

If `true` and `IxLoad` encounters a transaction failure, it continues processing SIP requests. If `false` and `IxLoad` encounters a transaction failure, it stops processing SIP requests. (Default = "1").

EXAMPLE

```
$Activity_SIPClient1 agent.pm.stateMachine.config \  
-bNextOnFail true \  
-nTimersT4 5000 \  
-nReRegDuration 0 \  
-nTimersT1 500 \  
-nTimersT2 4000 \  
-bUseTimer false \  
-nTimeout 30000 \  
-nTimersTD 32000 \  
-nTimersTC 180000 \  
-bRecv5xx false
```

SEE ALSO

SIP Client Agent



Media Settings

Media Settings—Selects and configures the streaming audio files for the multisession that the client will play over RTP.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.pm.mediaSettings.config
```

DESCRIPTION

A SIP client's advanced configuration options are set by modifying the options of the `pm.mediaSettings` option of the `SIP Client Agent` object.

SUBCOMMANDS

None.

OPTIONS

`szCodecName`

Codec to be used to encode waveform audio files listed in the Audio Clips Pool. The choices are:

Value	Description
"G711ALaw"	(default) G.711 A-law
"G711ULaw"	G.711 mu-law
"G729A"	G.729A
"G729B"	G.729B
"G726"	G.726
"G723_1"	G.723.1

`szCodecDetails`

Displays the properties of the codec such as the number of bytes per frame of compressed audio, and the rate at which packets are sent over the connection. (Default = {}).

`szCodecDetails` from Media Settings has a special format: `BFval1PTval2`, where:

Value	Description
"val1"	Number of codec bytes per frame (only the rtp payload; do not add the 12 bytes for the rtp header)
"val2"	The packet time

These two options specify information about the packetization.

`szCodecDescr`

Codec description. (Default = {}).

`szBitRate`

This specifies the bit rate of the codec being used. Possible values are:

Codec	Bit Rate
G711Alaw	64 kbps
G711Ulaw	64 kbps
G723.1	5.3 kbps 6.3 kbps
G726	40 kbps
G729A	8 kbps
G729B	8 kbps
AMR	4.75 kbps 5.15 kbps 5.9 kbps 6.7 kbps 7.4 kbps 7.95 kbps 10.2 kbps 12.2 kbps
iLBC	13.33 kbps 15.2 kbps

`bModifyPowerLevel`

If `true`, `IxLoad` modifies the volume of the compressed audio. (Default = "0").

`szPowerLevel`

If `bModifyPowerLevel` is `true`, this parameter specifies the amount of gain (voladded to compressed audio. The choices are :



Value	Description
"PLO"	(default) 0 dB
"PL_10"	-10 dB
"PL_20"	-20 dB
"PL_30"	-30 dB

`bUseJitter`

Enables or disables use of the jitter buffer. (Default = "0").

`bJitMs`

Defines the method used to set the jitter buffer size.

Value	Description
0	(Default). Jitter buffer size is set by <code>nJitterBuffer</code> .
1	Jitter buffer size is set by <code>nJitterMs</code> .

`nJitterBuffer`

Number of packets to buffer in order to reduce jitter. Minimum = "0," maximum = "3." (Default = 0).

`nJitterMs`

Jitter Buffer size, in milliseconds. Minimum = "1," maximum = "3,000." (Default = "20").

`bUseCompensation`

Enables or disables use of the compensation jitter buffer. (Default = "0").

`bCompMs`

Defines the method used to set the compensation jitter buffer size.

Value	Description
0	(Default). Compensation jitter buffer size is set by <code>nCompJitterBuffer</code> .
1	Compensation jitter buffer size is set by <code>nCompJitterMs</code> .

nCompJitterBuffer

Compensation jitter buffer maximum size, in packets. Minimum = "0," maxi= "300." (Default = "50").

nCompJitterMs

Compensation jitter buffer maximum size, in milliseconds. Minimum = "0," maximum = "3,000." (Default = "1,000").

nCompMaxDropped

Maximum dropped consecutive packets. Minimum = "1," maximum = "100," (Default = "7").

bUseMOS

Enables or disables use of MOS. (Default = "0").

bMosOnMax

Defines whether MOS is calculated for a subset of streams or for all streams.

Value	Description
0	(Default). MOS calculation is applied to all streams.
1	MOS calculation is applied to the number of streams specified by nMosMax.

nMosMaxStreams

Maximum number of concurrent streams used in MOS calculation. Minimum = "1." (Default = "1").

nMosInterval

Frequency at which IxLoad samples the RTP streams to generate the MOS scores. Minimum = "2," maximum = "30." (Default = "3").

nDtmfDuration

Length of time allowed to play the DTMF sequence. Minimum = "60," maxi= "999." (Default = "100").

nDtmfInterdigits

Duration (in milliseconds) of the DTMF interdigit signal. Minimum = "30," max= "9999." (Default = "40").



bLimitDtmf

Enable or disable limitation on the number of DTMF streams to be processed. (Default = "1").

Value	Description
0	DTMF applied to all streams.
1	(Default) DTMF limited to number of streams specified by nDtmfStreams.

nDtmfStreams

Number of streams to which path confirmation will be applied. Minimum = "1," maximum = "900."
(Default = "10").

nPcInterval

If Synthetic path confirmation is selected, this is the interval at which IxLoad add the synthetic RTP packets to the stream. Minimum = "1," (Default = "500").

nSessionType

Type of voice session. The choices are:

Value	Description
"0"	(default) Plays audio file specified by szAudioFile.
"1"	Perform DTMF path confirmation.
"2"	Perform synthetic DTMF path confirmation.

szDtmfSeq

DTMF sequence used for path confirmation. (Default = "12,345").

szPeerCodecName

Name of codec used by peer. (Default = {}).

szPeerCodecDetails

Details of codec used by peer. (Default = {}).

szPeerDtmfSeq

DTMF sequence used by peer. (Default = {}).

nPeerDtmfDuration

DTMF duration used by peer. (Default = "0").

nPeerDtmfInterdigits

Inter-digits interval used by peer. (Default = "0").

audioClipsTable

This is a list of type `Audio Clips Pool`. This list contains the waveform audio files that the SIP message will send. (Default = {}).

bUseSilence

If enabled, IxLoad generates and sends artificial background noise during times of silence during a call.

bSilenceMode

Indicates the method used to generate the background noise. Possible Values are:

Value	Description
"0"	Comfort Noise silence type.
"1"	Null Data encoded silence type.

bRtpStartCollector

Specifies, whether the statistics for rtp should be collected or not. Possible values are:

Value	Description
0	Do not start
1	Start

EXAMPLE

```
$Activity_SIPClient1 agent.pm.mediaSettings.config \
```



```
-nPcInterval 500 \  
-nJitterBuffer 1 \  
-nDtmfInterdigits 40 \  
-nCompMaxDropped 7 \  
-nPeerDtmfDuration 0 \  
-nJitterMs 20 \  
-bSilenceMode 1 \  
-nAudioPoolTime 1178615586 \  
-szBitRate "64 kbps" \  
-nDtmfDuration 100 \  
-szPeerCodecName "" \  
-szSilenceFile "" \  
-bytesPerFrameBuffer "" \  
-groupBox_MOS1 false \  
-szPeerCodecDetails "" \  
-bMosOnMax 0 \  
-groupBox_JB1 false \  
-nMosInterval 3 \  
-nCompJitterBuffer 50 \  
-bUseJitter false \  
-szCodecName "G711ALaw" \  
-szPeerDtmfSeq "" \  
-bLimitDtmf true \  
-bUseMOS false \  
-bJitMs 0 \  
-szCodecDescr "ITU-T G.711 is a standard to  
represent 8 bit compressed pulse code modulation (PCM) samples for signals of  
voice frequencies, sampled at the rate of 8000 samples/second. G.711 encoder  
will create a 64 Kbps bitstream. A-Law G.711 PCM encoder converts 13 bit  
linear PCM samples into 8 bit compressed PCM (logarithmic form) samples, and  
the decoder does the conversion vice versa." \  

```

```
-bCompMs 0 \  
-nDtmfStreams 10 \  
-packetTimeBuffer "" \  
-szPowerLevel "PL_20" \  
-szDtmfSeq "12345" \  
-nCompJitterMs 1000 \  
-nPeerDtmfInterdigits 0 \  
-bRtpStartCollector false \  
-nMosMaxStreams 1 \  
-szCodecDetails "BF160PT20" \  
-nSessionType 0 \  
-bUseSilence false \  
-bModifyPowerLevel false \  
-bUseCompensation false
```

SEE ALSO

SIP Client Agent



Audio Clips Pool

Audio Clips Pool—Defines an audio file to be included in the list that the SIP cli will play.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.pm.mediaSettings.AudioClipsPool.config
```

DESCRIPTION

An `Audio Clips Pool` object is a part of a `mediaSettings` object, which is a mem of a `SIP Client Agent` object. Its options are configured as per the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`szWaveName`

Waveform audio (.wav) file. (Default = "...").

`szDataFormat`

Encoding format of waveform audio file. (Default = "...").

`nSampleRate`

Number of samples taken per second from the recording source. (Default = "0").

`nResolution`

Number of bits per sample. (Default = "0").

`nChannels`

Number of audio channels: "0" = mono, "1" = stereo. (Default = "0").

`nDuration`

Playing time of audio file. (Default = "0").

nSize

Size of audio file, in bytes. (Default = "0").

szRawWaveName

Name and path of wave file to be added to the list. (Default = "{}").

EXAMPLE

```
$Activity_SIPClient1 agent.pm.mediaSettings.AudioClipsPool.config \  
-szWaveName "meeting.wav" \  
-szDataFormat "PCM" \  
-nSampleRate "8000" \  
-nResolution "16" \  
-nChannels "1" \  
-nDuration "30" \  
-nSize "65748" \  
-szRawWaveName "/files/audio/wav/meeting.wav"
```

SEE ALSO

Content of Messages



Video Settings

Video Settings—Contains the controls that you can use to define the parameters of the synthetic video the SIP server generates for a MEDIASESSION scenario.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.pm.videoSettings.config \
```

DESCRIPTION

Video Settings is configured and added to an SIP activity.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`videoBitrate`

Bit rate of generated (synthetic) video data.

`videoBitrateLimit`

The `videoBitrate` limit in Kbps.

EXAMPLE

```
$Activity_SIPClient1 agent.pm.videoSettings.config \
    -videoBitrate                128.0 \
    -videoBitrateLimit           0
$Activity_SIPClient1 agent.pm.scenarios.clear
```

SEE ALSO

Media Settings



Scenarios

Scenarios—Creates the list of SIP commands that the client will send to a SIP server.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SIPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_SIPClient1 agent.pm.scenarios.appendItem
```

DESCRIPTION

A command is added to the Scenarios object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

SIP command to be executed. One of the following:

Command	Description
REGISTRATION	Registers the SIP client with a registrar. This option also accepts IPV6 addresses that are enclosed in square brackets.
ORIGINATECALL	Sets up a multimedia session with the specified destination. This option also accepts IPV6 addresses that are enclosed in square brackets.
THINK	Pause during command list processing. You should include a {Think} command whenever necessary to allow the destination to process the preceding commands. You can configure a pause of fixed length or of random length.
ENDCALL	Terminates the SIP session.
REDIRECTION	Redirects the request for a SIP session from one proxy to another. This option also accepts IPV6 addresses that are enclosed in square brackets.
VOICESESSION	Plays one of the waveform audio files listed in the Audio Clips Pool on the Media Settings tab. The SIP client sends the file to the destination configured for the previous Originate Call command in the command list.
GENERATEMF	Generates multi-frequency tone sequences. The sequences are encoded using a G.711 voice codec and sent in-band over RTP.
GENERATEDT MF	Generates dual-tone multi-frequency sequences. The sequences are encoded using a G.711 voice codec and sent in-band over RTP.

Command	Description
GENERATETONE	Generates tone sequences. The sequences are encoded using a G.711 voice codec and sent in-band over RTP.
DETECTDTMF	Detects the tones generated by the DETECTDTMF, DETECTMF, or DETECTTONE commands.
MEDIASESSION	Simulates a call made using a video phone. MEDIASESSION transaudio similar to the VOICESESSION command, and (optionally) generates simulated video data. MEDIASESSION must be preceded by ORIGINATECALL and succeeded by ENDCALL.

Arguments for id = REGISTRATION

bUseDest

If true, the registration is sent to the address specified by szDestination. If false, the registration is sent to the Registrar configured by the General Settings com(Default = "1").

szDestination

Registrar that the registration will be sent to. This option also accepts IPV6 addresses that are enclosed in square brackets. (Default = "127.0.0.1:5060").

Arguments for id = ORIGINATECALL

symDestination

Destination of the call. If the destination is an external host, specify its address or host name and port number. If the destination is an IxLoad SIP server agent, specify the name of the agent. This option also accepts IPV6 addresses that are enclosed in square brackets. (Default = "None").

Arguments for id = THINK

nThinkMin

Minimum length of the pause, in milliseconds. To configure a fixed-length pause, enter the same value in this field and nThinkMax. (Default = "1,000").

nThinkMax

Maximum length of the pause, in milliseconds. To configure a fixed-length pause, enter the same value in this field and nThinkMin. (Default = "1,000").

Arguments for id = ENDCALL

None.



Arguments for id = REDIRECTION

szDestination

Address of the proxy to which the request is to be redirected to. This option also accepts IPV6 addresses that are enclosed in square brackets. (Default = "127.0.0.1:5060").

Arguments for id = VOICESESSION

szAudioFile

Waveform audio file that will be played during the session. This must be an `szWaveName` object contained within the `Audio Clips Pool` object. (Default = "<None>").

nPlayMode

If `true`, the audio file plays for a fixed number of times. If `false`, the audio file plays continuously. (Default = "0").

nRepeatCount

If `nPlayMode` is `true`, this parameter sets the number of times that the audio file will play. (Default = "1").

nPlayTime

Length of time to play the audio file. Specify the units of time in the `nTimeUnit`.

nTimeUnit

Units of time used to set the audio file play time (`nPlayTime`). The choices are:

Value	Description
"0"	(default) Seconds
"1"	Minutes
"2"	Hours
"3"	Days

Arguments for id = GENERATEMF

szMfSeq

The sequence of MF digits to be generated.

nMfDuration

Length of time allowed to play the MF sequence. Minimum = "10", Maximum = "990".

nInterMfInterval

Duration (in milliseconds) of the MF interdigit signal. Minimum = "10", Maxi = "9990".

nMfAmplitude

The amplitude of the signal generated by the sending sequence. Minimum = "-30", Maximum = "-10".

nPlayMode

The play mode to play the MF tones. Possible values are:

Value	Description
0	Generate for a specified period of time
1	Repeat for a specified number of times

nRepeatCount

Number of times to repeat the generation of the sequence.

nPlayTime

The time units to play the specified sequence.

nTimeUnit

Signifies the time unit type. Possible values are:

Value	Description
"0"	(default) Seconds
"1"	Minutes
"2"	Hours
"3"	Days

Arguments for id = GENERATEDTMF

szDtmfSeq

The dtmf sequence to be generated.



nDtmfDuration

Length of time allowed to play the DTMF sequence. Minimum = "10", Maximum = "990".

nDtmfInterdigits

Duration (in milliseconds) of the DTMF interdigit signal. Minimum = "10", Max = "9990".

nDtmfAmplitude

The signal amplitude generated for the stream containing the digits.

nPlayMode

The play mode to play the DTMF tones. Possible values are:

Value	Description
0	Generate the specified sequence for a specified number of times
1	Generate the specified sequence for a specified time

nRepeatCount

Number of time to repeat the generation of the specified sequence.

Arguments for id = GENERATETONE

nToneName

This is the id for the tone. Possible values are:

Value	Description
0	"600-10"
1	"1400-10"
2	"2500-10"
3	"550-20"
4	"1350-20"
5	"2450-20"
6	"650-30"
7	"2550-30"

Value	Description
0	"600-10"
8	"1450-30"
9	"3400-10"
10	"3400-30"
11	"2100-10"
12	"2150-30"
13	"400-10"
14	"450-30"
15	"Confirmation Tone"
16	"Call Waiting Tone"
17	"TN_1"
-1	"Custom Tone"

nPlayMode

The play mode to play the MF tones. Possible values are:

Value	Description
0	Generate for a specified period of time
1	Repeat for a specified number of times

nRepeatCount

Number of times to repeat the generation of the sequence.

nPlayTime

The time units to play the specified sequence.

nTimeUnit

Signifies the time unit type. Possible values are:

Value	Description
"0"	(default) Seconds
"1"	Minutes
"2"	Hours

"3"	Days
-----	------

nToneDuration

The duration of a tone with only one frequency.

nFrequency1

For a single tone, this is the frequency of the signal used to generate the tone. For a dual tone, this is the frequency of the signal used to generate the lower band of the tone.

nFrequency2

For a dual tone, this is the frequency of the signal used to generate the upper band of the tone.

nAmplitude1

Amplitude of the nFrequency1 signal.

nAmplitude2

Amplitude of the nFrequency2 signal.

nOnTime

For a cadenced tone, this is the amount of time the tone signal or signals are played.

nOffTime

For a cadenced tone, this is the amount of time the tone signal or signals are muted.

nRepetitionCount

For a cadenced tone, this specifies the number of times that the On Time / Off Time cycle is repeated.

nToneType

The format of the tone. Possible values:

Value	Description
0	"Single Tone"
1	"Dual Tone"
2	"Single Tone Cadence"
3	"Dual Tone Cadence"

Arguments for id = DETECTDTMF

nDTMFDetectionMode

Method used to detect tones. Possible values are:

Value	Description
0	detect continously for a specified time
1	detect exactly a specified numebr of digits
2	detect a specified sequence

szDtmfSeq

Sequence of digits to detect.

nDetectTime

The number of time units to sustain the detect operation.

nDetectTimeUnit

Signifies the time unit type. Possible values are:

Value	Description
"0"	(default) Seconds
"1"	Minutes
"2"	Hours
"3"	Days

nDtmfCount

The exact number of digits to detect.

nFirstDTMFTimeout

The maximum time for the first digit to arrive and to be decoded.



nInterDTMFInterval

The maximum time between the arrival of digits.

Arguments for id = MEDIASESSION

nRepeatCount

Number of times to repeat the generation of the sequence.

nWavDuration

The time duration of a wave.

nTimeUnit

Signifies the time unit type. Possible values are:

Value	Description
"0"	(default) Seconds
"1"	Minutes
"2"	Hours
"3"	Days

nPlayMode

The play mode to play the MF tones. Possible values are:

Value	Description
0	Generate for a specified period of time
1	Repeat for a specified number of times

synthVideo

If enabled, the SIP client generates video data and transmits it to the server along with the audio to simulate a video phone call. To configure the video parameters refer `Video Settings`.

szAudioFile

The name of the audio file that will be played.

szTotalTime

The total time for which an audio file will be played.

nTotalTime

The total time for which an .wav file will be played.

nPlayTime

The time units to play the specified sequence.

EXAMPLE

```
$Activity_SIPClient1 agent.pm.scenarios.appendItem \  
-id "ORIGINATECALL" \  
-symDestination "Traffic2_SIPServer1:5060" \  
-bNextCommandIsDetect false \  
-isLastCmd false \  
-useDhcpForOriginate false \  
-hasVideo false \  
-_gbDhcpServerPortForOriginate false \  
-dhcpServerPortForOriginate 5060  
$Activity_SIPClient1 agent.pm.scenarios.appendItem \  
-id "ENDCALL" \  
-isLastCmd true \  
-szDummy03 ""
```

SEE ALSO

Video Settings



SIP Server Agent

SIP Server Agent - create a SIP server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_SIPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_SIPServer1 agent.config
```

DESCRIPTION

An SIP server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time .

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
#####
# Activity SIPServer1 of NetTraffic Traffic2@Network2
#####
set Activity_SIPServer1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType                "SIP Server" ]
```

```
set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]

$Activity_SIPServer1 config \

    -enable                true \

    -name                  "SIPServer1" \

    -timeline              $_Match_Longest_

$Activity_SIPServer1 agent.config \

    -enable                true \

    -name                  "SIPServer1"

$Activity_SIPServer1 agent.pm.generalSettings.config \

    -dhcpServerPort        5060 \

    -ipv6Form              0 \

    -bRemoveCredent        false \

    -bRegBefore            false \

    -type_of_service_for_rtp "Best Effort (0x0)" \

    -_gbDhcpServerPort     false \

    -nUdpMaxSize           1024 \

    -regInterval           0 \

    -szAuthDomain          "domain\[0000-\]" \

    -vlan_priority_sip     0 \

    -useDhcp               false \

    -enableTosSIP          false \

    -nUdpPort              5060 \

    -ipPreference          0 \

    -_gbIpPreference       false \

    -nPrefQop              0 \

    -szRegistrar           "127.0.0.1:5060" \

    -enableVlanPriority_for_sip false \

    -nTcpPort              5060 \
```



```
-szTransport                "UDP" \  
-szAuthPassword             "password\[0000-\\]" \  
-type_of_service_for_sip    "Best Effort (0x0)" \  
-szAuthUsername             "user\[0000-\\]" \  
-enableTosRTP               false \  
-compressZeros              false  
  
$Activity_SIPServer1 agent.pm.mediaSettings.config \  
-nPcInterval                500 \  
-nJitterBuffer              1 \  
-nDtmfInterdigits           40 \  
-nCompMaxDropped            7 \  
-nPeerDtmfDuration          0 \  
-nJitterMs                  20 \  
-bSilenceMode                1 \  
-nAudioPoolTime             1178615588 \  
-szBitRate                   "64 kbps" \  
-nDtmfDuration              100 \  
-szPeerCodecName             "" \  
-szSilenceFile               "" \  
-bytesPerFrameBuffer         "" \  
-groupBox_MOS1               false \  
-szPeerCodecDetails          "" \  
-bMosOnMax                   0 \  
-groupBox_JB1                false \  
-nMosInterval                3 \  
-nCompJitterBuffer           50 \  
-bUseJitter                  false \  
-szCodecName                  "G711ALaw" \  

```

```

-szPeerDtmfSeq          "" \
-bLimitDtmf            true \
-bUseMOS               false \
-bJitMs                0 \
-szCodecDescr          "ITU-T G.711 is a standard to
represent 8 bit compressed pulse code modulation (PCM) samples for signals of
voice frequencies, sampled at the rate of 8000 samples/second. G.711 encoder
will create a 64 Kbps bitstream. A-Law G.711 PCM encoder converts 13 bit
linear PCM samples into 8 bit compressed PCM (logarithmic form) samples, and
the decoder does the conversion vice versa." \
-bCompMs               0 \
-nDtmfStreams          10 \
-packetTimeBuffer      "" \
-szPowerLevel          "PL_20" \
-szDtmfSeq             "12345" \
-nCompJitterMs        1000 \
-nPeerDtmfInterdigits 0 \
-bRtpStartCollector    false \
-nMosMaxStreams        1 \
-szCodecDetails        "BF160PT20" \
-nSessionType          0 \
-bUseSilence           false \
-bModifyPowerLevel     false \
-bUseCompensation      false
$Activity_SIPServer1 agent.pm.contentOfMessages.config \
-bFolding              false \
-bBestPerformance      1 \
-szTO                  "<sip:id\[50000-\]@IP>" \
-bOptional             false \
-szCONTACT              "<sip:id\[50000-\]@IP>" \
-bAdvisable            false \

```



```
-bCompact                false \  
-szFROM                  "<sip:id\[50000-\]@IP">" \  
-szREQUESTURI           "sip:IP" \  
-bScattered              false  
  
$Activity_SIPServer1 agent.pm.contentOfMessages.rulesTable.clear  
$Activity_SIPServer1 agent.pm.stateMachine.config \  
-nActiveTimeout          0 \  
-bUasStateless           false \  
-nActiveTimeoutValue     0 \  
-nTimersT4                5000 \  
-nTimersT1                500 \  
-nTimersT2                4000 \  
-nTimersTD                32000 \  
-nActiveTimeoutTU        0 \  
-nTimersTC                180000  
  
$Activity_SIPServer1 agent.pm.videoSettings.config \  
-videoBitrate            128.0 \  
-videoBitrateLimit       0  
  
$Activity_SIPServer1 agent.pm.scenarios.clear  
$Activity_SIPServer1 agent.pm.scenarios.appendItem \  
-id                       "RECEIVEUSING180" \  
-bNextCommandIsDetect     false \  
-szDummy10                "" \  
-isLastCmd                false  
  
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.clear  
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.appendItem \  
-id                       "TypeOfServiceForRTP" \  
-tos_val_for_rtp          "Best Effort (0x0)"
```

```
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.appendItem \  
    -id                    "TypeOfServiceForRTP" \  
    -tos_val_for_rtp      "Class 1 (0x20)" \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.appendItem \  
    -id                    "TypeOfServiceForRTP" \  
    -tos_val_for_rtp      "Class 2 (0x40)" \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.appendItem \  
    -id                    "TypeOfServiceForRTP" \  
    -tos_val_for_rtp      "Class 3 (0x60)" \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.appendItem \  
    -id                    "TypeOfServiceForRTP" \  
    -tos_val_for_rtp      "Class 4 (0x80)" \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.appendItem \  
    -id                    "TypeOfServiceForRTP" \  
    -tos_val_for_rtp      "Express Forwarding (0xA0)" \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.appendItem \  
    -id                    "TypeOfServiceForRTP" \  
    -tos_val_for_rtp      "Control (0xC0)" \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_rtp.appendItem \  
    -id                    "TypeOfServiceForRTP" \  
    -tos_val_for_rtp      "Best Effort (0x0)" \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.clear \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.appendItem \  
    -id                    "TypeOfServiceForSIP" \  
    -tos_val_for_sip      "Best Effort (0x0)" \  
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.appendItem \  
    -id                    "TypeOfServiceForSIP" \  
    -tos_val_for_sip      "Class 1 (0x20)"
```



```
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.appendItem \  
-id "TypeOfServiceForSIP" \  
-tos_val_for_sip "Class 2 (0x40)"  
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.appendItem \  
-id "TypeOfServiceForSIP" \  
-tos_val_for_sip "Class 3 (0x60)"  
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.appendItem \  
-id "TypeOfServiceForSIP" \  
-tos_val_for_sip "Class 4 (0x80)"  
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.appendItem \  
-id "TypeOfServiceForSIP" \  
-tos_val_for_sip "Express Forwarding (0xA0)"  
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.appendItem \  
-id "TypeOfServiceForSIP" \  
-tos_val_for_sip "Control (0xC0)"  
$Activity_SIPServer1 agent.pm.predefined_tos_for_sip.appendItem \  
-id "TypeOfServiceForSIP" \  
-tos_val_for_sip "Best Effort (0x0)"
```

SEE ALSO

`ixNetTraffic`

General Settings

General Settings—Sets the SIP server agent's general configuration options.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_SIPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_SIPServer1 agent.pm.generalSettings.config
```

DESCRIPTION

A SIP server's advanced configuration options are set by modifying the options of the `pm.generalSettings` option of the `SIP Server Agent` object.

SUBCOMMANDS

None.

OPTIONS

The SIP server agent's General Settings options are the same as for the SIP client agent. See the SIP Client.

EXAMPLE

```
$Activity_SIPServer1 agent.pm.generalSettings.config \  
-dhcpServerPort          5060 \  
-ipv6Form                 0 \  
-bRemoveCredent          false \  
-bRegBefore               false \  
-type_of_service_for_rtp "Best Effort (0x0)" \  
-_gbDhcpServerPort        false \  
-nUdpMaxSize              1024 \  
-regInterval              0 \  
-szAuthDomain             "domain\[0000-\]" \  
-vlan_priority_sip        0 \  
-useDhcp                  false \  
-enableTosSIP             false \  
-nUdpPort                 5060 \  

```



```
-ipPreference           0 \  
-_gbIpPreference       false \  
-nPrefQop              0 \  
-szRegistrar           "127.0.0.1:5060" \  
-enableVlanPriority_for_sip  false \  
-nTcpPort              5060 \  
-szTransport           "UDP" \  
-szAuthPassword        "password\[0000-\]" \  
-type_of_service_for_sip "Best Effort (0x0)" \  
-szAuthUsername        "user\[0000-\]" \  
-enableTosRTP          false \  
-compressZeros         false
```

SEE ALSO

SIP Server Agent

Content of Messages

Content of Messages—Specifies the content of the SIP messages sent by the server.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_SIPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_SIPServer1 agent.pm.contentOfMessages.config
```

DESCRIPTION

A SIP server's advanced configuration options are set by modifying the options of the `pm.contentOfMessages` option of the `SIP Server Agent` object.

SUBCOMMANDS

None.

OPTIONS

The SIP server agent's Content of Messages options are the same as for the SIP client agent. See `Content of Messages`.

SEE ALSO

`SIP Server Agent`



State Machine

State Machine—Configures the SIP server agent's internal timers and other parameters of its state machine.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_SIPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_SIPServer1 agent.pm.stateMachine.config
```

DESCRIPTION

A SIP server's state machine parameters are set by modifying the options of the `pm.StateMachine` option of the `SIP Server Agent` object.

SUBCOMMANDS

None.

OPTIONS

The SIP server agent's State Machine options are the same as for the SIP client agent with one addition (below). See `Content of Messages`.

`bUasStateless`

If `true`, the SIP server behaves as a stateless User Agent Server (UAS).

A stateless UAS does not maintain transaction states. It replies to requests norbut discards any state that would ordinarily be retained by a UAS after a response has been sent.

If a stateless UAS receives a retransmission of a request, it regenerates the response and resends it, just as if it were replying to the first instance of the request. (Default = "0").

EXAMPLE

```
$Activity_SIPServer1 agent.pm.stateMachine.config \  
-nActiveTimeout          0 \  
-bUasStateless           false \  
-nActiveTimeoutValue    0 \  
-nTimersT4               5000 \  
-nTimersT1               500 \  
-nTimersT2               4000 \  
-nTimersTD               32000 \  

```

-nActiveTimeoutTU	0 \
-nTimerSTC	180000

SEE ALSO

SIP Client Agent



Media Settings

Media Settings—Selects and configures the streaming audio files for the multisession that the server will play over RTP.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_SIPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_SIPServer1 agent.pm.mediaSettings.config
```

DESCRIPTION

A SIP server's advanced configuration options are set by modifying the options of the `pm.mediaSettings` option of the `SIP Server Agent` object.

SUBCOMMANDS

None.

OPTIONS

The SIP server agent's Media Settings options are the same as for the SIP client agent. See `Media Settings`.

SEE ALSO

`SIP Server Agent`

Scenarios

Scenarios—Creates the list of SIP commands that the server will send to a SIP server.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_SIPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_SIPServer1 agent.pm.scenarios.appendItem
```

DESCRIPTION

A command is added to the Scenarios object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

SIP command to be executed. One of the following:

Command	Description
RECEIVEUSING180	Causes the SIP server to respond to an INVITE by returning a 180 (Ringing) response, which indicates that it is trying to alert the user. The 180 response is routed back through the proxies in the reverse direc
RECEIVEUSING100AND180	Causes the SIP server to respond to an INVITE by first returning a 100 response, which indicates that the request has been received by the next-hop server and that some unspecified action is being taken on behalf of this call (for example, a database is being consulted). The server then responses with a 180 (Ringing) response, which indicates that it is trying to alert the user. The 180 response is routed back through the proxies in the reverse direction.
SEND6XX	Causes the SIP server to respond to an INVITE by first returning a 604 (Does not Exist Anywhere) response. 6xx-series responses are failure responses that indicate that the server has definitive information about a particular user, not just the particinstance indicated in the Request-URI.



Command	Description
VOICESESSION	<p>Plays one of the waveform audio files listed in the Audio Clips Pool on the Media Settings tab. The SIP server sends the file to the origin of the SIP call.</p> <p>Audio Pool File: Select one of the waveform audio files listed in the Audio Pool File on the Media Settab.</p> <p>Play: Select this option if you want the SIP server to play the clip a fixed number of times. Configure the number of times in the field.</p> <p>Repeat Continuous for: Select this option if you want the SIP server to play the clip continuously for some number of seconds, minutes, hours, or days. Select the units of time from the drop-down list, then configure the number of seconds, minutes, hours, or days that the clip will play in the field.</p>

Arguments for id = RECEIVEUSING180

None.

Arguments for id = RECEIVEUSING100AND180

None.

Arguments for id = SEND6XX

None.

Arguments for id = VOICESESSION

`szAudioFile`

Waveform audio file that will be played during the session. This must be an `szWaveName` object contained within the `Audio Clips Pool` object. (Default = "<None>").

`nPlayMode`

If `true`, the audio file plays for a fixed number of times. If `false`, the audio file plays continuously. (Default = "0").

`nRepeatCount`

If `nPlayMode` is `true`, this parameter sets the number of times that the audio file will play. (Default = "1").

`nPlayTime`

Length of time to play the audio file. Specify the units of time in the `nTimeUnit`.

nTimeUnit

Units of time used to set the audio file play time (nPlayTime). The choices are:

Value	Description
"0"	(default) Seconds
"1"	Minutes
"2"	Hours
"3"	Days

EXAMPLE

```
$Activity_SIPServer1 agent.pm.scenarios.appendItem \  
  -id "RECEIVEUSING180" \  
  -bNextCommandIsDetect false \  
  -szDummy10 "" \  
  -isLastCmd false
```

SEE ALSO

SIP Client Agent



Using Variables in SIP Fields

You can insert variables into various fields on the SIP client tabs, such as the `Username`, `Password`, and `Domain` fields on the SIP client General Settings tab and the `Userinfo` (header) fields on the Content of Messages. You can use the variables to generate large numbers of unique user names, passwords, and domain names or header fields.

You can use the following variables:

- Numbers 0-9
- Letters A-Z and a-z

The letter variables are case-sensitive; IxLoad considers the variable strings “AA” and “aa” to be different.

You can combine the variables with fixed text to create the user names, password and domain names. For example, you can enter `user[00-]` to create a range of unique user names that begin with the characters “user” (user00, user01, and so on).

To insert the variables into a field, enclose them in square brackets (`[]`). To specify a range, separate the minimum and maximum values with a hyphen (-). For example, `[00-10]` specifies a range of 00 through 10.

The number of variables you insert determines the width of the generated strings. For example, the variable “00” can generate the strings 00 - 99. The variable string “000” can generate the strings 000 - 999.

Similarly, “AA” can generate strings that consist of all the two-letter combinations from AA to ZZ. “AAA” can generate strings that consist of all the three letter combinations from AAA to ZZZ.

You can use a single variable string and allow IxLoad to generate strings up to the maximum value of the string, or you can use two variable strings together to restrict the generated strings to a certain range.

See the following example:

`[0-]` will generate all the values 0 - 9 (0, 1, 2, 3 . . . 9).

`[0-5]` will generate all the values 0 - 5.

`[00-]` will generate all the values 00 - 99 (00, 01, 02, 03. . . 97, 98, 99).

`[00-50]` will generate all the values 0 - 50.

`[A-]` will generate all the values A - Z (A, B, C . . . Z).

`[A-K]` will generate all the values A - K.

`[AA-]` will generate all the values AA - ZZ (AA, AB, AC. . . ZX, ZY, ZZ).

`[AA-KK]` will generate all the values AA - KK.

When IxLoad has generated the final string, if the test configuration requires additional strings, IxLoad returns to the starting value of the variable and continues to generate strings until no more are required. In this case, the generated strings will not be unique.

For example, if a SIP test requires 256 user names and the `Username` field is configured as:

`User[00-]`

IxLoad generates the strings User00 - User99, then repeats and again generates strings User00 - User99, then generates the final group of strings User00 - User56.

IxLoad generates the SIP Username, Password, and Domain fields simultaneously and associates one value from each to form each user name–password– domain combination used in the test.

For example, the first generated user name will be associated with the first generated password and the first generated domain. The second generated user name will be associated with the second generated password and the second generated domain, and so on until all the necessary strings have been generated.

If a SIP Username, Password, and Domain fields contain variables while the remaining fields contain a fixed value (no variable), IxLoad associates the identical value from the fixed field to all the generated values.

See the following example:

Field Values	Associated Strings
Username = User[00-]	User00 + PassAA + Domainaa
Password = Pass[AZ-]	User01 + PassAB + Domainab
Domain = Domain[az-]	User02 + PassAC + Domainac
	...
Username = User[00-]	User00 + PassAA + MyDomain
Password = Pass[AZ-]	User01 + PassAB + MyDomain
Domain = MyDomain	User02 + PassAC + MyDomain
	...
Username = User[00-]	User00 + Pass + MyDomain
Password = Pass	User01 + Pass + MyDomain
Domain = MyDomain	User02 + Pass + MyDomain
	...



Bulk SIP Statistics

For the Bulk SIP statistics, see the following:

Bulk SIP Client Statistics (on page 950)

Bulk SIP Server Statistics (on page 963)

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program
Files\Ixia\IxLoad\Results\\<repository>_<testconfiguration>_<yyyy><mm><
dd>_<hh><mm><ss>
```

Bulk SIP Client Statistics

The table below lists the Bulk SIP Client Statistics.

Statistic	Description
Call-related Statistics	
SIP calls initiated	Number of SIP calls initiated.
SIP calls completed	Number of SIP calls established.
SIP calls active	Number of SIP calls active.
Transaction-related Statistics	
SIP INVITE client transactions initiated	Number of SIP INVITE transactions initiated by the client.
SIP INVITE client transactions succeeded	Number of INVITE transactions initiated by the client that succeeded.
SIP INVITE client transactions failed	Number of INVITE transactions initiated by the client that failed for all reasons.
SIP INVITE client transactions failed (TIMER B)	Number of INVITE transactions initiated by the client that failed because Timer B (transaction timeouts timer) expired.
SIP INVITE client transactions failed (TRANSPORT ERROR)	Number of INVITE transactions initiated by the client that failed due to TCP or UDP errors.
SIP INVITE client transactions failed (TRANSACTION TIMEOUT TIMER)	Number of INVITE transactions initiated by the client that failed because the transaction timeout timer expired.

SIP INVITE client transactions failed (5xx)	Number of INVITE transactions initiated by the client that failed due to 5xx-series (server error) errors.
SIP NON-INVITE client transactions initiated	Number of SIP NON-INVITE transactions initiated by the client.
SIP NON-INVITE client transactions succeeded	Number of SIP NON-INVITE transactions initiated by the client that succeeded.
SIP NON-INVITE client transactions failed	Number of SIP NON-INVITE transactions initiated by the client that failed.
SIP NON-INVITE client transactions failed (TIMER F)	Number of NON-INVITE transactions initiated by the client that failed because Timer F (non-INVITE transaction timeout timer) expired.
SIP NON-INVITE client transactions failed (TRANSPORT ERROR)	Number of NON-INVITE transactions initiated by the client that failed due to TCP or UDP errors.
Message-related Statistics	
SIP INVITE requests sent	Number of SIP INVITE messages sent by the client.
SIP ACK requests sent	Number of SIP ACK messages sent by the client.
SIP BYE requests sent	Number of SIP BYE messages sent by the client.
SIP REGISTER requests sent	Number of SIP REGISTER messages sent by the client.
SIP INVITE messages retransmitted	Number of INVITE messages initiated by the client that had to be re-transmitted.
SIP NON-INVITE requests retransmitted	Number of NON-INVITE transactions initiated by the client that had to be re-transmitted.
SIP INVITE requests unexpected	Number of SIP INVITE requests that the client did not expect to receive.
SIP ACK requests unexpected	Number of SIP ACK requests that the client did not expect to receive.
SIP BYE requests unexpected	Number of SIP BYE requests that the client did not expect to receive.
SIP CANCEL requests unexpected	Number of SIP CANCEL requests that the client did not expect to receive.
SIP UNKNOWN messages unexpected	Number of SIP UNKNOWN messages that the client did not expect to receive.
SIP UNKNOWN requests unexpected	Number of SIP UNKNOWN requests that the client did not expect to receive.
SIP 1xx responses expected	Number of SIP 1xx-series responses the client received that it expected.
SIP 1xx responses unexpected	Number of SIP 1xx-series responses the client received that it did not expect.



SIP 2xx responses expected	Number of SIP 2xx-series responses the client received that it expected.
SIP 2xx responses unexpected	Number of SIP 2xx-series responses the client received that it did not expect.
SIP 3xx responses expected	Number of SIP 3xx-series responses the client received that it expected.
SIP 3xx responses unexpected	Number of SIP 3xx-series responses the client received that it did not expect.
SIP 4xx responses expected	Number of SIP 4xx-series responses the client received that it expected.
SIP 4xx responses unexpected	Number of SIP 4xx-series responses the client received that it did not expect.
SIP 5xx responses expected	Number of SIP 5xx-series responses the client received that it expected.
SIP 5xx responses unexpected	Number of SIP 5xx-series responses the client received that it did not expect.
SIP 6xx responses expected	Number of SIP 6xx-series responses the client received that it expected.
SIP 6xx responses unexpected	Number of SIP 6xx-series responses the client received that it did not expect.
RTP: Global Stream Transmit Statistics	
RTP Bytes Sent	Total number of bytes sent, including header and payload.
RTP Packets Sent	Total number of packets sent.
RTP Tx Jitter (ns)	Average amount of transmit jitter, in nanoseconds.
RTP Tx Packets Dropped	Number of packets transmitted by the client that were dropped.
RTP: Global Stream Statistics	
RTP Dropped Packets	Number of RTP packets dropped.
RTP Bytes Received	Number of RTP bytes received.
RTP Packets Received	Number of RTP packets received.
RTP Payload Bytes Received	Number bytes received in RTP payloads.
RTP Bad Packets Received	Number of defective RTP packets received.
RTP Lost Packets	Number of packets lost.
RTP Misordered Packets Received	Number of packets received out of order.
RTP Duplicate Packets Received	Number of duplicate packets received.
RTP Jitter Min	Smallest amount of jitter detected.
RTP Jitter Max	Largest amount of jitter detected.
RTP Packets With Jitter Up To 1ms	Packets received with jitter of up to 1ms.
RTP Packets With Jitter Up To 3ms	Packets received with jitter of 1-3ms.

RTP Packets With Jitter Up To 5ms	Packets received with jitter of 3-5ms.
RTP Packets With Jitter Up To 10ms	Packets received with jitter of 5-10ms.
RTP Packets With Jitter Up To 20ms	Packets received with jitter of 10-20ms
RTP Packets With Jitter Up To 40ms	Packets received with jitter of 20-40ms
RTP Packets With Jitter More Than 40ms	Packets received with jitter of more than 40ms.
RTP DTMF Digits Detected	Total number of path confirmation DTMF tone sequences received.
RTP DTMF Digits Matched	Number of DTMF sequences received that matched the sequence specified on the Detect DTMF command. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP DTMF Digits Not Matched	Number of DTMF sequences received that did not match the sequence specified on the Detect DTMF command. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP Good DTMF Sequences Detected	Total number of correct path confirmation DTMF tone sequences received.
RTP Bad DTMF Sequences Detected	Total number of incorrect path confirmation DTMF tone sequences received.
RTP DTMF Detection Timeout	Number of DTMF detection attempts (by the Detect DTMF command) that ended because one of the timeout timers expired.
RTP DTMF Digits Sent	Number of DTMF digits sent by Generate DTMF commands. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP DTMF Sequences Sent	Number of DTMF sequences sent by Generate DTMF commands. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP MF Digits Sent	Number of MF digits sent by Generate MF commands.
RTP MF Sequences Sent	Number of MF sequences sent by Generate MF commands.
RTP Custom Tones Sent	Number of custom tones sent by Generate Tone commands.
RTP Packets Dropped By Jitter Buffer	Number of packet dropped from the jitter buffer because they arrived later than expected.
Note: In the CSV files, global MOS scores are represented as whole numbers (for example, "345"); in StatViewer (they are represented as floating-point numbers (for example, "3.45").	



RTP MOS Average Instant	Average MOS score at the time of the sampling interval.
RTP MOS Worst Instant	Lowest MOS score at the time of the sampling interval.
RTP MOS Best Instant	Highest MOS score at the time of the sampling interval.
RTP MOS Worst	Lowest MOS score recorded during the test.
RTP MOS Best	Highest MOS score recorded during the test.
RTP MOS Average Per Call	Average MOS score per call.
RTP MOS Worst Per Call	Lowest MOS score per call.
RTP MOS Best Per Call	Highest MOS score per call.
RTP Calls With Continuous Path Confirmation	Number of calls on which path confirmation continued throughout the call.
RTP Calls With Interrupted Path Confirmation	Number of calls on which path confirmation was interrupted during the call.
RTP Calls Without Path Confirmation	Number of calls on which there was no path confirmation.
Transport Statistics	
SIP Bytes Transmitted	Total number of SIP bytes transmitted.
SIP Bytes Received	Total number of SIP bytes received.
SIP Signaling UDP Packets Transmitted	Number UDP packets transmitted for SIP signaling purposes.
SIP Signaling UDP Packets Received	Number UDP packets received for SIP signaling purposes.
Per-Stream Statistics	
RTP Path Confirmation Status	Status of path confirmation on the stream. Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
RTP MOS	Average MOS score recorded on the stream.
RTP Worst MOS	Lowest MOS score recorded on the stream.
RTP Best MOS	Highest MOS score recorded on the stream.
RTP Bytes	Number of bytes transmitted on the stream.
RTP Packets	Number of packets transmitted on the stream.
RTP Bad Packets	Number of bad packets transmitted on the stream.
RTP Lost Packets	Number of packets lost on the stream.
RTP Missorder Packets	Number of packets received out of order on the stream.
RTP Duplicate Packets	Number of duplicate packets received on the stream.

RTP Packets With Jitter Up To 1ms	Number of packets received on the stream with jitter up to 1 millisecond.
RTP Packets With Jitter Up To 3ms	Number of packets received on the stream with jitter up to 3 milliseconds.
RTP Packets With Jitter Up To 5ms	Number of packets received on the stream with jitter up to 5 milliseconds.
RTP Packets With Jitter Up To 10ms	Number of packets received on the stream with jitter up to 10 milliseconds.
RTP Packets With Jitter Up To 20ms	Number of packets received on the stream with jitter up to 20 milliseconds.
RTP Packets With Jitter Up To 40ms	Number of packets received on the stream with jitter up to 40 milliseconds.
RTP Packets With Jitter More Than 40ms	Number of packets received on the stream with jitter over 40 milliseconds.
RTP Average Jitter (ns)	Average jitter, in nanoseconds.
RTP Min Jitter (ns)	Lowest jitter recorded, in nanoseconds.
RTP Max Jitter (ns)	Largest jitter recorded, in nanoseconds.
RTP DTMFs Detected	Total number of path confirmation DTMF tone sequences sent.
RTP DTMFs Matched	Number of DTMF sequences received that matched the sequence specified on the Detect DTMF command. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP DTMFs Not Matched	Number of DTMF sequences received that did not match the sequence specified on the Detect DTMF command. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP Good DTMF Sequences Detected	Total number of correct path confirmation DTMF tone sequences received.
RTP Bad DTMF Sequences Detected	Total number of incorrect path confirmation DTMF tone sequences received.
RTP DTMF Detection Timeout	Number of DTMF detection attempts (by the Detect DTMF command) that ended because one of the timeout timers expired.
RTP DTMF Digits Sent	Number of DTMF digits sent by Generate DTMF commands. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP DTMF Sequences Sent	Number of DTMF sequences sent by Generate DTMF commands. This statistic is not related to the option to perform path confirmation using DTMF digits.



RTP MF Digits Sent	Number of MF digits sent by Generate MF commands.
RTP MF Sequences Sent	Number of MF sequences sent by Generate MF commands.
RTP Custom Tones Sent	Number of custom tones sent by Generate Tone commands.
RTP Packets Dropped By Jitter Buffer	Total number of packets dropped from the jitter buffer because they were received late.
Video Statistics	
Video Total Bytes Sent	Total video bytes sent by the server.
Video Total Packets Sent	Total video packets sent by the server.
Video Tx Jitter (ns)	Variation in video packet transmission times, in nanoseconds.
Video Tx Packets Dropped	Number of video packets dropped before transmission.
Video Global Stream Statistics	
Video Frame Stats Disabled	<p>Initially, this statistic displays no value.</p> <p>If the received data rate exceeds the cut-off threshold, IxLoad stops computing the I-, P-, and B-frame statistics and this statistic will display "YES".</p> <p>The value will remain YES until the end of the iteration. Once frame statistics computation is disabled during a run, it remains disabled throughout the remainder of the run.</p> <p>Prior to starting the next run (or the next iteration of the same test), this statistic will be cleared and IxLoad will again begin computing the frame statistics. It will continue to compute the frame statistics as long as the bit rate remains below the cut-off threshold.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>

Video Quality Metrics Disabled	<p>Initially, this statistic displays no value.</p> <p>If the received data rate exceeds the cut-off threshold, IxLoad stops computing the Quality Metrics, and this statistic will display “YES”.</p> <p>The value will remain YES until the end of the iteration. Once the Quality Metrics computation is disabled during a run, it remains disabled throughout the remainder of the run.</p> <p>Prior to starting the next run (or the next iteration of the same test), this statistic will be cleared and IxLoad will again begin computing the Quality Metrics. It will continue to compute the metrics as long as the bit rate remains below the cut-off threshold.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Video Total Bytes Rcvd	Total number of video bytes received by the client.
Video Total packets Rcvd	Total number of video packets received by the client.
Video Total Loss	Total number video packets lost.
Video Unexpected UDP Packets Received	Number of UDP video packets received packets during a time when no channels are active.
Video Overload Packets Dropped	Number of RTP video packets dropped because a port did not have enough computing power to process them.
Video Total RTP Packets Lost	Total number of RTP video packets lost while using RTP over UDP transport.
Video Total Out Of Order RTP Packets	Total number of RTP video packets received in the wrong order while using RTP over UDP transport.
Video Total Duplicate RTP Packets	Total number of duplicate video RTP packets received.
Video Global Jitter	<p>Average variation in arrival times of video packets on all streams.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
Video Jitter less than 50 us	Number of video packets received with 0 to 50 microseconds of jitter.
Video Jitter between 50 - 100 us	Number of video packets received with 50 to 100 microseconds of jitter.
Video Jitter between 100 - 500 us	Number of video packets received with 100 - 500 microseconds of jitter.



Video Jitter between 500 us - 2 ms	Number of video packets received with 500 microseconds to 2 milliseconds of jitter.
Video Jitter between 2 - 5 ms	Number of video packets received with 2 to 5 milliseconds of jitter.
Video Jitter between 5 - 10 ms	Number of video packets received with 5 to 10 milliseconds of jitter.
Video Jitter greater than 10 ms	Number of video packets received with more than 10 milliseconds of jitter.
Video Inter Packet Arrival Time between 0 - 2 ms	Number of video packets that arrived less than 2 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 2 - 5 ms	Number of video packets that arrived between 2 and 5 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 5 - 10 ms	Number of video packets that arrived between 5 and 10 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 10 - 25 ms	Number of video packets that arrived between 10 and 25 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 25 - 50 ms	Number of video packets that arrived between 25 and 50 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 50 - 100 ms	Number of video packets that arrived between 50 and 100 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 100 - 200 ms	Number of video packets that arrived between 100 and 200 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 200 - 500 ms	Number of video packets that arrived between 200 and 500 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time greater than 500 ms	Number of video packets that arrived more than 500 milliseconds after the preceding packet was received.
Video Per-Stream Statistics	
Video Active	Indicates whether the video stream is active or not: 0 = inactive 1 = active Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
Video Stream Name	Name of video stream. Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .

Video Flow ID	<p>Number identifying the flow used by the video stream.</p> <p>A flow consists of the packets flowing between a source IP:port and a destination IP:port.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Video Transport	<p>Type of transport used on the video stream.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Video Codec	<p>Video codec used on the video stream.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Video Stream Bit Rate	Bit rate used on video stream.
Video MDI-DF	Media Delay Index Delay Factor (MDI-DF) experienced on video stream.
Video MIN MDI-DF	Smallest MDI Delay Factor experienced on video stream.
Video MAX MDI-DF	Largest MDI Delay Factor experienced on video stream.
Video AVG-MDI-DF	Average MDI Delay Factor experienced on video stream.
Video MDI-MLR	Media Delay Index Media Loss Rate experienced on video stream.
Video Bytes	Number of bytes received on the video stream.
Video I Frames Rcvd	<p>Number of I-frames received on the video stream.</p> <p>An I-frame is encoded with no reference to any previous or subsequent frames.</p>
Video P Frames Rcvd	<p>Number of P-frames received on the video stream.</p> <p>A P-frame is encoded relative to the previous reference frame.</p>
Video B Frames Rcvd	<p>Number of B-frames received on the video stream.</p> <p>A B-frame is encoded relative to the previous reference frame, the subsequent reference frame, or both</p>
Video Packets	Number of packets received on the video stream.
Video Loss	Number of packets lost on the video stream.



Video Jitter	Number of packets with jitter received on the video stream.
Video Inter Pkt Arrival Time	Amount of time between received video packets, in milliseconds.
Video Min Inter Pkt Arrival Time	Smallest amount of time between received video packets, in milliseconds.
Video Max Inter Pkt Arrival Time	Largest amount of time between received video packets, in milliseconds.
Video Packet Latency (ns)	Average packet latency on the video stream.
Video Min Packet Latency (ns)	Smallest packet latency on the video stream.
Video Max Packet Latency (ns)	Longest packet latency on the video stream.
Video Join Latency (ms)	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP JOIN (broadcast channel) or RTSP PLAY (VoD channel) and the time it received the first byte of video data.
Video I Join Latency (ms)	<p>Amount of time, in milliseconds, elapsed between the time the client sent an IGMP JOIN (broadcast channel) or RTSP PLAY (VoD channel) and the time it received the first I frame.</p> <p>This statistic is computed for MPEG2 transport streams carrying MPEG2 video data.</p>
Video Leave Latency (ms)	<p>Amount of time, in milliseconds, elapsed between the time the client sent an IGMP LEAVE (broadcast channel) or RTSP PAUSE (VoD channel) and the time it received the last byte of video data.</p> <p>Leave latency has a maximum timeout of 10 seconds; if the client continues to receive data 10 seconds after it has sent the Leave command, the latency is measured as 10 seconds.</p>
Video Channel Switch Latency	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP LEAVE (broadcast channel) or RTSP PAUSE (VoD channel) to stop receiving one video stream and the time it received the first byte of data of a new video stream.
Video Channel Gap Duration	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP LEAVE (broadcast channel) and received the last byte of the old video stream and the time it received the first byte of data of a new video stream.

Video Channel Overlap Duration	Amount of time, in milliseconds, elapsed after sending an IGMP LEAVE (broadcast channel) that the client was simultaneously receiving both the old and new video streams.
Video Control Sent	Indicates the type of video control command that has most recently been sent: 0 = LEAVE or PAUSE/TEARDOWN sent 1 = JOIN or PLAY sent
Video Data Rcvd	Indicates whether or not video data is being received: 0 = no data received 1 = data received
Video RTP Packets Lost	Number of RTP video packets lost.
Video RTP Packets Out of Order	Number of RTP video packets received out of order.
Video RTP Packets Duplicated	Number of duplicate RTP video packets received.
Video Quality Statistics	
Video JB Packets Accepted	Number of video packets accepted into the jitter buffer.
Video JB Packets Early	Number of video packets that arrived earlier than expected in the jitter buffer.
Video JB Packets Discarded	Total number of video packets that were discarded. This statistic is the total of: JB Packets Discarded (Underrun) and JB Packets Discarded (Overrun).
Video JB Packets Discarded (Underrun)	Number of video packets discarded because they arrived after their expected time slot.
Video JB Packets Discarded (Overrun)	Number of video packets discarded because the jitter buffer was full.
Video MOS_V	Mean Opinion Score for Video. This score is computed from the Video Service Quality statistic to create a zero-to-five (0-5) assessment of the quality of the video stream.
Video Service Quality	A factor in the range from 0 to 120, which provides an assessment of the capability of the RTP channel to support video transmission.

Video Gap Video Service Quality	<p>Video Service Quality during the Gap state.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Burst Video Service Quality	<p>Video Service Quality during the Burst state.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Burst Count	<p>Number of times the stream entered the Burst state.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Avg Gap Len (Pkts)	<p>The average gap length, in packets.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Avg Burst Len (Pkts)	<p>The average burst length, in packets.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Degradation (Loss)	<p>The amount of the overall video quality degradation that can be attributed to packet loss.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Degradation (Discard)	<p>The amount of the overall video quality degradation that can be attributed to packets being discarded from the jitter buffer.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>

Video Degradation (Video Codec)	The amount of the overall quality degradation that can be attributed to video codec selection. This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).
Video Current JB Packets Accepted	Number of video packets accepted into the jitter buffer during the current statistics Update Interval.
Video Current JB Packets Discarded	Number of video packets discarded from the jitter buffer during the current statistics Update Interval.
Video Current JB Packets Lost	Number of video packets lost during the current statistics Update Interval.
Video Current Video Service Quality	Video Service Quality during the current statistics update interval.

Bulk SIP Server Statistics

The table below lists the Bulk SIP Server Statistics.

Statistic	Description
Call-related Statistics	
SIP calls received	Number of SIP calls received.
SIP calls completed	Number of SIP calls completed.
SIP calls active	Number of SIP calls active.
Transaction-related Statistics	
SIP INVITE server transactions received	Number of INVITE transactions received by the server.
SIP INVITE server transactions succeeded	Number of INVITE transactions received by the server that succeeded.
SIP INVITE server transactions failed	Number of INVITE transactions received by the server that failed for all reasons.
SIP INVITE server transactions failed (TIMER H)	Number of INVITE transactions initiated by the server that failed because Timer H (wait time for ACK receipt) expired.
SIP INVITE server transactions failed (TRANSPORT ERROR)	Number of INVITE transactions initiated by the server that failed due to TCP or UDP errors.



SIP NON-INVITE server transactions received	Number of NON-INVITE transactions received by the server.
SIP NON-INVITE server transactions succeeded	Number of SIP NON-INVITE transactions initiated by the server that succeeded.
SIP NON-INVITE server transactions failed	Number of SIP NON-INVITE transactions initiated by the server that failed.
Message-related Statistics	
SIP NON-INVITE requests retransmitted	Number of NON-INVITE requests that were re-transmitted.
SIP REGISTER Requests sent	Number of REGISTER requests sent.
SIP 1xx responses expected	Number of 100-series responses that the server expected to receive.
SIP 2xx responses expected	Number of 200-series responses that the server expected to receive.
SIP 3xx responses expected	Number of 300-series responses that the server expected to receive.
SIP 4xx responses expected	Number of 400-series responses that the server expected to receive.
SIP 5xx responses expected	Number of 500-series responses that the server expected to receive.
SIP 6xx responses expected	Number of 600-series responses that the server expected to receive.
SIP 300-699 responses retransmitted	Number of 3xx- to 6xx-series responses that had to be retransmitted by the server.
SIP INVITE requests expected	Number of INVITE requests that the server expected to receive.
SIP ACK requests expected	Number of ACK requests that the server expected to receive.
SIP BYE requests expected	Number of BYE requests that the server expected to receive.
SIP 1xx responses sent	Number of 1xx-series responses sent by the server.
SIP 1xx responses unexpected	Number of 1xx-series responses sent by the server that it did not expect to send.
SIP 2xx responses sent	Number of 2xx-series responses sent by the server.
SIP 2xx responses unexpected	Number of 2xx-series responses sent by the server that it did not expect to send.
SIP 3xx responses sent	Number of 3xx-series responses sent by the server.
SIP 3xx responses unexpected	Number of 3xx-series responses sent by the server that it did not expect to send.
SIP 4xx responses sent	Number of 4xx-series responses sent by the server.

SIP 4xx responses unexpected	Number of 4xx-series responses sent by the server that it did not expect to send.
SIP 5xx responses sent	Number of 5xx-series responses sent by the server.
SIP 5xx responses unexpected	Number of 5xx-series responses sent by the server that it did not expect to send.
SIP 6xx responses sent	Number of 6xx-series responses sent by the server.
SIP 6xx responses unexpected	Number of 6xx-series responses sent by the server that it did not expect to send.
SIP INVITE requests unexpected	Number of SIP INVITE requests that the server did not expect to receive.
SIP ACK requests unexpected	Number of SIP ACK requests that the server did not expect to receive.
SIP BYE requests unexpected	Number of SIP BYE requests that the server did not expect to receive.
SIP CANCEL requests unexpected	Number of SIP CANCEL requests that the server did not expect to receive.
SIP UNKNOWN requests unexpected	Number of SIP requests that the server did not expect to receive.
SIP UNKNOWN messages unexpected	Number of SIP messages that the server sent that it did not expect to send.
RTP: Global Stream Transmit Statistics	
RTP Bytes Sent	Total number of bytes sent, including header and payload.
RTP Packets Sent	Total number of packets sent.
RTP Tx Jitter (ns)	Average amount of transmit jitter, in nanoseconds.
RTP Tx Packets Dropped	Number of packets transmitted by the client that were dropped.
RTP: Global Stream Statistics	
RTP Dropped Packets	Number of RTP packets dropped.
RTP Bytes Received	Number of RTP bytes received.
RTP Packets Received	Number of RTP packets received.
RTP Payload Bytes Received	Number bytes received in RTP payloads.
RTP Bad Packets Received	Number of defective RTP packets received.
RTP Lost Packets	Number of packets lost.
RTP Misordered Packets Received	Number of packets received out of order.
RTP Duplicate Packets Received	Number of duplicate packets received.
RTP Jitter Min	Smallest amount of jitter detected.
RTP Jitter Max	Largest amount of jitter detected.



RTP Packets With Jitter Up To 1ms	Packets received with jitter of up to 1ms.
RTP Packets With Jitter Up To 3ms	Packets received with jitter of 1-3ms.
RTP Packets With Jitter Up To 5ms	Packets received with jitter of 3-5ms.
RTP Packets With Jitter Up To 10ms	Packets received with jitter of 5-10ms.
RTP Packets With Jitter Up To 20ms	Packets received with jitter of 10-20ms
RTP Packets With Jitter Up To 40ms	Packets received with jitter of 20-40ms
RTP Packets With Jitter More Than 40ms	Packets received with jitter of more than 40ms.
RTP DTMFs Detected	Total number of path confirmation DTMF tone sequences sent.
RTP DTMFs Matched	Number of DTMF sequences received that matched the sequence specified on the Detect DTMF command. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP DTMFs Not Matched	Number of DTMF sequences received that did not match the sequence specified on the Detect DTMF command. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP Good DTMF Sequences Detected	Total number of correct path confirmation DTMF tone sequences received.
RTP Bad DTMF Sequences Detected	Total number of incorrect path confirmation DTMF tone sequences received.
RTP DTMF Detection Timeout	Number of DTMF detection attempts (by the Detect DTMF command) that ended because one of the timeout timers expired.
RTP DTMF Digits Sent	Number of DTMF digits sent by Generate DTMF commands. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP DTMF Sequences Sent	Number of DTMF sequences sent by Generate DTMF commands. This statistic is not related to the option to perform path confirmation using DTMF digits.
RTP MF Digits Sent	Number of MF digits sent by Generate MF commands.
RTP MF Sequences Sent	Number of MF sequences sent by Generate MF commands.
RTP Custom Tones Sent	Number of custom tones sent by Generate Tone commands.
RTP Packets Dropped By Jitter Buffer	Number of packet dropped from the jitter buffer because they arrived later than expected.

Note: In the CSV files, global MOS scores are represented as whole numbers (for example, “345”); in StatViewer (they are represented as floating-point numbers (for example, “3.45”).	
RTP MOS Average Instant	Average MOS score at the time of the sampling interval.
RTP MOS Worst Instant	Lowest MOS score at the time of the sampling interval.
RTP MOS Best Instant	Highest MOS score at the time of the sampling interval.
RTP MOS Worst	Lowest MOS score recorded during the test.
RTP MOS Best	Highest MOS score recorded during the test.
RTP MOS Average Per Call	Average MOS score per call.
RTP MOS Worst Per Call	Lowest MOS score per call.
RTP MOS Best Per Call	Highest MOS score per call.
RTP Calls With Continuous Path Confirmation	Number of calls on which path confirmation continued throughout the call.
RTP Calls With Interrupted Path Confirmation	Number of calls on which path confirmation was interrupted during the call.
RTP Calls Without Path Confirmation	Number of calls on which there was no path confirmation.
Transport Statistics	
SIP Bytes Transmitted	Total number of SIP bytes transmitted.
SIP Bytes Received	Total number of SIP bytes received.
SIP Signaling UDP Packets Transmitted	Number UDP packets transmitted for SIP signaling purposes.
SIP Signaling UDP Packets Received	Number UDP packets received for SIP signaling purposes.
Per-Stream Statistics	
RTP Path Confirmation Status	Status of path confirmation on the stream. Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
RTP MOS	Average MOS score recorded on the stream.
RTP Worst MOS	Lowest MOS score recorded on the stream.
RTP Best MOS	Highest MOS score recorded on the stream.
RTP Bytes	Number of bytes transmitted on the stream.
RTP Packets	Number of packets transmitted on the stream.
RTP Bad Packets	Number of bad packets transmitted on the stream.
RTP Lost Packets	Number of packets lost on the stream.
RTP Missorder Packets	Number of packets received out of order on the stream.

RTP Duplicate Packets	Number of duplicate packets received on the stream.
RTP Packets With Jitter Up To 1ms	Number of packets received on the stream with jitter up to 1 millisecond.
RTP Packets With Jitter Up To 3ms	Number of packets received on the stream with jitter up to 3 milliseconds.
RTP Packets With Jitter Up To 5ms	Number of packets received on the stream with jitter up to 5 milliseconds.
RTP Packets With Jitter Up To 10ms	Number of packets received on the stream with jitter up to 10 milliseconds.
RTP Packets With Jitter Up To 20ms	Number of packets received on the stream with jitter up to 20 milliseconds.
RTP Packets With Jitter Up To 40ms	Number of packets received on the stream with jitter up to 40 milliseconds.
RTP Packets With Jitter More Than 40ms	Number of packets received on the stream with jitter over 40 milliseconds.
RTP Average Jitter (ns)	Average jitter, in nanoseconds.
RTP Min Jitter (ns)	Lowest jitter recorded, in nanoseconds.
RTP Max Jitter (ns)	Largest jitter recorded, in nanoseconds.
RTP DTMFs Detected	Total number of path confirmation DTMF tone sequences sent.
RTP Good DTMF Sequences Detected	Total number of correct path confirmation DTMF tone sequences received.
RTP DTMFs Matched	<p>Number of DTMF sequences received that matched the sequence specified on the Detect DTMF command.</p> <p>This statistic is not related to the option to perform path confirmation using DTMF digits.</p>
RTP DTMFs Not Matched	<p>Number of DTMF sequences received that did not match the sequence specified on the Detect DTMF command.</p> <p>This statistic is not related to the option to perform path confirmation using DTMF digits.</p>
RTP Good DTMF Sequences Detected	Total number of correct path confirmation DTMF tone sequences received.
RTP Bad DTMF Sequences Detected	Total number of incorrect path confirmation DTMF tone sequences received.
RTP DTMF Detection Timeout	Number of DTMF detection attempts (by the Detect DTMF command) that ended because one of the timeout timers expired.
RTP DTMF Digits Sent	<p>Number of DTMF digits sent by Generate DTMF commands.</p> <p>This statistic is not related to the option to perform path confirmation using DTMF digits.</p>

RTP DTMF Sequences Sent	<p>Number of DTMF sequences sent by Generate DTMF commands.</p> <p>This statistic is not related to the option to perform path confirmation using DTMF digits.</p>
RTP MF Digits Sent	Number of MF digits sent by Generate MF commands.
RTP MF Sequences Sent	Number of MF sequences sent by Generate MF commands.
RTP Custom Tones Sent	Number of custom tones sent by Generate Tone commands.
RTP Packets Dropped By Jitter Buffer	Total number of packets dropped from the jitter buffer because they were received late.
Video Statistics	
Video Total Bytes Sent	Total video bytes sent by the server.
Video Total Packets Sent	Total video packets sent by the server.
Video Tx Jitter (ns)	Variation in video packet transmission times, in nanoseconds.
Video Tx Packets Dropped	Number of video packets dropped before transmission.
Video Global Stream Statistics	
Video Frame Stats Disabled	<p>Initially, this statistic displays no value.</p> <p>If the received data rate exceeds the cut-off threshold, IxLoad stops computing the I-, P-, and B-frame statistics and this statistic will display "YES".</p> <p>The value will remain YES until the end of the iteration. Once frame statistics computation is disabled during a run, it remains disabled throughout the remainder of the run.</p> <p>Prior to starting the next run (or the next iteration of the same test), this statistic will be cleared and IxLoad will again begin computing the frame statistics. It will continue to compute the frame statistics as long as the bit rate remains below the cut-off threshold.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>

Video Quality Metrics Disabled	<p>Initially, this statistic displays no value.</p> <p>If the received data rate exceeds the cut-off threshold, IxLoad stops computing the Quality Metrics, and this statistic will display “YES”.</p> <p>The value will remain YES until the end of the iteration. Once the Quality Metrics computation is disabled during a run, it remains disabled throughout the remainder of the run.</p> <p>Prior to starting the next run (or the next iteration of the same test), this statistic will be cleared and IxLoad will again begin computing the Quality Metrics. It will continue to compute the metrics as long as the bit rate remains below the cut-off threshold.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Video Total Bytes Rcvd	Total number of video bytes received by the client.
Video Total packets Rcvd	Total number of video packets received by the client.
Video Total Loss	Total number video packets lost.
Video Unexpected UDP Packets Received	Number of UDP video packets received packets during a time when no channels are active.
Video Overload Packets Dropped	Number of RTP video packets dropped because a port did not have enough computing power to process them.
Video Total RTP Packets Lost	Total number of RTP video packets lost while using RTP over UDP transport.
Video Total Out Of Order RTP Packets	Total number of RTP video packets received in the wrong order while using RTP over UDP transport.
Video Total Duplicate RTP Packets	Total number of duplicate video RTP packets received.
Video Global Jitter	<p>Average variation in arrival times of video packets on all streams.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
Video Jitter less than 50 us	Number of video packets received with 0 to 50 microseconds of jitter.
Video Jitter between 50 - 100 us	Number of video packets received with 50 to 100 microseconds of jitter.
Video Jitter between 100 - 500 us	Number of video packets received with 100 - 500 microseconds of jitter.

Video Jitter between 500 us - 2 ms	Number of video packets received with 500 microseconds to 2 milliseconds of jitter.
Video Jitter between 2 - 5 ms	Number of video packets received with 2 to 5 milliseconds of jitter.
Video Jitter between 5 - 10 ms	Number of video packets received with 5 to 10 milliseconds of jitter.
Video Jitter greater than 10 ms	Number of video packets received with more than 10 milliseconds of jitter.
Video Inter Packet Arrival Time between 0 - 2 ms	Number of video packets that arrived less than 2 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 2 - 5 ms	Number of video packets that arrived between 2 and 5 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 5 - 10 ms	Number of video packets that arrived between 5 and 10 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 10 - 25 ms	Number of video packets that arrived between 10 and 25 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 25 - 50 ms	Number of video packets that arrived between 25 and 50 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 50 - 100 ms	Number of video packets that arrived between 50 and 100 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 100 - 200 ms	Number of video packets that arrived between 100 and 200 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time between 200 - 500 ms	Number of video packets that arrived between 200 and 500 milliseconds after the preceding packet was received.
Video Inter Packet Arrival Time greater than 500 ms	Number of video packets that arrived more than 500 milliseconds after the preceding packet was received.
Video Per-Stream Statistics	
Video Active	Indicates whether the video stream is active or not: 0 = inactive 1 = active Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
Video Stream Name	Name of video stream. Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .

Video Flow ID	<p>Number identifying the flow used by the video stream.</p> <p>A flow consists of the packets flowing between a source IP:port and a destination IP:port.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Video Transport	<p>Type of transport used on the video stream.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Video Codec	<p>Video codec used on the video stream.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Video Stream Bit Rate	<p>Bit rate used on video stream.</p>
Video MDI-DF	<p>Media Delay Index Delay Factor (MDI-DF) experienced on video stream.</p>
Video MIN MDI-DF	<p>Smallest MDI Delay Factor experienced on video stream.</p>
Video MAX MDI-DF	<p>Largest MDI Delay Factor experienced on video stream.</p>
Video AVG-MDI-DF	<p>Average MDI Delay Factor experienced on video stream.</p>
Video MDI-MLR	<p>Media Delay Index Media Loss Rate experienced on video stream.</p>
Video Bytes	<p>Number of bytes received on the video stream.</p>
Video I Frames Rcvd	<p>Number of I-frames received on the video stream.</p> <p>An I-frame is encoded with no reference to any previous or subsequent frames.</p>
Video P Frames Rcvd	<p>Number of P-frames received on the video stream.</p> <p>A P-frame is encoded relative to the previous reference frame.</p>
Video B Frames Rcvd	<p>Number of B-frames received on the video stream.</p> <p>A B-frame is encoded relative to the previous reference frame, the subsequent reference frame, or both.</p>
Video Packets	<p>Number of packets received on the video stream.</p>
Video Loss	<p>Number of packets lost on the video stream.</p>
Video Jitter	<p>Number of packets with jitter received on the video stream.</p>

Video Inter Pkt Arrival Time	Amount of time between received video packets, in milliseconds.
Video Min Inter Pkt Arrival Time	Smallest amount of time between received video packets, in milliseconds.
Video Max Inter Pkt Arrival Time	Largest amount of time between received video packets, in milliseconds.
Video Packet Latency (ns)	Average packet latency on the video stream.
Video Min Packet Latency (ns)	Smallest packet latency on the video stream.
Video Max Packet Latency (ns)	Longest packet latency on the video stream.
Video Join Latency (ms)	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP JOIN (broadcast channel) or RTSP PLAY (VoD channel) and the time it received the first byte of video data.
Video I Join Latency (ms)	<p>Amount of time, in milliseconds, elapsed between the time the client sent an IGMP JOIN (broadcast channel) or RTSP PLAY (VoD channel) and the time it received the first I frame.</p> <p>This statistic is computed for MPEG2 transport streams carrying MPEG2 video data.</p>
Video Leave Latency (ms)	<p>Amount of time, in milliseconds, elapsed between the time the client sent an IGMP LEAVE (broadcast channel) or RTSP PAUSE (VoD channel) and the time it received the last byte of video data.</p> <p>Leave latency has a maximum timeout of 10 seconds; if the client continues to receive data 10 seconds after it has sent the Leave command, the latency is measured as 10 seconds.</p>
Video Channel Switch Latency	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP LEAVE (broadcast channel) or RTSP PAUSE (VoD channel) to stop receiving one video stream and the time it received the first byte of data of a new video stream.
Video Channel Gap Duration	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP LEAVE (broadcast channel) and received the last byte of the old video stream and the time it received the first byte of data of a new video stream.
Video Channel Overlap Duration	Amount of time, in milliseconds, elapsed after sending an IGMP LEAVE (broadcast channel) that the client was simultaneously receiving both the old and new video streams.



Video Control Sent	Indicates the type of video control command that has most recently been sent: 0 = LEAVE or PAUSE/TEARDOWN sent 1 = JOIN or PLAY sent
Video Data Rcvd	Indicates whether or not video data is being received: 0 = no data received 1 = data received
Video RTP Packets Lost	Number of RTP video packets lost.
Video RTP Packets Out of Order	Number of RTP video packets received out of order.
Video RTP Packets Duplicated	Number of duplicate RTP video packets received.
Video Quality Statistics	
Video JB Packets Accepted	Number of video packets accepted into the jitter buffer.
Video JB Packets Early	Number of video packets that arrived earlier than expected in the jitter buffer.
Video JB Packets Discarded	Total number of video packets that were discarded. This statistic is the total of: JB Packets Discarded (Underrun) and JB Packets Discarded (Overrun).
Video JB Packets Discarded (Underrun)	Number of video packets discarded because they arrived after their expected time slot.
Video JB Packets Discarded (Overrun)	Number of video packets discarded because the jitter buffer was full.
Video MOS_V	Mean Opinion Score for Video. This score is computed from the Video Service Quality statistic to create a zero-to-five (0-5) assessment of the quality of the video stream.
Video Service Quality	A factor in the range from 0 to 120, which provides an assessment of the capability of the RTP channel to support video transmission.
Video Gap Video Service Quality	Video Service Quality during the Gap state. This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).

Video Burst Video Service Quality	<p>Video Service Quality during the Burst state.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Burst Count	<p>Number of times the stream entered the Burst state.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Avg Gap Len (Pkts)	<p>The average gap length, in packets.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Avg Burst Len (Pkts)	<p>The average burst length, in packets.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Degradation (Loss)	<p>The amount of the overall video quality degradation that can be attributed to packet loss.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Degradation (Discard)	<p>The amount of the overall video quality degradation that can be attributed to packets being discarded from the jitter buffer.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>
Video Degradation (Video Codec)	<p>The amount of the overall quality degradation that can be attributed to video codec selection.</p> <p>This statistic is cumulative, and is reset to zero (0) whenever you join or pause the channel/stream (Multicast/VOD mode) or change to the channel/stream (Broadcast mode).</p>



Video Current JB Packets Accepted	Number of video packets accepted into the jitter butter during the current statistics Update Interval.
Video Current JB Packets Discarded	Number of video packets discarded from the jitter butter during the current statistics Update Interval.
Video Current JB Packets Lost	Number of video packets lost during the current statistics Update Interval.
Video Current Video Service Quality	Video Service Quality during the current statistics update interval.

11

DDoS

This section describes the DDoS Tcl API objects.

In this chapter:

API Overview	978
DDoS Client Agent	979
DDoS Command	981
DDoS Statistics	987

API Overview

DDoS protocol commands are organized as a simple structure.

- DDoS Client Agent
 - DDoS Command

Objectives

The objectives (userObjective) you can set for DDoS are listed below. Test objectives are set in the ixTimeline object.

- simulatedUsers

DDoS Client Agent

The DDoS Client Agent defines a simulated user performing DDoS attacks. Refer to `DDoS Client Agent` for a full description of this command. The impropoptions of this command are:

Option	Usage
enable	Enables the use of the DDoS client agent.
name	The name associated with the client agent.

DDoS Command

Each client command is a single attack. Refer to `DDoS Command` for a full description of this command. The important options of this command are:

Option	Usage
id	The type of DDoS attack to be mounted.
<options>	Options specific to the type of DDoS attack.

DDoS Client Agent

DDoS Client Agent - configure a DDoS client

SYNOPSIS

```
set clientTraffic [::IxLoad new ixClientTraffic options]
$clientTraffic agentList.appendItem options...
```

DESCRIPTION

An DDoS client agent is added to the `agentList` option of the `ixCustomPortMap` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. It is customary to set all the options of the client agent durthe `appendItem` call (see the example below).

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands. For example, the first agent uses an index of 0 and its name may be modified as follows:

```
$clientTraffic agentList(0).config -name "different"
```

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set clnt_traffic \
    [::IxLoad new ixClientTraffic -name "clnt_traffic"]
#-----
# Create an DDoS client agent
#-----
```



```
$clnt_traffic agentList.appendItem \  
    -name      "my_DDoS_client" \  
    -protocol  "DDoS" \  
    -type      "Client"
```

SEE ALSO

DDoS Command, ixCustomPortMap

DDoS Command

DDoS Command - configure the command that a DDoS client will execute

SYNOPSIS

```
set clientTraffic [::IxLoad new ixClientTraffic options]
$clientTraffic agentList.appendItem options...
$clientTraffic agentList(0).pm.commands.appendItem options...
```

DESCRIPTION

An DDoS client command is added to the `pm.commands` option of the DDoS Client Agent object using its `appendItem`. Note the use of the `'pm.'` component in the name. See the following example:

```
set clientTraffic [::IxLoad new ixClientTraffic options...]
$clientTraffic agentList.appendItem ...
$clientTraffic agentList(0).pm.commands.appendItem \
  -id "SynFloodAttack" ...
```

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands. For example, the first action of the first agent's name may be modified by:

```
$clientTraffic agentList(0).pm.commands(0).config \
  -id "RstFloodAttack"
```

SUBCOMMANDS

None.

OPTIONS

`id`

The name of the type of attack. The options available for this command vary by type of attack and are documented following this option. One of:

option	Usage
SynFloodAttack	SYN Flood Attack
RstFloodAttack	RST Flood Attack
SmurfAttack	Smurf Attack
LandAttack	Land Attack
PingSweepAttack	Ping Sweep Attack
UDPScanAttack	UDP Scan Attack
TCPScanAttack	TCP Scan Attack
TearDropAttack	Tear Drop Attack
PingOfDeathAttack	Ping of Death Attack
ArpFloodAttack	ARP Flood Attack



option	Usage
EvasiveUDPAttack	Evasive UDP Attack
UDPFloodAttack	UDP Flood Attack
UnreachableHostAttack	Unreachable Host Attack
XMasTreeAttack	XMAS Tree Attack

options for: SynFloodAttack

 RstFloodAttack

 TCPScanAttack

 TearDropAttack

 UDPFloodAttack

destinationHostsFrom

The first IPv4 destination address used in the attack. (Default = 198.18.0.101).

destinationHostsTo

The last IPv4 destination address used in the attack. (Default = 198.18.0.108).

destinationPortsFrom

The first destination port number used in the attack. (Default = 1,024).

destinationPortsTo

The last destination port number used in the attack. (Default = 2,048).

packetRate

The number of transmitted packets per second; a value of 0 means the maximum rate possible. Note: this appears under the label `burstRate` in the IxLoad GUI. (Default = 0).

sourceHostsFrom

If `sourceNetworkConfig` is `useCustomNetwork`, then this is the first IPv4 source address used in the attack. (Default = 198.18.0.1).

sourceHostsTo

If `sourceNetworkConfig` is `useCustomNetwork`, then this is the last IPv4 source address used in the attack. (Default = 198.18.0.8).

`sourceNetworkConfig`

The basic means by which source addresses are chosen. One of:

option	Usage
<code>useCustomNetwork</code>	Use the values in <code>sourceHostsFrom</code> and <code>sourceHostsT</code>
<code>useClientNetwork</code>	Use the range of values for the client network that this activity is associated with.

`sourcePortsFrom`

The first source port number used in the attack. (Default = 1,024).

`sourcePortsTo`

The last source port number used in the attack. (Default = 2,048).

options for SmurfAttack

`destinationHostsFrom`
`destinationHostsTo`
`numberOfPackets`
`packetRate`

See the definitions under “SynFloodAttack” above.

`sourceHosts`

If `sourceNetworkConfig` is `useCustomNetwork`, then this is the IPv4 source address used in the attack. (Default = 198.18.255.255).

`sourceNetworkConfig`

The basic means by which the source address is chosen. One of:

option	Usage
<code>useCustomNetwork</code>	Use the value in <code>sourceHosts</code> .
<code>useClientNetwork</code>	Use the first of the range of values for the client network that this activity is associated with.

options for LandAttack

destinationHostsFrom
destinationHostsTo
destinationPortsFrom
destinationPortsTo
numberOfPackets
packetRate

See the definitions under “SynFloodAttack” above.

options for: PingSweepAttack

PingOfDeathAttack

UnreachableHostAttack

destinationHostsFrom
destinationHostsTo
sourceHostsFrom
sourceHostsTo
numberOfPackets
packetRate
sourceNetworkConfig

See the definitions under “SynFloodAttack” above.

packetSize

The packet size used in the attack. (Default =64) .

options for: UDPScanAttack

XMasTreeAttack

destinationHostsFrom
destinationHostsTo
destinationPortsFrom
destinationPortsTo
sourceHostsFrom
sourceHostsTo
sourcePortsFrom
sourcePortsTo
numberOfPackets
packetRate
sourceNetworkConfig

See the definitions under “SynFloodAttack” above.

packetSize

The packet size used in the attack. (Default =64)

options for ArpFloodAttack

destinationHostsFrom
destinationHostsTo
sourceHostsFrom
sourceHostsTo
numberOfPackets
packetRate
sourceNetworkConfig

See the definitions under “SynFloodAttack” above.

arpType

The type of ARP message to be used in the attack. One of:

option	Usage
ARP_REQUEST	(default) An ARP request.
ARP_RESPONSE	An ARP response.

destinationMACFrom

The first destination MAC address to be used in the attack. (Default = “00:00:00:00:00:00”).

destinationMACTo

The last destination MAC address to be used in the attack. (Default = “FF:FF:FF:FF:FF:FF”).

sourceMACFrom

The first source MAC address to be used in the attack. (Default = “00:00:00:00:00:00”).

sourceMACTo

The last source MAC address to be used in the attack. (Default = “FF:FF:FF:FF:FF:FF”).

options for EvasiveUDPAttack

destinationHostsFrom
destinationHostsTo
sourceHostsFrom
sourceHostsTo
numberOfPackets
packetRate
sourceNetworkConfig



See the definitions under "SynFloodAttack" above.

```
destinationMACFrom  
destinationMACTo  
sourceMACFrom  
sourceMACTo
```

See the definitions under "ArpFloodAttack" above.

EXAMPLE

```
#-----  
# Add actions to the client agent  
#-----  
$clnt_traffic agentList(0).pm.commands.appendItem \  
    -id                "SynFloodAttack" \  
    -destinationHostsFrom "1.1.1.1" \  
    -destinationHostsTo  "1.1.1.2" \  
    -destinationPortsFrom 1024 \  
    -destinationPortsTo  2048 \  
    -sourceHostsFrom      "20.0.1.1" \  
    -sourceHostsTo        "20.0.1.8"\  
    -sourcePortsFrom      1024 \  
    -sourcePortsTo        2048\  
    -numberOfPackets      100 \  
    -packetRate            10 \  
    -sourceNetworkConfig  "useCustomNetwork"
```

SEE ALSO

DDoS Client Agent

DDoS Statistics

The table below lists the statistics that IxLoad provides for each attack. The default statistics are preconfigured to display in the StatViewer graphs for a DDoS activity. The custom statistics are statistics you can add to the StatViewer graphs.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test configuration>_<yyyy><mm><dd>_<hh><mm><ss>

Statistic	Description
Default Statistics	
Successful Packets	Number of attack packets sent.
Bytes Sent	Number of bytes contained in attack packets sent.
Failed Packets	<p>The reporting of Failed Packets depends on the attack type:</p> <p>ARP Flood attack:</p> <p>In an ARP Flood attack, packets are always reported as successful; no packets are ever reported as failed.</p> <p>All other attacks:</p> <p>If ARP resolution is successful (the target resides on the same network as the IxLoad DDoS client and is up), all the packets sent are reported as successful even if the target does not listen or respond to the attack.</p> <p>Packets are reported as failed if either:</p> <ul style="list-style-type: none"> - ARP resolution failed, or - The target resides on the different network and is unreachable (there is no route to it configured).
Custom Statistics	
Round Time	Amount of time required to run the attacks in the attack list. One round is one pass through the attack list, from the first attack to the last.
Load Size	Total number of users simulated by the attack.
Attempted Rounds	Number of attempted passes through the attacks in the attack list. One round is one pass through the attack list, from the first attack to the last.
Successful Rounds	Number of successful passes through the attacks in the attack list. One round is one pass through the attack list, from the first attack to the last.

12

DHCP

This section describes the DHCP Tcl API objects.

In this chapter:

Overview	990
DHCP Client Agent	994
DHCP Command List	996
Advanced Options	1004
Relay Agent	1007
Option	1010
IP Address	1023
Using Variables in DHCP Fields	1025
DHCP Statistics	1026

Overview

The IxLoad DHCP API consists of a client agent and its commands.

Objectives

The objectives (userObjective) you can set for DHCP are listed below. Test objectives are set in the ixTimeline object.

- transactionRate
- simulatedUsers

DHCP Client Agent

The DHCP Client Agent command defines a simulated user using DHCP to establish and terminate sessions DHCP. Refer to `DHCP Client Agent` on page 19-6 for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time.
protocol	Protocol used by the client agent.
type	Defines the agent as either a client or server.

DHCP Command List

The DHCP Command List creates the list of DHCP commands that the client will send to a DHCP server. Refer to `DHCP Command List` on page 19-17 for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	Command that client will send.

Advanced Options

The Advanced Options sets the DHCP client agent's global configuration options. Refer to *Advanced Options* on page 19-23 for a full description of this command. The most significant options of this command are listed below.

Option	Description
clientPort	UDP port that the client listens on for DHCP and BOOTP responses.
serverPort	UDP port that the server listens on.
numRetransmit	Number of times that the client will retransmit a request for which it has not received a response.
initialTimeout	Length of time that the client waits for a response to a request.
timeoutIncrFactor	<p>If the client has retransmitted a timed-out request, then this parameter increments the Initial Timeout value. IxLoad uses the following formula for calculating the timeout interval:</p> <pre>maxTimeOut = 1000 * initialTimeOut * (1 - pow(incrementnumRetransmissions+1)) / (1 - increment - else - maxTimeOut = 1000 * initialTimeOut * (numRetransmis+ 1); (pow(a,b) means a raised to the power of b.)</pre> <p>The maximum timeout interval is 2147483647.</p>
maximumDHCPMsgSize	Maximum size of a DHCP packet that the client will accept, including IP and UDP headers.
vendorClass	Text string identifying the vendor type and configuration of the DHCP client.
optionsOverload	If <code>true</code> , then this indicates to the server that the client will allow option overloading.
broadcastBit	If <code>true</code> , then this sets the client's Broadcast bit to 1 in the 'flags' field in any DHCPDISCOVER or DHCPREQUEST messages that client sends.

Relay Agent

The Relay Agent command enables the DHCP client agent to function as a DHCP relay agent. Refer to *Relay Agent* on page 19-25 for a full description of this command. The most significant options of this command are listed below.

Option	Description
enableRelay	If <code>true</code> , then the DHCP client will emulate DHCP relay agents.

Option	Description
enableCircuitId	If <code>true</code> , then the DHCP agent includes the Circuit ID sub-option in DHCP messages.
circuitId	If <code>circuitId</code> is <code>true</code> , then this option sets the circuit ID.
circuitIdGroupSize	Number of DHCP clients sharing the same Circuit ID.
enableRemoteId	If <code>true</code> , then the DHCP agent includes the Remote ID sub-option in DHCP messages.
remoteId	If <code>remoteId</code> is <code>true</code> , then this option sets the remote ID.
remoteIdGroupSize	Number of DHCP clients sharing the same Remote ID.
relayAgentIPAddr	IP address of the first emulated DHCP Relay Agent.
numRelayAgent	Number of DHCP Relay Agents to emulate.

Option

The Option object configures a DHCP option. Refer to `Option` on page 19-27 for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	DHCP option

Option Set

The Option Set object configures the list of DHCP options that the DHCP com will use. Refer to `Option Set` on page 19-32 for a full description of this command. The options supported are listed below.

Option (Option code)
Protocol-Mandated Options
Pad (0) (not visible to the end user)
Requested IP Address (50)
IP Address Lease Time (51)
Overload "sname" or "file" (52)
DHCP Message Type (53) (not visible to the end user)
DHCP Server Identification (54)
Parameter Request List (55)
DHCP Error Message (56)
DHCP Maximum Message Size (57)

Option (Option code)
DHCP Renewal (T1) Time (58)
DHCP Rebinding (T2) Time (59)
(Vendor) Class Identifier (60)
Client Identifier (61)
End (255) (not visible to the end user)
Informational Options
Subnet Mask Value (1)
Time Offset in Seconds from UTC (2)
Router addresses (3)
DNS Server addresses (6)
Hostname string (12)
DNS domain name of the Client (15)
Interface MTU Size (26)
All Subnets are Local (27)
Broadcast Address (28)
Perform Mask Discovery (29)
Perform Router Discovery (31)
ARP Cache Timeout (35)
Vendor Specific Information (43)
User Class Information (77)

Option Set Manager

The Option Set Manager object configures the list of Option Sets. Refer to `Option Set Manager` on page 19-34 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>optionSetList</code>	List of <code>Option Sets</code> .



DHCP Client Agent

DHCP Client Agent

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPClient1 agent.config
```

DESCRIPTION

A DHCP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

`loopValue`

If this option is enabled (1), then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (0), then the client will progress through the command list only once, and then go idle. (Default = 0).

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity DHCPClient1 of NetTraffic Traffic1@Network1
```

```
#####  
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "dhcp Client" ]  
$Activity_DHCPClient1 agent.config \  
    -enable                          true \  
    -name                             "DHCPClient1"
```

SEE ALSO

`ixNetTraffic`

DHCP Command List

DHCP Command List—Creates the list of DHCP commands that the client will send to a DHCP server.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPCClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPCClient1 agent.pm.DHCPCommandList.appendItem
```

DESCRIPTION

A command is added to the DHCP Command List object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

DHCP command to be executed. One of the following:

Command	Description
DHCPDiscover	Broadcasts a DHCPDISCOVER message—a broadcast to locate available servers. The client will then wait to receive one or more DHCPOFFER messages and select one of these offers.
DHCPRequest	<ul style="list-style-type: none">▪ Sends a DHCPREQUEST message. The DHCPREQUEST message can be used to perform several tasks:▪ It can request the offered parameters from one server and implicitly decline offers from all others.▪ It can confirm the correctness of a previously allocated address (for example, after a reboot).▪ It can extend the lease on a particular network address.
DHCPDecline	Sends a DHCPDECLINE message. A DHCP client sends a DHCPDECLINE when it knows that an offered IP address is already in use.
DHCPRelease	Sends a DHCPRELEASE message. If a client no longer requires use of its assigned network address (for example, because it is shutting down), it sends a DHCPRELEASE message to the server.

Command	Description
DHCPInform	Sends a DHCPINFORM message. After sending the comthe client waits for a DHCPACK message from the server. The DHCPINFORM message allows hosts that are not using DHCP to acquire IP addresses to still utilize its other configuration capabilities to request that a server send it parameters for how the network is to be used.
Bind	<p>An IxLoad command that is a composite of several DHCP commands that simulates a binding.</p> <p>The {Bind} command functions as follows:</p> <ol style="list-style-type: none"> 1. Send a DHCPDISCOVER message. 2. Wait to receive one or more DHCPOFFER messages from the servers. 3. Send a DHCPREQUEST message to one of the servers. 4. Wait to receive a DHCPACK message. 5. Validate the IP address received. If it is invalid, then send a DHCPDECLINE message to the server. <p>Using the {Bind} command is equivalent to issuing a DHCPcommand followed by a DHCPREQUEST (SELECTING) command and an optional DHCPDECLINE command.</p>
BindRelease	<p>An IxLoad command that is a composite of several DHCP commands that simulates a client binding, pausing execution, and then releasing its IP address. It is included to help speed your testing.</p> <p>Using the {BindRelease} command is equivalent to using {Bind}, {Think}, and DHCPRELEASE.</p>
BindRenew	<p>An IxLoad command that is a composite of several DHCP commands that simulates a client binding, pausing execution, and then renewing its IP address. It is included to help speed your testing.</p> <p>Using the {BindRenew} command is equivalent to using {Bind}, {Think}, and DHCPREQUEST (RENEWING).</p>
BOOTRequest	A BOOTP (Bootstrap Protocol) command that sends a BOOTREQUEST message and waits to receive a BOOTREmessage from a server.
Think	<p>Causes the client to become idle. {Think} is an internal IxLoad command intended to assist your testing; it is not a command defined in the DHCP protocol.</p> <p>If you specify identical values for the minimum and maximum times, the client will be idle for a fixed length of time. If you specify different values for the minimum and maximum times, IxLoad will select a value within the range and cause the client to be idle for that length of time.</p>
LoopBeginCommand	An IxLoad command that you can add to the Command List to cause the commands between it and the {LoopEndCommand} to be executed a specified number of times.
LoopEndCommand	Ends the list of commands that will be executed by the preced{Loop Begin} command.

Arguments for id = DHCPDiscover

optionSet

Name of option set. A value for this argument must one of the `name` objects from the `optionSet` object. Minimum length = 1. (Default = "Default Option Set for DHCPDISCOVER").

serverAlgo

Determines how the client selects the DHCP server from among those offered. Minimum = 1, maximum = 3. The choices are:

Value	Description
1	(default) The client selects the server that replies first.
2	The client selects the server whose IP address is specified for the <code>serverIPAddr</code> argument.
3	The client selects a server at random from a pool of those that reply. Specify the number of servers in the <code>upperLimit</code> field.

serverIPAddr

If `serverAlgo` is set to 2, this is the IP address of the DHCP server. Minimum length = 7, maximum length = 19. (Default = "10.0.1.1").

upperLimit

If `serverAlgo` is set to 3, this is the number of servers in the pool. Minimum = 1, maximum length = 2147483647. (Default = "5").

Arguments for id = DHCPRequest

sendState

State in which the client is to send the DHCPREQUEST message. See the state transition diagram in RFC 2131. Minimum = "1," maximum = "4." The choices are:

Value	Description
"1"	(default) Selecting
"2"	InitReboot
"3"	Renewing

"4"	Rebinding
-----	-----------

Arguments for id = DHCPDecline

optionSet

Name of option set. A value for this argument must one of the name objects from the optionSet object. Minimum length = 1. (Default = "Default Option Set for DHCPDECLINE").

Arguments for id = DHCPRelease

optionSet

Name of option set. A value for this argument must one of the name objects from the optionSet object. Minimum length = 1. (Default = "Default Option Set for DHCPRELEASE").

Arguments for id = DHCPInform

optionSet

Name of option set. A value for this argument must one of the name objects from the optionSet object. Minimum length = 1. (Default = "Default Option Set for DHCPINFORM").

clientIPAddr

IP address and subnet of client. The client will insert this address and subnet into the CIAddr field of the DHCPINFORM message. If the IP address has already been assigned, then this address will be ignored. Minimum length = "7," maxlength = "24." (Default = "10.0.0.1/8").

Arguments for id = Bind

optionSet

Name of option set. A value for this argument must be one of the name objects from the optionSet object. Minimum length = 1. (Default = "Default Option Set for {Bind}").

serverAlgo

Determines how the client selects the DHCP server from among those offered. Minimum = 1, maximum = 3. The choices are:

Value	Description
1	(default) The client selects the server that replies first.



2	The client selects the server whose IP address is specified for the <code>serverIPAddr</code> argument.
3	The client selects a server at random from a pool of those that reply. Specify the number of servers in the <code>upperLimit</code> field.

`serverIPAddr`

If `serverAlgo` is set to 2, this is the IP address of the DHCP server. Minimum length = 7, maximum length = 19. (Default = "10.0.1.1/8").

`upperLimit`

If `serverAlgo` is set to 3, this is the number of servers in the pool. Minimum = 1, maximum length = 2,147,483,647. (Default = "5").

Arguments for `id = BindRelease`

`timeToThinkMin`

Minimum length of time before the client releases the IP address. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

`timeToThinkMax`

Maximum length of time before the client releases the IP address. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

`optionSet`

Name of option set. A value for this argument must one of the `name` objects from the `optionSet` object. Minimum length = 1. (Default = "Default Option Set for {BindRelease}").

`serverAlgo`

Determines how the client selects the DHCP server from among those offered. Minimum = 1, maximum = 3. The choices are:

Value	Description
1	(default) The client selects the server that replies first.
2	The client selects the server whose IP address is specified for the <code>serverIPAddr</code> argument.
3	The client selects a server at random from a pool of those that reply. Specify the number of servers in the <code>upperLimit</code> field.

serverIPAddr

If `serverAlgo` is set to 2, this is the IP address of the DHCP server. Minimum length = 7, maximum length = 19. (Default = "10.0.1.1/8").

upperLimit

If `serverAlgo` is set to 3, this is the number of servers in the pool. Minimum = 1, maximum length = 2,147,483,647. (Default = "5").

Arguments for `id = BindRenew`

timeToThinkMin

Minimum length of time before the client releases the IP address. If you set a value for `timeToThinkMin`, you must also set `timeToThinkMax` to the same value. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

timeToThinkMax

Maximum length of time before the client releases the IP address. If you set a value for `timeToThinkMax`, you must also set `timeToThinkMin` to the same value. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

optionSet

Name of option set. A value for this argument must one of the `name` objects from the `optionSet` object. Minimum length = 1. (Default = "Default Option Set for {BindRenew}").

serverAlgo

Determines how the client selects the DHCP server from among those offered. Minimum = 1, maximum = 3. The choices are:

Value	Description
1	(default) The client selects the server that replies first.
2	The client selects the server whose IP address is specified for the <code>serverIPAddr</code> argument.
3	The client selects a server at random from a pool of those that reply. Specify the number of servers in the <code>upperLimit</code> field.

serverIPAddr



If `serverAlgo` is set to 2, this is the IP address of the DHCP server. Minimum length = 7, maximum length = 19. (Default = "10.0.1.1/8").

`upperLimit`

If `serverAlgo` is set to 3, this is the number of servers in the pool. Minimum = 1, maximum length = 2,147,483,647. (Default = "5").

Arguments for id = BOOTRequest

`clientIPAddr`

IP address and subnet of client. Minimum length = "7," maximum length = "24." (Default = "10.0.0.1/8").

`serverName`

Host name or IP address of the BOOTP server. In actual BOOTP implementation this field (SName) is normally used by a client to specify a particular server that it wants to receive a reply from.

If you enter a host name or IP address in this field, the client sends the BOOTREas a unicast message to the BOOTP server.

If you leave this field blank, the client sends the BOOTREQUEST as a broadcast message to the port number used by BOOTP to the broadcast address of the local network. Minimum length = "7," maximum length = "24." (Default = "10.0.0.1/8").

`bootFileType`

Indication to the BOOTP server as to the boot file that the client wants to receive. When the server receives the BOOTREQUEST, it determines which file contains the requested image, and uses Boot File Name field to send the name of the file to the client. Maximum length = 127. (Default="{ }").

`optionSet`

Name of option set. A value for this argument must be one of the name objects from the `optionSet` object. Minimum length = 1. (Default = "Default Option Set for {BOOTRequest}").

Arguments for id = Think

`timeToThinkMin`

Minimum length of time that the client is idle. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

`timeToThinkMax`

Maximum length of time that the client is idle. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

Arguments for id = LoopBeginCommand

loopCount

Number of times to repeat the enclosed commands. '0' treated as infinity. Mini= "0," maximum = "2,147,483,647." (Default = "5").

Arguments for id = LoopEndCommand

None.

EXAMPLE

```
$Activity_DHCPClient1 agent.pm.DHCPCommandList.appendItem \  
-id "DHCPDiscover" \  
-upperLimit 5 \  
-optionSet "Default Option Set for DHCPDISCOVER" \  
-serverAlgo 1 \  
-serverIPAddr "10.0.1.1"
```

SEE ALSO

DHCP Client Agent

Advanced Options

Advanced Options—Sets the DHCP client agent's global configuration options.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPClient1 agent.pm.advancedOptions.config
```

DESCRIPTION

A DHCP client's advanced configuration options are set by modifying the options of the `pm.advancedOptions` option of the `DHCP Client Agent`.

SUBCOMMANDS

None.

OPTIONS

`clientPort`

UDP port that the client listens on for DHCP and BOOTP responses. Minimum = 1, maximum = 65,535. (Default = 68).

`serverPort`

UDP port that the client addresses server requests to. Minimum = 1, maximum = 65535. (Default = 67).

`numRetransmit`

Number of times that the client will retransmit a request for which it has not received a response. Minimum = 0, maximum = 2,147,483. (Default = 3).

`initialTimeout`

Length of time that the client waits for a response to a request. If the Initial Timeout expires, the client retransmits the request (unless `numRetransmit` is 0). Minimum = 1, maximum = 2,147,483. (Default = 4).

`timeoutIncrFactor`

If the client has retransmitted a timed-out request, this parameter increments the Initial Timeout value. Minimum = 1, maximum = 2,147,483. (Default = 2).

maximumDHCPMsgSize

Maximum size of a DHCP packet that the client will accept, including IP and UDP headers. According to RFC 2131, the minimum message size that a client should accept is 576 octets. Minimum = 576, maximum = 65,536. (Default = 576).

vendorClass

Text string identifying the vendor type and configuration of the DHCP client. Minimum length = 0, maximum length = 255. (Default = "IXIA IxLoad DHCP Client")

optionsOverload

If true, indicates to the server that the client will allow option overloading. (Default = 0).

broadcastBit

If true, sets the client's Broadcast bit to 1 in the 'flags' field in any DHCPDIS or DHCPREQUEST messages that client sends. (Default = 0).

EXAMPLE

```
$Activity_DHCPClient1 agent.pm.advancedOptions.config \  
-clientPort                68 \  
-firstLoad                  false \  
-maxDHCPMsgSize            576 \  
-broadcastBit               false \  
-timeoutIncrFactor          2 \  
-numRetransmit              3 \  
-needValidation             false \  
-writeLeasesToFile          false \  
-serverPort                 67 \  
-optionsOverload            false \  
-memRequiredForOptions      52 \  
-vendorClass                 "IXIA IxLoad DHCP Client" \  
-initialTimeout             4 \  
-implicitLoopCheck          true
```

SEE ALSO

DHCP Client Agent



Relay Agent

Relay Agent—Enables the DHCP client agent to function as a DHCP relay agent and configuration the relay agent options.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPClient1 agent.pm.relayAgent.config
```

DESCRIPTION

A DHCP client's relay agent is configured by modifying the options of the `pm.relayAgent` option of the DHCP Client Agent object.

SUBCOMMANDS

None.

OPTIONS

`enableRelay`

If `true`, the DHCP client will emulate DHCP relay agents. (Default = 0).

`enableCircuitId`

If `true`, the DHCP agent includes the Circuit ID sub-option in DHCP messages. Use the `circuitId` option to configure the ID value. (Default = 0).

`circuitId`

If `circuitId` is `true`, this option sets the circuit ID. You can include variables to cause the client to generate large numbers of unique values. See *Using Variables in DHCP Fields* on page 19-38. Maximum length = 243. (Default = "123[000-999]").

`circuitIdGroupSize`

Number of DHCP clients sharing the same Circuit ID.

`enableRemoteId`

If `true`, the DHCP agent includes the Remote ID suboption in DHCP messages. Use the `remoteId` option to configure the ID value. (Default = 0).



remoteId

If `remoteId` is true, this option sets the remote ID. You can include variables to cause the client to generate large numbers of unique values. See *Using Variables in DHCP Fields* on page 19-38. Maximum length = 243. (Default = "Ixia-host-[0000-]").

remoteIdGroupSize

Number of DHCP clients sharing the same Remote ID.

relayAgentIPAddr

IP address of the first emulated DHCP Relay Agent. If you specify more than Relay Agent (the `numRelayAgent` option), IxLoad increments this address to create additional addresses for the agents. Minimum length = 7, maximum length = 24, (Default = "11.0.0.1/8").

numRelayAgent

Number of DHCP Relay Agents to emulate. Minimum = 1, maximum = 1,000,000. (Default = 1).

EXAMPLE

```
$Activity_DHCPClient1 agent.pm.relayAgent.config \  
-remoteId "Ixia-host-\[0000-\]" \  
-memRequired 0 \  
-circuitId "123\[000-999\]" \  
-relayAgentIPAddr "11.0.0.1/8" \  
-numVlans 1 \  
-enableRemoteId false \  
-remoteIdGroupSize 1 \  
-enableCircuitId false \  
-circuitIdGroupSize 1 \  
-enableRIdByteStream false \  
-vlanId 1 \  
-enableVlan false \  
-incrVlanId 1 \  
-enableCIdByteStream false \  
-numRelayAgent 1 \  

```

-enableRelay

0

SEE ALSO

DHCP Client Agent

Option

Option—Configures a DHCP option.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPClient1 agent.pm.optionSetMgr.optionSetList.appendItem
$Activity_DHCPClient1 agent.pm.optionSetMgr.optionSetList(0).option
```

DESCRIPTION

An `Option` object is an item in an `OptionsList`. An `Option` is added to an `Options List` using `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`id`

DHCP option. One of the following:

Value	Description
RequestedIPAddress	This option is used in a client request (DHCPDISCOVER) to allow the client to request that a particular IP address be assigned.
IPAddressLeaseTime	This option is used in a client request (DHCPDISCOVER or DHCPREQUEST) to allow the client to request a lease time for the IP address. In a server reply (DHCPOFFER), a DHCP server uses this option to specify the lease time it is willing to offer.
ParameterRequestList	This option is used by a DHCP client to request values for specified configuration parameters.
DHCPErrorMessage	This option is used by a DHCP server to provide an error message to a DHCP client in a DHCPNAK message, in the event of a failure. A client may use this option in a DHCPDmessage to indicate why the client declined the offered parameters.
DHCPRenewalTime	This option specifies the time interval from address assignuntil the client transitions to the RENEWING state.
DHCPRebindingTime	This option specifies the time interval from address assignuntil the client transitions to the REBINDING state.

VendorClassIdentifier	This option is used by DHCP clients to optionally identify the vendor type and configuration of a DHCP client.
ClientIdentifier	This option is used by DHCP clients to specify their unique identifier. DHCP servers use this value to index their data of address bindings. This value is expected to be unique for all clients in an administrative domain.
SubnetMaskValue	The subnet mask option specifies the client's subnet mask as per RFC 950.
TimeOffsetUTC	The time offset field specifies the offset of the client's subnet in seconds from Coordinated Universal Time (UTC). A positive offset indicates a location east of the zero meridian and a negative offset indicates a location west of the zero meridian.
RouterAddresses	The router option specifies a list of IP addresses for routers on the client's subnet. Enter the router addresses in order of preference.
DNSServerAddresses	The domain name server option specifies a list of Domain Name System name servers available to the client. List the servers in order of preference.
HostnameString	This option specifies the name of the client. The name may or may not be qualified with the local domain name.
DNSDomainNameClient	This option specifies the domain name that client should use when resolving hostnames via the Domain Name System.
InterfaceMTUSize	This option specifies the MTU to use on this interface. The MTU is specified as a 16-bit unsigned integer. The minimum value for MTU is 68.
SubnetsLocal	<p>This option specifies whether or not the client may assume that all subnets of the IP network to which the client is connected use the same MTU as the subnet of that network to which the client is directly connected.</p> <p>If this option is enabled, that indicates that all subnets share the same MTU.</p> <p>If this option is disabled, that indicates that the client should assume that some subnets of the directly connected network may have smaller MTUs.</p>
BroadcastAddress	This option specifies the broadcast address in use on the client's subnet.
PerformMaskDiscovery	<p>This option specifies whether or not the client should perform subnet mask discovery using ICMP.</p> <p>If this option is disabled, then it indicates that the client should not perform mask discovery.</p> <p>If this option is enabled, then it indicates that the client should perform mask discovery.</p>
PerformRouterDiscovery	<p>This option specifies whether or not the client should solicit routers using the Router Discovery mechanism defined in RFC 1256.</p> <p>If this option is disabled, then it indicates that the client should not perform router discovery.</p> <p>If this option is enabled, then it indicates that the client should perform router discovery.</p>



ARPCacheTimeOut	This option specifies the timeout in seconds for ARP cache entries.
VendorSpecificInfo	This option is used by clients and servers to exchange veninformation. The vendor is indicated in the venclass identifier option.
UserClassInfo	This option is used by a Dynamic Host Configuration Proto(DHCP) client to optionally identify the type or category of user or applications it represents.

Arguments for option = RequestedIPAddress

clientIPAddr

IP address requested by the client. Minimum length = 7, maximum length = 24. (Default = "10.0.0.1/8").

Arguments for option = IPAddressLeaseTime

interval

Duration of lease, in seconds. Minimum = 0, maximum = 4,294,967,295. (Default = 3,600).

Arguments for option = ParameterRequestList

options

List of options for requested parameters. This argument is a list of Option Choices objects. See [Option Choices](#) on page 19-35.

Arguments for option = DHCPErrorMessage

message

Text of error message. Minimum length = 1, maximum length = 255. (Default = "IP Address Rejected by IxLoad").

Arguments for option = DHCPRenewalTime

interval

Time, in seconds, from address assignment to transition to the RENEWING state. Minimum = 0, maximum = 4,294,967,295. (Default = 0).

Arguments for option = DHCPRebindingTime

interval

Time, in seconds, from address assignment to transition to the REBINDING state. Minimum = 0, maximum = 4,294,967,295. (Default = 0).

Arguments for option = VendorClassIdentifier

data

Text identifying vendor class. Minimum length = 1, maximum length = 255. (Default = "IXIA IxLoad DHCP Client").

Arguments for option = ClientIdentifier

identifier

Value for client identifier. Minimum = 1, maximum = 2,147,467,647. (Default = 1).

Arguments for option = SubnetMaskValue

mask

Subnet mask. Minimum length = 7, maximum length = 19. (Default = "255.0.0.0").

Arguments for option = TimeOffsetUTC

offset

Offset value. Minimum = -2,147,483,648, maximum = 2,147,483,647. (Default = 0).

Arguments for option = RouterAddresses

address

List of router IP addresses. This is a list of type IPAddress. IP Address on page 19-37. (Default={}).

Arguments for option = DNSServerAddresses

address

List of router IP addresses. This is a list of type IPAddress. IP Address on page 19-37. (Default = {}).

Arguments for option = HostnameString

hostName



Name of the client. You can use the following characters a-z, A-Z, 0-9, dash (-). Minimum length = 1, maximum length = 53. (Default = "IxLoad-DHCP-Cli").

Arguments for option = DNSDomainNameClient

domainName

Domain name. Minimum length = 1, maximum length = 255. (Default = "ixia").

Arguments for option = InterfaceMTUSize

size

MTU value. Minimum = 68, maximum = 65,535. (Default = 68).

Arguments for option = SubnetsLocal

val

Boolean value. 0 = false, 1 = true. (Default = 0).

Arguments for option = BroadcastAddress

address

Broadcast IP address. Minimum length = 7, maximum length = 19. (Default = "10.255.255.255").

Arguments for option = PerformMaskDiscovery

val

Boolean value. 0 = false, 1 = true. (Default = 0).

Arguments for option = PerformRouterDiscovery

val

Boolean value. 0 = false, 1 = true. (Default = 0).

Arguments for option = ARPCacheTimeOut

timeout

Timeout value. Minimum = 0, maximum = 4,294,967,295. (Default = 0).

Arguments for option = VendorSpecificInfo

info

Text string describing vendor information. Minimum length = 1. (Default = "None").

Arguments for option = UserClassInfo

info

Text string describing user class information. Minimum length = "1," maximum length = "254." (Default = "IXIA IxLoad DHCP Client").

EXAMPLE

```
$Activity_DHCPClient1 agent.pm.optionSetMgr.optionSetList.appendItem \
```

```
-id                "OptionSet" \  
-predefined        true \  
-decline           0 \  
-name              "Default Option Set for  
DHCPDISCOVER" \  
-inUse             1 \  
-bootRequest       0 \  
-bind              0 \  
-discover          true \  
-bindRelease       0 \  
-inform            0 \  
-bindRenew         0 \  
-release           0
```

```
$Activity_DHCPClient1  
agent.pm.optionSetMgr.optionSetList(0).optionsList.clear
```

```
$Activity_DHCPClient1  
agent.pm.optionSetMgr.optionSetList(0).optionsList.appendItem \  
-
```

```
-id                "RequestedIPAddress" \  
-clientIPAddr      "10.0.0.1/8"
```

```
$Activity_DHCPClient1  
agent.pm.optionSetMgr.optionSetList(0).optionsList.appendItem \  
-
```



```
-id "IPAddressLeaseTime" \  
-interval 3600
```

SEE ALSO

DHCP Client Agent

Option Set

Options Set—Configures the list of commands that an option list applies to.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPClient1 agent.pm.optionSet.config
```

DESCRIPTION

An `Options Set` is a list of `Options`, their arguments, and the commands for which those options are used. Configure the list using the same subcommands as for `ixConfig`.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOption` subcommands defined in the `ixConfig` command.

OPTIONS

`name`

Name of option set list. Minimum length = 1. (Default = "No Name")

`optionsList`

List of options and their arguments. See `Option` on page 19-27. (Default = "{}").

`predefined`

If `true`, then the options in this option set are predefined for the DHCP server to expose as available options. (Default = "0").

`inUse`

Minimum = 0, maximum = 1. (Default = 0).

`discover`

If `true`, then this option set can be used for the `DHCPDISCOVER` command. (Default = 0).

`inform`

If `true`, then this option set can be used for the `DHCPINFORM` command. (Default = 0).



decline

If true, then this option set can be used for the DHCPDECLINE command. (Default = 0).

release

If true, then this option set can be used for the DHCPRELEASE command. (Default = 0).

bind

If true, then this option set can be used for the (Bind} command. (Default = 0).

bindRelease

If true, then this option set can be used for the BindRelease command. (Default = 0).

bindRenew

If true, then this option set can be used for the BindRenew command. (Default = 0).

bootRequest

If true, then this option set can be used for the BOOTREQUEST command. (Default = 0).

EXAMPLE

```
$Activity_DHCPClient1 agent.pm.optionSet.config \  
-predefined                false \  
-decline                   false \  
-name                      "No Name" \  
-inUse                     0 \  
-bootRequest              false \  
-bind                      false \  
-discover                 false \  
-bindRelease              false \  
-inform                   false \  
-bindRenew                false \  
-release                  false
```

SEE ALSO

DHCP Client Agent



Option Set Manager

Options Set Manager—Configures the list of Option Sets.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPClient1 agent.pm.optionSetMgr.optionSetList.appendItem
$Activity_DHCPClient1 agent.pm.optionSetMgr.optionSetList(0).option
```

DESCRIPTION

To configure an Option Set Manager, use the `appendItem` command on the `pm.optionSetManager` component of the DHCP Client Agent.

SUBCOMMANDS

None.

OPTIONS

`optionSetList`

List of Option Sets. See Option Set.

EXAMPLE

See the example for `Option`.

SEE ALSO

DHCP Client Agent

Option Choices

Option Choices—Configures a list of DHCP options that the client agent requests values for.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPClient1 agent.pm.optionSetMgr.optionSetList.appendItem
$Activity_DHCPClient1 agent.pm.optionSetMgr.optionSetList(0).option
```

DESCRIPTION

The `Option` command includes a `Parameter Request List` option. `Parameter Request List` allows the client to send a list of DHCP options to the server and request the server to return the values that it supports for each option.

To specify the list of DHCP options that the client sends, use the `optionCode` parameter of the `Options Choices` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`optionCode`

DHCP option that the client requests values for. Minimum = 1, maximum = 59. The choices are:

Value	Description
51	(default) IP Address Lease Time
58	DHCP Renewal (T1) Time
59	DHCP Rebinding (T2) Time
1	Subnet Mask Value
2	Time Offset in Seconds from UTC
3	Router Addresses
6	DNS Server Addresses
12	Hostname String
15	DNS Domain Name of the Client
26	Interface MTU Size
27	All Subnets are Local



28	Broadcast Address
29	Perform Mask Discovery
31	Perform Router Discovery
35	ARP Cache Timeout
43	Vendor Specific Information

SEE ALSO

Option

IP Address

IP Address—Configures a list of IP addresses.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DHCPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_DHCPClient1 agent.pm.optionSetMgr.optionSetList(0).option
```

DESCRIPTION

Several DHCP Option commands includes options (RouterAddresses, DNSServ) that specify lists of IP addresses for various functions. To create those lists, you use IP Address, which is a list of type ixConfigSequenceContainer. Each element in the list is an IP address.

SUBCOMMANDS

None.

OPTIONS

address

IP address. Minimum length = 7, maximum length = 19. (Default = "0.0.0.0").

EXAMPLE

```
# Create a list of IP addresss

set ipAddrList [list \
                "198.18.0.1" \
                "10.205.17.71" \
                "10.205.17.176" \
                "198.18.0.101" \
                ]

# Go through the following loop adding the addresses one by one.

foreach {option id} [list RouterAddresses addresses DNSServerAddresses
addresses] {

    $clnt_traffic agentList(0).pm.optionSetMgr.optionSetList(0)
    \
    .optionsList.appendItem \
```



```
-id $option

set index [$clnt_traffic agentList(0).pm.optionSetMgr.optionSetList(0) \
.optionsList.indexCount]

incr index -1

foreach ip $ipAddrList {

    $clnt_traffic agentList(0).pm.optionSetMgr.optionSetList(0) \
    .optionsList($index).${id}.appendItem \

    -address $ip

}

}
```

SEE ALSO

Option, ixConfigSequenceContainer

Using Variables in DHCP Fields

You can insert variables into the Circuit ID and Remote ID fields on the DHCP client Relay Agent tab. You can use the variables to generate large numbers of unique circuit IDs and remote IDs.

You can use the following variables:

- Numbers 0-9
- Letters A-Z and a-z

The letter variables are case-sensitive; IxLoad considers the variable strings “AA” and “aa” to be different.

You can combine the variables with fixed text to create the circuit IDs and remote IDs. For example, you can enter `circuitID_[00-]` to create a range of unique IDs that begin with the characters “circuitID_” (circuitID_00, circuitID_01, and so on).

To insert the variables into a field, enclose them in square brackets ([]). To specify a range, separate the minimum and maximum values with a hyphen (-). For example, `[00-10]` specifies a range of 00 through 10.

The number of variables you insert determines the width of the generated strings. For example, the variable “00” can generate the strings 00 - 99. The variable string “000” can generate the strings 000 - 999.

Similarly, “AA” can generate strings that consist of all the two-letter combinations from AA to ZZ. “AAA” can generate strings that consist of all the three-letter combinations from AAA to ZZZ.

You can use a single variable string and allow IxLoad to generate strings up to the maximum value of the string or, you can use two variable strings together to restrict the generated strings to a certain range.

See the following example:

`[0-]` will generate all the values 0 - 9 (0, 1, 2, 3 . . . 9).

`[0-5]` will generate all the values 0 - 5.

`[00-]` will generate all the values 00 - 99 (00, 01, 02, 03. . . 97, 98, 99).

`[00-50]` will generate all the values 00 - 50.

`[A-]` will generate all the values A - Z (A, B, C . . . Z).

`[A-K]` will generate all the values A - K.

`[AA-]` will generate all the values AA - ZZ (AA, AB, AC. . . ZX, ZY, ZZ).

`[AA-KK]` will generate all the values AA - KK.

When IxLoad has generated the final string, if the test configuration requires additional strings, IxLoad returns to the starting value of the variable and continues to generate strings until no more are required. In this case, the generated strings will not be unique.

For example, if a DHCP test requires 256 circuit IDs and the Circuit ID field is configured as:

```
circuitID_[00-]
```

IxLoad will generate the strings `circuitID_00 - circuitID_99`, then repeat and again generate strings `circuitID_00 - circuitID_99`, then generate the final group of strings `circuitID_00 - circuitID_56`.

DHCP Statistics

The table below describes the DHCP client statistics.

For information on how the various DHCP options affect the size of a DHCP packet generated by IxLoad, see [Effect of Options on DHCP Packet Size](#) (on page 1034).

Statistic	Description
DHCP DHCPDISCOVER Response Time	<p>Amount of time elapsed between the time the client sent a DHCPDISCOVER request and the time it received an acceptable response to it. This statistic is updated when the client selects a DHCPOFFER, which can be affected by the server selection algorithm.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
DHCP DHCPREQUEST Response Time	<p>Amount of time elapsed between the time the client sent a DHCPREQUEST request and the time it received the first response to it.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
DHCP DHCPREQUEST (SELECTING) Response Time	<p>Amount of time elapsed between the time the client sent a DHCPREQUEST request in the Selecting state and the time it received the first response to it.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
DHCP DHCPREQUEST (INIT-REBOOT) Response Time	<p>Amount of time elapsed between the time the client sent a DHCPREQUEST request in the Init-Reboot state and the time it received the first response to it.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
DHCP DHCPREQUEST (RENEWING) Response Time	<p>Amount of time elapsed between the time the client sent a DHCPREQUEST request in the Renewing state and the time it received the first response to it.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>

DHCP DHCPREQUEST (REBINDING) Response Time	<p>Amount of time elapsed between the time the client sent a DHCPREQUEST request in the Rebinding state and the time it received the first response to it.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
DHCP DHCPINFORM Response Time	<p>Amount of time elapsed between the time the client sent a DHCPINFORM request and the time it received the first response to it.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
DHCP BOOTREQUEST Response Time	<p>Amount of time elapsed between the time the client sent a BOOTREQUEST request and the time it received the first response to it.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
DHCP Total Commands Sent	<p>Total number of commands sent by the client.</p> <p>Composite commands such as {Bind} are counted according to the number of actual DHCP commands they generate. For example, {Bind} normally generates two commands (DHCPDISCOVER and DHCPREQUEST) but it may generate three if it also sends a DHCPDECLINE.</p>
DHCP Total Commands Succeeded	Total number of commands sent by the client that succeeded.
DHCP Total Commands Failed	Total number of commands sent by the client that failed for all reasons.
DHCP Total Commands Failed (NAK Received)	Total number of commands sent by the client that failed because it received a NAK response.
DHCP Total Commands Failed (Timeout)	Total number of commands sent by the client that failed because it did not receive a response within the timeout period.
DHCP Total Commands Failed (Error)	Total number of commands sent by the client that failed because an error other than a timeout or NAK occurred.
DHCP Total Commands Retransmitted	Total number of commands that the client retransmitted.

DHCP Total Responses Matched	<p>Total number of responses received by the client in which the options in the response matched those it expected to receive.</p> <p>The expected options are provided on the Expected Options tab. If you provide a set of expected values or the server sends a set of values, a match between any of the expected values and received values is treated as success.</p>
DHCP Total Responses Mismatched	Total number of responses received by the client in which the options in the response did not match those it expected to receive.
DHCP DHCPDISCOVER Commands Sent	Total number of DHCPDISCOVER commands sent by the client.
DHCP DHCPDISCOVER Commands Succeeded	Total number of DHCPDISCOVER commands that succeeded.
DHCP DHCPDISCOVER Commands Failed	Total number of DHCPDISCOVER commands that failed for all reasons.
DHCP DHCPDISCOVER Commands Failed (Timeout)	Total number of DHCPDISCOVER commands that failed because the client did not receive a response within the timeout period.
DHCP DHCPDISCOVER Commands Failed (Error)	Total number of DHCPDISCOVER commands that failed because an error other than a NAK or timeout occurred.
DHCP DHCPDISCOVER Commands Retransmitted	Total number of DHCPDISCOVER commands that the client retransmitted.
DHCP DHCPDISCOVER Responses Matched	Total number of DHCPDISCOVER responses received by the client in which the options matched those that it expected to receive.
DHCP DHCPDISCOVER Responses Mismatched	Total number of DHCPDISCOVER responses received by the client in which the options did not match those that it expected to receive.
DHCP DHCPREQUEST Commands Sent	Total number of DHCPREQUEST commands sent by the client.
DHCP DHCPREQUEST Commands Succeeded	Total number of DHCPREQUEST commands that succeeded.
DHCP DHCPREQUEST Commands Failed	Total number of DHCPREQUEST commands that failed for all reasons.
DHCP DHCPREQUEST Commands Failed (NAK Received)	Total number of commands sent by the client that failed because it received a NAK response.
DHCP DHCPREQUEST Commands Failed (Timeout)	Total number of DHCPREQUEST commands that failed because the client did not receive a response within the timeout period.
DHCP DHCPREQUEST Commands Failed (Error)	Total number of DHCPREQUEST commands that failed because an error other than a NAK or timeout occurred.

DHCP DHCPREQUEST Commands Retransmitted	Total number of DHCPREQUEST commands that the client retransmitted.
DHCP DHCPREQUEST Responses Matched	Total number of DHCPREQUEST responses received by the client in which the options matched those that it expected to receive.
DHCP DHCPREQUEST Responses Mismatched	Total number of DHCPREQUEST responses received by the client in which the options did not match those that it expected to receive.
DHCP DHCPREQUEST (SELECTING) Commands Sent	Total number of DHCPREQUEST commands sent by the client while it was in the Selecting state.
DHCP DHCPREQUEST (SELECTING) Commands Succeeded	Total number of DHCPREQUEST commands sent by the client while it was in the Selecting state that succeeded.
DHCP DHCPREQUEST (SELECTING) Commands Failed	Total number of DHCPREQUEST commands sent by the client while it was in the Selecting state that failed for all reasons.
DHCP DHCPREQUEST (SELECTING) Commands Failed (NAK Received)	Total number of DHCPREQUEST commands sent by the client while it was in the Selecting state that failed because the client received a NAK response.
DHCP DHCPREQUEST (SELECTING) Commands Failed (Timeout)	Total number of DHCPREQUEST commands sent by the client while it was in the Selecting state that failed because the client did not receive a response within the timeout period.
DHCP DHCPREQUEST (SELECTING) Commands Failed (Error)	Total number of DHCPREQUEST commands sent by the client while it was in the Selecting state that failed because an error other than a NAK or timeout occurred.
DHCP DHCPREQUEST (SELECTING) Commands Retransmitted	Total number of DHCPREQUEST commands sent by the client while it was in the Selecting state that the client retransmitted.
DHCP DHCPREQUEST (SELECTING) Responses Matched	Total number of DHCPREQUEST responses received by the client while it was in the Selecting state in which the options matched those that it expected to receive.
DHCP DHCPREQUEST (SELECTING) Responses Mismatched	Total number of DHCPREQUEST responses received by the client while it was in the Selecting state in which the options did not match those that it expected to receive.
DHCP DHCPREQUEST (INIT-REBOOT) Commands Sent	Total number of DHCPREQUEST commands sent by the client while it was in the Init-Reboot state.
DHCP DHCPREQUEST (INIT-REBOOT) Commands Succeeded	Total number of DHCPREQUEST commands sent by the client while it was in the Init-Reboot state that succeeded.

DHCP DHCPREQUEST (INIT-REBOOT) Commands Failed	Total number of DHCPREQUEST commands sent by the client while it was in the Init-Reboot state that failed for all reasons.
DHCP DHCPREQUEST (INIT-REBOOT) Commands Failed (NAK Received)	Total number of DHCPREQUEST commands sent by the client while it was in the Init-Reboot state that failed because the client received a NAK response.
DHCP DHCPREQUEST (INIT-REBOOT) Commands Failed (Timeout)	Total number of DHCPREQUEST commands sent by the client while it was in the Init-Reboot state that failed because the client did not receive a response within the timeout period.
DHCP DHCPREQUEST (INIT-REBOOT) Commands Failed (Error)	Total number of DHCPREQUEST commands sent by the client while it was in the Init-Reboot state that failed because an error other than a NAK or timeout occurred.
DHCP DHCPREQUEST (INIT-REBOOT) Commands Retransmitted	Total number of DHCPREQUEST commands sent by the client while it was in the Init-Reboot state that the client retransmitted.
DHCP DHCPREQUEST (INIT-REBOOT) Responses Matched	Total number of DHCPREQUEST responses received by the client while it was in the Init-Reboot state in which the options matched those that it expected to receive.
DHCP DHCPREQUEST (INIT-REBOOT) Responses Mismatched	Total number of DHCPREQUEST responses received by the client while it was in the Init-Reboot state in which the options did not match those that it expected to receive.
DHCP DHCPREQUEST (RENEWING) Commands Sent	Total number of DHCPREQUEST commands sent by the client while it was in the Renewing state.
DHCP DHCPREQUEST (RENEWING) Commands Succeeded	Total number of DHCPREQUEST commands sent by the client while it was in the Renewing state that succeeded.
DHCP DHCPREQUEST (RENEWING) Commands Failed	Total number of DHCPREQUEST commands sent by the client while it was in the Renewing state that failed for all reasons.
DHCP DHCPREQUEST (RENEWING) Commands Failed (NAK Received)	Total number of DHCPREQUEST commands sent by the client while it was in the Renewing state that failed because the client received a NAK response.
DHCP DHCPREQUEST (RENEWING) Commands Failed (Timeout)	Total number of DHCPREQUEST commands sent by the client while it was in the Renewing state that failed because the client did not receive a response within the timeout period.
DHCP DHCPREQUEST (RENEWING) Commands Failed (Error)	Total number of DHCPREQUEST commands sent by the client while it was in the Renewing state that failed because an error other than a NAK or timeout occurred.

DHCP DHCPREQUEST (RENEWING) Commands Retransmitted	Total number of DHCPREQUEST commands sent by the client while it was in the Renewing state that the client retransmitted.
DHCP DHCPREQUEST (RENEWING) Responses Matched	Total number of DHCPREQUEST responses received by the client while it was in the Renewing state in which the options matched those that it expected to receive.
DHCP DHCPREQUEST (RENEWING) Responses Mismatched	Total number of DHCPREQUEST responses received by the client while it was in the Renewing state in which the options did not match those that it expected to receive.
DHCP DHCPREQUEST (REBINDING) Commands Sent	Total number of DHCPREQUEST commands sent by the client while it was in the Rebinding state.
DHCP DHCPREQUEST (REBINDING) Commands Succeeded	Total number of DHCPREQUEST commands sent by the client while it was in the Rebinding state that succeeded.
DHCP DHCPREQUEST (REBINDING) Commands Failed	Total number of DHCPREQUEST commands sent by the client while it was in the Rebinding state that failed for all reasons.
DHCP DHCPREQUEST (REBINDING) Commands Failed (NAK Received)	Total number of DHCPREQUEST commands sent by the client while it was in the Rebinding state that failed and for which the client received a NAK response.
DHCP DHCPREQUEST (REBINDING) Commands Failed (Timeout)	Total number of DHCPREQUEST commands sent by the client while it was in the Rebinding state that failed because the client did not receive a response within the timeout period.
DHCP DHCPREQUEST (REBINDING) Commands Failed (Error)	Total number of DHCPREQUEST commands sent by the client while it was in the Rebinding state that failed because an error other than a NAK or timeout occurred.
DHCP DHCPREQUEST (REBINDING) Commands Retransmitted	Total number of DHCPREQUEST commands sent by the client while it was in the Rebinding state that the client retransmitted.
DHCP DHCPREQUEST (REBINDING) Responses Matched	Total number of DHCPREQUEST responses received by the client while it was in the Rebinding state in which the options matched those that it expected to receive.
DHCP DHCPREQUEST (REBINDING) Responses Mismatched	Total number of DHCPREQUEST responses received by the client while it was in the Rebinding state in which the options did not match those that it expected to receive.
DHCP DHCPDECLINE Commands Sent	Total number of DHCPDECLINE commands sent by the client.
DHCP DHCPDECLINE Commands Send Failed	Total number of DHCPDECLINE commands that failed for all reasons.



DHCP DHCPRELEASE Commands Sent	Total number of DHCPRELEASE commands sent by the client.
DHCP DHCPRELEASE Commands Send Failed	Total number of DHCPRELEASE commands that failed for all reasons.
DHCP DHCPINFORM Commands Sent	Total number of DHCPINFORM commands sent by the client.
DHCP DHCPINFORM Commands Succeeded	Total number of DHCPINFORM commands that succeeded.
DHCP DHCPINFORM Commands Failed	Total number of DHCPINFORM commands that failed for all reasons.
DHCP DHCPINFORM Commands Failed (Timeout)	Total number of DHCPINFORM commands that failed because the client did not receive a response within the timeout period.
DHCP DHCPINFORM Commands Failed (Error)	Total number of DHCPINFORM commands that failed because an error other than a NAK or timeout occurred.
DHCP DHCPINFORM Commands Retransmitted	Total number of DHCPINFORM commands that the client retransmitted.
DHCP DHCPINFORM Responses Matched	Total number of DHCPINFORM responses received by the client in which the options matched those that it expected to receive.
DHCP DHCPINFORM Responses Mismatched	Total number of DHCPINFORM responses received by the client in which the options did not match those that it expected to receive.
DHCP BOOTREQUEST Commands Sent	Total number of BOOTREQUEST commands sent by the client.
DHCP BOOTREQUEST Commands Succeeded	Total number of BOOTREQUEST commands that succeeded.
DHCP BOOTREQUEST Commands Failed	Total number of BOOTREQUEST commands that failed for all reasons.
DHCP BOOTREQUEST Commands Failed (Timeout)	Total number of BOOTREQUEST commands that failed because the client did not receive a response within the timeout period.
DHCP BOOTREQUEST Commands Failed (Error)	Total number of BOOTREQUEST commands that failed because an error other than a NAK or timeout occurred.
DHCP BOOTREQUEST Commands Retransmitted	Total number of BOOTREQUEST commands that the client retransmitted.
DHCP BOOTREQUEST Responses Matched	Total number of BOOTREQUEST responses received by the client in which the options matched those that it expected to receive.
DHCP BOOTREQUEST Responses Mismatched	Total number of BOOTREQUEST responses received by the client in which the options did not match those that it expected to receive.
DHCP Total Number of DHCPOFFER Messages	Total number of DHCPOFFER commands received by the client.

DHCP Number of DHCPOFFER Messages Ignored	Total number of DHCPOFFER messages that the client ignored.
DHCP Total Number of DHCPACK Messages	Total number of DHCPACK commands received by the client.
DHCP Number of DHCPACK Messages Ignored	Total number of DHCPACK messages that the client ignored.
DHCP Total Number of DHCPNAK Messages	Total number of DHCPNAK messages received by the client.
DHCP Number of DHCPNAK Messages Ignored	Total number of DHCPNAK messages that the client ignored.
DHCP ICMP Echo Messages Received	Total number of ICMP Echo (ping) messages received by the client.
DHCP ICMP Echo Reply Messages Sent	Total number of ICMP Echo (ping) reply messages sent by the client.
DHCP ARP Request Messages Received	Total number of ARP requests received by the client.
DHCP ARP Reply Messages Sent	Total number of ARP replies sent by the client.
DHCP Valid IP Addresses Received	Total number of valid IP addresses received by the client.
DHCP Duplicate IP Addresses Received	Total number of duplicate IP addresses received by the client.
DHCP User Count	Number of DHCP users simulated by the client.
DHCP Total Transaction	Total number of DHCP transactions completed by the client. Note: DHCPRELEASE and DHCPDECLINE do not contribute to this statistic.
DHCP Number of Active Leases	Total number of IP address leases received that have not expired, been released (by sending DHCPRELEASE), or been declined (by sending DHCPDECLINE).
DHCP Number of Leases Expired	Total number of IP address leases that have expired.
DHCP Number of Clients Awaiting IP Address from Server	Total number of DHCP clients waiting to receive IP addresses from a server.
DHCP Total Bytes Transmitted	Total number bytes transmitted by the client. Note: All of the "Total Bytes" statistics count all the bytes in the packet, including the UDP and IP headers.
DHCP Total Bytes Received	Total number of bytes received by the client.
DHCP Total Bytes Transmitted and Received	Combined total of bytes transmitted and received by the client.



Effect of Options on DHCP Packet Size

The table below describes how the various DHCP options affect the size of a DHCP packet generated by IxLoad.

Description	Bytes
Size of headers and other fixed fields:	278
If "Allow Options Overload" is enabled, number of bytes added regardless of whether IxLoad actually overloads the options or not:	3
If the Maximum DHCP Message Size option is enabled, number of bytes added to a DHCP (not BOOTP) packet:	4
Number of bytes added for each option in the Option set used by a particular command:	Size of the option, including code, len, and data fields.
If the Host Name option is enabled, number of bytes added to the size of the (user-specified) data:	10
Note: If all options cannot fit into the packet and "Allow Options Overload" is enabled, IxLoad first tries to fit the extra options into "file" field of the DHCP packet header, and then into the "sname" field. Options that are placed in the "file" or the "sname" fields do not contribute to the packet size calculation.	



DNS

This section describes the DNS Tcl API objects.

In this chapter:

Overview	1036
DNS Client Agent	1040
DNS Server Agent	1047
DNS Statistics	1060

Overview

DNS protocol commands are organized as a simple structure.

- DNS Client Agent
- DNS Client Query
- DNS Client Advanced Options
 - - DNS Server Agent
 - - DNS Server Zone Management
 - - DNS Server Zone Configuration
 - - DNS Server Advanced Options
 - - DNS Server Resource Records

Objectives

The objectives (userObjective) you can set for DNS are listed below. Test objecare set in the ixTimeline object.

- simulatedUsers
- transactionRate
- throughputKBps
- transactionAttemptRate (displays as “Queries/Second” in the GUI)

DNS Client Agent

The DNS Client Agent defines a client performing DNS commands. Refer to `DNS Client Agent` for a full description of this command.

The important options and subobjects of this command are listed below.

Option	Usage
enable	Enables the use of the DNS client agent.
name	The name associated with the client agent.

Option	Usage
dnsConfig	A list of DNS Client Query objects.
advancedOptions	DNS Client Advanced Options.

DNS Client Query

Each DNS query is a single query command that a client makes of a server. Refer to `DNS Client Query` for a full description of this command. The important options of this command are listed below.

Option	Usage
hostName	The host name whose DNS entry is being queried for.
queryType	The type of DNS query to perform.
dnsServer	The name of the DNS server to query.
recursionDesired	Indicates DNS referrals are to be followed.
expect	The expected answer for the query. Optional.

DNS Client Advanced Options

The DNS client advanced options control network level operation of the client. Refer to `DNS Client Advanced Options` for a full description of this command. The important options of this command are listed below.

Option	Usage
lowerLayerTransport	Whether to use TCP or UDP.
responseTimeout	How long the client waits for a server response.
numberOfRetries	The number of retries to attempt.
noWaitForResp	Whether or not to wait for a response before sending the next query

DNS Server Agent

The DNS Server Agent defines the operation of the DNS server. Refer to `DNS Server Agent` for a full description of this command. The important options and subobjects of this command are listed below.

Option	Usage
enable	Enables the use of the HTTP server agent.
name	The name associated with the server agent.

Option	Usage
--------	-------



Option	Usage
zoneMgr	A list of possible zones that may be served by this server, as described in <code>DNS Server Zone Management</code> . Fill in this list first, and then select from the list in the <code>zoneConfig</code> option.
zoneConfig	A list actively servered zones, as described in <code>DNS Server Zone Configuration</code> .
advancedOptions	<code>DNS Server Advanced Options</code> .

DNS Server Zone Management

Each DNS Server zone management options define a particular zone that the server is authoritative for. The options of this object are the contents of a standard DNS server's SOH (Start of Authority) record. Refer to `DNS Server Zone Management` for a full description of this command. The important options of this command are listed below.

Option	Usage
name	The name of the zone.
masterServer	The master server IP address.
serial	The Start of Authority serial number.
expire	The Start of Authority expiration time.
resourceRecordList	A list of resource records for the zone, described in <code>DNS Server Resource Records</code> .

DNS Server Zone Configuration

The important options of this command are listed below.

Option	Usage
name	The name of a zone from the server agent's <code>zoneMgr</code> list to be served by the agent.

DNS Server Advanced Options

The DNS server advanced options control network level operation of the server. The important options of this command are:

Option	Usage
listeningPort	The port number that the server listens on.

DNS Server Resource Records

Each resource record represents a piece of information available from the server. The important options of this command are:

Option	Usage
id	The type of record.
hostName	The host name for the record, for use with type A and PTR records.
address	The address of a host, for use with type A records.
name	The name of a host, for use with MX and CNAME records.
mailServer	The address of a mail server, for use with MX records.
priority	The priority of a mail server, for use with MX records.
realName	The real server name, for use with CNAME records.
ipAddress	An IP address, for use with PTR records.
zoneName	The name of the zone, for use with NS records.
nameServer	The IP address of the name server for a zone, for use with NS records.



DNS Client Agent

DNS Client Agent - configure a DNS client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DNSClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_DNSClient1 agent.config
```

DESCRIPTION

A DNS client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity DNSClient1 of NetTraffic Traffic1@Network1
#####
set Activity_DNSClient1 [$Traffic1_Network1 activityList.appendItem \
    -protocolAndType                "DNS Client" ]
#####
```

```
# Timelinel for activities DNSClient1

#####

set Timelinel [::IxLoad new ixTimeline]

$Timelinel config \

    -rampUpValue          1 \
    -rampUpType           0 \
    -offlineTime          0 \
    -rampDownTime         20 \
    -standbyTime          0 \
    -iterations           1 \
    -rampUpInterval       1 \
    -sustainTime          20 \
    -timelineType         0 \
    -name                  "Timelinel"

$Activity_DNSClient1 config \

    -enable               true \
    -name                  "DNSClient1" \
    -enableConstraint     false \
    -userObjectiveValue   100 \
    -constraintValue      100 \
    -userObjectiveType    "simulatedUsers" \
    -timeline              $Timelinel

$Activity_DNSClient1 agent.config \

    -enable               true \
    -name                  "DNSClient1"

$Activity_DNSClient1 agent.pm.advancedOptions.config \

    -lowerLayerTransport  1 \
    -noWaitForResp        false \
```



```
-version                                0 \  
-responseTimeout                       20 \  
-implicitLoopCheck                     true \  
-numberOfRetries                       3  
$Activity_DNSClient1 agent.pm.seqGenExample.config \  
-dummy                                  ""  
$Activity_DNSClient1 agent.pm.dnsConfig.dnsQueries.clear  
$Activity_DNSClient1 agent.pm.dnsConfig.dnsQueries.appendItem \  
-id                                     "DnsQuery" \  
-expect                                 "" \  
-hostName                               "localhost" \  
-queryType                              "A" \  
-recursionDesired                      0 \  
-dnsServer                              "Traffic2_DNSServer1:53"
```

SEE ALSO

DNS Client Query, DNS Client Advanced Options, ixNetTraffic

DNS Client Query

DNS Client Query - configure a DNS query that the client will send

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DNSClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_DNSClient1 agent.pm.dnsConfig.dnsQueries.appendItem
```

DESCRIPTION

A DNS client query is added to the `pm.dnsConfig.dnsQueries` option of the DNS Client Agent object using its `appendItem`.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

None.

OPTIONS

`dnsServer`

The name of the DNS server to be queried. (Default = 'None').

`expect`

The expected answer for the query; optional. (Default = "").

`hostName`

The host name to be queried for. (Default = 'localhost'). If the `queryType` option is ENUM then the `hostName` option accepts only integers. You can use both independent and interdependent sequence generators if the `queryType` option is ENUM. See the Automatic Sequence Generators appendix for more information.

`loopValue`

If this option is enabled (1), then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (0), then the client will progress through the command list only once, and then go idle. (Default = 0).

`queryType`

The type of query to be performed. One of:



option	Usage
"A"	(default) An Address query.
"AAAA"	An IPV6 address retrieval query
"NS"	A Name Server query.
"CNAME"	A CName query.
"SOA"	A Start Of Authority query.
"PTR"	A Pointer query.
"MX"	A Mail eXchanger query.
"ENUM"	A query that resolves fully qualified telephone numbers to fully qualified domain name addresses.

recursionDesired

Indicates whether DNS referrals are to be followed or not. (Default = false).

EXAMPLE

```
$Activity_DNSClient1 agent.pm.dnsConfig.dnsQueries.appendItem \  
-id "DnsQuery" \  
-expect "" \  
-hostName "localhost" \  
-queryType "A" \  
-recursionDesired 0 \  
-dnsServer "Traffic2_DNSServer1:53"
```

SEE ALSO

DNS Client Agent

DNS Client Advanced Options

DNS Client Advanced Options - configure the DNS client's advanced options

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_DNSClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_DNSClient1 agent.pm.advancedOptions.config
```

DESCRIPTION

DNS advanced options are set through the `pm.advancedOptions` option of the `DNS Client Agent` object.

SUBCOMMANDS

None.

OPTIONS

`lowerLayerTransport`

The type of IP transport to be used for the queries in this client. One of:

option	Usage
"TCP"	TCP.
"UDP"	(default) UDP. The number of retries is only confor this option.

`numberOfRetries`

If `lowerLayerTransport` is `true`, this is the number of retries for the query. (Default = 3).

`responseTimeout`

The time, expressed in seconds, to wait for a DNS server response. (Default = 20).

`noWaitForResp`

This option accepts boolean value, `true` or `false`. This parameter is only effective when you set the client's `userObjectiveType` to `queriesPerSecond`. If `true`, the clidoes not wait for a response before sending the next query. Besides, fewer simulated users are created, and a higher `objectiveValue` (more `queriesPerSecis` likely to be achieved.

If `false`, the client waits for a response before sending the next query. More simuusers are created, and a lower `objectiveValue` (fewer `queriesPerSecond`) is likely to be achieved.

EXAMPLE

```
$Activity_DNSClient1 agent.pm.advancedOptions.config \  
-lowerLayerTransport          1 \  
-noWaitForResp                false \  
-version                       0 \  
-responseTimeout              20 \  
-implicitLoopCheck            true \  
-numberOfRetries               3
```

SEE ALSO

DNS Client Agent

DNS Server Agent

DNS Server Agent - configure a DNS server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_DNSServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_DNSServer1 agent.config
```

DESCRIPTION

A DNS server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this action. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
#####
# Activity DNSServer1 of NetTraffic Traffic2@Network2
#####
set Activity_DNSServer1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType                "DNS Server" ]
set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]
```



```
$Activity_DNSServer1 config \  
    -enable true \  
    -name "DNSServer1" \  
    -timeline $_Match_Longest_  
$Activity_DNSServer1 agent.config \  
    -enable true \  
    -name "DNSServer1"  
$Activity_DNSServer1 agent.pm.zoneConfig.zoneList.clear  
$Activity_DNSServer1 agent.pm.zoneConfig.zoneList.appendItem \  
    -id "ZoneList" \  
    -name "ixiacom.com"  
$Activity_DNSServer1 agent.pm.advancedOptions.config \  
    -listeningPort 53  
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices.clear  
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices.appendItem \  
    -id "Zone" \  
    -predefine true \  
    -serial 1234 \  
    -expire 8888 \  
    -name "localhost" \  
    -masterServer "ixia-dns-tester"  
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices(0).resourceRecordList.clear  
$Activity_DNSServer1  
agent.pm.zoneMgr.zoneChoices(0).resourceRecordList.appendItem \  
    -id "A" \  
    -hostName "localhost" \  
    -address "127.0.0.1"  
$Activity_DNSServer1  
agent.pm.zoneMgr.zoneChoices(0).resourceRecordList.appendItem \  
    -id "A" \  

```

```
-hostName                "host1" \  
-address                 "198.18.0.1" \  
  
$Activity_DNSServer1  
agent.pm.zoneMgr.zoneChoices(0).resourceRecordList.appendItem \  
  
-id                     "NS" \  
-nameServer             "198.18.0.2" \  
-zoneName               "localhost" \  
  
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices.appendItem \  
  
-id                     "Zone" \  
-predefine              true \  
-serial                 1234 \  
-expire                 8888 \  
-name                   "ixiacom.com" \  
-masterServer           "ixia-dns-tester" \  
  
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.clear  
  
$Activity_DNSServer1  
agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.appendItem \  
  
-id                     "A" \  
-hostName               "puppy1" \  
-address                "198.18.1.100" \  
  
$Activity_DNSServer1  
agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.appendItem \  
  
-id                     "A" \  
-hostName               "drowzee" \  
-address                "198.18.1.200" \  
  
$Activity_DNSServer1  
agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.appendItem \  
  
-id                     "CNAME" \  
-name                   "testName" \  
-realName               "realName"
```



```
$Activity_DNSServer1
agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.appendItem \

    -id                "NS" \
    -nameServer        "198.18.0.2" \
    -zoneName          "ixiacom.com"
```

SEE ALSO

DNS Server Zone Management, DNS Server Zone Configuration, DNS Server Advanced Options



```
-predefine true \  
-serial 1234 \  
-expire 8888 \  
-name "ixiacom.com" \  
-masterServer "ixia-dns-tester"
```

```
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.clear
```

```
$Activity_DNSServer1
```

```
agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.appendItem \  

```

```
-id "A" \  
-hostName "puppy1" \  
-address "198.18.1.100"
```

```
$Activity_DNSServer1
```

```
agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.appendItem \  

```

```
-id "A" \  
-hostName "drowzee" \  
-address "198.18.1.200"
```

```
$Activity_DNSServer1
```

```
agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.appendItem \  

```

```
-id "CNAME" \  
-name "testName" \  
-realName "realName"
```

```
$Activity_DNSServer1
```

```
agent.pm.zoneMgr.zoneChoices(1).resourceRecordList.appendItem \  

```

```
-id "NS" \  
-nameServer "198.18.0.2" \  
-zoneName "ixiacom.com"
```

SEE ALSO

DNS Server Agent, DNS Server Resource Record



DNS Server Zone Configuration

DNS Server Zone Configuration - setup the zones that the DNS server is authoritative for

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_DNSServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_DNSServer1 agent.pm.zoneConfig.zoneList.appendItem
```

DESCRIPTION

Each DNS server zone configuration list item represents a DNS domain that the server will respond to.

SUBCOMMANDS

None.

OPTIONS

name

The name of the domain, for example, "ixiacom.com." (Default = "Zone0").

EXAMPLES

```
$Activity_DNSServer1 agent.pm.zoneConfig.zoneList.appendItem \  
-id "ZoneList" \  
-name "ixiacom.com"
```

SEE ALSO

DNS Server Agent, DNS Server Resource Record

DNS Server Advanced Options

DNS Server Advanced Options

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_DNSServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_DNSServer1 agent.pm.advancedOptions.config
```

DESCRIPTION

DNS server advanced options are set through the `pm.advancedOptions` option of the `DNS Server Agent` object.

SUBCOMMANDS

None.

OPTIONS

`listenPort`

The port number that the server listens on for TCP and UDP requests. (Default = 53)

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
#####
# Activity DNSServer1 of NetTraffic Traffic2@Network2
#####
set Activity_DNSServer1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType                "DNS Server" ]
$Activity_DNSServer1 agent.config \
    -enable                          true \
    -name                            "DNSServer1"
$Activity_DNSServer1 agent.pm.advancedOptions.config \
    -listeningPort                   53
```

SEE ALSO

DNS Server Agent



DNS Server Resource Record

DNS Server Resource Record - add a resource record to the DNS server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_DNSServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices(0).resourceRe
```

DESCRIPTION

Each DNS server resource record list item represents a DNS domain that the server is authoritative over.

SUBCOMMANDS

None.

OPTIONS

id

Specifies the type of resource record defined. The remaining options in this comare dependent on this setting. One of:

option	Usage
A	Address record.
AAAA	IPV6 Address record
MX	Mail eXchanger record.
CNAME	Canonical Name record.
PTR	Pointer, or reverse DNS record.
NS	Name Server record.

Options for id = A

address

The IP address of a host. (Default = "").

hostName

The name of the host. (Default = "").

Options for id = AAAA

address

The IPV6 address of a host. (Default = "").

hostName

The name of the host. (Default = "").

Options for id = MX

mailServer

The name of the mail server. (Default = "").

name

The mail domain name. (Default = "").

priority

The priority associated with the mail server. (Default = "").

Options for id = CNAME

name

An alias of a host. (Default = "").

realName

The real name of the host, as it appears in an A record. (Default = "").

Options for id = PTR

hostName

The host name for the ipAddress. (Default = "").

ipAddress

The IP address for the reverse lookup. (Default = "").

Options for id = NS

nameServer

The IP address for the name server. (Default = "").

zoneName

The zone name being served. (Default = "").

EXAMPLES

```
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices.appendItem \
```

```
-id                "Zone" \
-predefine         true \
-serial            1234 \
-expire            8888 \
-name              "localhost" \
-masterServer      "ixia-dns-tester"
```

```
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices(0).resourceRecordList.clear
```

```
$Activity_DNSServer1
agent.pm.zoneMgr.zoneChoices(0).resourceRecordList.appendItem \
```

```
-id                "A" \
-hostName          "localhost" \
-address           "127.0.0.1"
```

```
$Activity_DNSServer1
agent.pm.zoneMgr.zoneChoices(0).resourceRecordList.appendItem \
```

```
-id                "A" \
-hostName          "host1" \
-address           "198.18.0.1"
```

```
$Activity_DNSServer1
agent.pm.zoneMgr.zoneChoices(0).resourceRecordList.appendItem \
```

```
-id                "NS" \
-nameServer        "198.18.0.2" \
-zoneName          "localhost"
```



```
$Activity_DNSServer1 agent.pm.zoneMgr.zoneChoices.appendItem \  
-id "Zone" \  
-predefine true \  
-serial 1234 \  
-expire 8888 \  
-name "ixiacom.com" \  
-masterServer "ixia-dns-tester"
```

SEE ALSO

DNS Server Agent, DNS Server Zone Management

DNS Statistics

For DNS client statistics, see DNS Client Statistics (on page 1060).

For DNS server statistics, see DNS Server Statistics (on page 1063).

DNS Client Statistics

The table below describes the DNS client statistics.

Statistic	Description
General Statistics	
DNS Total Queries Sent	Total number of DNS queries sent by the client.
DNS Total Queries Successful	Total number of DNS queries for which a valid response was received.
DNS Total Queries Retried	Total number of DNS queries that had to be re-sent at least once.
DNS Total Queries Failed	Total number of DNS queries that failed for all reasons.
DNS Total Queries Failed (Format Error)	Number of DNS queries that failed because the DNS server could not interpret the format of the query. Note: According to RFC 1034, the maximum host name length is 63 bytes. IxLoad does not enforce this limit, and allows you to create queries for host names larger than 63 bytes. If you configure the DNS client to send a query to a host name that is larger than 63 bytes, the DNS server responds with a Format Error.

DNS Total Queries Failed (Server Failure)	<p>Number of DNS queries that failed due to an error on the DNS server.</p> <p>Note: According to RFC 1034, the maximum host name length is 63 bytes. IxLoad does not enforce this limit, and allows you to create Resource Records that include host names larger than 63 bytes.</p> <p>If you configure the DNS client to send a query to server zone for a Resource Record that contains a host name that is larger than 63 bytes, the DNS server responds with a Server Failure.</p>
DNS Total Queries Failed (Name Error)	Number of DNS queries that failed because the DNS name does not exist.
DNS Total Queries Failed (Not Implemented)	Number of DNS queries that failed because the name server does not support the DNS request.
DNS Total Queries Failed (Refused)	Number of DNS queries that failed because the DNS server refused the request.
DNS Total Queries Failed (Other)	Number of DNS queries that failed for unknown reasons.
DNS Total Queries Failed (Timeout)	Number of DNS queries that failed because no response was received within the timeout period.
DNS Total Queries Failed (Aborted)	Number of aborted DNS queries.
A Record Statistics	
DNS (Type A) Queries Sent	Number of A record queries sent.
DNS (Type A) Queries Successful With Match	Number of A record queries for which the DNS client received the expected IP address.
DNS (Type A) Queries Successful Without Match	Number of A record queries which were processed without error but whose responses did not contain the expected IP address.
DNS (Type A) Queries Failed	Number of A record queries for which an invalid response was received, or no response was received.
AAAA Record Statistics	
DNS (Type AAAA) Queries Sent	Number of AAAA record queries sent.
DNS (Type AAAA) Queries Successful With Match	Number of AAAA record queries for which the DNS client received the expected IP address.
DNS (Type AAAA) Queries Successful Without Match	Number of AAAA record queries which were processed without error but whose responses did not contain the expected IP address.
DNS (Type AAAA) Queries Failed	Number of AAAA record queries for which an invalid response was received, or no response was received.
CNAME Record Statistics	
DNS (Type CNAME) Queries Sent	Number of canonical name record queries sent.
DNS (Type CNAME) Queries Successful With Match	Number of canonical name record queries for which the DNS server returned the expected host name.



DNS (Type CNAME) Queries Successful Without Match	Number of canonical name record queries which were processed without error but whose responses did not contain the expected host name.
DNS (Type CNAME) Queries Failed	Number of canonical name record queries for which an invalid response was received, or no response was received.
MX Record Statistics	
DNS (Type MX) Queries Sent	Number of mail exchange record queries sent.
DNS (Type MX) Queries Successful With Match	Number of mail exchange record queries for which the response contained the expected mail server host name.
DNS (Type MX) Queries Successful Without Match	Number of mail exchange record queries that were processed without error but for which the response did not contain the expected mail server host name.
DNS (Type MX) Queries Failed	Number of mail exchange record queries for which an invalid response was received, or no response was received.
PTR Record Statistics	
DNS (Type PTR) Queries Sent	Number of pointer record queries sent.
DNS (Type PTR) Queries Successful With Match	Number of pointer record queries for which the DNS client received the expected canonical host name for the supplied IP address.
DNS (Type PTR) Queries Successful Without Match	Number of pointer record queries that were processed correctly but whose responses did not contain the expected canonical host name.
DNS (Type PTR) Queries Failed	Number of pointer record queries for which an invalid response was received, or no response was received.
NS Record Statistics	
DNS (Type NS) Queries Sent	Number of name server record queries sent.
DNS (Type NS) Queries Successful With Match	Number of name server record queries for which the DNS server returned the name server expected for the supplied domain.
DNS (Type NS) Queries Successful Without Match	Number of name server record queries which the DNS server processed without error but whose responses did not contain the expected name server.
DNS (Type NS) Queries Failed	Number of name server record queries for which an invalid response was received, or no response was received.
SOA Record Statistics	
DNS (Type SOA) Queries Sent	Number of Start of Authority record queries sent.
DNS (Type SOA) Queries Successful With Match	Number of Start of Authority record queries for which the DNS client received the expected DNS server for the supplied domain.
DNS (Type SOA) Queries Successful Without Match	Number of Start of Authority record queries which were processed without error but whose responses did not contain the name of the expected DNS server.

NAPTR (ENUM) Query Statistics	
DNS (Type NAPTR) Queries Sent	Number of Naming Authority Pointer (ENUM) record queries sent.
DNS (Type NAPTR) Queries Successful With Match	Number of Naming Authority Pointer (ENUM) queries for which the response contained a string that matched the Expect field.
DNS (Type NAPTR) Queries Successful Without Match	Number of Naming Authority Pointer (ENUM) queries which were processed without error but for which the response did not contain a string that matched the Expect field.
DNS (Type NAPTR) Queries Failed	Number of Naming Authority Pointer (ENUM) queries for which an invalid response was received, or no response was received.
Bytes Transmitted and Received Statistics	
DNS Total Bytes Transmitted	Total bytes transmitted for all DNS queries, including re-tried queries.
DNS Total Bytes Received	Total bytes received for all DNS responses.
Test Objective Statistics	
DNS Bytes	Combined total number of DNS bytes transmitted and received.
DNS Throughput	Combined rate that the client received and transmitted DNS bytes.
DNS Transactions	Total number of DNS transactions completed. A DNS transaction consists of one query and one response to it.
DNS Transaction Rate	Rate at which the client completed DNS transactions completed.
DNS Simulated Users	Number of simulated users generating DNS queries.

DNS Server Statistics

The table below describes the DNS server statistics.

Statistic	Description
General Statistics	
DNS Total Queries Received	Total number of DNS queries received by the server.
DNS Total Queries Responded Successfully	Total number of DNS queries for which the server returned a valid response.
DNS Total Queries Failed	Total number of DNS queries which the server could not process for any reason.

DNS Total Queries Failed (Format Error)	<p>Number of DNS queries which the server could not process because it could not parse the query format.</p> <p>Note: According to RFC 1034, the maximum host name length is 63 bytes. IxLoad does not enforce this limit, and allows you to create queries for host names larger than 63 bytes.</p> <p>If you configure the DNS client to send a query to a host name that is larger than 63 bytes, the DNS server responds with a Format Error.</p>
DNS Total Queries Failed (Server Failure)	<p>Number of DNS queries that failed due to an error on the server.</p> <p>Note: According to RFC 1034, the maximum host name length is 63 bytes. IxLoad does not enforce this limit, and allows you to create Resource Records that include host names larger than 63 bytes.</p> <p>If you configure the DNS client to send a query to server zone for a Resource Record that contains a host name that is larger than 63 bytes, the DNS server responds with a Server Failure.</p>
DNS Total Queries Failed (Name Error)	Number of DNS queries that failed because the DNS name does not exist.
DNS Total Queries Failed (Not Implemented)	Number of DNS queries that failed because the name server does not support the DNS request.
DNS Total Queries Failed (Refused)	Number of DNS queries that failed because the server refused to serve the query.
DNS Total Queries Failed (Other)	Number of DNS queries that failed for unknown reasons.
A Record Statistics	
DNS (Type A) Queries Received	Number of A record queries received.
DNS (Type A) Queries Responded Successfully	Number of A record queries for which the DNS server returned a valid response.
DNS (Type A) Queries Failed	Number of A record queries that failed for any reason.
CNAME Record Statistics	
DNS (Type CNAME) Queries Received	Number of canonical name record queries received.
DNS (Type CNAME) Queries Responded Successfully	Number of canonical name record queries for which the DNS server returned a valid response.
DNS (Type CNAME) Queries Failed	Number of canonical name record queries that failed for any reason.
MX Record Statistics	
DNS (Type MX) Queries Received	Number of mail exchange record queries received.
DNS (Type MX) Queries Responded Successfully	Number of mail exchange record queries for which the DNS server returned a valid response.

DNS (Type MX) Queries Failed	Number of mail exchange record queries that failed for any reason.
PTR Record Statistics	
DNS (Type PTR) Queries Received	Number of pointer record queries received.
DNS (Type PTR) Queries Responded Successfully	Number of pointer record queries for which the DNS server returned a valid response.
DNS (Type PTR) Queries Failed	Number of pointer record queries that failed for any reason.
NS Record Statistics	
DNS (Type NS) Queries Received	Number of name server record queries received.
DNS (Type NS) Queries Responded Successfully	Number of name server record queries for which the DNS server returned a valid response.
DNS (Type NS) Queries Failed	Number of name server record queries that failed for any reason.
SOA Record Statistics	
DNS (Type SOA) Queries Received	Number of Start of Authority record queries received.
DNS (Type SOA) Queries Responded Successfully	Number of Start of Authority record queries for which the DNS server returned a valid response.
DNS (Type SOA) Queries Failed	Number of Start of Authority record queries that failed for any reason.
Bytes Transmitted and Received Statistics	
DNS Total Bytes Transmitted	Total bytes transmitted for all DNS queries, including re-tried queries.
DNS Total Bytes Received	Total bytes received for all DNS responses.
DNS Total Bytes Transmitted and Received	Combined total of bytes received in DNS queries and transmitted in DNS responses.



FTP

This section describes the FTP Tcl API objects.

In this chapter:

Overview	1067
Objectives	1067
FTP Client Agent	1067
FTP Server Agent	1068
FTP Client Agent	1070
FTP Server Agent	1084
FTP Statistics	1087

Overview

FTP protocol commands are organized as follows.

- FTP Client Agent
 - FTP Client Action
- FTP Server Agent

Objectives

The objectives (userObjective) you can set for FTP are listed below. Test objecare set in the ixTimeline object.

- connectionRate
- transactionRate
- simulatedUsers
- concurrentConnections
- throughputMBps
- throughputKBps

FTP Client Agent

The FTP Client Agent defines a simulated user performing FTP requests against one or more FTP servers. Refer to FTP Client Agent for a full description of this command. The important options of this command are listed in the table below:

Option	Usage
enable	Enables the use of the FTP client agent.
name	The name associated with the client agent.
actionList	The list of actions associated with the agent.
realFileList	Add real files in client and server plugins.
mode	The active/passive mode of the FTP interaction.
userName	The default user name for actions.
password	The default password for actions.



FTP Client Action

Each client action is a single step in the interaction. Refer to `FTP Client Action` for a full description of this command. The important subcommands and options of this command are listed below.

Subcommand	Usage
<code>checkConfig</code>	Checks the configuration of the action.

Option	Usage
<code>command arguments</code>	The FTP command, with optional arguments, to be executed.
<code>destination</code>	The name/address of the FTP server.
<code>userName</code>	The user name to use for login commands.
<code>password</code>	The password to use for login commands.

FTP Server Agent

The FTP Server Agent defines the operation of the FTP server. Refer to `FTP Server Agent` for a full description of this command. The important options of this command are listed in the table below:

Option	Usage
<code>enable</code>	Enables the use of the FTP server agent.
<code>name</code>	The name associated with the server agent.
<code>ftpPort</code>	The port number that the server will respond on.
<code>realFileList</code>	Add real files in client and server plugins.

realFileList

To add real files, use the `realFileList` is exposed in both client and server plugins. It is a sequence container of `RealFileObjects`. `RealFileObjects` have two configurables exposed, `page` and `payload`.

Option	Usage
<code>page</code>	Any linux file name (<code>client_file1</code>)
<code>payload</code>	The actual path of the file. ("C:\\Program Files\\Ixia\\IxLoad\\buildversion.ini")

For a sample script refer to the example section of FTP Client Agent .



FTP Client Agent

FTP Client Agent - create an FTP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_FTPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
Activity_FTPClient1 agent.config options...
```

DESCRIPTION

An FTP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

None.

OPTIONS

`actionList`

A list of actions that the agent should perform, of type `FTP Client Action`.

`enable`

Enables the use of this agent. (Default = `true`).

`enableEsm`

If `true`, the use of the `esm` option is enabled. (Default = `false`).

`enableTos`

Enables the setting of the TOS (Type of Service) bits in the header of the FTP packets. Use the `tos` option to specify the TOS bit setting.

0 (default) TOS bits not enabled.

1 TOS bits enabled.

`esm`

If `enableEsm` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size as 1,460 bytes. (`Default = 1,460`).

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, `IxLoad` sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (`Default = false`).

`fileList`

Represents a list of filenames which is of the form `#<a number>`. These can be referred in the `arguments` option in any of the `ixFtpAction` Objects.

`vlanPriority`

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.

`ipPreference`

If a mixture of IPv4 and IPv6 addresses are available on the client network, this parameter configures which address types the agent uses.

- 0 IPv4
- 1 IPv6
- 2 (default) Both, IPv4 first
- 3 Both, IPv6 first

`loopValue`

If this option is enabled (1), then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (0), then the client will progress through the command list only once, and then go idle. (`Default = 0`).

`mode`

The mode that the client will use to access the server: To establish an FTP conthe client connects from a random unprivileged port (port `n`, where `n` is greater than 1,024) to the FTP server's command port, normally port 21. What happens next depends on whether the client is in active or passive mode. The choices are listed below:

Option	Usage
"ACTIVE"	(Default). The client sends the PORT command and waits for an OK response from the server.



Option	Usage
"PASSIVE"	In Passive mode, the client initiates both connecto the server.

name

The name associated with this object, which must be set at object creation.

password

Enter the password for the default user name in `userName`. When you use a LOGIN action in the action list, this password will be used by default. (Default = "noreply@ixiacom.com").

You can insert sequence generators into this field to create unique entries automatically. For information on how to use sequence generators, see <X-ref>"Using Automatic Sequence Generators" on page -1.

tos

If `enableTos` is `true`, this option specifies the IP Precedence / TOS (Type of Serbit setting and Assured Forwarding classes. (Default="Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The choices are:

"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control

userName

Enter the default user name that the client will use to login to the FTP server. When you use a LOGIN action in the action list, this user name will be used by default. Ixia servers currently only accept a user name of 'root.' (Default = "root").

You can insert sequence generators into this field to create unique entries autoFor information on how to use sequence generators, see <X-ref>"Using Automatic Sequence Generators" on page -1.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
```

```
#-----  
# Activity FTPClient1 of NetTraffic Traffic1@Network1  
#-----  
set Activity_FTPClient1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "FTP Client" ]  
  
#-----  
# Timeline1 for activities FTPClient1  
#-----  
set Timeline1 [::IxLoad new ixTimeline]  
$Timeline1 config \  
    -rampUpValue                    1 \  
    -rampUpType                      0 \  
    -offlineTime                    0 \  
    -rampDownTime                   20 \  
    -standbyTime                    0 \  
    -iterations                     1 \  
    -rampUpInterval                 1 \  
    -sustainTime                    20 \  
    -timelineType                   0 \  
    -name                           "Timeline1"  
  
$Activity_FTPClient1 config \  
    -enable                         true \  
    -name                           "FTPClient1" \  
    -enableConstraint               false \  
    -userObjectiveValue             100 \  

```



```
-constraintValue                100 \  
-userObjectiveType             "simulatedUsers" \  
-timeline                       $Timeline1  
  
$Activity_FTPClient1 agent.config \  
-userName                       "root" \  
-enableTos                      false \  
-loopValue                      true \  
-enable                         true \  
-ipPreference                   2 \  
-name                           "FTPClient1" \  
-vlanPriority                    0 \  
-tos                             0 \  
-fileList                       "'/#1', '/#4', '/#16', '/#64',  
'/#256', '/#1024', '/#4096', '/#16384', '/#65536', '/#262144', '/#1048576'" \  
-enableEsm                      false \  
-mode                           "ACTIVE" \  
-esm                            1460 \  
-password                       "noreply@ixiacom.com" \  
-enableVlanPriority              false  
  
$Activity_FTPClient1 agent.actionList.clear  
set my_ixFtpAction [::IxLoad new ixFtpAction]  
$my_ixFtpAction config \  
-userName                       "root" \  
-destination                    "Traffic2_FTPServer1:21" \  
-sessionId                      "1" \  
-command                        "{Get}" \  
-arguments                      "/#4096" \  
-password                       "noreply@ixiacom.com"
```

```
$Activity_FTPClient1 agent.actionList.appendItem -object $my_ixFtpAction
```

SEE ALSO

FTP Client Action, ixNetTraffic



FTP Client Action

FTP Client Action - define the commands that the FTP client will execute

SYNOPSIS

```
set clientTraffic [::IxLoad new ixClientTraffic options]
$clientTraffic agentList.appendItem options...
$clientTraffic agentList(0).actionList.appendItem options...

set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_FTPClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_FTPClient1 agent.actionList.appendItem -object $my_ixFtpAction
```

DESCRIPTION

An FTP client action is added to the `actionList` option of the FTP Client Agent `activityList` object using its `appendItem`. See the following example:

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition the following commands are available. Unless otherwise described, no value is returned and an exception is raised for any error found.

checkConfig

This subcommand checks the configuration of an individual action.

OPTIONS

arguments

This option contains an argument that is used by the various commands defined in the `command` option. The type of the value depends on the command:

Command option	Usage
"CD"	The path to switch to.
"{Get}"	The path to where the file is stored.
"LOGIN"	N/A.
"{Put}"	The path/size of the file to be sent to the server.
"QUIT"	N/A.
"RETRIEVE"	The path to where the file is stored.
"STORE"	The path/size of the file to be sent to the server.

Command option	Usage
"{Think}"	The number of milliseconds to pause before exethe next command in the action list.

command

Selects the FTP command to be used. One of:

Option	Usage
"CD"	Changes the current working directory to the value in the <code>arguments</code> option.
"{Get}"	(Default). Retrieves the file specified in the <code>argumen</code> option. {Get} is not a standard FTP command; it allows you to retrieve a file from an Ixia server without having to log in.
"LOGIN"	Logs in to the FTP server using the name and password in the <code>us</code> and <code>password</code> options.
"{Put}"	Copies the file specified in the <code>arguments</code> option from the client to the server. {Put} is not a standard FTP command; it allows you to store a file on an Ixia server without having to log in.
"QUIT"	Logs out of the FTP server.
"RETRIEVE"	Retrieves the file specified in the <code>arguments</code> option.
"STORE"	Copies the file specified in the <code>arguments</code> option from the client to the server.
"{Think}"	Adds a pause (think time) before the next command is executed. Specthe duration of the pause in the <code>arguments</code> option.
"{LoopBegin}"	An IxLoad command that you can add to the Command List to cause the commands between it and the {Loop End} to be executed a specinumber of times.
"{LoopEnd}"	Ends the list of commands that will be executed by the preceding {Loop Begin} command.

destination

Either the IP address of a real FTP server or the value of the `-name` option of an FTP Server Agent. If the FTP server listens on a port other than the standard (21), enter a colon after the IP address and then enter the port number. When using an FTP Server Agent, the port number must agree with that defined by the Server Agent. See the following example:

```
192.168.0.1:21
```

The `destination` option also accepts IPv6 addresses. IxLoad supports all forms of IPv6 addressing except `::dotted-quad` notation (for example, `:::1.2.3.4`).

password

The password for the user name. Ixia servers accept any password.



userName

The user name that the client will use to log in to the FTP server. Ixia servers only accept a user name of root.

EXAMPLE

```
set my_ixFtpAction [::IxLoad new ixFtpAction]

$my_ixFtpAction config \

    -userName                "root" \
    -destination              "Traffic2_FTPServer1:21" \
    -sessionId                "1" \
    -command                  "{Get}" \
    -arguments                "/#4096" \
    -password                 "noreply@ixiacom.com"

$Activity_FTPClient1 agent.actionList.appendItem -object $my_ixFtpAction
```

SEE ALSO

FTP Client Agent

realFileList

realFileList - Configure a list of files to for {PUT}/STORE commands.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_FTPServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_FTPServer1 agent.config options...
```

DESCRIPTION

A list of type `RealFileObject`. `RealFileObject` has two options – ‘payloadFile’ and ‘name’. This name can be referred to in the arguments of a {PUT}/STORE actions

This is applicable for both client and server agents.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

payloadFile

Name of the file to be uploaded.

name

This represents the actual path of the `payloadFile`.

EXAMPLE

```
$Activity_FTPClient1 agent.config \  
-userName                "root" \  
-enableTos                false \  
-loopValue                true \  
-enable                   true \  
-ipPreference             2 \  
-name                     "FTPClient1" \  
-vlanPriority              0 \  
-tos                       0
```



```
-fileList                "'/#1', '/#4', '/#16', '/#64',  
'/#256', '/#1024', '/#4096', '/#16384', '/#65536', '/#262144', '/#1048576'" \  
  
-enableEsm                false \  
  
-mode                    "ACTIVE" \  
  
-esm                    1460 \  
  
-password                "noreply@ixiacom.com" \  
  
-enableVlanPriority      false
```

```
$Activity_FTPClient1 agent.actionList.clear
```

```
$Activity_FTPClient1 agent.realFileList.clear
```

```
set my_RealFileObject [::IxLoad new RealFileObject]
```

```
$my_RealFileObject config \  

```

```
-payloadFile              "<Dummy File>" \  

```

```
-page                    "/#1"
```

```
$Activity_FTPClient1 agent.realFileList.appendItem -object $my_RealFileObject
```

```
set my_RealFileObject1 [::IxLoad new RealFileObject]
```

```
$my_RealFileObject1 config \  

```

```
-payloadFile              "<Dummy File>" \  

```

```
-page                    "/#4"
```

```
$Activity_FTPClient1 agent.realFileList.appendItem -object  
$my_RealFileObject1
```

```
set my_RealFileObject2 [::IxLoad new RealFileObject]
```

```
$my_RealFileObject2 config \  

```

```
-payloadFile              "<Dummy File>" \  

```

```
-page                                "/#16"

$Activity_FTPClient1 agent.realFileList.appendItem -object
$my_RealFileObject2

set my_RealFileObject3 [::IxLoad new RealFileObject]
$my_RealFileObject3 config \
    -payloadFile                      "<Dummy File>" \
    -page                              "/#64"

$Activity_FTPClient1 agent.realFileList.appendItem -object
$my_RealFileObject3

set my_RealFileObject4 [::IxLoad new RealFileObject]
$my_RealFileObject4 config \
    -payloadFile                      "<Dummy File>" \
    -page                              "/#256"

$Activity_FTPClient1 agent.realFileList.appendItem -object
$my_RealFileObject4

set my_RealFileObject5 [::IxLoad new RealFileObject]
$my_RealFileObject5 config \
    -payloadFile                      "<Dummy File>" \
    -page                              "/#1024"

$Activity_FTPClient1 agent.realFileList.appendItem -object
$my_RealFileObject5

set my_RealFileObject6 [::IxLoad new RealFileObject]
```



```
$my_RealFileObject6 config \  
    -payloadFile                "<Dummy File>" \  
    -page                       "/#4096"  
  
$Activity_FTPClient1 agent.realFileList.appendItem -object  
$my_RealFileObject6  
  
set my_RealFileObject7 [::IxLoad new RealFileObject]  
  
$my_RealFileObject7 config \  
    -payloadFile                "<Dummy File>" \  
    -page                       "/#16384"  
  
$Activity_FTPClient1 agent.realFileList.appendItem -object  
$my_RealFileObject7  
  
set my_RealFileObject8 [::IxLoad new RealFileObject]  
  
$my_RealFileObject8 config \  
    -payloadFile                "<Dummy File>" \  
    -page                       "/#65536"  
  
$Activity_FTPClient1 agent.realFileList.appendItem -object  
$my_RealFileObject8  
  
set my_RealFileObject9 [::IxLoad new RealFileObject]  
  
$my_RealFileObject9 config \  
    -payloadFile                "<Dummy File>" \  
    -page                       "/#262144"  
  
$Activity_FTPClient1 agent.realFileList.appendItem -object  
$my_RealFileObject9
```

```
set my_RealFileObject10 [::IxLoad new RealFileObject]

$my_RealFileObject10 config \

    -payloadFile                "<Dummy File>" \
    -page                        "/#1048576"

$Activity_FTPClient1 agent.realFileList.appendItem -object
$my_RealFileObject10

set my_RealFileObject11 [::IxLoad new RealFileObject]

$my_RealFileObject11 config \

    -payloadFile
"E:\\perforce\\ixweb\\3.30\\build_cs.log" \
    -page                        "/C_file1"

$Activity_FTPClient1 agent.realFileList.appendItem -object
$my_RealFileObject11
```

SEE ALSO

FTP Client Agent, FTP Server Agent



FTP Server Agent

FTP Server Agent - configure an FTP server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_FTPServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_FTPServer1 agent.config options...
```

DESCRIPTION

An FTP server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this server agent. (Default = true).

`enableEsm`

If true, the use of the `esm` option is enabled. (Default = false).

`enableTos`

Enables the setting of the TOS (Type of Service) bits in the header of the FTP packets. Use the `tos` option to specify the TOS bit setting.

0 (default) TOS bits not enabled.

1 TOS bits enabled.

`esm`

If `enableEsm` is true, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size is 1,460 bytes. (Default = 1,460).

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = `false`).

`vlanPriority`

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.

`ftpPort`

The port number to which the FTP server will respond. To specify multiple lisports, enter the port numbers, separated by commas (,). You can specify up to 50 listening ports. (Default = `21`).

`name`

The name associated with this object, which must be set at object creation.

`tos`

If `enableTos` is `true`, this option specifies the IP Precedence / TOS (Type of Serbit setting and Assured Forwarding classes. (Default="Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
```



```
#-----  
# Activity FTPServer1 of NetTraffic Traffic2@Network2  
#-----  
set Activity_FTPServer1 [$Traffic2_Network2 activityList.appendItem \  
    -protocolAndType                "FTP Server" ]  
set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]  
$Activity_FTPServer1 config \  
    -enable                          1 \  
    -name                            "FTPServer1" \  
    -timeline                        $_Match_Longest_  
$Activity_FTPServer1 agent.config \  
    -enableTos                       0 \  
    -enable                          1 \  
    -name                            "FTPServer1" \  
    -vlanPriority                     0 \  
    -tos                             0 \  
    -ftpPort                         21 \  
    -enableEsm                       0 \  
    -esm                             1460 \  
    -enableVlanPriority               0  
$Activity_FTPServer1 agent.realFileList.clear  
set my_RealFileObject11 [::IxLoad new RealFileObject]  
$my_RealFileObject11 config \  
    -payloadFile                     "<Dummy File>" \  
    -page                            "/#1"  
$Activity_FTPServer1 agent.realFileList.appendItem -object  
$my_RealFileObject11
```

SEE ALSO

ixServerTraffic

FTP Statistics

For the FTP statistics, see the following:

FTP Client Statistics (on page 1088)

FTP Server Statistics (on page 1090)



FTP Client Statistics

The table below lists the statistics that IxLoad reports for FTP clients. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

For TCP statistics, see Appendix D, TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Conditional Views	Description
FTP Bytes	--	Total number of FTP bytes sent.
FTP Control Bytes Received	All	Number of bytes received on the control connections by the FTP client.
FTP Control Bytes Sent	All	Number of bytes transmitted on the control connections by the FTP client.
FTP Control Conn Requested	All	Number of requests to establish control connections sent by the clients.
FTP Control Conn Established	All	Number of control connections established by the clients.
FTP Control Conn Failed	All	Number of control connections that could not be established.
FTP Control Conn Failed (Rejected)	All	Number of control connections that could not be established because the server rejected the connection request.
FTP Control Conn Failed (Other)	All	Number of control connections that could not be established for reasons other than rejection by the server.
FTP Control Conn Active	All	Number of control connections actively transferring FTP commands.
FTP Data Conn Established	All	Number of data connections established.
FTP Data Conn Established (Active Mode)	All	Number of data connections established in Active mode.
FTP Data Conn Requested (Passive Mode)	All	Number of data connections requested in Passive mode.

FTP Data Conn Established (Passive Mode)	All	Number of data connections established in passive mode.
FTP Data Conn Failed (Passive Mode)	All	Number of data connections that failed.
FTP Data Conn Active	All	Number of data connections active.
FTP File Uploads Requested	All	Number of requests to upload files sent by the clients.
FTP File Uploads Successful	All	Number of uploads that completed successfully.
FTP File Uploads Failed	All	Number of upload attempts that failed.
FTP File Downloads Requested	All	Number of requests to download files sent by the clients.
FTP File Downloads Successful	All	Number of downloads that completed successfully.
FTP File Downloads Failed	All	Number of download attempts that failed.
FTP Data Bytes Sent	All	Number of bytes transmitted on the data connections by the FTP client
FTP Data Bytes Received	All	Number of bytes received on the data connections by the FTP client.
FTP Control Bytes Sent	All	Number of bytes received on the control connections by the FTP client.
FTP Control Bytes Received	All	Number of bytes received on the control connections by the FTP client.
FTP Simulated Users	--	Number of users to be simulated during the test.
FTP Connections	--	Number of FTP connections between clients and servers, including both control and data connections.
FTP Transactions	--	Number of transactions completed by the clients.
FTP Bytes	All	Total number of FTP bytes sent.
FTP Throughput	--	Rate, in bytes per second, at which the client sent and received FTP data.
FTP Throughput (Kbps)	All	Rate, in kilobits per second, at which the client sent and received FTP data. This statistic is only available in Conditional View.



FTP Connection Rate	All	Rate at which the client established FTP connections.
FTP Transaction Rate	All	Average rate at which the clients completed FTP transactions.
FTP Concurrent Sessions	All	Number of simultaneous FTP sessions active.
FTP Control Connection Latency (ms)	All	Average amount of latency on control connections, in milliseconds. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
FTP Data Connection Latency (Passive Mode) (ms)	All	Average amount of latency (in milliseconds) on data connections that were established in Passive mode. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.

FTP Server Statistics

The table below lists the statistics that IxLoad reports for FTP servers. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

The Conditional Views column indicates which views a statistic is available in:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test
configuration>_<yyy><mm><dd>_<hh><mm><ss>
```

For TCP statistics, see Appendix D, TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Conditional Views	Description
FTP Control Conn Received	IP, VLAN	Number of requests to establish control connections received by the servers.

FTP Control Conn Established	IP, VLAN	Number of control connections established by the servers.
FTP Control Conn Rejected	IP, VLAN	Number of requests to establish control connections rejected by the servers.
FTP Control Conn Active	IP, VLAN	Number of control connections actively transferring FTP commands.
FTP Data Conn Established	IP, VLAN	Number of data connections established by the server (active and passive mode).
FTP Data Conn Requested (Active Mode)	IP, VLAN	Number of requests to establish data connections in active mode received by the servers.
FTP Data Conn Established (Active Mode)	IP, VLAN	Number of data connections established in active mode.
FTP Data Conn Failed (Active Mode)	IP, VLAN	Number of data connections opened in active mode that failed.
FTP Data Conn Established (Passive Mode)	IP, VLAN	Number of data connections established in passive mode.
FTP Data Conn Active	IP, VLAN	Number of data connections actively uploading or downloading data.
FTP File Uploads Requested	IP, VLAN	Number of requests to upload data received by the servers.
FTP File Uploads Successful	IP, VLAN	Number of uploads that completed successfully.
FTP File Uploads Failed	IP, VLAN	Number of uploads that failed.
FTP File Downloads Requested	IP, VLAN	Number of requests to download files received by the servers.
FTP File Downloads Successful	IP, VLAN	Number of downloads that completed successfully.
FTP File Downloads Failed	IP, VLAN	Number of downloads that failed.
FTP Data Bytes Sent	--	Number of bytes sent by the servers on data connections.
FTP Data Bytes Received	--	Number of bytes received by the servers on data connections.
FTP Control Bytes Sent	--	Number of bytes sent by the servers on control connections.
FTP Control Bytes Received	--	Number of bytes received by the servers on control connections.
FTP Data Connection Latency (Active Mode) (ms)	IP, VLAN	<p>Average amount of latency (in milliseconds) on data connections opened in active mode.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>

15

HTTP

This section describes the HTTP Tcl API objects.

In this chapter:

Overview	1094
HTTP Client Agent	1100
HTTP Server Agent	1118
Using Your Own Web Pages In IxLoad	1146
Using Sequence Generators in HTTP Client Commands and Server	
Header Name=Value Fields	1148
Statistics	1151



Overview

HTTP protocol commands are organized as:

- HTTP Client Agent
 - HTTP Client Action
- HTTP Server Agent
 - ixCookieContents
 - ixResponseHeader
 - ixWebPageObject
 - CustomPayloadObject

Additional topics included are:

- Supported Ciphers— describes the set of supported encryption ciphers.
- Using Your Own Web Pages In IxLoad describes how to use your own Web pages in the server's emulation.
- Using Sequence Generators in HTTP Client Commands and Server Header Name=Value Fields— describes how to use variables to generate large numbers of difobjects.

Objectives

The objectives (userObjective) you can set for HTTP are listed below. Test objectives are set in the ixTimeline object.

- connectionRate
- connectionAttemptRate
- transactionRate
- simulatedUsers
- concurrentConnections
- throughputMBps
- throughputKBps

HTTP Client Agent

The HTTP Client Agent defines a simulated user performing HTTP requests against one or more HTTP servers. Refer to `HTTP Client Agent` for a full description of this command. The important subcommands and options of this command are:

HTTP Client Agent Subcommands

Subcommand	Usage
importCertificate	Reads and decodes a certificate from a file.
importPrivateKey	Reads and decodes a private key from a file.

HTTP Client Agent Options

Option	Usage
enable	Enables the use of the HTTP client agent.
name	The name associated with the client agent.
actionList	The list of actions associated with the agent.
httpVersion	The version of HTTP emulated by the client.
browserEmulation	The type of browser (IE versus Netscape) that will be emulated.
maxSessions	The maximum number of sessions that a client can have open at one time.
keepAlive maxPersistentRequests	Controls persistent connection operation.
followHttpRedirects	Allows the emulation to follow redirects from the server.
enableDecompressSupport	Decompress pages.
enableIntegrityCheckSupport	Calculates and compare checksums for received pages.
enableHttpProxy httpProxy enableHttpsProxy httpsProxy	Controls HTTP and HTTPS proxy operation.
enableCookieSupport cookieJarSize cookieRejectProbability	Controls cookie support and operation.
enableSsl sslVersion	Enables the use of SSL sessions.
clientCiphers	The encryption ciphers that the browser emulation will offer during an SSL transaction.
sequentialSessionReuse	Indicates how many times the same set of keys may be used.
certificate privateKey privateKeyPassword	The data for the user's certificate and private key, along with the password used to decode both.
advancedoptions	Required for large headers. It has two options: <code>enableLargeHeader</code> and <code>maxHeaderLen</code> .



HTTP Client Action

Each client action is a single step in the interaction. Refer to `HTTP Client Action` for a full description of this command. The important subcommands and options of this command are:

HTTP Client Action Subcommands

Subcommand	Usage
<code>checkConfig</code>	Checks the configuration of the action.

HTTP Client Action Options

Option	Usage
<code>command arguments</code>	The HTTP command, with optional arguments, to be executed.
<code>destination</code>	The name/address of the HTTP server.
<code>namevalueargs</code>	Name/value pair arguments for GET, HEAD, PUT, and POST commands.
<code>pageObject</code>	The page at the destination to be retrieved.

HTTP Server Agent

The HTTP Server Agent defines the operation of the HTTP server. Refer to `HTTP Server Agent` for a full description of this command. The important subcommand options of this command are:

HTTP Server Agent Sub commands

Subcommand	Usage
<code>importCertificate</code>	Reads and decodes a certificate from a file.
<code>importPrivateKey</code>	Reads and decodes a private key from a file.

HTTP Server Agent options

Option	Usage
<code>enable</code>	Enables the use of the HTTP server agent.
<code>name</code>	The name associated with the server agent.
<code>acceptSslConnections</code>	Specifies if SSL connections are accepted or not.
<code>httpPort</code> <code>httpsPort</code>	The port numbers to respond on for normal and SSL sessions, respectively.
<code>webPageList</code>	The Web pages that the server will respond to—a list of objects of type <code>ixWebPageObject</code> .

Option	Usage
responseHeaderList	The response headers that will be associated with returned Web pages—a list of objects of type <code>ixResponseHeader</code> .
cookieList	The cookies that the server will supply—a list of objects of type <code>ixCookieContents</code> .
customPayloadList	The custom payloads associated with returned Web pages—a list of objects of type <code>CustomPayloadObject</code> .
minResponseDelay maxResponseDelay	Minimum and maximum length of time, in milliseconds, that the HTTP server delays sending a response.
docrootfile	Selects the file (zip or tar) that defines default directory path for actual files stored on the HTTP server.
enableEsm	If true, the use of the esm option is enabled. (Default = false).
esm	If <code>enableEsm</code> is true, this option specifies the TCP Maximum Segment Size in the MSS (RX) field.
enableMD5Checksum	Causes server to calculate checksums for pages sent to client.
enableVlanPriority, vlanPriority	This parameter sets the VLAN priority for the activity. (Default = false). When <code>enableVlanPriority</code> is true, this option accepts the <code>vlan</code> priority value.
enable tos, tos	If <code>enableTos</code> is true, the <code>tos</code> option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "0").
integrityCheckOption	Type of checksum sent for pages from docroot file.
tcpCloseOption	This option helps the server to close connections. It accepts integer value. (Default = 0).
hrequestTimeout	The amount of time that the server will wait for input on an open connection before closing the session with a '408' error.
certificate privateKey privateKeyPassword	The data used for creating the server certificate and private key, along with the password used to decode password-protected values.
ServerCipher	The server encryption ciphers offered during an SSL transaction.
enableDHsupport	Enables Diffie-Hellman support for DH keys and ADH or EDH ciphers.



Option	Usage
dhParams	If the EnableDH support option is selected, this parameter specifies the file that contains a DSA key and certificate that is converted to a DH key.
enablePerServerPerURLstat	If true, the statistics from a returned page records the server IP address from where the page is sent.

ixWebPageObject

The `ixWebPageObject` is a specification of a Web page reference that the server will respond to. Refer to `ixWebPageObject` for a full description of this command. The important options of this command are:

ixWebPageObject Options

Option	Usage
MD5Option	Type of checksum sent for page.
page	A pattern of URL references that will be satisfied by this Web page.
payloadType payloadSize payloadFile	Indicates the type and size of data to be returned.
response	A reference to a response header item, of type <code>ixResponseHeader</code> , to be used as part of this response.
cookie	A reference to a cookie item, of type <code>ixCookieContents</code> , to be used as part of this response.

customPayloadObject

The `CustomPayloadObject` is used to configure a custom payload object. This object is declared in the `payloadFile` option of `ixWebPageObject`. Refer to `CustomPayloadObject` for a full description of this command.

ixResponseHeader

The `ixResponseHeader` object is a single header item that is associated with a Web page returned by the server. Refer to `ixResponseHeader` for a full description of this command. The important options of this command are:

ixResponseHeader Options

Option	Usage
name	The name in the <code>name==value</code> response.

Option	Usage
value	The value in the <code>name==value</code> response.
domain path	Specifies the range of URLs that will trigger this cookie to be returned to the client.

ixCookieContents

The `ixCookieContents` object is a single cookie that will be used in a Web page object. Refer to `ixCookieContent` for a full description of this command. The important options of this command are:

ixCookieContents Options

Option	Usage
name	The name of the cookie returned.
value	The value that is returned with the cookie.
domain path	Specifies the range of URLs that will trigger this cookie to be returned to the client.
maxAge	The maximum valid age for the cookie.

HTTP Client Agent

HTTP Client Agent

SYNOPSIS

```
set HTTP_client_client_network [::IxLoad new ixNetTraffic]
set Activity_newAgent1 [$HTTP_client_client_network activityList.appendItem
option...]
$Activity_newAgent1 agent.config \
```

DESCRIPTION

An HTTP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Two subcommands are available to load certificates and private keys: `importCertificate` and `importPrivateKey`.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition, the following commands are available. Unless otherwise described, no values are returned and an exception is raised for any error found.

```
importCertificate file
```

Imports a certificate from a disk `file`, setting the `certificate` option with the result. `True` is returned if the import succeeded and `false` otherwise. `IxLoad` can import ASCII PEM (Privacy Enhanced Mail) or binary (PKCS#12) certificates and keys; it converts binary certificates and keys into ASCII PEM format.

- PEM uses Base64 encoding, and is optimized for sending binary data in 7-bit transport environments like the Internet.
- PKCS #12 (Public Key Cryptography Standard #12) is an industry standard format used to transfer certificates and their corresponding private keys from one computer to another, or from a computer to removable media. If this format is imported, the `privateKeyPassword` must be set.



Note: Even though the certificate and key are stored in the same file, you must import each one separately.

Example

```
$Activity_newClientActivity1 agent.importCertificate "C:/Program
Files/Ixia/IxLoad/3.40.49.32-
EB/Client/Plugins/agent/HTTP_Common/SSL_Certificates/Unsecured_RSA_cert_512.p
em"
```

`importPrivateKey file`

This subcommand performs the same function, but for the private key. The decoded and decrypted values are set into the `password` option and `true` or `false` are returned to indicate success and failure, respectively.

```
$Activity_newClientActivity1 agent.importPrivateKey "C:/Program
```

```
Files/Ixia/IxLoad/3.40.49.32-
EB/Client/Plugins/agent/HTTP_Common/SSL_Certificates/Unsecured_RSA_key_512.pe
m"
```

OPTIONS

`actionList`

A list of actions that the agent should perform, of type `HTTP Client Action`. Actions are normally added using the `appendItem` subcommand.

`browserEmulation`

The type of browser that the client will emulate. One of:

Option	Usage
<code>::HTTP_Client(kBrowserTypeNone)</code> or " <code><Custom></code> "	No browser is emulated by the client. The headers may be entered in the <code>headerList</code> option.
<code>::HTTP_Client(kBrowserTypeIE5)</code> " <code>Microsoft IE 5.x</code> "	Microsoft Internet Explorer 5.x browser is emulated by the client.
<code>::HTTP_Client(kBrowserTypeMozilla)</code> " <code>Mozilla</code> "	Netscape, Mozilla, and Firefox browsers are emulated by the client.
<code>::HTTP_Client(kBrowserTypeIE6)</code> " <code>Microsoft IE 6.x</code> "	(Default) Microsoft Internet Explorer 6.x browser is emulated by the client.
<code>::HTTP_Client(kBrowserTypeFirefox)</code> " <code>Firefox</code> "	The Firefox browser is emulated by the client.
<code>::HTTP_Client(kBrowserTypeSafari)</code> " <code>Safari</code> "	The Safari browser is emulated by the client.

`certificate`

If `enableSsl` is `true`, this is a certificate to be used by the client if requested by the server. The certificate must be an X.509 certificate in binary format, fully decoded. The `importCertificate` subcommand can read and decode a certificate held in a disk file. (Default = "").

`clientCiphers`



If `enableSsl` is true, this is a ':' separated list of encryption ciphers that will be supported by the client. See Supported Ciphers for a list of supported ciphers. (Default = "DEFAULT").

`cookieJarSize`

If `enableCookieSupport` is true, this option indicates the number of cookies that will be saved for each client. The maximum value of this is 300. (Default = 10).

`cookieRejectProbability`

If `enableCookieSupport` is true, then this option indicates the probability, from 0 to 1, that a client will reject a request for a cookie's contents from the server. (Default = 0.0).

`enable`

Enables the use of this action. (Default = true).

`enableCookieSupport`

If true, then the client will support cookie retention, as indicated in the `cookieJarSize` and `cookieRejectProbability`. (Default = false).

`piggybackAck`

If true, the client includes the ACK for the previous packet in the same packet as the next packet.. (Default = true).

`enableDecompressSupport`

If true, the client decodes pages that have been encoded using a supported encoding method such as gzip or deflate. (Default = false).

`enableEsm`

If true, the use of the `esm` option is enabled. (Default = false).

`enableHttpProxy`

If true, the client will retrieve pages from an HTTP proxy device defined in `httpProxy` instead of the target specified in the URL. (Default = false).

`enableHttpsProxy`

If true, the client will retrieve secure (SSL) pages from an HTTPS proxy device defined in `httpsProxy` instead of the target specified in the URL. (Default = false).

`enableIntegrityCheckSupport`

If `true`, the client calculates a checksum for a received page and compares it with the checksum received from the server. (Default = `false`).

`enableLargeHeader`

If enabled, this specifies whether IxLoad will support large headers. It accepts boolean value of `True` or `False`. (Default = `false`).

`enableSsl`

If `true`, then the client will support SSL interactions. The operation of SSL mode is controlled by the `certificate`, `clientCiphers`, `privateKey`, `privateKeyPassword`, `sequentialSessionReuse` (`sic`), and `sslVersion` options. (Default = `0`).

`enableTos`

Enables the setting of the TOS (Type of Service) bits in the header of the HTTP packets. Use the `tos` option to specify the TOS bit setting.

0 (default) TOS bits not enabled.

1 TOS bits enabled.

`esm`

If `enableEsm` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size is 1,460 bytes. (Default = 1,460).

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = `false`).

`vlanPriority`

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.

`followHttpRedirects`

If `true`, the client follows HTTP Redirect commands from the server. An HTTP Redirect is a response status code from the server in the range 300-399 that defines the reason for redirection (for example, "301 Moved Permanently") and supplies an alternative location (specified in the Location HTTP header) from which the client can retrieve the page. (Default = `false`).



headerList

If `browserEmulation` is set to "None," then this list of headers will be transmitted as part of a client request. This list is of type `ixResponseHeader`; items are added to the list via the `appendItem` subcommand. Each element of the list must be of the form "key: value" without any spaces in the key. (Default = None).

httpProxy

If `enableHttpProxy` is true, this option is the name of a HTTP proxy device (typically, a caching device) that will be used instead of the target specified in the URL. It should be of the form: <IP address>:<port>; for example, 192.168.3.1:8080. (Default = "").

httpsProxy

If `enableHttpsProxy` is true, this option is the name of a HTTPS proxy device (typically, a caching device) that will be used instead of the target specified in the URL for secure (SSL) pages. It should be of the form: <IP address>:<port>; for example, 192.168.3.1:8080. (Default = "").

httpVersion

Select the version of the HTTP protocol that you want to use in the test. One of:

Option	Usage
"1.0"	(Default) Under HTTP 1.0 without Keep-Alive, when a user clicks on a link for a Web page, a TCP connection request is sent by the client to the server. When the server accepts the connection, the client sends an HTTP GET request to download the Web page from the server. The client acknowledges receipt of the page by sending an ACK to the server. After making a single HTTP request, the client closes the TCP connection. After the server has sent the entire page, it will also close the connection from its side. See the description of <code>keepAlive</code> for a description of its effect on HTTP 1.0
"1.1"	Most browsers use HTTP 1.1. If a client and server use HTTP 1.1, multiple HTTP requests can be sent by the client on a single TCP connection. This saves processing power, since fewer TCP connections need to be established. HTTP 1.1 also allows for persistent connections, enabling connections to stay up for (relatively) long periods of time. In HTTP 1.1, the server initiates the closing of the TCP connection by sending a FIN message.

ipPreference

If a mixture of IPv4 and IPv6 addresses are available on the client network, this parameter configures which address types the agent uses.

- 0 IPv4
- 1 IPv6
- 2 (default) Both, IPv4 first
- 3 Both, IPv6 first

keepAlive

This option is only applicable if `httpVersion` is set to "1.0."

If this option is set to true, the client adds the Connection: Keep-Alive header to its request. Each request from a client creates a new socket connection to the server. The client reads from that socket connection to get the response. If keepAlive is not set, the server closes the connection. If the client needs to make a new request, it will establish a new connection.

If the client sends the Keep-Alive header, the server keeps the connection open. When the client sends another request, it uses the same connection. This will continue until either the client or the server decides that the session is over, and one of them closes the connection. (Default = false).

loopValue

If this option is enabled (1), then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (0) then the client will progress through the command list only once, and then go idle. (Default = 0).

maxHeaderLen

Specifies the length of header data. It accepts integer values. Minimum = 1,024, maximum = 1,0240. (Default = 1,024).

maxPersistentRequests

This option is only applicable if `httpVersion` is set to "1.1" or `httpVersion` is set to "1.0" with `keepAlive` set to true. This option enables you to control the number of transactions that can occur during a single connection.

A value of 0 indicates the maximum possible, in which case IxLoad will create as many transactions as possible for each connection.

If you enter a value to limit the number of transactions, IxLoad limits the number of transactions that can occur during a single TCP connection. If a user reaches the maximum number of transactions and needs to continue communicating with the server, it will close the connection and open a new one. (Default = 1).

maxPipeline

This option enables you to control the maximum number of requests that the client will send before waiting for a response. Minimum = 1, maximum = 1,000. (Default = 1).

HTTP pipelining allows a client to send multiple HTTP requests before it has received a response to the first request. A client that does not use HTTP pipelining waits for a response to a request before it sends the next request.

This option is only applicable if `httpVersion` is set to "1.1" or `httpVersion` is set to "1.0" with `keepAlive` set to true.

Setting `maxPipeline` to 1 (the default) effectively disables pipelining; the client will send only one request before stopping to wait for a response.



Setting `maxPipeline` to a value greater than 1 reduces the maximum number of concurrent connections that a test can attain.

If pipelining is enabled, `IxLoad` pipelines all requests: GET, PUT, POST, HEAD, and DELETE.

Note: If you enable pipelining, you should also consider the value you will enter in the `maxPersistentRequests` field, because it may override the value for `maxPipeline`. For example, if you set the value of `maxPersistentRequests` to '5' instead of "Maximum possible" and set the `maxPipeline` value to 100, pipelining will effectively be nullified because the client will allow only 5 requests to be sent by over an HTTP connection.

`maxSessions`

This value determines the maximum number of connections that a single user can have open at any given time. For example, clients may open multiple connections when their command list contains URLs for multiple servers.

The value for this parameter has an effect on the total number of users that can be configured; increasing the number of concurrent connections decreases the number of users that can be configured. Setting this parameter to 1 allows the maximum numbers of users to be created.

`IxLoad` enforces these limits for clients. For servers, the limits are the same but not enforced. (Default = 3).

`name`

The name associated with this object, which must be set at object creation.

`privateKey`

If `enableSsl` is `true`, this is a user's private key. The password must be in binary format, fully decoded. The `importPrivateKey` subcommand can read and decode a certificate held in a disk file. (Default = "").

`privateKeyPassword`

The password used to decode a certificate and private key, when using the `importCertificate` or `importPrivateKey` subcommands. (Default = 0).

`sequentialSession`

Reuse

If `enableSsl` is `true`, this option indicates the number of times that a set of keys will be reused after its initial usage. For example, if this value is set to 3, then the keys will be used for four total sessions. (Default = 0).

`sslVersion`

If `enableSsl` is `true`, this is the SSL version be supported by the client. One of

Option	Usage
<code>\$.:HTTP_Client(kSslVersion2)</code>	SSL version 2.0.
<code>\$.:HTTP_Client(kSslVersion3)</code>	SSL version 3.0.
<code>\$.:HTTP_Client(kTlsVersion1)</code>	(Default) TLS version 1.0.

`tcpCloseOption`

This option selects the method used to close connections.

- 0 (Default) Connections are closed using three way handshake.
- 1 Connections are closed by sending Reset (RST) segments instead of Finish (FIN) segments
- 2 Connections are closed using a four-way handshake

`tos`

If `enableTos` is `true`, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

- "Best Effort (0x0)" (Default) routine priority
- "Class 1 (0x20)" Priority service, Assured Forwarding class 1
- "Class 2 (0x40)" Immediate service, Assured Forwarding class 2
- "Class 3 (0x60)" Flash, Assured Forwarding class 3
- "Class 4 (0x80)" Flash-override, Assured Forwarding class 4
- "Express Forwarding (0xA0)" Critical-ecp
- "Control (0xC0)" Internet-control

`urlStatsCount`

Number of URL statistics to display in Statistics (StatViewer) window. During a test, IxLoad displays statistics for one or more URLs in the Statistics window at the bottom of the main IxLoad window. You can use this field to restrict the number of per-URL statistics that are displayed, so that the window displays statistics only for the URLs that are most important to you. If you select a large number of URLs, the Statistics window can become difficult to read. Maximum = 1,000, (default=10).

Value Extraction Settings

`varExtract_enable`

If enabled, IxLoad searches for a match for the string configured in the fields and, if found, applies the value to the variable. It accepts `true` or `false` value.



`varExtract_varName`

This represents the name of the variable.

`varExtract_prefix`

This indicates the characters preceding the value string in the response.

You can specify up to 512 characters, which can be any valid printable ASCII characters.

`varExtract_suffix`

This indicates the characters following the value string in the response.

You can specify up to 512 characters, which can be any valid printable ASCII characters.

`varExtract_location`

This indicates where to search for the value string. It can take three different values: `Header`, `Body`, or `Both`.

STATISTICS

EXAMPLE

```
$Activity_HTTPClient1 agent.config \  
-vlanPriority 0 \  
-enableDecompressSupport 0 \  
-enableHttpsProxy 0 \  
-enableSsl 0 \  
-enableUnidirectionalClose 0 \  
-ipPreference 2 \  
-loopValue 1 \  
-enableLargeHeader 0 \  
-maxPersistentRequests 1 \  
-enableEsm 0 \  
-certificate "" \  
-sequentialSessionReuse 0 \  
-tos 0 \  
-maxPipeline 1 \  
-maxHeaderLen 1024 \  
-maxSessions 3 \  
-enableHttpProxy 0 \  
-enableTos 0 \  
-cookieRejectProbability 0.0 \  
-browserEmulation 3 \  
-cookieJarSize 10 \  
-privateKey "" \  
-commandTimeout 600 \  
-enableIntegrityCheckSupport 0 \  
-commandTimeout_ms 0 \  
-privateKeyPassword "" \  
-urlStatsCount 10 \  
-followHttpRedirects 0 \  
-tcpCloseOption 0 \  
-enableVlanPriority 0 \  
-esm 1460 \  
-httpVersion 0 \  
-sslVersion 3 \  
-enableCookieSupport 0 \  
-piggybackAck true \  
-clientCiphers "DEFAULT" \  
-httpProxy "0.0.0.0" \  
-keepAlive 0 \  
-enableCRCCheckSupport 0 \  
-httpsProxy "0.0.0.0"
```

SEE ALSO

HTTP Client Action, ixNetTraffic, HTTP Client Action, Supported Ciphers



HTTP Client Profile

HTTP Client Profile - configure the an HTTP client's functionality.

SYNOPSIS

```
set HTTP_client_client_network [::IxLoad new ixNetTraffic]
set Activity_newAgent1 [$HTTP_client_client_network activityList.appen
set CustomCommandProfile1 [::IxLoad new ixHttpCommandProfile]
$Activity_newAgent1 agent.profileList.appendItem -object
$CustomCommandProfile1
```

DESCRIPTION

An HTTP client profile is added to the `profileList` option of the HTTP Client Agent object using the `appendItem` subcommand from the HTTP Client Agent.

Request Headers

The request header is a list of type `ixConfigSequenceContainer` used to hold objects of type `ixResponseHeader`. The elements in this list describe the responses of the Web server to HTTP requests as per the profile specified in the cli. You can map multiple header responses to one common profile ID.

```
# Request Headers
```

```
set my_ixHttpRequestString [::IxLoad new ixHttpRequestString]
$my_ixHttpRequestString config \
    -data "Accept: */*"
$Activity_newAgent1 agent.headerList.appendItem -object
$my_ixHttpRequestString
```

Substring Matching

The HTTP client filter strings received in responses from the server.

```
# Substring Match
```

```
set CustomCommandProfile1 [::IxLoad new ixHttpCommandProfile]
$CustomCommandProfile1 config \
    -name "CustomCommandProfile1" \
    -substringMatchEnabled true \
    -userID "test" \
    -id 0 \
    -substring "a" \
```

```
-basicAuthenticationEnabled      true \  
-password                        "test" \  
-caseInsensitiveMatch           true  
  
$CustomCommandProfile1 requestHeaders.clear
```

SUBCOMMANDS

None.

OPTIONS

Substring options

name

This is the name of the profile that needs to be matched.

caseInsensitivematch

If this is enabled, then IxLoad ignores the case of the characters in a substring match. The value is 0 for enabled and 1 if disabled. (Default = 0).

substring

This is the server response text string to be matched, when enabled. Minimum = 0, maximum =1,024. (Default=0).

substringMatchEnabled

If enabled, the response to any command that uses this profile is searched for the text string in the substring field. The value is true if enabled and false if disabled. (Default = false).

basicAuthentication

If this is enabled the client sends an authorization header to the server requesting for a page. The userid and password is sent to the server against Authorization header like userID: password after base64 encoding. (Default = 0).

userID

Identification of the client sending the basicAuthentication request. (Default = userid).



password

Password of the client sending the `basicAuthentication` request. (Default = `pass`).

randomPageGenEnabled

If enabled, the GET requests that contain sequence generators in the `pageObject` field send requests for pages in a random order.

If disabled, GET requests that use sequence generators generate requests in alphabetic or numerical order. The value is true if enabled and false if disabled. (Default = false).

Request Header options

data

This contains the name and the value of the header. The request header maps to the profile list through the profile ID declared in the client action. The request header is specified for GET, HEAD, PUT, POST, DELETE and their SSL counter

HTTP Client Action

HTTP Client Action - configure the actions that an HTTP client will perform.

SYNOPSIS

```
set HTTP_client_client_network [::IxLoad new ixNetTraffic]
set Activity_newAgent1 [$HTTP_client_client_network activityList.appendItem
set my_ixHttpAction [::IxLoad new ixHttpAction]
$Activity_newAgent1 agent.actionList.appendItem -object $my_ixHttpAction
```

DESCRIPTION

An HTTP client action is added to the `actionList` option of the HTTP Client Agent object using the `appendItem` subcommand from the HTTP Client Agent. See the following example:

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_HTTPClient1 [$Traffic1_Network1 activityList.appendItem \
set my_ixHttpAction [::IxLoad new ixHttpAction]
$my_ixHttpAction config \
    -profile 0 \
    -namevalueargs "" \
    -destination "Traffic2_HTTPServer1:80" \
    -abort "None" \
    -command "GET" \
    -arguments "" \
    -pageObject "/1b.html"
$Activity_HTTPClient1 agent.actionList.appendItem -object $my_ixHttpAction
```

Each member of the list may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command. In addition, the following commands are available. Unless otherwise described, no value is returned and an exception is raised for any error found.

`checkConfig`

This subcommand checks the configuration of an individual action.

OPTIONS

abort

This option allows you to abort an operation at one of two places during the interThe following commands support the `abort` option: `GET`, `PUT`, `POST`, `HEAD`, `DELETE`, `GET(SSL)`, `PUT(SSL)`, `POST(SSL)`, `HEAD(SSL)`, and `DELETE(SSL)`.

The types of aborts available are:

Option	Usage
::HttpAction(kAbortNone) or "None"	Do not abort transaction. (Default)
::HttpAction(kAbortBefore) or "AbortBeforeRequest"	Abort the operation immediately after the TCP connection.
::HttpAction(kAbortAfter) or "AbortAfterRequest"	Abort the operation after the operation has been sent to the HTTP server. This option is not valid for SSL connections.

arguments

This option contains an argument that is used by the various commands defined in `command`. The type of the value depends on the command:

Option	Usage
"GET", "GET(SSL)"	N/A.
"DELETE"	N/A.
"HEAD", "HEAD(SSL)"	N/A.
"PUT", "PUT(SSL)"	The name and path of the file to be posted on the server.
"POST", "POST(SSL)"	The name and path of the file to be posted on the server.
"{Think}"	The number of milliseconds to pause before executing the next command in the action list.

command

Selects the HTTP command to be used. One of:

Option	Usage
"GET"	(Default) Retrieves the page specified in the <code>pageObject</code> option.
"GET(SSL)"	Retrieves the page specified in the <code>pageObject</code> option, using SSL. This command must be used if <code>enableSsl</code> is set in the HTTP Client Action.

Option	Usage
"DELETE"	Requests that the server delete the page specified in the <code>pageObject</code> option.
"HEAD"	Retrieves only the HTTP headers for the page specified in <code>pageOb</code> option.
"HEAD(SSL)"	Retrieves only the HTTP headers for the page specified in <code>pageOb</code> option. This command must be used if <code>enableSsl</code> is set in the HTTP Client Action.
"PUT"	Stores the page specified in the <code>pageObject</code> option on the server at the path specified in the <code>arguments</code> option.
"PUT(SSL)"	Stores the page specified in the <code>pageObject</code> option on the server at the path specified in the <code>arguments</code> option. This command must be used if <code>enableSsl</code> is set in the HTTP Client Action.
"POST"	Creates a new object linked to the item specified in the <code>pageObject</code> option. The <code>arguments</code> option can be used to set the object's <code>mesfield</code> .
"POST(SSL)"	Creates a new object linked to the item specified in the <code>pageObject</code> option. The <code>arguments</code> option can be used to set the object's <code>mesfield</code> . This command must be used if <code>enableSsl</code> is set in the HTTP Client Action.
"{Think}"	Adds a pause (think time) before the next command is executed. Specify the duration of the pause in the <code>arguments</code> option.
"{LoopBegin}"	An <code>IxLoad</code> command that you can add to the Command List to cause the commands between it and the <code>{Loop End}</code> to be executed a <code>specnumber</code> of times.
"{LoopEnd}"	Ends the list of commands that will be executed by the preceding <code>{Loop Begin}</code> command.

`destination`

Either the IP address of a real HTTP server or the value of the `-name` option of an HTTP Server Agent. If the HTTP server listens on a port other than the standard (80), enter a colon after the IP address and then enter the port number. If an HTTP Server Agent is used, the port number should agree with the port number associated with the Server Agent. If you are testing an SLB with a virtual IP address (VIP), enter its address here. See the following example:

```
192.168.0.1:80
```

The `destination` option also accepts IPv6 addresses. `IxLoad` supports all forms of IPv6 addressing except `::dotted-quad` notation (for example, `:::1.2.3.4`).

`namevalueargs`

Name=value arguments for GET, HEAD, POST and PUT commands. Name=value arguments are optional and specify parameter names and values; they can occur in any order. To enter multiple name=value arguments, separate the arguments with ampersands (&). For example:

```
name1=value1&name2=value2& . .
```

You can include sequence generators and system variables in the name=value arguments. (Default= "")



pageObject

This option contains a page stored on the HTTP server specified in the Destination field. Three formats are available:

- /#n identifies a target that contains n bytes of HTTP data. For example, /#1 is 1 byte of HTTP data. In addition to the default sizes listed, you can cause the HTTP Server agent to generate a custom-size target by specifying the size using the same convention used for the default sizes. For example, to specify a target of 16 bytes, use /#16.
- /nk.htm identifies a target that is an HTML page that is n kilobytes in size. For example, /4k.htm is an 4096-byte HTML page.
- If you have added customized pages, type its path and name into the Page/Object field. See *Using Your Own Web Pages In IxLoad* for a description of how to use your own custom pages.

You can also include variables in this parameter. See *Using Sequence Generators in HTTP Client Commands and Server Header Name=Value Fields* on page 6-57.

profile

When a HTTP Client Profile is created there is an associated ID, created for each profile. This is incremented by one for each profile. This profile ID maps the Substring match and Request Header together. (Default= -1).

EXAMPLE

```
#-----
# Add actions to this client agent
#-----

set my_ixHttpAction [::IxLoad new ixHttpAction]

$my_ixHttpAction config \
    -profile                0 \
    -namevalueargs         "" \
    -destination           "Traffic2_HTTPServer1:80" \
    -abort                 "None" \
    -command               "GET" \
    -arguments             "" \
    -pageObject            "/1b.html"

$Activity_HTTPClient1 agent.actionList.appendItem -object $my_ixHttpAction
```

SEE ALSO

HTTP Client Agent, Using Your Own Web Pages In IxLoad



HTTP Server Agent

HTTP Server Agent - configure an HTTP server.

SYNOPSIS

```
set HTTP_server_server_network [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$HTTP_server_server_network activityList.
appendItem
$Activity_newServerActivity1 agent.config
```

DESCRIPTION

An HTTP server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

The set of Web pages available through the server is described in the `webPageList` option, which references response headers held in the `responseHeaderList` option and cookies held in the `cookList`.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

In addition, two subcommands are available to load certificates and private keys: `importCertificate` and `importPrivateKey`.

```
importCertificate file
```

Imports a certificate from a disk file, setting the certificate option with the result. For more information, see the description under HTTP Client Agent (on page 1100).

```
importPrivateKey file
```

This subcommand performs the same function, but for the private key. For more information, see the description under HTTP Client Agent (on page 1100).

OPTIONS

```
acceptSslConnections
```

If `true`, the server will accept incoming SSL connections. (Default = `false`).

```
piggybackAck
```

If `true`, the server includes the ACK for the previous packet in the same packet as the next packet..
(Default = `true`).

`minResponseDelay`

Minimum length of time, in milliseconds, that the HTTP server delays sending a response.

`maxResponseDelay`

Maximum length of time, in milliseconds, that the HTTP server delays sending a response.

`privateKey`

If the `acceptSslConnections` parameter is `true`, this parameter specifies a private key in ASCII PEM (Privacy Enhanced Mail) or binary (PKCS#12) format that is used to create a server private key.

`privateKeyPassword`

If the `privateKey` is password-protected (PKCS#12 format), this parameter defines a password for retrieving the key.

`certificate`

If the `acceptSslConnections` parameter is `true`, this parameter specifies a certificate in ASCII PEM (Privacy Enhanced Mail) format that is used to create a server certificate.

`enableDHsupport`

Enables Diffie-Hellman support for DH keys and ADH or EDH ciphers. (Default = `false`).

`dhParams`

If the `EnableDH` support option is selected, this parameter specifies the file that contains a DSA key and certificate. The DSA key is converted to a DH key that can be used in a DH key exchange with an SSL client when the selected cipher is ADH or EDH.

`ServerCiphers`

Defines the server cipher which is one of these listed under `Supported Ciphers`.

`cookieList`

This is a list of type `ixConfigSequenceContainer` used to hold objects of type `ixCookieObject`. The elements in this list describe the cookies that the server sends to clients. (Default = `{}`).

`docrootfile`



Selects the file (zip or tar) that defines default directory path for actual files stored on the HTTP server. HTTP clients can retrieve these files. To retrieve the files specify in the `pageObject` option in the client's command list.

`docrootChunkSize`

If `enableChunkEncoding` is `true`, this option defines the chunk size used for pages in the Docroot file. Specify this value as a min-max range. (Default = 521-1024)

`enable`

Enables the use of this server agent. (Default = `true`).

`enableEsm`

If `true`, the use of the `esm` option is enabled. (Default = `false`).

`enableTos`

Enables the setting of the TOS (Type of Service) bits in the header of the HTTP packets. Use the `tos` option to specify the TOS bit setting.

- 0 (default) TOS bits not enabled.
- 1 TOS bits enabled.

`esm`

If `enableEsm` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size is 1,460 bytes. (Default = 1,460).

`enableMD5Checksum`

If `true`, the server calculates checksums for the pages it sends to the client. See `integrityCheckOption` in HTTP Server Agent and `MD5option` in `ixWebPageOb` (Default = `false`).

`enablePerServerPerURLstat`

If `true`, the statistics from a returned page, records the server IP address from where the page is sent. (Default = `false`).

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = `false`).

`vlanPriority`

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.

`httpPort`

The port number to which the HTTP server will respond to non-SSL requests. To specify multiple listening ports, enter the port numbers, separated by commas (,). You can specify up to 50 listening ports. (Default = 80).

`httpsPort`

The port number to which the HTTP server will respond to SSL requests. To specify multiple listening ports, enter the port numbers, separated by commas (,). You can specify up to 50 listening ports. (Default = 443).

`integrityCheckOption`

Type of checksum calculated for pages requested from the `docroot` file. In order to send checksums, `enableMD5Checksum` must be `true`. Valid values for this option are the following strings:

Option	Usage
Custom MD5	(Default) MD5 checksum in IxLoad-specific header.
Standard MD5	MD5 checksum in RFC 2616-compliant header.
Standard & Custom MD5	MD5 checksum in both IxLoad-specific and RFC-compliant headers
Disable MD5	No checksum is sent.

`enableChunkEncoding`

If `true`, Chunk Transfer-Encoding is enabled. (Default = `false`).

`name`

The name associated with this object, which must be set at object creation.

`requestTimeout`

The amount of time that the server will wait for input on an open connection before closing the session with a '408' error. The legal values are from 1 to 64,000 seconds. (Default = 300).

`responseHeaderList`

This is a list of type `ixConfigSequenceContainer` used to hold objects of type `ixResponseHeader`. The elements in this list describe the responses of the Web server to requests—both returned page contents and other messages. (Default = `{}`).



tcpCloseOption

This option helps the server to close connections. It accepts integer value. (Default = 0).

- 0 (Default) Connections are closed using three way handshake
- 1 Connections are closed by sending Reset (RST) segments instead of Finish (FIN) segments
- 2 Connections are closed using a four-way handshake

tos

If `enableTos` is `true`, this option specifies the IP Precedence / TOS (Type of Service setting and Assured Forwarding classes. (Default = "0"). The choices are:

- 0 (Default) (0x000) routine
- 32 (0x0020) priority service, Assured Forwarding class 1
- 64 (0x0040) immediate service, Assured Forwarding class 2
- 96 (0x0060) flash, Assured Forwarding class 3
- 128 (0x0080) flash-override, Assured Forwarding class 4
- 160 (0x00A0) critical-ecp
- 192 (0x00C0) internet-control

urlStatsCount

Number of URL statistics to display in Statistics (StatViewer) window. During a test, IxLoad displays statistics for one or more URLs in the Statistics window at the bottom of the main IxLoad window. You can use this field to restrict the number of per-URL statistics that are displayed, so that the window displays statistics only for the URLs that are most important to you. If you select a large number of URLs, the Statistics window can become difficult to read. Maximum = 1,000, (Default = 10).

webPageList

This is a list of type `ixConfigSequenceContainer` used to hold objects of type `ixWebPageObject`. The elements in this list describe the headers of the Web pages returned by the server. (Default = {}).

customPayloadList

This is a list of type `ixConfigSequenceContainer` used to hold objects of type `CustomPayloadObject`. The elements in this list describe the payload of the Web pages returned by the server. (Default = {}). Two predefined `CustomPayload` Objects exist, `AsciiCustomPayload` and `HexCustomPayload`.

STATISTICS

EXAMPLE

```
$Activity_HTTPServer1 agent.config \  
-vlanPriority          0 \  
-maxResponseDelay     0 \  
-docrootChunkSize     "512-1024" \  
-enablePerServerPerURLstat 0 \  
-enableEsm            0 \  
-certificate          "" \  
-tos                  0 \  
-enableMD5Checksum    false \  
-httpPort             "80" \  
-httpsPort            "443" \  
-esm                  1460 \  
-enableTos            0 \  
-integrityCheckOption "Custom MD5" \  
-enableChunkEncoding  false \  
-privateKey           "" \  
-privateKeyPassword   "" \  
-urlStatsCount        10 \  
-tcpCloseOption       0 \  
-enableVlanPriority   0 \  
-docrootfile          "" \  
-dhParams              "" \  
-requestTimeout       300 \  
-ServerCiphers        "DEFAULT" \  
-acceptSslConnections 0 \  
-piggybackAck         true \  
-enableDHsupport      0 \  
-minResponseDelay     0
```

SEE ALSO

`ixServerTraffic`, `ixCookieContent`, `ixResponseHeader`, `ixWebPageObject`



ixCookieContent

ixCookieContent—Defines a cookie response for a Web page.

SYNOPSIS

```
$UserCookie cookieContentList.appendItem -object $lastName  
$Activity_newServerActivity1 agent.cookieList.appendItem -object $User
```

DESCRIPTION

The `ixCookieContent` command is used to construct a cookie response associated with a cookie, described in a `ixCookieObject`. The list of cookie contents are noradded to a `ixCookieObject` at the time of its creation.

```
set firstName [::IxLoad new ixCookieContent]  
  
$firstName config \  
  
    -domain                "" \  
    -name                  "firstName" \  
    -maxAge                 "" \  
    -value                 "Joe" \  
    -other                  "" \  
    -path                   ""  
  
$UserCookie cookieContentList.appendItem -object $firstName
```

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

domain

The domain for which the cookie is valid. If omitted, it applies to the entire domain for the HTTP server. (Default = "").

maxAge

The lifetime of the cookie, in seconds. After the time elapses, the client should discard the cookie. A value of zero means the cookie should be discarded immelf blank, the cookie is discarded at the end of the browser session. (Default = "").

name

The name part of the name = value pair being defined. (Default = "name").

other

A comment associated with the cookie. (Default = "").

path

The subset of URLs to which this cookie applies. If omitted, it applies to all URLs for the server. (Default = "")

value

The value part of the name = value pair being defined. (Default = "value").

EXAMPLE

```
set firstName [::IxLoad new ixCookieContent]

$firstName config \

    -domain          "" \
    -name            "firstName" \
    -maxAge          "" \
    -value           "Joe" \
    -other           "" \
    -path            ""

$UserCookie cookieContentList.appendItem -object $firstName

set lastName [::IxLoad new ixCookieContent]

$lastName config \

    -domain          "" \
    -name            "lastName" \
    -maxAge          "" \
    -value           "Smith" \
    -other           "" \
    -path            ""
```



```
$UserCookie cookieContentList.appendItem -object $lastName
```

```
$Activity_newServerActivity1 agent.cookieList.appendItem -object $UserCookie
```

SEE ALSO

HTTP Server Agent

ixCookieObject

ixCookieObject—Defines a cookie.

SYNOPSIS

```
set HTTP_server_server_network [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$HTTP_server_server_network
activityList.appendItem
$Activity_newServerActivity1 agent.cookieList.appendItem -object $UserCookie
```

DESCRIPTION

The ixCookieObject command is used to construct a cookie for the server. The list of cookie contents are normally added to a HTTP Server Agent at the time of its creation.

```
set UserCookie [::IxLoad new CookieObject]

$UserCookie config \

    -mode                3 \
    -type                2 \
    -name                "UserCookie" \
    -description        "Name of User"
```

The cookies are referenced by the actual Web page in an ixWebPageObject included in the webPageList option of the HTTP Server Agent.

SUBCOMMANDS

The options for this command are configured and read using the standard config, cget, and getOptions subcommands defined in the ixConfig command.

OPTIONS

cookieContentList

This is a list of type ixConfigSequenceContainer used to hold objects of type ixCookieContent. The elements in this list describe the cookie contents associated with this cookie. (Default = {}).

description

A description for the cookie. (Default = "").

mode



Determines how the cookies in the cookie group should be handled by the HTTP server that receives them. One of:

Option	Usage
<code>::CookieObject(kModeIgnore)</code>	Causes the server to discard these cookies when it receives them from the client. The IxLoad HTTP server does not add cookies received with this mode to its statistics. Therefore, the statistics for the numof cookies sent by the client will be greater than the number of cookies received by the IxLoad HTTP server.
<code>::CookieObject(kModeReflectSetCookie1)</code>	Causes the server to return the cookies to the client in a Set-Cookie format header.
<code>::CookieObject(kModeReflectSetCookie2)</code>	Causes the server to return the received cookie data to the client in a Set-Cookie2 format header.
<code>::CookieObject(kModeNormal)</code>	(default) Causes the server to perform the functions described by the cookies.

name

The name part of the cookie object being defined. (Default = "name").

readOnly

Indicates that the cookie may not be deleted without resetting this flag. (Default = false)

type

The type of the cookie. One of:

Option	Usage
<code>::CookieObject(kTypeSetCookie1)</code> "1"	Use the original cookie specification, as per RFC 2109.
<code>::CookieObject(kTypeSetCookie2)</code> "2"	(default) Use the cookie 2 specification, as per RFC 2965.

EXAMPLE

```
set UserCookie [::IxLoad new CookieObject]

$UserCookie config \
    -mode                3 \
    -type                2 \
    -name                "UserCookie" \
    -description         "Name of User"
```

```
$UserCookie cookieContentList.clear
```

SEE ALSO

[HTTP Server Agent](#)



ixResponseHeader

ixResponseHeader—Defines a response for a Web page.

SYNOPSIS

```
set HTTP_server_server_network [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$HTTP_server_server_network activityL
options...
$Activity_newServerActivity1 agent.responseHeaderList.appendItem options...
```

DESCRIPTION

The `ixResponseHeader` command is used to describe the responses of the Web server to request both returned page contents and other messages. See the following example:

```
set 200_OK [::IxLoad new ResponseHeader]

$200_OK config \

    -mimeType                "text/plain" \
    -expirationMode          0 \
    -code                     "200" \
    -name                     "200_OK" \
    -lastModifiedMode        1 \
    -lastModifiedIncrementEnable false \
    -lastModifiedDateValue   "2005/02/02 21:55:04" \
    -lastModifiedIncrementFor 1 \
    -expirationDateTimeValue "2005/03/04 21:55:04" \
    -expirationAfterRequestValue 3600 \
    -expirationAfterLastModifiedValue 3600 \
    -lastModifiedIncrementBy 5 \
    -description             "OK"

$200_OK responseList.clear
```

The response header referenced by the actual Web page in an `ixWebPageObject` included in the `responseHeaderList` option of the HTTP Server Agent.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOption` subcommands defined in the `ixConfig` command.

OPTIONS

`code`

The number returned by response. HTTP response codes are defined in Section 10 of RFC 2616. (Default = "200").

`description`

A commentary description for the response. (Default = "OK").

`expirationAfterLastModifiedValue`

If `expirationMode` is set to "AfterLastModified," this is the number of seconds after the page's last modified value, at which time the page will expire. The last modified value is set in `lastModifiedIncrementBy`, and `lastModifoptions`. (Default = 3,600).

`expirationAfterRequestValue`

If `expirationMode` is set to "AfterRequest," this is the number of seconds after which the page will expire. (Default = 3,600).

`expirationDateTime`
Value

If `expirationMode` is set to "DateTime," this is the date and time at which the page will expire. The format of this field is "YYYY/MM/DD HH:MM:SS." For example, "2004/12/31 23:59:59." (Default = "2004/12/31 23:59:59").

`expirationMode`

The means by which the page's expiration is published. One of:

Option	Usage
<code>\$.:ResponseHeader</code> (<code>kExpirationModeNever</code>)	(Default) The page never expires.
<code>\$.:ResponseHeader</code> (<code>kExpirationModeDateTime</code>)	The page expires after a certain date and time, specified in <code>e</code> .
<code>\$.:ResponseHeader</code> (<code>kExpirationModeAfterRequest</code>)	The page expires after a certain amount of time, specified in <code>e</code> .



Option	Usage
\$::ResponseHeader (kExpirationModeAfterLastModified)	The page expires after a certain amount of time following the last modified date and time, specified in e.

lastModifiedDateTimeValue

If lastModifiedMode is set to "DateTime," then this is the value to be returned for the last modified date/time. This value may be incremented for subsequent responses through use of the lastModifiedIncrementEnable, lastModifiedIncrementBy and lastModifiedIncrementFor options. (Default = "2004/12/31 23:59:59").

lastModifiedIncrementBy

If lastModifiedMode is set to "DateTime" and lastModifiedIncrementEnable is set to true, then this is the number of seconds to increment the lastModifiedIncrementBy (Default = 5).

lastModifiedIncrementEnable

If lastModifiedMode is set to "DateTime," this option enables the incrementing of lastModifiedDateTimeValue by lastModifiedIncrementBy as modified by 1. (Default = false).

lastModifiedIncrementFor

If lastModifiedMode is set to "DateTime" and lastModifiedIncrementEnable is set to true, then this is the number of times that the page is referenced before the last modified date/time is incremented as specified in lastModifiedIncrementBy. (Default = 1).

lastModifiedMode

This option determines if and how the last modified field will be published for the page. One of:

Option	Usage
\$::ResponseHeader (kLastModifiedModeNever)	(Default) No last modification time is published
\$::ResponseHeader (kLastModifiedModeDateTime)	A last modification date/time is published as specified in lastModifiedIncrementBy and lastModifiedIncrementFor

contentType

The MIME type for the page. The types: "text/plain," "text/html," and "text/xml" are predefined, but any legal type may be set. (Default = "text/plain").

name

The name of the response. (Default = "200_OK").

responseList

A list of additional headers, to be sent with the response. This list is of type `ixConfigSequenceContainer`; items are added to the list via the `appendItem` sub-command. Each element of the list must be of the form "key: value" where `key` is a value HTTP header key. (Default = {}).

Items are added to this list using the `-data` option. See the following example:

```
$responseHeader responseList.appendItem -data "key1:value1"
```

You can also include variables in this parameter. See [Using Sequence Generators in HTTP Client Commands and Server Header Name=Value Fields on page 6-57](#).

EXAMPLE

```
set 200_OK [::IxLoad new ResponseHeader]

$200_OK config \

    -mimeType                "text/plain" \
    -expirationMode          0 \
    -code                    "200" \
    -name                    "200_OK" \
    -lastModifiedMode        1 \
    -lastModifiedIncrementEnable false \
    -lastModifiedDateValue   "2005/02/02 21:55:04" \
    -lastModifiedIncrementFor 1 \
    -expirationDateTimeValue "2005/03/04 21:55:04" \
    -expirationAfterRequestValue 3600 \
    -expirationAfterLastModifiedValue 3600 \
    -lastModifiedIncrementBy 5 \
    -description             "OK"

$200_OK responseList.clear
```

SEE ALSO

HTTP Server Agent, `ixWebPageObject`



PageObject

ixWebPageObject —Defines a Web page supported by an HTTP Server Agent.

SYNOPSIS

```
set HTTP_server_server_network [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$HTTP_server_server_network
activityOptions...
$Activity_newServerActivity1 agent.webPageList.appendItem options...
```

DESCRIPTION

The `PageObject` command is used to describe the Web pages that are available from the Web server, along with the response header described in `ixResponseHeader` and cookie described in `ixCookieContent`. See the following example:

```
set my_PageObject [::IxLoad new PageObject]

$my_PageObject config \

    -Md5Option          "0" \
    -payloadType        "range" \

    -payloadFile        "<specify file>" \

    -page               "/1b.html" \

    -payloadSize        "1-1" \

    -customPayloadId    -1 \

    -response           $200_OK

$Activity_newServerActivity1 agent.webPageList.appendItem -object
$my_PageObject
```

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`cookie`

This option links to an element in the `cookieList` of the enclosing HTTP Server Agent. It should match the contents of the `name` field of one of the `ixCookieContent` members of that list. (Default = "").

`response`



This option links to an element in the `responseHeaderList` of the enclosing HTTP Server Agent. It should match an `ixResponseHeader` object. (Default = "").

chunkSize

If `enableChunkEncoding` is `true` (in HTTP Server Agent (on page 1118)), this option defines the chunk size used for the page, in bytes. (Default = "1").

Md5Option

Type of checksum generated for page object and sent along with page data to the client. In order to send checksums, `enableMD5Checksum` in the HTTP Server Agent must be `true`. The valid values for this option are:

Option	Usage
0	(Default) Custom MD5. MD5 checksum in IxLoad-specific header.
1	Standard MD5. MD5 checksum in RFC 2616-compliant header.
2	Standard & Custom MD5. MD5 checksum in both IxLoad-specific and RFC-compliant headers
3	Disable MD5. No checksum is sent.

page

The URL of the HTML page that clients can retrieve from the HTTP Server Agent. The path is relative to the root directory of the Ixia server port. You can enter an asterisk (*) at any point on the path, and the server will treat it as a match for any number of directories.

For example, if you configure the page's URL as `/home/liesl/*/pup.html`, a server would serve the page if it received any of the following GET requests:

```
/home/liesl/pics/pup.html  
/home/liesl/0/temp/pup.html  
/home/liesl/pup.html  
(Default = "/newPage.html").
```

payloadFile

If `payloadType` is set to "file," this field specifies the file that will be returned. Make sure to specify the entire path to the file in this specification. Also note that the directory separator `'\'` must be represented as `'\\'` within the string. To download the file to the server, see *Using Your Own Web Pages In IxLoad*. (Default = "<specify file>").

If `payloadType` is set to "customPayload", this field specifies an existing custom payload type or a new one.

payloadSize

If `payloadType` is set to "range," this field specifies the amount of data returned. Specify the size of the data as a minimum size and a maximum size. For exampto specify a minimum size of 1,024 bytes and a maximum of 2,048, specify `1,024-2,048`. To specify a single fixed amount of data, specify the a single value. (Default = 4,096).

payloadType

Indicates the type of payload that will be returned for this page reference. One of:

Option	Usage
<code>\$.:PageObject</code> (<code>kPayloadTypeRange</code>) or "range"	(Default) Causes the Server Agent to generate data automatThe value in <code>payloadSize</code> indicates the amount of data to return.
<code>\$.:PageObject</code> (<code>kPayloadTypeFile</code>) or "file"	Causes the Server Agent to return the actual file indicated in the <code>payloadFile</code> option. See Using Your Own Web Pages In IxLoad for instructions on making your own pages available on the Server Agent.
"customPayload"	Causes the server to return a response that contains syn(generated) data that includes a payload that you creSpecify the payload in the <code>payloadFile</code> option.

EXAMPLE

```
set my_PageObject [::IxLoad new PageObject]

$my_PageObject config \

    -payloadType           "range" \
    -payloadFile           "<specify file>" \
    -page                  "/1b.html" \
    -payloadSize           "1-1" \
    -response              $200_OK

$Activity_newServerActivity1 agent.webPageList.appendItem -object
$my_PageObject
```

SEE ALSO

HTTP Server Agent, `ixResponseHeader`, `ixCookieContent`, Using Your Own Web Pages In IxLoad



CustomPayloadObject

CustomPayloadObject —Defines a custom payload object. A custom payload can contain up to 4096 bytes of ASCII or hexadecimal data.

SYNOPSIS

```
set HTTP_server_server_network [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$HTTP_server_server_network
activityList.appendItem options...
$Activity_HTTPServer1 agent.customPayloadList.appendItem -object
$AsciiCustomPayload...
```

DESCRIPTION

The CustomPayloadObject is used to configure a custom payload object. This object is declared in the payloadFile option of ixWebPageObject.

SUBCOMMANDS

The options for this command are configured and read using the standard config, cget, and getOptions subcommands defined in the ixConfig command.

OPTIONS

repeat

This field determines how often the custom payload appears in the payload space.

If set to true, IxLoad divides the total payload space into 4096-byte blocks, and inserts the custom payload into the first block and into all subsequent 4096-byte blocks. If the custom payload is shorter than 4096 bytes, IxLoad pads the remaining space with zeroes (0).

If set to false, IxLoad inserts the custom payload once, either at the beginning of the payload space or at offset value. If the custom payload is shorter than the total payload space, IxLoad pads the remaining space with generated data (the same type of data that would be generated if you set payloadType to Range).

name

This indicates the name of the customPayloadObject.

asciiPayloadValue

According to the option specified in payloadMode, this option accepts the custom payload value in ASCII.

payloadMode

Specifies 0 (ASCII) or 1 (hexadecimal) value.



offset

Defines the number of bytes from the beginning of the payload field where the payload is inserted.

hexPayloadValue

According to the option specified in `payloadMode`, this option accepts the custom payload value in hexadecimal.

payloadPosition

This can be one of:

`startWith`: inserts a payload at the beginning of the payload field of the response.

`endWith`: inserts a payload at the end of the payload field.

`insertAtMiddle`: inserts a payload at a location within the payload field.

EXAMPLE

```
set my_PageObject1 [::IxLoad new PageObject]

$my_PageObject1 config \

    -payloadType                "customPayload" \
    -payloadFile                 "AsciiCustomPayload" \
    -page                        "/4k.html" \
    -payloadSize                 "4096-4096" \
    -customPayloadId             0 \
    -response                     $200_OK1

$Activity_HTTPServer1 agent.webPageList.appendItem -object $my_PageObject1

set AsciiCustomPayload [::IxLoad new CustomPayloadObject]

$AsciiCustomPayload config \

    -repeat                       false \
    -name                          "AsciiCustomPayload" \
    -asciiPayloadValue             "Ixia-Ixload-Http-Server-Custom-
Payload" \
    -payloadmode                   0 \
    -offset                         1 \
```

```
-hexPayloadValue      "" \  
-payloadPosition      "Start With" \  
-id                    0  
  
$Activity_HTTPServer1 agent.customPayloadList.appendItem -object  
$AsciiCustomPayload
```

SEE ALSO

HTTP Server Agent, ixWebPageObject



Supported Ciphers

The following ciphers are supported by IxLoad HTTPS clients and servers.

SSL 2.0 Cipher Suites

Cipher Suite	Description
RC4-MD5	RC4 data encryption using 128-bit keys and MD5 message digest.
EXP-RC4-MD5	Export version of RC4-MD4 using 40-bit keys.
IDEA-CBC-MD5	IDEA data encryption using 128-bit keys with Cipher Block Chaining and MD5 message digest.
DES-CBC-MD5	DES data encryption using 64-bit keys with Cipher Block Chaining and MD5 message digest.
DES-CBC3-MD5	Triple-DES data encryption using 192-bit keys with Cipher Block Chaining and MD5 message digest.

SSL 3.0 Cipher Suites

Cipher Suite	Description
NULL-MD5	No data encryption, MD5 message digest.
NULL-SHA	No data encryption, SHA-1 message digest.
EXP-RC4-MD5	Export version of RC4-MD5 using 40-bit keys.
RC4-MD5	RC4 data encryption using 128-bit keys and MD5 message digest.
RC4-SHA	RC4 data encryption using 128-bit keys and SHA-1 message digest.
EXP-RC2-CBC-MD5	Exportable cipher using RC2 data encryption with 40-bit keys, Cipher Block Chaining, and MD5 message digest.
IDEA-CBC-SHA	IDEA encryption with Cipher Block Chaining, RSA authentication, and SHA-1 message digest.
EXP-DES-CBC-SHA	Export version of DES-CBC-SHA using 40-bit keys.
DES-CBC-SHA	DES encryption using 168-bit keys, Cipher Block Chaining, RSA authentication, and SHA-1 message digest.
DES-CBC3-SHA	Triple-DES encryption using 168-bit keys, Cipher Block Chaining, and SHA-1 message digest.
EXP-EDH-DSS-DES-CBC-SHA	Export version of EDH-DSS-DES-CBC-SHA using 40-bit keys.

Cipher Suite	Description
EDH-DSS-DES-CBC3-SHA	Ephemeral Diffie-Hellman key exchange with DSS authentication, Triple-DES encryption with Cipher Block Chaining, and SHA-1 message digest.
EXP-EDH-RSA-DES-CBC-SHA	Export version of EDH-RSA-DES-CBC-SHA using 40-bit keys.
EDH-RSA-DES-CBC-SHA	DES encryption with Cipher Block Chaining, RSA authentication, Ephemeral Diffie-Hellman key exchange, and SHA-1 message digest.
EDH-RSA-DES-CBC3-SHA	Triple-DES encryption with Cipher Block Chaining, RSA authentication, Ephemeral Diffie-Hellman key exchange, and SHA-1 message digest.
EXP-ADH-RC4-MD5	Exportable cipher using RC4 encryption with 40-bit keys, Anonymous Diffie-Hellman key exchange, and MD5 message digest.
EXP-ADH-DES-CBC-SHA	Export version of ADH-DES-CBC-SHA using 40-bit keys.
ADH-DES-CBC-SHA	DES encryption with Cipher Block Chaining, Anonymous Diffie-Hellman key exchange, and SHA-1 message digest.
ADH-DES-CBC3-SHA	Triple-DES encryption with Cipher Block Chaining, Anonymous Diffie-Hellman key exchange, and SHA-1 message digest.
EXP1024-DES-CBC-SHA	Exportable cipher with DES encryption and Cipher Block Chaining, RSA authentication, and SHA-1 message digest.
EXP1024-RC4-SHA	Exportable cipher with RC4 encryption, RSA authentication, and SHA-1 message digest.
EXP1024-DHE-DSS-DES-CBC-SHA	Exportable cipher with DES encryption and Cipher Block Chaining, Ephemeral Diffie-Hellman key exchange, DSS authentication, and SHA-1 message digest.
EXP1024-DHE-DSS-RC4-SHA	Exportable cipher with RC4 encryption, DSS authentication, Ephemeral Diffie-Hellman key exchange, and SHA-1 message digest.
DHE-DSS-RC4-SHA	RC4 encryption using 128-bit keys, DSS authentication, Diffie-Hellman key exchange, and SHA-1 message digest.

TLS 1.0 Cipher Suites

Cipher Suite	Description
NULL-MD5	No encryption, RSA authentication and MD5 message digest.

Cipher Suite	Description
NULL-SHA	No encryption, RSA authentication and SHA-1 message digest.
EXP-RC4-MD5	Export version of RC4-MD5.
RC4-MD5	RC4 encryption using 128-bit keys, RSA authentication, and MD5 message digest.
RC4-SHA	RC4 encryption using 128-bit keys, RSA authentication, and SHA-1 message digest.
EXP-RC2-CBC-MD5	Exportable cipher with RC2 encryption using 40-bit keys and Cipher Block Chaining, RSA authentication, and MD5 message digest.
IDEA-CBC-SHA	IDEA encryption with Cipher Block Chaining, RSA authentication, and MD5 message digest.
EXP-DES-CBC-SHA	Export version of DES-CBC-SHA using 40-bit keys.
DES-CBC-SHA	DES encryption with Cipher Block Chaining, RSA authentication, and SHA-1 message digest.
DES-CBC3-SHA	Triple-DES encryption with Cipher Block Chaining, RSA authentication, and SHA-1 message digest.
EXP-EDH-DSS-DES-CBC-SHA	Exportable cipher with DES encryption using 40-bit keys and Cipher Block Chaining, DSS authentication, and SHA-1 message digest.
EDH-DSS-DES-CBC3-SHA	Triple-DES encryption with Cipher Block Chaining, DSS authentication, Ephemeral Diffie-Hellman key exchange, and SHA-1 message digest.
EXP-EDH-RSA-DES-CBC-SHA	Exportable cipher with DES encryption using 40-bit keys and Cipher Block Chaining, RSA authentication, and SHA-1 message digest.
EDH-RSA-DES-CBC-SHA	DES encryption with Cipher Block Chaining, RSA authentication, Ephemeral Diffie-Hellman key exchange, and SHA-1 message digest.
EDH-RSA-DES-CBC3-SHA	Triple-DES encryption with Cipher Block Chaining, RSA authentication, Ephemeral Diffie-Hellman key exchange, and SHA-1 message digest.
EXP-ADH-RC4-MD5	Export version of ADH-RC4-MD5.
ADH-RC4-MD5	RC4 encryption with 128-bit keys, Anonymous Diffie-Hellman key exchange, and MD5 message digest.
EXP-ADH-DES-CBC-SHA	Export version of ADH-DES-CBC-SHA using 40-bit keys.

Cipher Suite	Description
ADH-DES-CBC-SHA	DES encryption with Cipher Block Chaining, Anonymous Diffie-Hellman key exchange, and SHA-1 message digest.
ADH-DES-CBC3-SHA	Triple-DES encryption with Cipher Block Chaining, Anonymous Diffie-Hellman key exchange, and SHA-1 message digest.
EXP1024-DES-CBC-SHA	Exportable cipher with DES encryption and Cipher Block Chaining, RSA authentication, and SHA-1 message digest.
EXP1024-RC4-SHA	Exportable cipher using RC4 encryption with 56-bit keys, RSA authentication, and SHA-1 message digest.
EXP1024-DHE-DSS-DES-CBC-SHA	Exportable cipher using DES encryption and Cipher Block Chaining, Diffie-Hellman key exchange, and SHA-1 message digest.
EXP1024-DHE-DSS-RC4-SHA	Export version of DHE-DSS-RC4-SHA using 56-bit keys.
DHE-DSS-RC4-SHA	RC4 encryption with 128-bit keys, DSS authentication, Diffie-Hellman key exchange, and SHA-1 message digest.

Ciphers Selected from the Generic Ciphers List

Cipher Attribute	Selected Cipher Suite
DEFAULT	<p>Default list of cipher suites. Includes all cipher suites with the following attributes, listed in order of decreasing preference:</p> <ol style="list-style-type: none"> 1. ALL cipher suites (see below). 2. Cipher suites without ADH key exchange (you cannot add ADH ciphers to the list separately). 3. Cipher suites with RC4 encryption and RSA authentication. <p>SSL v2 cipher suites.</p>
ALL	Includes all cipher suites except those in the eNULL suite.
HIGH	Cipher suites with keys larger than 128 bits.
MEDIUM	Cipher suites with 128-bit keys.
LOW	Cipher suites with 40- or 56-bit keys, but not including exportable cipher suites.
EXP,EXPORT	Exportable cipher suites.
EXPORT40	Exportable cipher suites with 40-bit keys.
EXPORT56	Exportable cipher suites with 56-bit keys.



Cipher Attribute	Selected Cipher Suite
eNULL,NULL	Cipher suites with no encryption.
aNULL	Cipher suites with anonymous key exchange (Anonymous Diffie-Hellman).
kRSA,RSA	Cipher suites with RSA key exchange.
kEDH	Cipher suites with Ephemeral Diffie-Hellman key exchange.
aRSA	Cipher suites with RSA authentication.
aDSS,DSS	Cipher suites with DSS authentication.
TLSv1	TLS v.1 cipher suites.
SSLv3	SSL v.3 cipher suites.
SSLv2	SSL v.2 cipher suites.
DH	Cipher suites with Diffie-Hellman key exchange (including Anonymous Diffie-Hellman).
ADH	Cipher suites with Anonymous Diffie-Hellman key exchange.
3DES	Cipher suites with Triple-DES encryption.
DES	Cipher suites with DES encryption (not including those with Triple-DES).
RC4	Cipher suites with RC4 encryption.
RC2	Cipher suites with RC2 encryption.
IDEA	Cipher suites with IDEA encryption.
MD5	Cipher suites with MD5 message digest.
SHA1,SHA	Cipher suites with SHA-1 message digest.

Using Your Own Web Pages In IxLoad

IxLoad includes a series of Web pages of various sizes that you can select for hosting on HTTP server agents and configure HTTP clients to download. In addito the Web pages supplied with IxLoad, you can create your own Web pages or any other type of files (such as GIF, JPEG, EXE, BIN) and host them on IxLoad HTTP server agents.

To use your own Web pages, you create the Web pages (or files) you want to use, and add them to the following directory on your IxLoad client PC:

```
C:\Program  
Files\Ixia\IxLoad\<<version>\Client\Data\HTTP_Server\PagePackage\userpages
```

If you want to store pages in a hierarchy of directories, you can create subdirectories underneath the `\userpages` directory, and store your pages in the subdirectory. IxLoad includes some sample HTML pages you can use as-is or as the basis for creating your own pages.

When you configure a test in IxLoad, IxLoad downloads the pages to the CPUs on each port in the test where an IxLoad HTTP server agent is configured.

The RAM used to store the pages on the Ixia port is shared by other IxLoad processes. The larger the total size of the pages you use, the less space is available for other IxLoad processes, which reduces the number of sessions you can create.

As a guide, if your Load Module has 128 MB of RAM or less and you restrict the total size of the HTML pages to approximately 5MB, the page size will not affect the number of sessions you can create. If your Load Module has 256 MB of RAM, you can use up to 10MB of pages safely. In no case should the total size of your pages exceed 16MB.

 To use your own pages in IxLoad:

1. The default HTML pages are stored in the following directories:

```
C:\Program
Files\Ixia\IxLoad\
```

```
C:\Program
Files\Ixia\IxLoad\
```

Ixia suggests that you keep the default Web pages. If you delete them and you have a client agent that is configured to retrieve one of the deleted pages, the missing page will result in page retrieval errors in the HTTP server and client statistics.

2. Create the HTML pages you want to use, and copy them to the following directory:

```
C:\Program
Files\Ixia\IxLoad\
```

If you want to store pages in subdirectories, consider `\userpages` as the root directory, and create the subdirectories underneath `\userpages`. Ensure that the combined length of the file name and directory path does not exceed 255 characters.

3. If you have not already configured an HTTP server agent, configure one now.
4. Reboot the ports that host the HTTP server agent. If there are just a few ports involved, then use the `ixTest enableResetPorts` option. If there are a large number of ports affected, then it may be better to restart IxServer at the `chasAnytime` you change the pages available on the HTTP server agent, you must reboot the ports.
5. The next time you run your program, IxLoad will download your HTML files to the chassis and the pages will be available on all the HTTP server agents in your test.



Using Sequence Generators in HTTP Client Commands and Server Header Name=Value Fields

Several HTTP fields allow you to include variables in order to generate large numbers of different objects such as page names or HTTP header values.

If the destination of an HTTP client command is an IxLoad HTTP server, you can insert variables into the Page/Object fields to cause the HTTP server to return dynamically-generated pages with unique names.

- You can use the following variables:
- Numbers 0-9
- Letters A-Z and a-z

The letter variables are case-sensitive; IxLoad considers the variable strings “AA” and “aa” to be different.

You can combine the variables with fixed text to create the page names. For example, you can enter page[00-] to create a range of unique user names that begin with the characters “page” (page00, page01, and so on).

To insert the variables into a field, enclose them in square brackets ([]). To specify a range, separate the minimum and maximum values with a hyphen (-). For example, [00-10] specifies a range of 00 through 10.

The number of variables you insert determines the width of the generated strings. For example, the variable “00” can generate the strings 00 - 99. The variable string “000” can generate the strings 000 - 999.

Similarly, “AA” can generate strings that consist of all the two-letter combinations from AA to zz. “AAA” can generate strings that consist of all the three-letter combinations from AAA to zzz.

You can use a single variable string and allow IxLoad to generate strings up to the maximum value of the string or, you can use two variable strings together to restrict the generated strings to a certain range.

See the following example:

[0-] will generate all the values 0 - 9 (0, 1, 2, 3 . . . 9).
[0-5] will generate all the values 0 - 5.

[00-] will generate all the values 00 - 99 (00, 01, 02, 03. . . 97, 98, 99).
[00-50] will generate all the values 00 - 50.

[A-] will generate all the values A - z (A, B, C . . . z).
[A-K] will generate all the values A - K.

[AA-] will generate all the values AA - zz (AA, AB, AC. . . zx, zy, zz).
[AA-KK] will generate all the values AA - KK.

When IxLoad has generated the final string, if the test configuration requires additional strings, IxLoad returns to the starting value of the variable and continues to generate strings until no more are required. In this case, the generated strings will not be unique.

If you include more than one variable, the number of unique pages IxLoad generates is equivalent to the multiple of the maximum values of all the variables. For example, if you specify the page name as:

Page[01-10]_of_[01-99].

the IxLoad server can potentially generate 990 (10*99) unique pages with names "Page01_of_01" through "Page10_of_99." The server will only generate all the pages if it receives requests for all of them from clients.



Note: If you include multiple variables, the start and end fields must be the same width.

You can also configure the HTTP server to include wildcards in page names. For example, you can configure the server for a Web page named "Page*_of_*" where '*' is considered to be the wildcard. A client that used the variables in the previous example would receive all the pages requested.

Using System Variables

In addition to the letter and number variables, there are several system variables you can use. If you include the system variables in a page name, IxLoad replaces the system variable with the appropriate value from the current test configuration. You can use the letter, number, and system variables with HTTP name = value arguments.

You specify the system variables by enclosing them in parentheses (). Like the letter number variables, you must use the system variables in the page name in the URL, not in the path. Table 6-15 describes the system variables.

System Variables for Use in Page Names

System Variable	Description
(\$port-id)	ID of the Ixia port that the client is running on. When the IxLoad HTTP server returns the page, it expands (\$port-id) into <Chassis-Card-Port>.
(\$user-id)	Integer value representing the user that the client is simulating.
(\$sourceip-int)	Integer representation of the source IP address of the simulated user. For 128-bit IPv6 addresses, this is in the format of four integers of the form int-int-int-int.
(\$sourceip)	Source IP address, in dotted-decimal format, of the simulated user.



System Variable	Description
(\$sourceport)	TCP session Source port. Note: You must use (\$sourceport) only in the page name portion of the URL, not in the path. See the following example: Correct: /dir/filename(\$sourceport).html Incorrect: /dir(\$sourceport)/filename.html

Example 1

```
http://ixiacom.com/page($sourceip)-($sourceport).html
```

If this page was retrieved by client 1 from source IP 192.168.1.1 and source port 3589, the page that would be returned would be:

```
http://ixiacom.com/page192.168.1.1-3589.html
```

If the same command was used by a different client from source IP 192.168.2.10 and source port 46990, the page that would be returned would be:

```
http://ixiacom.com/page192.168.2.10-46990.html
```

Example 2

Suppose that a command uses the POST/GET method with a name=value argument of:

```
?user=customer($user-id)&password=pwd&clientport=($sourceport)
```

Then the command list:

```
POST/GET, servername, myfile.html, argument
```

could expand to:

```
POST/GET, servername, myfile.html, ?user=customer1&pass
```

for the first simulated user, and to:

```
POST/GET, servername, myfile.html, ?user=customer99&pass
```

for the 99th simulated user.

Example 3

Suppose that you needed to use a GET command to generate unique user names and passwords for a use on a login page. You could create a page URL of:

```
http://server/login.html?user=user($port-id)_($user-id)&password=pwd
```

IxLoad expands (\$port-id) to chassis-card-port and (\$user-id) to the ID of the simulated user on the port. The pages retrieved might be:

```
http://server/login.html?user=user0-1-1_11&password=pwd
```

```
http://server/login.html?user=user0-1-2_47&password=pwd
```

A command that uses (\$port-id) and (\$user-id) in this way ensures that the user names generated are unique throughout all the ports used in the test.

Statistics

The HTTP statistics are listed in this section.

HTTP Server Statistics

The table below describes the statistics that IxLoad records for the HTTP servers. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\ixia\IxLoad\<version>\Results\<repository>_<test
configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

Statistic	Description
HTTP Requests Received	Number of HTTP requests received by the servers. The statistics show the number of requests for each URL (page).
HTTP Requests Successful	Number of complete and positive HTTP responses (2xx- and 3xx-range responses) sent to the clients. The statistics show the number of requests for each URL (page).
HTTP Requests Failed	Number of HTTP requests from the clients that failed for any reason. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (404)	Number of HTTP requests that failed due to missing files (error 404). The statistics show the number of requests for each URL (page)
HTTP Requests Failed (50x)	Number of HTTP requests that failed due to lack of resources (500-series errors). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Write Error)	Number of HTTP requests that failed due to a socket write error. The statistics show the number of requests for each URL (page).



HTTP Requests Failed (Aborted)	Number of HTTP requests that failed because the HTTP transaction was aborted. The statistics show the number of requests for each URL (page).
HTTP Sessions Rejected (503)	Service Unavailable. Number of HTTP sessions that could not be established due to lack of resources on the server.
HTTP Sessions Timeouts (408)	<p>Number of HTTP 408 responses sent. This statistic includes all 408 responses sent regardless of whether they were received for a pending HTTP request or not.</p> <p>IxLoad counts 408 responses differently depending on whether or not a client has a pending HTTP request:</p> <p>If a client has an HTTP request pending and it receives a 408 response, IxLoad increments the HTTP Received 408, HTTP Requests Failed (4xx), and HTTP Requests Failed statistics.</p> <p>If a client does not have an HTTP request pending and it receives a 408 response, IxLoad only increments the HTTP Received 408 statistic.</p>
HTTP Transactions Active	Number of HTTP transactions transferring HTTP commands or data.
HTTP Responses Sent (1xx)	<p>Number of 100-series responses sent.</p> <p>100-series responses indicate a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line.</p> <p>Refer to RFC 2616, Section 10, for a full description.</p>
HTTP Responses Sent (2xx)	<p>Number of 200-series responses sent.</p> <p>200-series responses indicate that the client's request was successfully received, understood, and accepted.</p> <p>Refer to RFC 2616, Section 10, for a full description.</p>
HTTP Responses Sent (3xx)	<p>Number of 300-series (Redirection) responses sent.</p> <p>300-series responses indicate that further action needs to be taken by the user agent in order to fulfill the request.</p> <p>Refer to RFC 2616, Section 10, for a full description.</p>
HTTP Responses Sent (4xx)	<p>Number of 400-series (Bad Request) responses received.</p> <p>400-series responses indicate that the request could not be understood by the server due to malformed syntax.</p> <p>Refer to RFC 2616, Section 10, for a full description.</p>
HTTP Responses Sent (5xx)	<p>Number of 500-series (Server Error) responses sent.</p> <p>500-series responses indicate that the server is aware that it has erred or is incapable of performing the request.</p> <p>Refer to RFC 2616, Section 10, for a full description.</p>
HTTP Responses Sent (Other)	Number of responses sent that were not 100-, 200-, 300-, 400-, or 500-series responses.
Throughput Statistics	

HTTP Bytes Received	<p>Number of HTTP bytes received by the servers.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic due to increases caused by retransmits.</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not.</p>
HTTP Bytes Sent	<p>Number of HTTP bytes sent by the servers.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic (increased by retransmits) or less than this statistic (decreased by broken or reset connections).</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not.</p>
HTTP Content Bytes Received	Number of bytes received that were HTTP content.
HTTP Content Bytes Sent	Number of bytes sent that were HTTP content.
Cookie Statistics	
HTTP Cookies Received	Number of cookies received by the server.
HTTP Cookies Sent	Number of cookies sent by the server.
HTTP Cookies Received With Matching ServerID	<p>Number of cookies received in which the server ID matched the server name.</p> <p>Server IDs are unique per Ixia port, and can be sent in a cookie as a VALUE for a server NAME in a NAME=VALUE pair. The servers track these IDs, and when a server NAME received from a client matches one tracked by the server, the server tries to match the server ID that was sent as the VALUE.</p>
HTTP Cookies Received With Non-matching ServerID	Number of cookies received in which the server ID did not match the server name.
Transfer Encoding Statistics	
HTTP Chunked Encoded Responses Sent	Number of HTTP responses sent that used chunked-transfer encoding.
HTTP Total Chunks Sent	Total number of chunked-transfer chunks sent.



HTTP Server Conditional View Statistics

The table below describes the conditional view statistics that IxLoad records for the HTTP servers. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

The Conditional Views column in the table indicates the views in which a statistic is available:

IP: per-IP view
User: per-User view
VLAN: per-VLAN view
All: all views

Statistic	Conditional Views	Description
HTTP TCP Connections Accepted	IP, VLAN	Number of requests to establish TCP connections accepted by the server. This statistic is only available in Conditional View.
HTTP TCP Connections Closed	IP, VLAN	Number of TCP connections that ended normally. This statistic is only available in Conditional View.
HTTP TCP Connections Failed	IP, VLAN	Number of TCP connections that did ended abnormally, for any reason. This statistic is only available in Conditional View.
HTTP TCP Connections Failed Due to Socket Error	IP, VLAN	Number of TCP connections that did ended abnormally due to a socket error. This statistic is only available in Conditional View.
HTTP Server Sessions Timeouts (408)	IP, VLAN	Number of HTTP 408 responses sent. This statistic includes all 408 responses sent regardless of whether they were received for a pending HTTP request or not. IxLoad counts 408 responses differently depending on whether or not a client has a pending HTTP request: If a client has an HTTP request pendand it receives a 408 response, IxLoad increments the HTTP Received 408, HTTP Requests Failed (4xx), and HTTP Requests Failed statistics. If a client does not have an HTTP request pending and it receives a 408 response, IxLoad only increthe HTTP Received 408 statistic.

HTTP Server Sessions Rejected (503)	IP, VLAN	Service Unavailable. Number of HTTP sessions that could not be established due to lack of resources on the server.
HTTP Server Requests Received	IP, VLAN	Number of HTTP requests received by the servers. The statistics show the number of requests for each URL (page).
HTTP Server Requests Successful	IP, VLAN	Number of complete and positive HTTP responses (2xx- and 3xx-range responses) sent to the clients. The statistics show the number of requests for each URL (page).
HTTP Server Responses Sent (1xx)	IP, VLAN	Number of 100-series responses sent. 100-series responses indicate a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line. Refer to RFC 2616, Section 10, for a full description.
HTTP Server Responses Sent (2xx)	IP, VLAN	Number of 200-series responses sent. 200-series responses indicate that the client's request was successfully received, understood, and accepted. Refer to RFC 2616, Section 10, for a full description.
HTTP Server Responses Sent (3xx)	IP, VLAN	Number of 300-series (Redirection) responses sent. 300-series responses indicate that further action needs to be taken by the user agent in order to fulfill the request. Refer to RFC 2616, Section 10, for a full description.
HTTP Server Responses Sent (4xx)	IP, VLAN	Number of 400-series (Bad Request) responses received. 400-series responses indicate that the request could not be understood by the server due to malformed syntax. Refer to RFC 2616, Section 10, for a full description.
HTTP Server Responses Sent (5xx)	IP, VLAN	Number of 500-series (Server Error) responses sent. 500-series responses indicate that the server is aware that it has erred or is incapable of performing the request. Refer to RFC 2616, Section 10, for a full description.
HTTP Server Responses Sent (Other)	IP, VLAN	Number of responses sent that were not 100-, 200-, 300-, 400-, or 500-series responses.



HTTP Server Requests Failed	IP, VLAN	Number of HTTP requests from the clients that failed for any reason. The statistics show the number of requests for each URL (page).
HTTP Server Requests Failed (404)	IP, VLAN	Number of HTTP requests that failed due to missing files (error 404). The statistics show the number of requests for each URL (page)
HTTP Server Requests Failed (50x)	IP, VLAN	Number of HTTP requests that failed due to lack of resources (500-series errors). The statistics show the number of requests for each URL (page).
HTTP Server Requests Failed (Write Error)	IP, VLAN	Number of HTTP requests that failed due to a socket write error. The statistics show the number of requests for each URL (page).
HTTP Server Bytes Received	IP, VLAN	<p>Number of HTTP bytes received by the servers.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic due to increases caused by retransmits.</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not.</p>
HTTP Server Bytes Sent	IP, VLAN	<p>Number of HTTP bytes sent by the servers.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic (increased by retransmits) or less than this statistic (decreased by broken or reset connections).</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not.</p>
HTTP Server Transactions Active	IP, VLAN	Number of HTTP transactions transferring HTTP commands or data.
HTTP Server Cookies Sent	IP, VLAN	Number of cookies sent by the server.
HTTP Server Cookies Received	IP, VLAN	Number of cookies received by the server.
HTTP Server Cookies Received With Matching ServerID	IP, VLAN	<p>Number of cookies received in which the server ID matched the server name.</p> <p>Server IDs are unique per Ixia port, and can be sent in a cookie as a VALUE for a server NAME in a NAME=VALUE pair. The servers track these IDs, and when a server NAME received from a client matches one tracked by the server, the server tries to match the server ID that was sent as the VALUE.</p>
HTTP Server Cookies Received With Non-matching ServerID	IP, VLAN	Number of cookies received in which the server ID did not match the server name.
HTTP Server Content Bytes Received	IP, VLAN	Number of bytes received that were HTTP content.

HTTP Server Content Bytes Sent	IP, VLAN	Number of bytes sent that were HTTP content.
--------------------------------	----------	--

HTTP Server per-URL Statistics

The table below describes the statistics that IxLoad records for the HTTP servers. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

Statistic	Description
<i>The following statistics are available only for the first 'N' distinct URLs configured in the list of HTTP Server Web Pages, where 'N' is the value for 'Max Number of URLs' in the 'Per URL Stat Settings' on the HTTP Server's Advanced Options tab.</i>	
HTTP Requests Received	Number of HTTP requests received by the servers. The statistics show the number of requests for each URL (page).
HTTP Requests Successful	Number of complete and positive HTTP responses (2xx- and 3xx-range responses) sent to the clients. The statistics show the number of requests for each URL (page).
HTTP Requests Failed	Number of HTTP requests from the clients that failed for any reason. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (404)	Number of HTTP requests that failed due to missing files (error 404). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (50x)	Number of HTTP requests that failed due to lack of resources (500-series errors). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Write Error)	Number of HTTP requests that failed due to a socket write error. The statistics show the number of requests for each URL (page).
HTTP Responses Sent	Total number of HTTP responses of all types sent.
HTTP Responses Sent (1xx)	Number of 100-series responses sent. 100-series responses indicate a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line. Refer to RFC 2616, Section 10, for a full description.

HTTP Responses Sent (2xx)	Number of 200-series responses sent. 200-series responses indicate that the client's request was successfully received, understood, and accepted. Refer to RFC 2616, Section 10, for a full description.
HTTP Responses Sent (3xx)	Number of 300-series (Redirection) responses sent. 300-series responses indicate that further action needs to be taken by the user agent in order to fulfill the request. Refer to RFC 2616, Section 10, for a full description.
HTTP Responses Sent (4xx)	Number of 400-series (Bad Request) responses received. 400-series responses indicate that the request could not be understood by the server due to malformed syntax. Refer to RFC 2616, Section 10, for a full description.
HTTP Responses Sent (5xx)	Number of 500-series (Server Error) responses sent. 500-series responses indicate that the server is aware that it has erred or is incapable of performing the request. Refer to RFC 2616, Section 10, for a full description.
HTTP Responses Sent (Other)	Number of responses sent that were not 100-, 200-, 300-, 400-, or 500-series responses.
HTTP Chunked Encoded Responses Sent	Number of HTTP responses sent that used chunked-transfer encoding.
HTTP Total Chunks Sent	Total number of chunked-transfer chunks sent.
HTTP Average Chunk Size	Average size of the chunks sent.
HTTP Average Chunks per Response	Average number of chunks sent for each HTTP response.

HTTP Client Statistics

The table below lists the statistics IxLoad reports for HTTP clients. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.



Note: The HTTP client statistics do not include the bytes transmitted and received for the SSL handshake.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test
configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Description
Transaction Statistics	
HTTP Requests Sent	Number of HTTP requests sent by the clients. The statistics show the number of requests for each URL (page).
HTTP Requests Successful	Number of positive HTTP responses (2xx- and 3xx-range responses) received by the clients. The statistics show the number of requests for each URL (page).
HTTP Intermediate Responses Received (1xx)	Number of 100-series (Informational) responses received. 100-series responses indicate a provisional response, consisting only of the Status-Line and optional headers, and terminated by an empty line. Refer to RFC 2616, Section 10, for a full description.
HTTP Requests Successful (2xx)	Number of 200-series (Successful) responses received. 200-series responses indicate that the client's request was successfully received, understood, and accepted.
HTTP Requests Successful (3xx)	Number of 300-series (Redirection) responses received. 300-series responses indicate that further action needs to be taken by the user agent in order to fulfill the request.
HTTP Requests Successful (301)	Number of 301 (Moved Permanently) responses received. 301 responses indicate that the requested resource has been assigned a new permanent URI and any future references to this resource should use one of the returned URIs.
HTTP Requests Successful (302)	Number of 302 (Found) responses received. 302 responses indicate that the requested resource resides temporarily under a different URI.
HTTP Requests Successful (303)	Number of 303 (See Other) responses received. 303 responses indicate that the response to the request can be found under a different URI and should be retrieved using a GET method on that resource.
HTTP Requests Successful (307)	Number of 307 (Temporary Redirect) responses received. 307 responses indicate that the requested resource resides temporarily under a different URI.
HTTP Requests Failed	Number of HTTP requests that failed for any reason. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Write)	Number of HTTP requests that failed due to a socket write error. The statistics show the number of requests for each URL (page).



HTTP Requests Failed (Read)	Number of HTTP requests that failed due to a socket read error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Bad Header)	Number of HTTP requests that failed due to a defective HTTP header. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (4xx)	Number of 4xx-range responses received by the clients in response to an HTTP request. The statistics show the number of requests for each URL (page). 408 responses are counted separately by the HTTP Session Timeout (408) statistic and may or may not also be included in the HTTP Requests Failed (4xx) count. See the description of HTTP Session Timeout (408) for more information.
HTTP Requests Failed (400)	Bad Request. Number of requests that failed due to a syntax error in the URL. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (401)	Unauthorized. Number of requests that failed due to because the server did not receive the correct user name or password from the browser. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (403)	Forbidden. Number of requests that failed due to because the name or password supplied by the browser are incorrect. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (404)	Not Found. Number of requests that failed because requested object is not stored on the server on the path supplied. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (407)	Proxy Authentication Required. Number of requests that failed because access to the URL requires authentication with a proxy server.
HTTP Requests Failed (408)	Timeout. Number of requests that failed due to communications between the client and server taking too long. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (4xx other)	Number of HTTP requests that failed for reasons other than a Bad Request (400), Unauthorized (401), Forbidden (403), Not Found (404), or Timeout (408) error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (5xx)	Number of HTTP requests that failed due to lack of resources on the server (HTTP 500-series errors). This statistic is only incremented if the client had issued a request to the server before receiving the 5xx response. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (505)	HTTP Version not Supported. Number of requests that failed because the server does not support the HTTP version used by the client. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (5xx other)	Number of requests that failed for reasons other than an HTTP version mis-match (505). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (other)	Number of requests that failed that could not be classified.
HTTP Requests Failed (Timeout)	Number of HTTP requests that failed because the clients did not receive a response within 600 seconds. The statistics show the number of requests for each URL (page).

HTTP Requests Failed (Aborted)	Number of HTTP requests that ended prematurely due to events outside HTTP or TCP. For example, if any HTTP requests are pending when the Ramp-Down period ends, those requests are aborted by IxLoad. The statistics show the number of requests for each URL (page).
HTTP Session Timeouts (408)	<p>Number of HTTP 408 responses received. This statistic includes all 408 responses received regardless of whether they were received for a pending HTTP request or not.</p> <p>IxLoad counts 408 responses differently depending on whether or not a client has a pending HTTP request:</p> <ul style="list-style-type: none"> ▪ If a client has an HTTP request pending and it receives a 408 response, IxLoad increments the HTTP Received 408, HTTP Requests Failed (4xx), and HTTP Requests Failed statistics. <p>If a client does not have an HTTP request pending and it receives a 408 response, IxLoad only increments the HTTP Received 408 statistic.</p>
HTTP Sessions Rejected (503)	Service Unavailable. Number of HTTP sessions that could not be established due to lack of resources on the server.
HTTP Aborted Before Request	Number of HTTP requests aborted just before sending the request on an open TCP connection.
HTTP Aborted After Request	Number of HTTP requests aborted just after sending the request on an open TCP connection.
HTTP Transactions Active	Number of HTTP transactions transferring HTTP commands or data.
HTTP Users Active	Number of HTTP users simulated.
Throughput Statistics	
HTTP Bytes Sent	<p>Number of HTTP bytes transmitted by the clients.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic (increased by retransmits) or less than this statistic (decreased by broken or reset connections).</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not (HTTP only).</p>
HTTP Bytes Received	<p>Number of HTTP bytes received by the clients.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic due to increases caused by retransmits.</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not (HTTP only).</p>
HTTP Content Bytes Sent	Number of bytes of HTTP data sent.
HTTP Content Bytes Received	Number of bytes of HTTP data received.
HTTP Decompressed Content Bytes Received	Number of bytes of HTTP data decompressed.
Latency Statistics	

HTTP Connect Time (ms)	<p>Average time elapsed between the time the client sends a SYN packet and the time it receives the SYN/ACK.</p> <p>The units for this statistic are microseconds.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
HTTP Time To First Byte (ms)	<p>Average time elapsed before clients received the first byte of an HTTP response.</p> <p>The units for this statistic are microseconds.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
HTTP Time To Last Byte (ms)	<p>Average time elapsed before clients received the last byte of an HTTP response.</p> <p>The units for this statistic are microseconds.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
Cookie Statistics	
HTTP Cookies Received	Number of cookies received by the clients.
HTTP Cookies Sent	Number of cookies sent by the clients.
HTTP Cookies Rejected	<p>Number of cookies rejected by the clients. Clients may reject cookies for the following reasons:</p> <ul style="list-style-type: none"> ▪ Cookie jar is full ▪ Path specified in the cookie is not a subset of the URI requested ▪ Domain of the requesting host does not match the cookie domain <p>Cookie Reject Probability was greater than 0 and cookie was randomly selected for rejection</p>
HTTP Cookies Rejected - (Path Match Failed)	Number of cookies rejected by the clients because the path specified in the cookie was not available on the server.
HTTP Cookies Rejected - (Domain Match Failed)	Number of cookies rejected by the clients because the cookie was sent by a server outside the domain specified in the cookie.
HTTP Cookies Rejected - (Cookiejar Overflow)	Number of cookies rejected by the clients because their cookie jars were full.
HTTP Cookies Rejected - (Probabalistic Reject)	Number of cookies rejected because the clients were configured to reject a percentage of all cookies at random.
HTTP Cookie headers Rejected - (Memory Overflow)	<p>Number of "Set-Cookie" or "Set-Cookie2" headers which were not processed fully due to insufficient memory.</p> <p>If Large Header Support is enabled, IxLoad monitors the available memory while it processes a "Set-Cookie" or "Set-Cookie2" header.</p> <p>If the amount of free memory declines to the point that IxLoad cannot continue processing the header, IxLoad drops the remainder of the header. If this occurs, IxLoad cannot determine the number of cookies that were in the un-processed portion of the header, so the Cookies Rejected total may be inaccurate; the Memory Overflow statistic is an indication of this.</p>
Test Objective Statistics	

HTTP Simulated Users	Number of users to be simulated during the test.
HTTP Concurrent Connections	Number of concurrent connections maintained during the test.
HTTP Connections	Total number of connections established by the clients.
HTTP Connection Attempts	Total number of connections attempted.
HTTP Transactions	Total number of transactions completed by the clients.
HTTP Bytes	Amount of HTTP data sent and received by the clients, in bytes.
Test Objective Rate Statistics	
HTTP Connection Rate	Rate at which the client established HTTP connections.
HTTP Connection Attempt Rate	Rate at which the client attempted to establish HTTP connections.
HTTP Transaction Rate	Rate at which the client completed HTTP transactions.
HTTP Throughput	Rate at which the client sent and received HTTP traffic.
Content Encoding Statistics	
Note: HTTP 1.0 does not support compression (content encoding). If you run a test using HTTP 1.0, the content encoding statistics are zero.	
Content-Encoded Responses Received	Total number of encoded (compressed) pages received.
Gzip Content-Encoding Received	Number of gzip-encoded pages received.
Deflate Content-Encoding Received	Number of deflate-encoded pages received.
Unrecognized Content-Encoding Received	Number of pages received encoded with an unknown encoding method.
Content-Encoded Responses Decode Successful	Total number of pages successfully decoded.
Gzip Content-Encoding Decode Successful	Number of gzip-encoded pages successfully decoded.
Deflate Content-Encoding Decode Successful	Number of deflate-encoded pages successfully decoded.
Content-Encoded Responses Decode Failed	Total number of pages that could not be decoded.
Gzip Content-Encoding Decode Failed	Number of gzip-encoded pages that could not be decoded for all reasons.
Deflate Content-Encoding Decode Failed	Number of deflate-encoded pages that could not be decoded for all reasons.
Gzip Content-Encoding Decode Failed - Data Error	Number of gzip-encoded pages that could not be decoded because the files were corrupted.
Gzip Content-Encoding Decode Failed - Decoding Error	Number of gzip-encoded pages that could not be decoded due to an internal error in IxLoad.
Deflate Content-Encoding Decode Failed - Data Error	Number of deflate-encoded pages that could not be decoded because the files were corrupted.

Deflate Content-Encoding Decode Failed - Decoding Error	Number of deflate-encoded pages that could not be decoded due to an internal error in IxLoad.
Transfer Encoding Statistics	
Chunked Transfer-Encoded Responses Received	Number of pages received with chunked transfer encoding.
Chunked Transfer-Encoding Decode Successful	Number of chunked transfer-encoded pages successfully decoded.
Chunked Transfer-Encoding Decode Failed	Number of chunked transfer-encoded pages that could not be decoded.
Integrity Check Statistics	
Content-MD5 Responses Received	Number of page checksums received in Content-MD5 headers.
Content-MD5 Check Successful	Number of checksums calculated by the client that matched the Content-MD5 checksums in the response headers.
Content-MD5 Check Failed	Number of checksums calculated by the client that did not match the Content-MD5 checksums in the response headers.
Custom-MD5 Responses Received	Number of page checksums received in Custom-MD5 (IxLoad-specific) headers.
Custom-MD5 Check Successful	Number of checksums calculated by the client that matched the Custom-MD5 checksums in the response headers.
Custom-MD5 Check Failed	Number of checksums calculated by the client that did not match the Content-MD5 checksums in the response headers.

HTTP Client Conditional Statistics

The table below lists the conditional statistics IxLoad reports for HTTP clients. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.



Note: The HTTP client statistics do not include the bytes transmitted and received for the SSL handshake.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test
configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

The Conditional Views column in the table indicates the views in which a statistic is available:

IP: per-IP view
User: per-User view

VLAN: per-VLAN view
 All: all views

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Conditional Views	Description
Test Objective Statistics		
HTTP Client Concurrent Connections	All	Number of concurrent connections maintained during the test.
HTTP Client Connections	All	Total number of connections established by the clients.
HTTP Client Connection Attempts	All	Total number of connections attempted.
HTTP Client Transactions	All	Total number of transactions completed by the clients.
HTTP Client Bytes Received and Transmitted	All	Amount of HTTP data sent and received by the clients, in bytes.
Transaction Statistics		
HTTP Client Requests Sent	All	Number of HTTP requests sent by the clients. The statistics show the number of requests for each URL (page).
HTTP Client Requests Successful	All	Number of positive HTTP responses (2xx- and 3xx-range responses) received by the clients. The statistics show the number of requests for each URL (page).
HTTP Client Intermediate Responses Received (1xx)	All	Number of 100-series (Informational) responses received. 100-series responses indicate a provisional response, consisting only of the Status-Line and optional headers, and terminated by an empty line. Refer to RFC 2616, Section 10, for a full description.
HTTP Client Requests Successful (2xx)	All	Number of 200-series (Successful) responses received. 200-series responses indicate that the client's request was successfully received, understood, and accepted.
HTTP Client Requests Successful (3xx)	All	Number of 300-series (Redirection) responses received. 300-series responses indicate that further action needs to be taken by the user agent in order to fulfill the request.



HTTP Client Requests Successful (301)	All	<p>Number of 301 (Moved Permanently) responses received.</p> <p>301 responses indicate that the requested resource has been assigned a new permanent URI and any future references to this resource should use one of the returned URIs.</p>
HTTP Client Requests Successful (302)	All	<p>Number of 302 (Found) responses received.</p> <p>302 responses indicate that the requested resource resides temporarily under a different URI.</p>
HTTP Client Requests Successful (303)	All	<p>Number of 303 (See Other) responses received.</p> <p>303 responses indicate that the response to the request can be found under a different URI and should be retrieved using a GET method on that resource.</p>
HTTP Client Requests Successful (307)	All	<p>Number of 307 (Temporary Redirect) responses received.</p> <p>307 responses indicate that the requested resource resides temporarily under a different URI.</p>
HTTP Client Requests Failed	All	<p>Number of HTTP requests that failed for any reason. The statistics show the number of requests for each URL (page).</p>
HTTP Client Requests Failed (Write)	All	<p>Number of HTTP requests that failed due to a socket write error. The statistics show the number of requests for each URL (page).</p>
HTTP Client Requests Failed (Read)	All	<p>Number of HTTP requests that failed due to a socket read error. The statistics show the number of requests for each URL (page).</p>
HTTP Client Requests Failed (Bad Header)	All	<p>Number of HTTP requests that failed due to a defective HTTP header. The statistics show the number of requests for each URL (page).</p>
HTTP Client Requests Failed (4xx)	All	<p>Number of 4xx-range responses received by the clients in response to an HTTP request. The statistics show the number of requests for each URL (page).</p> <p>408 responses are counted separately by the HTTP Session Timeout (408) statistic and may or may not also be included in the HTTP Requests Failed (4xx) count. See the description of HTTP Session Timeout (408) for more information.</p>
HTTP Client Requests Failed (400)	All	<p>Bad Request. Number of requests that failed due to a syntax error in the URL. The statistics show the number of requests for each URL (page).</p>

HTTP Client Requests Failed (401)	All	Unauthorized. Number of requests that failed due to because the server did not receive the correct user name or password from the browser. The statistics show the number of requests for each URL (page).
HTTP Client Requests Failed (403)	All	Forbidden. Number of requests that failed due to because the name or password supplied by the browser are incorrect. The statistics show the number of requests for each URL (page).
HTTP Client Requests Failed (404)	All	Not Found. Number of requests that failed because requested object is not stored on the server on the path supplied. The statistics show the number of requests for each URL (page).
HTTP Client Requests Failed (407)	All	Proxy Authentication Required. Number of requests that failed because access to the URL requires authentication with a proxy server.
HTTP Client Requests Failed (408)	All	Timeout. Number of requests that failed due to communications between the client and server taking too long. The statistics show the number of requests for each URL (page).
HTTP Client Requests Failed (4xx other)	All	Number of HTTP requests that failed for reasons other than a Bad Request (400), Unauthorized (401), Forbidden (403), Not Found (404), or Timeout (408) error. The statistics show the number of requests for each URL (page).
HTTP Client Requests Failed (5xx)	All	Number of HTTP requests that failed due to lack of resources on the server (HTTP 500-series errors). This statistic is only incremented if the client had issued a request to the server before receiving the 5xx response. The statistics show the number of requests for each URL (page).
HTTP Client Requests Failed (505)	All	HTTP Version not Supported. Number of requests that failed because the server does not support the HTTP version used by the client. The statistics show the number of requests for each URL (page).
HTTP Client Requests Failed (5xx other)	All	Number of requests that failed for reasons other than an HTTP version mis-match (505). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (other)	All	Number of requests that failed that could not be classified.
HTTP Client Requests Failed (Timeout)	All	Number of HTTP requests that failed because the clients did not receive a response within 600 seconds. The statistics show the number of requests for each URL (page).



HTTP Client Requests Failed (Aborted)	All	Number of HTTP requests that ended prematurely due to events outside HTTP or TCP. For example, if any HTTP requests are pending when the Ramp-Down period ends, those requests are aborted by IxLoad. The statistics show the number of requests for each URL (page).
HTTP Client Session Timeouts (408)	All	Timeout. Number of requests that failed due to communications between the client and server taking too long. The statistics show the number of requests for each URL (page).
HTTP Client Sessions Rejected (503)	All	Service Unavailable. Number of HTTP sessions that could not be established due to lack of resources on the server.
HTTP Client Aborted Before Request	All	Number of HTTP requests aborted just before sending the request on an open TCP connection.
HTTP Client Aborted After Request	All	Number of HTTP requests aborted just after sending the request on an open TCP connection.
HTTP Client Transactions Active	All	Number of HTTP transactions transferring HTTP commands or data.
Throughput Statistics		
HTTP Client Bytes Sent	All	Number of HTTP bytes transmitted by the clients. If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic (increased by retransmits) or less than this statistic (decreased by broken or reset connections). SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not (HTTP only).
HTTP Client Bytes Received	All	Number of HTTP bytes received by the clients. If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic due to increases caused by retransmits. SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not (HTTP only).
HTTP Client Content Bytes Sent	All	Number of bytes of HTTP data sent.
HTTP Client Content Bytes Received	All	Number of bytes of HTTP data received.
Cookie Statistics		
HTTP Client Cookies Received	All	Number of cookies received by the clients.
HTTP Client Cookies Sent	All	Number of cookies sent by the clients.

HTTP Client Cookies Rejected	All	<p>Number of cookies rejected by the clients. Clients may reject cookies for the following reasons:</p> <ul style="list-style-type: none"> ▪ Cookie jar is full ▪ Path specified in the cookie is not a subset of the URI requested ▪ Domain of the requesting host does not match the cookie domain <p>Cookie Reject Probability was greater than 0 and cookie was randomly selected for rejection</p>
HTTP Client Cookies Rejected - (Path Match Failed)	All	Number of cookies rejected by the clients because the path specified in the cookie was not available on the server.
HTTP Client Cookies Rejected - (Domain Match Failed)	All	Number of cookies rejected by the clients because the cookie was sent by a server outside the domain specified in the cookie.
HTTP Client Cookies Rejected - (Cookiejar Overflow)	All	Number of cookies rejected by the clients because their cookie jars were full.
HTTP Client Cookies Rejected - (Probabalistic Reject)	All	Number of cookies rejected because the clients were configured to reject a percentage of all cookies at random.
Content Encoding Statistics		
HTTP Client Transfer-Encoding Received	All	Number of pages received with chunked transfer encoding.
HTTP Client Transfer-Encoding Decode Successful	All	Number of chunked transfer-encoded pages successfully decoded.
HTTP Client Transfer-Encoding Decode Failed	All	Number of chunked transfer-encoded pages that could not be decoded.
Latency Statistics		
HTTP Client Connect Time (us)	All	<p>Average time elapsed between the time the client sends a SYN packet and the time it receives the SYN/ACK.</p> <p>The units for this statistic are nanoseconds.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
HTTP Client Time To First Byte (us)	All	<p>Average time elapsed before clients received the first byte of an HTTP response.</p> <p>The units for this statistic are nanoseconds.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>



HTTP Client Time To Last Byte (us)	All	Average time elapsed before clients received the last byte of an HTTP response. The units for this statistic are nanoseconds. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
Test Objective Statistics		
HTTP Client Connection Rate	All	Rate at which the client established HTTP connections.
HTTP Client Connection Attempt Rate	All	Rate at which the client attempted to establish HTTP connections.
HTTP Client Transaction Rate	All	Rate at which the client completed HTTP transactions.
HTTP Client Throughput (Kbps)	All	Rate at which the client sent and received HTTP traffic.
Chunk-Transfer Encoding Statistics		
Note: HTTP 1.0 does not support chunked-transfer encoding. If you run a test using HTTP 1.0, the chunked-transfer encoding statistics are zero.		
HTTP Client Chunk Transfer-Encoding Headers Received	All	Number of pages received with chunked transfer encoding.
HTTP Client Chunk Transfer-Encoding Decode Successful	All	Number of chunked transfer-encoded pages successfully decoded.
HTTP Client Chunk Transfer-Encoding Decode Failed	All	Total number of pages that could not be decoded.
HTTP Client Total Chunks Received	All	Total number of chunks received by the client.
HTTP Client Average Chunk Size	All	Average size of the chunks received.
HTTP Client Average Chunks per Response	All	Average number of chunks received for each HTTP response.
Content Encoding Statistics		
Note: HTTP 1.0 does not support compression (content encoding). If you run a test using HTTP 1.0, the content encoding statistics are zero.		
HTTP Client Decoded Content bytes	All	Number of bytes decoded.
HTTP Client Compression Ratio	All	Average ratio of uncompressed content bytes to compressed content bytes (uncomp/comp) in compressed pages.
HTTP Client Content-Encoded Responses Received	All	Total number of encoded (compressed) pages received.
HTTP Client Gzip Content-Encoding Received	All	Number of gzip-encoded pages received.

HTTP Client Deflate Content-Encoding Received	All	Number of deflate-encoded pages received.
HTTP Client Unrecognized Content-Encoding Received	All	Number of pages received encoded with an unknown encoding method.
Content-Encoded Responses Decode Successful	All	Total number of pages successfully decoded.
HTTP Client Gzip Content-Encoding Decode Successful	All	Number of gzip-encoded pages successfully decoded.
HTTP Client Deflate Content-Encoding Decode Successful	All	Number of deflate-encoded pages successfully decoded.
Content-Encoded Responses Decode Failed	All	Number of deflate-encoded pages that could not be decoded for all reasons.
HTTP Client Gzip Content-Encoding Decode Failed	All	Number of gzip-encoded pages that could not be decoded for all reasons.
HTTP Client Deflate Content-Encoding Decode Failed	All	Number of deflate-encoded pages that could not be decoded for all reasons.
HTTP Client Gzip Content-Encoding Decode Failed Data Error	All	Number of gzip-encoded pages that could not be decoded because the files were corrupted.
HTTP Client Gzip Content-Encoding Decode Failed Decoding Error	All	Number of gzip-encoded pages that could not be decoded due to an internal error in IxLoad.
HTTP Client Deflate Content-Encoding Decode Failed Data Error	All	Number of deflate-encoded pages that could not be decoded because the files were corrupted.
HTTP Client Deflate Content-Encoding Decode Failed Decoding Error	All	Number of deflate-encoded pages that could not be decoded due to an internal error in IxLoad.
Integrity Check Statistics		
HTTP Client Content-MD5 Responses Received	All	Number of page checksums received in Content-MD5 headers.
HTTP Client Content-MD5 Check Successful	All	Number of checksums calculated by the client that matched the Content-MD5 checksums in the response headers.
HTTP Client Content-MD5 Check Failed	All	Number of checksums calculated by the client that did not match the Content-MD5 checksums in the response headers.
HTTP Client Custom-MD5 Responses Received	All	Number of page checksums received in Custom-MD5 (IxLoad-specific) headers.
HTTP Client Custom-MD5 Check Successful	All	Number of checksums calculated by the client that matched the Custom-MD5 checksums in the response headers.
HTTP Client Custom-MD5 Check Failed	All	Number of checksums calculated by the client that did not match the Content-MD5 checksums in the response headers.

HTTP Client per-URL Statistics

The table below lists the per-URL statistics IxLoad reports for HTTP clients. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.



Note: The HTTP client statistics do not include the bytes transmitted and received for the SSL handshake.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test
configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Description
Transaction Statistics	
HTTP Requests Sent	Number of HTTP requests sent by the clients. The statistics show the number of requests for each URL (page).
HTTP Requests Successful	Number of positive HTTP responses (2xx- and 3xx-range responses) received by the clients. The statistics show the number of requests for each URL (page).
HTTP Requests Failed	Number of HTTP requests that failed for any reason. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Write)	Number of HTTP requests that failed due to a socket write error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Read)	Number of HTTP requests that failed due to a socket read error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Bad Header)	Number of HTTP requests that failed due to a defective HTTP header. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (4xx)	Number of 4xx-range responses received by the clients in response to an HTTP request. The statistics show the number of requests for each URL (page). 408 responses are counted separately by the HTTP Session Timeout (408) statistic and may or may not also be included in the HTTP Requests Failed (4xx) count. See the description of HTTP Session Timeout (408) for more information.
HTTP Requests Failed (400)	Bad Request. Number of requests that failed due to a syntax error in the URL. The statistics show the number of requests for each URL (page).

HTTP Requests Failed (401)	Unauthorized. Number of requests that failed due to because the server did not receive the correct user name or password from the browser. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (403)	Forbidden. Number of requests that failed due to because the name or password supplied by the browser are incorrect. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (404)	Not Found. Number of requests that failed because requested object is not stored on the server on the path supplied. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (407)	Proxy Authentication Required. Number of requests that failed because access to the URL requires authentication with a proxy server.
HTTP Requests Failed (408)	Timeout. Number of requests that failed due to communications between the client and server taking too long. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (4xx other)	Number of HTTP requests that failed for reasons other than a Bad Request (400), Unauthorized (401), Forbidden (403), Not Found (404), or Timeout (408) error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (5xx)	Number of HTTP requests that failed due to lack of resources on the server (HTTP 500-series errors). This statistic is only incremented if the client had issued a request to the server before receiving the 5xx response. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (505)	HTTP Version not Supported. Number of requests that failed because the server does not support the HTTP version used by the client. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (5xx other)	Number of requests that failed for reasons other than an HTTP version mis-match (505). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Timeout)	Number of HTTP requests that failed because the clients did not receive a response within 600 seconds. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Aborted)	Number of HTTP requests that ended prematurely due to events outside HTTP or TCP. For example, if any HTTP requests are pending when the Ramp-Down period ends, those requests are aborted by IxLoad. The statistics show the number of requests for each URL (page).
HTTP Aborted Before Request	Number of HTTP requests aborted just before sending the request on an open TCP connection.
HTTP Aborted After Request	Number of HTTP requests aborted just after sending the request on an open TCP connection.
HTTP Responses Received With Match	Number of responses received that matched the Profile search string.
HTTP Responses Received Without Match	Number of responses received that did not match the Profile search string.



HTTP Intermediate Responses Received (1xx)	<p>Number of 100-series (Informational) responses received.</p> <p>100-series responses indicate a provisional response, consisting only of the Status-Line and optional headers, and terminated by an empty line.</p> <p>Refer to RFC 2616, Section 10, for a full description.</p>
HTTP Requests Successful (2xx)	<p>Number of 200-series (Successful) responses received.</p> <p>200-series responses indicate that the client's request was successfully received, understood, and accepted.</p>
HTTP Requests Successful (3xx)	<p>Number of 300-series (Redirection) responses received.</p> <p>300-series responses indicate that further action needs to be taken by the user agent in order to fulfill the request.</p>
HTTP Requests Successful (301)	<p>Number of 301 (Moved Permanently) responses received.</p> <p>301 responses indicate that the requested resource has been assigned a new permanent URI and any future references to this resource should use one of the returned URIs.</p>
HTTP Requests Successful (302)	<p>Number of 302 (Found) responses received.</p> <p>302 responses indicate that the requested resource resides temporarily under a different URI.</p>
HTTP Requests Successful (303)	<p>Number of 303 (See Other) responses received.</p> <p>303 responses indicate that the response to the request can be found under a different URI and should be retrieved using a GET method on that resource.</p>
HTTP Requests Successful (307)	<p>Number of 307 (Temporary Redirect) responses received.</p> <p>307 responses indicate that the requested resource resides temporarily under a different URI.</p>
HTTP Requests Failed (other)	Number of requests that failed that could not be classified.
Chunked-Transfer Encoding Statistics	
HTTP Chunk Encoded Responses Received	Number of pages received with chunked transfer encoding.
HTTP Chunk Encoded Responses Successful	Number of chunked transfer-encoded pages successfully decoded.
HTTP Chunk Encoded Responses Failed	Number of chunked transfer-encoded pages that could not be decoded.
HTTP Total Chunks Received	Total number of chunks received by the client.
HTTP Average Chunk Size	Average size of the chunks received.
HTTP Average Chunks per Response	Average number of chunks received for each HTTP response.
Decompression and Integrity Statistics	
<p>Note: HTTP 1.0 does not support compression (content encoding). If you run a test using HTTP 1.0, the content encoding statistics are zero.</p>	
HTTP Gzip-Encoded Responses Received	Number of gzip-encoded pages received.

HTTP Gzip-Encoded Responses Successful	Number of gzip-encoded pages successfully decoded.
HTTP Gzip-Encoded Responses Failed	Number of gzip-encoded pages that could not be decoded for all reasons.
HTTP Deflate-Encoded Responses Received	Number of deflate-encoded pages received.
HTTP Deflate-Encoded Responses Successful	Number of deflate-encoded pages successfully decoded.
HTTP Deflate-Encoded Responses Failed	Number of deflate-encoded pages that could not be decoded for all reasons.
HTTP Content-MD5 Responses Received	Number of page checksums received in Content-MD5 headers.
HTTP Content-MD5 Responses Successful	Number of checksums calculated by the client that matched the Content-MD5 checksums in the response headers.
HTTP Content-MD5 Responses Failed	Number of checksums calculated by the client that did not match the Content-MD5 checksums in the response headers.
HTTP Custom MD5 Responses Received	Number of page checksums received in Custom-MD5 (IxLoad-specific) headers.
HTTP Custom MD5 Responses Successful	Number of checksums calculated by the client that matched the Custom-MD5 checksums in the response headers.
HTTP Custom MD5 Responses Failed	Number of checksums calculated by the client that did not match the Content-MD5 checksums in the response headers.
Average Compression Ratio	Average ratio of uncompressed content bytes to compressed content bytes (uncomp/comp) in compressed pages.

TCP Reset Statistics

Under some scenarios, the number of RSTs may not match between the client and server.

For example, an Abort following a request generates two RSTs. On the client side, when the first RST is sent, the socket context is destroyed and hence only one RST is included in the client's TCP stats. However, on the server, receiving the first RST doesn't destroy the socket context immediately and so the second RST received is the one that is updated.

IxLoad Statistics Interpolation

IxLoad statistics are interpolated. Because statistics collection points may not fall on run state-change boundaries, when the last statistics collected from the previous state and the first statistics collected from the current state are interpolated, they may not show the true condition of the current state (RU = Ramp Up, SU = Sustain, RD = Ramp Down).

For example, when the statistics from the last connection point in the SU state and the first collection point in the RD state are interpolated, they may show transactions continuing to increase, when in fact they have stopped.

The interpolated statistics for the first and second collection points within a state will show the true condition of that state. For example, when the first and second sets of statistics collected from the RD state were interpolated, they would show that transactions had stopped. See the figure below.

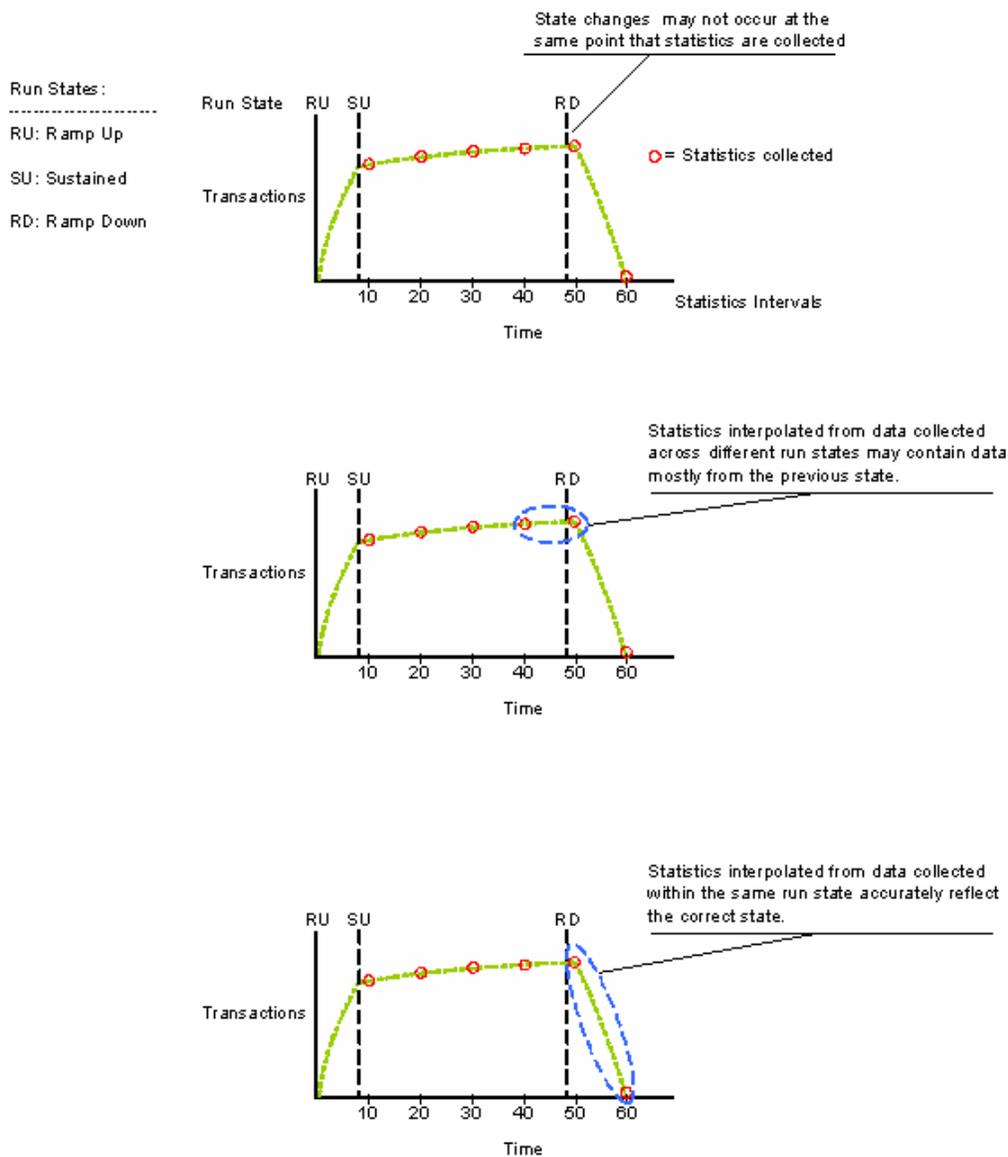


Figure 1: Statistics Collection and Interpolation in IxLoad



16

IMAP

This section describes the IMAP Tcl API objects.

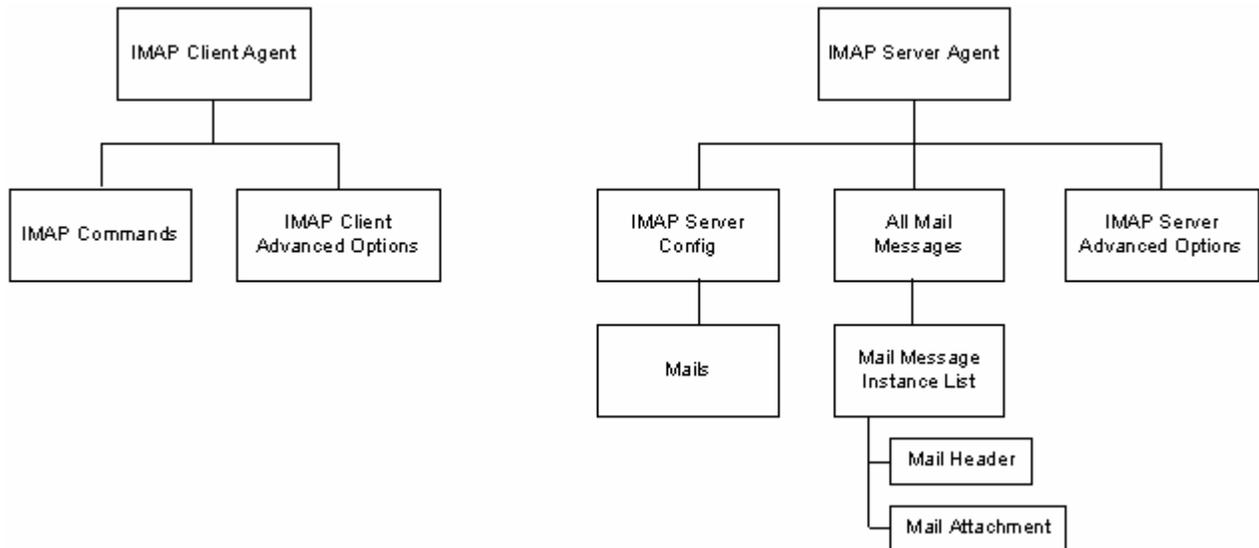
In this chapter:

API Overview	1179
IMAP Client Agent	1185
IMAP Server Agent	1197
Using Auto-Generated Strings	1218
IMAP Statistics	1219

API Overview

IMAP protocol commands are organized as shown in the figure below

An additional section, Using Auto-Generated Strings, pertains to several commands.



Objectives

The objectives (userObjective) you can set for IMAP are listed below. Test objectives are set in the ixTimeline object.

- connectionRate
- transactionRate
- simulatedUsers
- concurrentConnections
- throughputMBps
- throughputKBps

IMAP Client Commands

This section describes the IMAP client commands.



IMAP Client Agent

The IMAP Client Agent command defines a simulated user performing IMAP requests against one or more IMAP servers. Refer to `IMAP Client Agent` for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time.
protocol	Protocol used by the agent. For an IMAP agent, this must be "IMAP."
type	Agent type. For a client agent, this must be "Client."

IMAP Commands

This command defines the list of commands that the client sends to the server. Refer to `IMAP Commands` for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	IMAP command to be executed.

IMAP Client Advanced Options

The IMAP client Advanced Options control network level operation of the client. Refer to `IMAP Client Advanced Options` for a full description of this command. The most significant options of this command are listed below.

Option	Description
enableEsm	If set to 1 (<code>true</code>), the use of the <code>esm</code> option is enabled.
esm	If <code>enableEsm</code> is <code>true</code> , this option specifies the TCP Maximum Segment Size in the MSS (RX) field.
commandTimeout	Amount of time allowed for a command to be completed.

IMAP Server Commands

This section lists the IMAP server commands.

IMAP Server Agent

The IMAP Server Agent command defines the operation of an IMAP server. Refer to `IMAP Server Agent` for a full description of this command. The most significant options of this command are listed below.

Option	Usage
enable	Enables the use of this agent.
name	The name of this server agent, which must be set at agent creation time .
protocol	Protocol used by the agent. For an IMAP agent, this must be "IMAP."
type	Agent type. For a server agent, this must be "Server."

IMAP Server Advanced Options

The IMAP Server Advanced Options command configures the TCP Maximum Segment Size advertised by the server. Refer to `IMAP Server Advanced Options` for a full description of this command. The most significant options of this command are listed below.

Option	Description
enableEsm	If set to 1 (<code>true</code>), the use of the <code>esm</code> option is enabled.
esm	If <code>enableEsm</code> is <code>true</code> , this option specifies the TCP Maximum Segment Size in the MSS (RX) field.

IMAP Server Config

The IMAP Server Config command specifies the list of mail messages available on an IMAP server. Refer to `IMAP Server Config` for a full description of this command. The most significant options of this command are listed below.

Option	Description
mails	List of mail messages available on the server.

Mails

Mails define the number and type of mail messages in a list. Refer to `Mails` for a full description of this command. The most significant options of this command are listed below.

Option	Description
mail_name	Name of mail message type in list.
mail_mesg_count	Number of mail messages of the type specified by the <code>mail_name</code> option.



Mail Message Instance List

Mail Message Instance List configures a mail message. Refer to `Mail Message Instance List` for a full description of this command. The most significant options of this command are listed below.

Option	Description
mail_message_name	Name of mail message.
description	Description of mail message.
cpyfrom	Existing message to be copied to create the new message.
mail_format	Format of the mail message.
mail_size	Size of the mail message in bytes.
headerList	List of <code>Header List</code> objects included with message.
attachmentList	List of <code>Attachment List</code> objects included with message.

Header List

Header List configures a list of headers to be included with a mail message. Refer to `Header List` for a full description of this command. The most significant options of this command are listed below.

Option	Description
field_name	"Name" portion of header to be included in header list.
field_body	"Value" portion of header to be included in header list.

Attachment List

Attachment List configures an attachment to be included with a mail message. Refer to `Attachment List` for a full description of this command. The most significant options of this command are listed below.

Option	Description
attachment_type	Source of the data contained in the attachment.
attachment_data_type	If the <code>attachment_type</code> option is set to 1 (generated data), this option defines the format of the data contained in the attach
attachment_size_range	If the <code>attachment_type</code> option is set to 1 (generated data), this option defines the size range of the attachment, in bytes.
number_of_attachment	If the <code>attachment_type</code> option is set to 1 (generated data), this option defines the number of attachments included with the message.

Option	Description
attach_filename	If the attachment_type option is set to 2 (actual file), this option defines the name and path of the file.

All Mail Messages

All Mail Messages configures a list of Mail Message Instance Lists. Refer to `All Mail Messages` for a full description of this command. The most significant options of this command are listed below.

Option	Description
mailMessageInstList	Name of the list of mail messages.

Mail Message Instance List

Mail Message Instance List configures a list of mail messages. Refer to `Mail Message Instance List` for a full description of this command. The most significant options of this command are listed below.

Option	Description
mail_message_name	Name of mail message.
description	Description of mail message.
cpyfrom	Existing message to be copied to create the new message.
mail_format	Format of the mail message.
mail_size	Size of the mail message in bytes.
mail_size_fixed_len	If the mail_size option is set to Fixed (1), this option specifies the length of the mail message, in bytes.
mail_size_random_min_len	If the mail_size option is set to Random (2), this option specifies the lower bound of the range of the mail message length, in bytes.
mail_size_random_max_len	If the mail_size option is set to Random (2), this option specifies the upper bound of the range of the mail message length, in bytes.
Recent	Flag indicating that message is new.
Seen	Flag indicating that message has been read.
Answered	Flag indicating that message has been answered.
Deleted	Flag indicating that message has been deleted.
Draft	Flag indicating that message has not been completed.
headerList	List of <code>Header List</code> objects included with message.
attachmentList	List of <code>Attachment List</code> objects included with message.



IMAP Client Agent

IMAP Client Agent - create an IMAP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IMAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_IMAPClient1 agent.config
```

DESCRIPTION

An HTTP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time .

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity IMAPClient1 of NetTraffic Traffic1@Network1
#####
set Activity_IMAPClient1 [$Traffic1_Network1 activityList.appendItem \
```



```
-protocolAndType                                "IMAP Client" ]

#####

# Timeline1 for activities IMAPClient1

#####

set Timeline1 [::IxLoad new ixTimeline]

$Timeline1 config \

    -rampUpValue                                1 \

    -rampUpType                                  0 \

    -offlineTime                                 0 \

    -rampDownTime                               20 \

    -standbyTime                                0 \

    -iterations                                 1 \

    -rampUpInterval                             1 \

    -sustainTime                                20 \

    -timelineType                               0 \

    -name                                        "Timeline1"

$Activity_IMAPClient1 config \

    -enable                                      true \

    -name                                        "IMAPClient1" \

    -enableConstraint                           false \

    -userObjectiveValue                         100 \

    -constraintValue                            100 \

    -userObjectiveType                          "simulatedUsers" \

    -timeline                                    $Timeline1

$Activity_IMAPClient1 agent.config \

    -enable                                      true \

    -name                                        "IMAPClient1"

$Activity_IMAPClient1 agent.pm.advOptions.config \
```

```
-commandTimeout          120 \  
-vlan_priority           0 \  
-ipPreference            2 \  
-implicitLoopCheck      true \  
-enableEsm               false \  
-esm                     1460 \  
-enableVlanPriority      false  
  
$Activity_IMAPClient1 agent.pm.ipHistory.clear  
$Activity_IMAPClient1 agent.pm.imapCommands.clear  
$Activity_IMAPClient1 agent.pm.imapCommands.appendItem \  
-id                      "GETMAILS" \  
-Username                 "user\[00-\]" \  
-Message_data_items      "(BODY.PEEK\[\\])" \  
-Password                 "password\[00-\]" \  
-Mailbox_name             "INBOX" \  
-imapServerIp             "Traffic2_IMAPServer1:143"
```

SEE ALSO

ixNetTraffic



IMAP Commands

IMAP Commands—Adds an IMAP client command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IMAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_IMAPClient1 agent.pm.imapCommands.appendItem
```

DESCRIPTION

An `imapCommands` object is added to the `commandList` option of the IMAP Client Agent object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

IMAP command to be executed. One of the following:

Command	Description
CAPABILITY	Requests a list of capabilities that the IMAP server supports. This command is sent to the server that the client has logged in to using a preceding OPEN command.
NOOP	Does not perform any function other than to contact the server. NOOP can be used as a periodic poll for new messages or message status updates during a period of inactivity or to reset an inactivity timer on the server.
LOGOUT	Informs the server that the client is finished using the connectThe server will send a BYE untagged response before the (tagged) OK response, and then close the network connection.
CLOSE	Permanently removes all messages that have the Deleted flag set from the currently selected mailbox, and returns to the authenticated state from the selected state. No untagged EXPUNGE responses are sent.

EXPUNGE	Permanently removes all messages that have the Deleted flag set from the currently selected mailbox. The server for each message that it removes, the server sends an untagged EXPUNGE response to the client. After it has removed all the Deleted-flagged messages, it returns an OK response.
GETMAILS	Retrieves mail messages from a server. {GetMails} is an IxLoad command that combines the functionality of multiple IMAP commands into a single command. {GetMails} performs the following IMAP commands: LOGIN SELECT UID FETCH LOGOUT
DELMAILS	Deletes all mail messages from a selected mailbox. {Delete} is an IxLoad command that combines the functionality of multiple IMAP commands into a single command. {DeleteMails} performs the following IMAP commands: UID STORE EXPUNGE
OPEN	Establishes a TCP connection to an IMAP server. OPEN is not an IMAP command.
LOGIN	Identifies the client to the server and carries the plaintext password authenticating the user.
SELECT	Selects a mailbox so that messages in it can be accessed. The IxLoad IMAP server returns a simple OK response if the mailbox name is valid.
FETCH	Retrieves data associated with a message in the mailbox. The data items to be fetched can be either a single atom or a parenlist.
LIST	Returns a subset of names from the complete set of all names available to the client.
STORE	Alters data associated with a message in the mailbox.
CREATE	Creates a mailbox with the given name.
THINK	The {Think} command causes the client to become inactive. {Think} is an internal IxLoad command intended to assist your testing; it is not a command defined in the IMAP protocol. If you specify identical values for the minimum and maximum times, the client will be inactive for a fixed length of time. If you specify different values for the minimum and maximum times, IxLoad will select a value within the range and cause the client to be inactive for that length of time.
DELETE	Permanently removes a mailbox with the given name. The server returns a simple OK response if the mailbox name is RFC-compliant and the command syntax is correct. Attempting to delete the INBOX or a mailbox name that does not exist is an error.
LoopBeginCommand	An IxLoad command that you can add to the Command List to cause the commands between it and the {Loop End} to be executed a specified number of times.



LoopEndCommand	Ends the list of commands that will be executed by the preced{Loop Begin} command.
----------------	--

Arguments for id = CAPABILITY

None.

Arguments for id = NOOP

None.

Arguments for id = LOGOUT

None.

Arguments for id = CLOSE

None.

Arguments for id = EXPUNGE

None.

Arguments for id = GETMAILS

imapServerIp

The IP address of the IMAP server, or the name of the Ixia IMAP server activity. (Default = "None").

Username

User name used to log in to the IMAP server. You can include variables (as the default value does) to generate multiple unique usernames. See *Using Auto-Generated Strings* on page 16-39. (Default = "user[00-]").

Password

Password used to log in to the IMAP server. You can include variables (as the default value does) to generate multiple unique passwords. See *Using Auto-Generated Strings* on page 16-39. (Default = "password[00-]").

Mailbox_name

Mailbox to retrieve mail from. (Default = "INBOX").

Message_data_items

Message data item names to be retrieved such as ["FLAGS","ENVELOPE"]. (Default = "BODY.PEEK[]").

Arguments for id = DELMAILS

Mailbox_name

Mailbox to delete mail from. (Default = "INBOX").

Arguments for id = OPEN

IMAP_Server_IP

The IP address of the IMAP server, or the name of the Ixia IMAP server activity. (Default = "0.0.0.0").

Arguments for id = LOGIN

Username

User name used to log in to the IMAP server. You can include variables (as the default value does) to generate multiple unique user names. See *Using Auto-Generated Strings* on page 16-39. (Default = "user[00-]").

Password

Password used to log in to the IMAP server. You can include variables (as the default value does) to generate multiple unique passwords. See *Using Auto-Generated Strings* on page 16-39. (Default = "password[00-]").

Arguments for id = SELECT

Mailbox_name

Mailbox selected by command. (Default = "INBOX").

Arguments for id = FETCH

Message_sequence_set

Sequence number set specifying the messages to be retrieved. (Default = "1-1").

Message_data_items

Message data item names to be retrieved. (Default = "FULL").

Arguments for id = LIST

Reference_name

Name of a mailbox or a level of mailbox hierarchy. (Default = "~").

Mail_box_name_with_wildcards

Name of the mailbox to be accessed, and wildcard characters.

The wildcard character "*" matches zero or more characters at this position.

The wildcard character "%" is similar to "*", but it does not match a hierarchy delimiter. If the "%" wildcard is the last character of a mailbox name argument, matching levels of hierarchy are also returned. (Default = "").

Arguments for id = STORE

Message_sequence_set

Sequence number set specifying the messages to be retrieved. (Default = "1-1").

Data_items

Action to be performed on message flags for the affected messages. (Default = "+FLAGS").

- **FLAGS:** Replace the flags for the message (other than \Recent) with the flag selected in the Flags parameter. The new value of the flags is returned as if a FETCH of those flags was done.
- **FLAGS.SILENT:** Equivalent to FLAGS, but without returning a new value.
- **+FLAGS:** Adds the flag selected in the Flags parameter to the message. The new value of the flags is returned as if a FETCH of those flags was done.
- **+FLAGS.SILENT:** Equivalent to +FLAGS, but without returning a new value.
- **-FLAGS:** Removes the flag selected in the Flags parameter from the message. The new value of the flags is returned as if a FETCH of those flags was done.
- **-FLAGS.SILENT:** Equivalent to -FLAGS, but without returning a new value.

Flags

Flag to be added or removed (action depends on setting of Data Items parameter) on the messages specified by Message Sequence Set parameter. (Default = "Answered").

- **Answered:** Message has been answered.
- **Flagged:** Message is marked for urgent or special attention.
- **Deleted:** Message is marked for deletion, to be removed by an EXPUNGE command at a later time.
- **Draft:** Message has not been completely composed (marked as a draft).
- **Seen:** Message has been read.



-Password	"password\[00-\]" \
-Mailbox_name	"INBOX" \
-imapServerIp	"Traffic2_IMAPServer1:143"

SEE ALSO

IMAP Client Agent, Using Auto-Generated Strings

IMAP Client Advanced Options

IMAP Client Advanced Options - configure an IMAP client's advanced options

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IMAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_IMAPClient1 agent.pm.advOptions.config
```

DESCRIPTION

IMAP client advanced options are set through the `pm.advOptions` option of the IMAP Client Agent object (see the example below).

SUBCOMMANDS

None.

OPTIONS

`commandTimeout`

Time, in seconds, to wait for a response to an IMAP command. Minimum = 1, Maximum = 2,147,483. (Default = 12).

`enableEsm`

If `true`, the use of the `esm` option is enabled. (Default = `false`).

`esm`

If `enableEsm` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size as 1,460 bytes. (Default = 1,460).

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = `false`).

`vlanPriority`

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.



ipPreference

This option indicates the order by which the IMAP client will use the subnets, if there is a mixture of IPv4 and IPv6 subnets in the network. The values are: IpPreferenceV4, IpPreferenceV6, IpPreferenceV4Any, IpPreferenceV6Any.

loopValue

If this option is enabled (1), then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (0), then the client will progress through the command list only once, and then go idle. (Default = 0).

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]

#####

# Activity IMAPClient1 of NetTraffic Traffic1@Network1

#####

set Activity_IMAPClient1 [$Traffic1_Network1 activityList.appendItem \

    -protocolAndType                "IMAP Client" ]

$Activity_IMAPClient1 agent.pm.advOptions.config \

    -commandTimeout                 120 \

    -vlan_priority                   0 \

    -ipPreference                    2 \

    -implicitLoopCheck               true \

    -enableEsm                       false \

    -esm                             1460 \

    -enableVlanPriority               false
```

SEE ALSO

IMAP Client Agent

IMAP Server Agent

IMAP Server Agent - configure an IMAP server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IMAPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IMAPServer1 agent.config
```

DESCRIPTION

An IMAP server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this action. (Default = true).

`mailConfig`

This is a list of type `IMAP Server Config`. The elements in this list are the messages that a IMAP user will receive when he queries the mailbox. (Default = {}).

`name`

The name of this server agent, which must be set at agent creation time.

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
```



```
#####  
# Activity IMAPServer1 of NetTraffic Traffic2@Network2  
#####  
set Activity_IMAPServer1 [$Traffic2_Network2 activityList.appendItem \  
    -protocolAndType                "IMAP Server" ]  
set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]  
$Activity_IMAPServer1 config \  
    -enable                          true \  
    -name                            "IMAPServer1" \  
    -timeline                        $_Match_Longest_  
$Activity_IMAPServer1 agent.config \  
    -enable                          true \  
    -name                            "IMAPServer1"  
$Activity_IMAPServer1 agent.pm.advOptions.config \  
    -vlan_priority                   0 \  
    -esm                             1460 \  
    -enableEsm                       true \  
    -enableVlanPriority              true \  
    -listening_port                  "143"  
$Activity_IMAPServer1 agent.pm.imapServerConfig.mails.clear  
$Activity_IMAPServer1 agent.pm.imapServerConfig.mails.appendItem \  
    -id                              "mailMessageList" \  
    -mail_name                       "Simple" \  
    -mail_mesg_count                 10  
$Activity_IMAPServer1 agent.pm.allMailMessages.mailMessageInstList.clear  
$Activity_IMAPServer1 agent.pm.allMailMessages.mailMessageInstList.appendItem \  
    -id                              "mailMessage" \  
    -mail_message_name               "Simple" \  

```

```
-mail_format 1 \  
-description "100 bytes plain text body" \  
-custom_mail_body_use_real_file 0 \  
-Answered 1 \  
-Deleted 1 \  
-custom_mail_body_filename "" \  
-mail_size 1 \  
-custom_mail_body_content "" \  
-Flagged 1 \  
-cpyfrom "Simple" \  
-custom_mail_body_encode 0 \  
-mail_size_fixed_len 100 \  
-Draft 1 \  
-Seen 1 \  
-mail_size_random_min_len 1 \  
-mail_size_random_max_len 4096 \  
-mail_body_type 0 \  
-Recent 1  
  
$Activity_IMAPServer1  
agent.pm.allMailMessages.mailMessageInstList(0).headerList.clear  
  
$Activity_IMAPServer1  
agent.pm.allMailMessages.mailMessageInstList(0).headerList.appendItem \  
-id "mailHeader" \  
-field_body "fromName@company.com" \  
-field_name "From"  
  
$Activity_IMAPServer1  
agent.pm.allMailMessages.mailMessageInstList(0).headerList.appendItem \  
-id "mailHeader" \  
-field_body "fromName@company.com" \  
-field_name "To"
```



```
$Activity_IMAPServer1
agent.pm.allMailMessages.mailMessageInstList(0).attachmentList.clear

$Activity_IMAPServer1 agent.pm.allMailMessages.mailMessageInstList.appendItem
\

  -id                "mailMessage" \
  -mail_message_name "SimpleLarge" \
  -mail_format        1 \
  -description        "4k bytes plain text body" \
  -custom_mail_body_use_real_file 0 \
  -Answered           1 \
  -Deleted            1 \
  -custom_mail_body_filename "" \
  -mail_size           1 \
  -custom_mail_body_content "" \
  -Flagged            1 \
  -cpyfrom            "Simple" \
  -custom_mail_body_encode 0 \
  -mail_size_fixed_len 4096 \
  -Draft              1 \
  -Seen               1 \
  -mail_size_random_min_len 1 \
  -mail_size_random_max_len 4096 \
  -mail_body_type     0 \
  -Recent             1

$Activity_IMAPServer1
agent.pm.allMailMessages.mailMessageInstList(1).headerList.clear

$Activity_IMAPServer1
agent.pm.allMailMessages.mailMessageInstList(1).headerList.appendItem \

  -id                "mailHeader" \
  -field_body        "fromName@company.com" \
  -field_name        "From"
```

```
$Activity_IMAPServer1  
agent.pm.allMailMessages.mailMessageInstList(1).headerList.appendItem \  

```

```
-id                "mailHeader" \  
-field_body        "fromName@company.com" \  
-field_name        "To"
```

```
$Activity_IMAPServer1  
agent.pm.allMailMessages.mailMessageInstList(1).attachmentList.clear
```

```
$Activity_IMAPServer1  
agent.pm.allMailMessages.mailMessageInstList(4).attachmentList.appendItem \  

```

```
-id                "mailAttachment" \  
-attachment_data_type 1 \  
-number_of_attachment "1-1" \  
-attach_filename     "" \  
-attachStr           "" \  
-attachment_size_range "100-100" \  
-attachment_type     "Generated data"
```

SEE ALSO

ixNetTraffic, IMAP Server Config, MailMessage



IMAP Server Advanced Options

IMAP Server Advanced Options - configure an IMAP server's advanced options

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IMAPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IMAPServer1 agent.pm.advOptions.config
```

DESCRIPTION

IMAP client advanced options are set through the `pm.advOptions` option of the `IMAP Client Agent` object.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enableEsm`

If set to 1 (`true`), the use of the `esm` option is enabled. (Default = 0).

`esm`

If `enableEsm` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size as 1,460 bytes. Minimum = 64, Maximum = 1,460. (Default = 1,460).

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = `false`).

`vlanPriority`

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.

`listening_port`

Port that the IMAP server listens on. To specify multiple ports, separate the port numbers with commas (.). You can specify up to 50 listening ports. (Default = 143).

EXAMPLE

```
$Activity_IMAPServer1 agent.pm.advOptions.config \  
-vlan_priority          0 \  
-esm                    1460 \  
-enableEsm              true \  
-enableVlanPriority     true \  
-listening_port         "143"
```

SEE ALSO

IMAP Server Agent



IMAP Server Config

IMAP Server Config—Specifies the list of mail messages available on an IMAP server.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IMAPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IMAPServer1 agent.pm.imapServerConfig.mails.appendItem
```

DESCRIPTION

IMAP Server Config defines the list of mail messages available on an IMAP server.

SUBCOMMANDS

None.

OPTIONS

mails

List of mail messages available on the server. This is a list of objects of type Mails. (Default = "").

EXAMPLE

```
$Activity_IMAPServer1 agent.pm.imapServerConfig.mails.appendItem \  
-id "mailMessageList" \  
-mail_name "Simple" \  
-mail_mesg_count 10
```

SEE ALSO

IMAP Server Agent, Mails

Mails

Mails—Number and type of mail messages available on an IMAP server.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IMAPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IMAPServer1 agent.pm.imapServerConfig.mails.appendItem
```

DESCRIPTION

Mail Message List defines a list of mail messages.

SUBCOMMANDS

None.

OPTIONS

mail_name

Name of mail message type in list. See the table below for a list of the preconfigured mail messages supplied with IxLoad. (Default = "").

Message Name	Description
Simple	Plain text message, 100 bytes in size.
SimpleLarge	Plain text message, 4,096 bytes in size.
HTMLSmall	HTML-format message, 1,024 bytes in size.
HTMLRandom	HTML-format message, size varies randomly between 1,024 and 32,768 bytes.
AttachmentSmall	Plain text message 100 bytes in size, with one plain text attachof 1,024 bytes.
AttachmentLarge	HTML-format message 1,024 bytes in size, with one HTML-format attachment of 65,536 bytes.
RandomSmall	Message body that varies randomly between plain text and HTML format, varying in size between 100 and 1,024 bytes, and with from one to four plain text or HTML-format attachments. The plain text attachments range from 100 to 1,024 bytes in size, and the HTML attachments range from 512 to 4,096 bytes in size.
RandomLarge	Message body that varies randomly between plain text and HTML format, varying in size between 1,024 and 16,384 bytes, and with from one to eight plain texts or HTML-format attachments. The plain text attachments range from 1,024 to 16,384 bytes in size, and the HTML attachments range from 4,096 to 262,144 bytes in size.

mail_mesg_count



Number of mail messages of the type specified by the `mail_name` option. (Default = "10").

EXAMPLE

```
$Activity_IMAPServer1 agent.pm.imapServerConfig.mails.appendItem \  
-id "mailMessageList" \  
-mail_name "Simple" \  
-mail_mesg_count 10
```

SEE ALSO

IMAP Server Agent, MailMessage



-cpyfrom	"Simple" \
-custom_mail_body_encode	0 \
-mail_size_fixed_len	100 \
-Draft	1 \
-Seen	1 \
-mail_size_random_min_len	1 \
-mail_size_random_max_len	4096 \
-mail_body_type	0 \
-Recent	1

SEE ALSO

IMAP Server Agent, Mail Message Instance List

Mail Message Instance List

Mail Message Instance List—Configures one or more mail messages.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IMAPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IMAPServer1
agent.pm.allMailMessages.mailMessageInstList(0).headerList.appendItem
```

DESCRIPTION

Mail Message Instance List defines a list of Mail Message Instance Lists.

SUBCOMMANDS

None.

OPTIONS

mail_message_name

Name of mail message. (Default = "Simple").

description

Description of mail message. (Default = "100 byte plain text body").

cpyfrom

Existing message to be copied to create the new message. (Default = "Simple").

mail_format

Format of the mail message. (Default = "1").

The formats available are:

Format	Description
1	Plain: The message body contains only ASCII characters and no formatting or disinformation. RFC 2822 describes this format.
2	HTML: The message body contains HTML tags for formatting and display. An HTML message is identified by the MIME type <code>text/html</code> .



3	Random: Message bodies are a random mixture of plain and HTML formats.
---	--

`mail_size`

Size of the mail message in bytes. (Default = "1").

Specify the size as follows:

Size	Description
1	Fixed: The size of the message body is fixed at a single size. Use the <code>mail_size_fixed_len</code> option to specify the size.
2	Random: The size of the message body varies randomly between a minimum and a maximum size. Use the <code>mail_size_random_min_len</code> and <code>mail_size_random_max_len</code> .

`mail_size_fixed_len`

If the `mail_size` option is set to Fixed (1), this option specifies the length of the mail message, in bytes. Minimum = "1" Maximum = "2,147,483" (Default = "100").

`mail_size_random_min_len`

If the `mail_size` option is set to Random (2), this option specifies the lower bound of the range of the mail message length, in bytes. Minimum = "1" Maximum = "2,147,483" (Default = "1").

`mail_size_random_max_len`

If the `mail_size` option is set to Random (2), this option specifies the upper bound of the range of the mail message length, in bytes. Minimum = "1" Maximum = "2,147,483" (Default = "4,096").

`mail_body_type`

The mail body type can be default, imported data, or custom. You cannot import files through Tcl so you can work only with default or custom data. The value for default is 1 and custom is 2. Custom data are composed of data that you provide. If 2 is specified, then you need to specify the applicable `custom mail body options`. (Default = 1).

Recent

Flag indicating that message is new. The choices for setting this flag are:

Flag	Description
1	Always Set: Flag is always set.
2	Not Set: Flag is never set.

3	Random: Flag is randomly set.
4	Toggle: Reverses flag setting; if flag is not set, sets it; if flag is set, un-sets it.

Seen

Flag indicating that message has been read. See the description of the Recent flag for a description of the choices for setting this flag.

Answered

Flag indicating that message has been answered. See the description of the Recent flag for a description of the choices for setting this flag.

Deleted

Flag indicating that message has been deleted. See the description of the Recent flag for a description of the choices for setting this flag.

Draft

Flag indicating that message has not been completed. See the description of the Recent flag for a description of the choices for setting this flag.

`custom_mail_body_use_real_file`

This option accepts boolean value of 0 or 1. If zero is given, there is no need to specify a file name. You have to enter the mail message text in `custom_mail_body_content`. If 1 is given, a file name is specified in the `custom_mail_body_filename`.

`custom_mail_body_encode`

This option specifies the encoding option for the real file. For boolean value 0, IxLoads encodes the file using the default encoding. For already encoded files, you choose boolean value 1.

`custom_mail_body_filename`

This option specifies the absolute path for the real file. See the following example: `"c:\temp.txt" \`

`custom_mail_body_content`

This option accepts the mail message text. Example: "abcd123."

`headerList`

List of Header List objects included with message.



attachmentList

List of Attachment List objects included with message.

EXAMPLE

```
$Activity_IMAPServer1 agent.pm.allMailMessages.mailMessageInstList.appendItem
\  
-id                "mailMessage" \  
-mail_message_name "Simple" \  
-mail_format       1 \  
-description       "100 bytes plain text body" \  
-custom_mail_body_use_real_file 0 \  
-Answered         1 \  
-Deleted          1 \  
-custom_mail_body_filename "" \  
-mail_size         1 \  
-custom_mail_body_content "" \  
-Flagged          1 \  
-cpyfrom          "Simple" \  
-custom_mail_body_encode 0 \  
-mail_size_fixed_len 100 \  
-Draft            1 \  
-Seen             1 \  
-mail_size_random_min_len 1 \  
-mail_size_random_max_len 4096 \  
-mail_body_type   0 \  
-Recent          1
```

SEE ALSO

IMAP Server Agent



Header List

Header List—Configures a list of headers to be included with a mail message.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IMAPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IMAPServer1
agent.pm.allMailMessages.mailMessageInstList(0).headerList.appendItem
```

DESCRIPTION

Header List defines a list of headers to be included with a mail message.

SUBCOMMANDS

None.

OPTIONS

field_name

“Name” portion of header to be included in header list. (Default = “From”). The fields names available are listed below.

Field	Description
From	“From” header
To	“To” header
Cc	“Cc” header
Bcc	“Bcc” header
Subject	“Subject” header

In addition you can include your own custom headers by editing the header fields. You can enter any printable US ASCII characters into the fields, except the space () and the colon (:). The custom headers accepts MIME type headers also. MIME type headers start with “Content-”.

field_body

“Value” portion of header to be included in header list. (Default = “fromName@company.com”).

EXAMPLE

```
$Activity_IMAPServer1
agent.pm.allMailMessages.mailMessageInstList(0).headerList.appendItem \

    -id                                "mailHeader" \
```

```
-field_body          "fromName@company.com" \  
-field_name          "From"  
  
$Activity_IMAPServer1  
agent.pm.allMailMessages.mailMessageInstList(0).headerList.appendItem \  
-id                  "mailHeader" \  
-field_body          "fromName@company.com" \  
-field_name          "To"  
  
$Activity_IMAPServer1  
agent.pm.allMailMessages.mailMessageInstList(0).attachmentList.clear
```

SEE ALSO

IMAP Server Agent, MailMessage



Attachment List

Attachment List—Configures a list of attachments to be included with a mail message.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IMAPServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IMAPServer1 agent.pm.allMailMessages.mailMessageIn
```

DESCRIPTION

Mail Attachment defines an attachment to be included with a mail message.

SUBCOMMANDS

None.

OPTIONS

`attachment_type`

Source of the data contained in the attachment. (Default = "1"). The fields names available are listed below.

Field	Description
1	Generated data: The attachment contains data generated automatically by IxLoad.
2	File: The attachment is an actual file.

`attachment_data_type`

If the `attachment_type` option is set to 1 (generated data), this option defines the format of the data contained in the attachment. (Default = "1"). The data types available are listed below.

Data Type	Description
1	Plain: The attachment contains only ASCII characters and no formatting or display information.
2	HTML: The attachment contains HTML tags for formatting and display. An HTML attachment is identified by the MIME type text/html.
3	Random: Attachment is a random mixture of plain and HTML formats.

`attachment_size_range`

If the `attachment_type` option is set to 1 (generated data), this option defines the size range of the attachment, in bytes. The format of the range is "min-max"; use the left-most value to set the minimum size and the right-most value to set the maximum size. (Default = "1-1").

number_of_attachment

If the `attachment_type` option is set to 1 (generated data), this option defines the number of attachments included with the message. The format of the range is "min-max"; use the left-most value to set the minimum number of attachments and the right-most value to set the maximum number. (Default = "1-3").

attach_filename

If the `attachment_type` option is set to 2 (actual file), this option defines the name and path of the file. (Default = "").

EXAMPLE

```
$Activity_IMAPServer1
agent.pm.allMailMessages.mailMessageInstList(4).attachmentList.appendItem \

  -id                "mailAttachment" \
  -attachment_data_type 1 \
  -number_of_attachment "1-1" \
  -attach_filename    "" \
  -attachStr          "" \
  -attachment_size_range "100-100" \
  -attachment_type     "Generated data"
```

SEE ALSO

IMAP Server Agent, MailMessage

Using Auto-Generated Strings

In some of the fields in the IMAP client and server Activities, you can include variables that will cause IxLoad to automatically generate multiple values for the field. For example, the IMAP Username and Password fields both support the inclusion of variables.

You can use the following types of variables:

Fixed Numeric Range

This is a variable of the format [xxx-yyy]. The first value in the variable, xxx, defines the initial value of the range and the number of digits in the range (3, in the case of xxx). The second value defines the maximum value in the range.

For example, if you specify the following user name: user_[001-100], Ixload will generate the user names user_001, user_002, user_003, and so on up to user_100, which will be the final unique user name generated. If more user names are required (because there are more users than user names), IxLoad returns to the initial values of the range and repeats the generation process.

Open-Ended Numeric Range

This is a variable of the format [xx-]. The value in the variable determines the initial value of the range and the number of digits in the range (2, in the case of xx). The maximum value of the range will be reached when all digits are 9.

For example, if you specify the following user name: user_[01-], Ixload will generate the user names user_01, user_02, user_03, and so on up to user_99, which will be the final user unique name generated. If more user names are required (because there are more users than user names), IxLoad returns to the initial values of the range and repeats the generation process.

Fixed Alphabetic Range

This is a variable of the format [aaa-zzz]. The first value in the variable, aaa, defines the initial value of the range and the number of letters in the range (3, in the case of aaa). The second value defines the final value in the range.

For example, if you specify the following user name: user_[aaa-bbb], Ixload will generate the user names user_aaa, user_aab, user_abb, user_aba, user_baa, user_bba, and user_bbb, which will be the final unique user name generated. If more user names are required (because there are more users than user names), IxLoad returns to the initial values of the range and repeats the generation process.

IxLoad treats lowercase letters and uppercase letters as different values. For example, the all-lowercase range [aaa-bbb] does not generate values of AAA, BBB, or any other uppercase letters.

Open-Ended Alphabetic Range

This is a variable of the format [aa-]. The value in the variable determines the initial value of the range and the number of letters in the range (2, in the case of aa). The maximum value of the range will be reached when all digits are 9.

For example, if you specify the following user name: user_[aa-], Ixload will generate the user names user_aa, user_ab, user_ac, and so on up to user_zz, which will be the final unique user name

generated. If more user names are required (because there are more users than user names), IxLoad returns to the initial values of the range and repeats the generation process.

IxLoad treats lowercase letters and uppercase letters as different values. For example, the all-lowercase range [aa-] does not generate values of AA, BB, or any other uppercase letters.

IMAP Statistics

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<>version>\Results\<>repository>_<test
configuration>_
<yyyy><mm><dd>_<hh><mm><ss>
```

For IMAP client statistics, see IMAP Client statistics (on page 1219).

For IMAP server statistics, see IMAP Server Statistics (on page 1222).

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

IMAP Client Statistics

The table below lists the statistics that IxLoad reports for IMAP clients.

Statistic	Description
IMAP Sessions Requested	Number IMAP sessions requested by the client. This statistic is the total of: IMAP Sessions Established + IMAP Sessions Failed. An IxLoad IMAP "session" is the sequence of client/server interactions that take place from the time that a TCP connection is established until it is terminated.
IMAP Sessions Established	Number of IMAP sessions successfully established by the client (the client received an 'OK' response from the server).
IMAP Sessions Failed	Number of IMAP sessions that the client was unable to establish (the client did not receive an 'OK' response from the server).
IMAP Total Bytes Sent	Total number of bytes sent by the client in IMAP commands, responses, and messages. This statistic counts all the bytes in the IMAP packet including the terminating CRLF.
IMAP Total Bytes Received	Total number of bytes received by the client in IMAP commands, responses, and messages. This statistic counts all the bytes in the IMAP packet including the terminating CRLF.
IMAP Mail Bytes Received	Total number of bytes contained in the mail messages retrieved by the client.



IMAP Total Mails Received	Total number of mail messages retrieved by the client.
IMAP Commands Timeout	Total number of IMAP commands for which the client did not receive a response within the timeout period.
IMAP capability Command Sent	Total number of CAPABILITY commands sent by the client.
IMAP noop Command Sent	Total number of NOOP commands sent by the client.
IMAP login Command Sent	Total number of LOGIN commands sent by the client.
IMAP logout Command Sent	Total number of LOGOUT commands sent by the client.
IMAP list Command Sent	Total number of LIST commands sent by the client.
IMAP select Command Sent	Total number of SELECT commands sent by the client.
IMAP fetch Command Sent	Total number of FETCH commands sent by the client.
IMAP store Command Sent	Total number of STORE commands sent by the client.
MAP create Command Sent	Total number of CREATE commands sent by the client.
IMAP delete Command Sent	Total number of DELETE commands sent by the client.
IMAP close Command Sent	Total number of CLOSE commands sent by the client.
IMAP expunge Command Sent	Total number of EXPUNGE commands sent by the client.
MAP uid fetch Command Sent	Total number of UID FETCH commands sent by the client.
IMAP uid store Command Sent	Total number of UID STORE commands sent by the client.
IMAP capability Command Failed	Total number of CAPABILITY commands sent by the client that failed for any reason.
IMAP noop Command Failed	Total number of NOOP commands sent by the client that failed for any reason.
IMAP login Command Failed	Total number of LOGIN commands sent by the client that failed for any reason.
IMAP logout Command Failed	Total number of LOGOUT commands sent by the client that failed for any reason.
IMAP list Command Failed	Total number of LIST commands sent by the client that failed for any reason.
IMAP select Command Failed	Total number of SELECT commands sent by the client that failed for any reason.
IMAP fetch Command Failed	Total number of FETCH commands sent by the client that failed for any reason.
IMAP store Command Failed	Total number of STORE commands sent by the client that failed for any reason.
IMAP create Command Failed	Total number of CREATE commands sent by the client that failed for any reason.
IMAP delete Command Failed	Total number of DELETE commands sent by the client that failed for any reason.
IMAP close Command Failed	Total number of CLOSE commands sent by the client that failed for any reason.

IMAP expunge Command Failed	Total number of EXPUNGE commands sent by the client that failed for any reason.
IMAP uid fetch Command Failed	Total number of UID FETCH commands sent by the client that failed for any reason.
IMAP uid store Command Failed	Total number of UID STORE commands sent by the client that failed for any reason.
IMAP capability Command OK	Total number of CAPABILITY commands for which the client received an OK response.
IMAP noop Command OK	Total number of NOOP commands for which the client received an OK response.
IMAP login Command OK	Total number of LOGIN commands for which the client received an OK response.
IMAP logout Command OK	Total number of LOGOUT commands for which the client received an OK response.
IMAP list Command OK	Total number of LIST commands for which the client received an OK response.
IMAP select Command OK	Total number of SELECT commands for which the client received an OK response.
IMAP fetch Command OK	Total number of FETCH commands for which the client received an OK response.
IMAP store Command OK	Total number of STORE commands for which the client received an OK response.
IMAP create Command OK	Total number of CREATE commands for which the client received an OK response.
IMAP delete Command OK	Total number of DELETE commands for which the client received an OK response.
IMAP close Command OK	Total number of CLOSE commands for which the client received an OK response.
IMAP expunge Command OK	Total number of EXPUNGE commands for which the client received an OK response.
IMAP uid fetch Command OK	Total number of UID FETCH commands for which the client received an OK response.
IMAP uid store Command OK	Total number of UID STORE commands for which the client received an OK response.
IMAP Total Bytes Sent and Received	Combined total of all bytes transmitted and received by the client in IMAP commands, responses, mail messages, and attachments
IMAP Transaction Rate	Rate at which the client completed IMAP transactions.
IMAP Transactions	Total number of IMAP transactions of all types.
IMAP Connection Rate	Rate at which the client established IMAP connections.
IMAP Connections	Total number of IMAP connections established.



IMAP Total Connections	Total number of IMAP connections of established. "Connection" refers to the entire sequence of client/server interactions from the initial establishment of the connection to the server until its termination.
IMAP Concurrent Connection	Number of concurrent IMAP connections active.
IMAP Simulated Users	Number of IMAP users simulated by the client.

IMAP Server Statistics

The table below lists the statistics that IxLoad reports for IMAP servers.

Statistic	Description
IMAP Session Requests Received	Number of requests to establish IMAP sessions received by the server. "Session" refers to the sequence of client/server interactions from the time that a TCP connection is established until the time that TCP connection terminates.
IMAP Session Requests Completed	Number of requested IMAP sessions successfully established by the server.
IMAP Session Requests Failed	Number of requested IMAP sessions that the server failed to establish.
IMAP Total Mail Bytes Sent	Total number of bytes sent by the server in IMAP responses and messages. This statistic counts all the bytes in the IMAP packet including the terminating CRLF.
IMAP Total Mails Sent	Total number of mail messages sent over IMAP connections.
IMAP Total Attachments Sent	Total number of attachments sent over IMAP connections.
IMAP Total Mails with Attachments Sent	Total number of mail messages sent that included one or more attachments.
IMAP Total Bytes Sent	Total number of bytes sent by the server in IMAP commands, responses, and messages.
IMAP Total Bytes Received	Total number of bytes received by the server in IMAP commands, responses, and messages.
IMAP Total Bytes Sent and Received	Combined total of all bytes transmitted and received by the server in IMAP commands, responses, and mail messages.
IMAP capability Command Received	Total number of CAPABILITY commands received by the server.
IMAP noop Command Received	Total number of NOOP commands received by the server.

IMAP login Command Received	Total number of LOGIN commands received by the server.
IMAP logout Command Received	Total number of LOGOUT commands received by the server.
IMAP list Command Received	Total number of LIST commands received by the server.
IMAP select Command Received	Total number of SELECT commands received by the server.
IMAP fetch Command Received	Total number of FETCH commands received by the server.
IMAP store Command Received	Total number of STORE commands received by the server.
IMAP create Command Received	Total number of CREATE commands received by the server.
IMAP delete Command Received	Total number of DELETE commands received by the server.
IMAP close Command Received	Total number of CLOSE commands received by the server.
IMAP expunge Command Received	Total number of EXPUNGE commands received by the server.
IMAP uid_fetch Command Received	Total number of UID FETCH commands received by the server.
IMAP uid_store Command Received	Total number of UID STORE commands received by the server.
IMAP capability Response Sent	Total number of CAPABILITY responses sent by the server.
IMAP noop Response Sent	Total number of NOOP responses sent by the server.
IMAP login Response Sent	Total number of LOGIN responses sent by the server.
IMAP logout Response Sent	Total number of LOGOUT responses sent by the server.
IMAP list Response Sent	Total number of LIST responses sent by the server.
IMAP select Response Sent	Total number of SELECT responses sent by the server.
IMAP fetch Response Sent	Total number of FETCH responses sent by the server.
IMAP store Response Sent	Total number of STORE responses sent by the server.
IMAP create Response Sent	Total number of CREATE responses sent by the server.
IMAP delete Response Sent	Total number of DELETE responses sent by the server.



IMAP close Response Sent	Total number of CLOSE responses sent by the server.
IMAP expunge Response Sent	Total number of EXPUNGE responses sent by the server.
IMAP uid_fetch Response Sent	Total number of UID FETCH responses sent by the server.
IMAP uid_store Response Sent	Total number of UID STORE responses sent by the server.
IMAP capability Sent Failed	Total number of CAPABILITY responses that the server failed to send.
IMAP noop Sent Failed	Total number of NOOP responses that the server failed to send.
IMAP login Sent Failed	Total number of LOGIN responses that the server failed to send.
IMAP logout Sent Failed	Total number of LOGOUT responses that the server failed to send.
IMAP list Sent Failed	Total number of LIST responses that the server failed to send.
IMAP select Sent Failed	Total number of SELECT responses that the server failed to send.
IMAP fetch Sent Failed	Total number of FETCH responses that the server failed to send.
IMAP store Sent Failed	Total number of STORE responses that the server failed to send.
IMAP create Sent Failed	Total number of CREATE responses that the server failed to send.
IMAP delete Sent Failed	Total number of DELETE responses that the server failed to send.
IMAP close Sent Failed	Total number of CLOSE responses that the server failed to send.
IMAP expunge Sent Failed	Total number of EXPUNGE responses that the server failed to send.
IMAP uid_fetch Sent Failed	Total number of UID FETCH responses that the server failed to send.
IMAP uid_store Sent Failed	Total number of UID STORE responses that the server failed to send.

17

IPTV/ Video

This section describes the IPTV/Video Tcl API objects.



In this chapter:

Overview	1227
Video Client Agent	1236
Video Server Agent	1277

Overview

This section describes the IPTV / Video commands.

Video

The IxLoad video API consists of a client agent, a server agent, and their com



Note: Do not run video tests from the `../3rdParty/Tcl8.4.7/bin` directory. During Download-on-Demand (DOD), a `.tgz` file is created which uses a python tarfile which in turn uses `zlib.dll`. The `../3rdParty/Tcl8.4.7/bin` directory contains a local `zlib.dll` inside which confuses the DOD process.

IPTV

The IPTV client and server API structure is similar to the video API structure with some additions.

IPTV Mode Server and Client

The IxLoad Video client and server can operate in either of two modes:

Video to emulate a standard multicast/unicast video client and server. The option for Video is 0.

IPTV to emulate an IPTV client and server. The option for IPTV is 1. For exam

```
$Activity_IPTV_VideoServer1 agent.pm.videoConfig.config \  
-serverMode 1
```

Video Server

In the IPTV mode, the IxLoad video server can be configured to emulate two types of IPTV servers: a combination A/D Server or a V server.

- In an actual IPTV implementation an **A** (Acquisition) server packages RTP streams into multicast UDP packets and streams them onto the distribution network.
- A **D** (Distribution) server caches a certain amount of the multicast video data being streamed over the network. When a user changes a channel, the **D** server sends a short unicast burst of the new channel's video traffic for the user to view while the system switches the user from the previous channel's multicast group to the new channel's group.
- A **V** server provides Video-on-Demand service to an IPTV client.

This is explained in the `type` option in `Video Properties` and `Stream`.



Video Client

In IPTV mode, the IxLoad video client emulates an IPTV client. In IPTV mode, all the same commands are available as in Video mode, except that the `Join` command is replaced with the `ICCCommand` for testing multicast performance. This is explained in the `Commands` section.

IPTV Options

The IPTV Options configure the options specific to the video client in IPTV mode. Refer `IPTV Options` for detailed information.

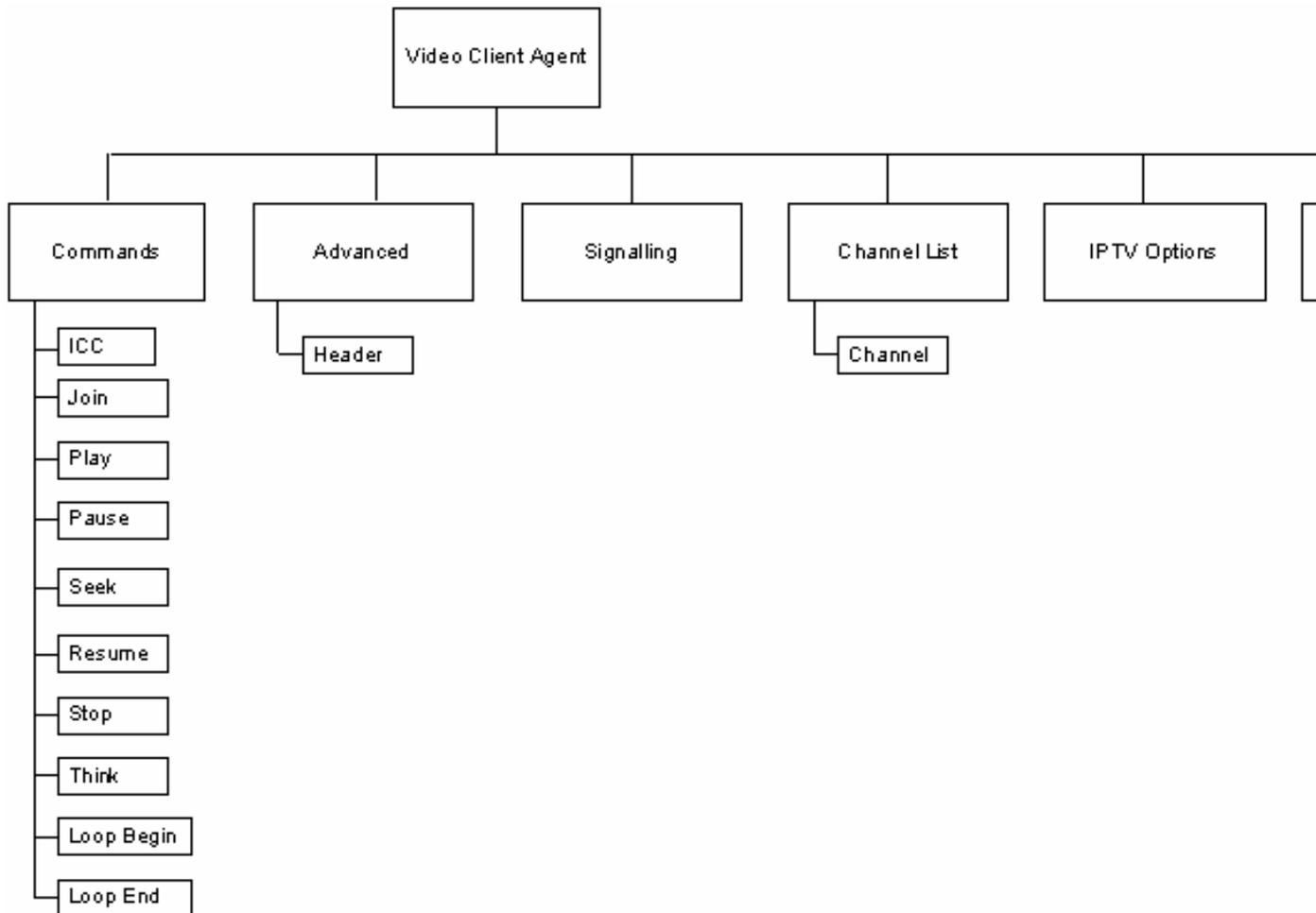
Objectives

The objectives (userObjective) you can set for Video are listed below. Test objectives are set in the `ixTimeline` object.

- `simulatedUsers`
- `streams`
- `connectionRate`
- `transactionRate`

Video Client API Structure

The figure below shows the structure of the video client API.



Video Client Commands

This section lists the video client commands.

Video Client Agent

The Video Client Agent command defines a simulated user viewing video clips from a video-on-demand (VOD) server or real-time streaming video from a broadcast-type video source. Refer to `Video Client Agent` for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
protocol	Protocol used by the agent.



Option	Description
type	Type of agent.
name	The name associated with this object, which must be set at object creation time .

Commands

This command defines the list of commands that the client sends to the server. Refer to `Commands` for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	Video command to be executed.

Advanced

This command defines the list the client's global configuration options. Refer to `Advanced` for a full description of this command. The most significant options of this command are:

Option	Description
enableEsm	Enables <code>esm</code> option.
esm	TCP Maximum Segment Size.
transport	Type of transport to be used.
enableTosRTSP	Enables TOS bits in RTSP packets.
type_of_service_for_rtsp	Specifies the TOS bit setting.
rtsp_header	Type of header used to identify the video player simulated by the Video client agent.
header_values	List of headers included with RTSP requests that the client sends to the server.
users_allowed	Number of users allowed.

Header

This command creates a list of RTSP headers to define a Video client as a custom video player. Refer to `Header` for a full description of this command. The most significant options of this command are:

Option	Description
name	Name of the header.
value	Value for header.

Signaling

This command configures the multicast signaling options. Refer to `Signaling` for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>enable_custom</code>	If enabled, the custom client profiles that have been configured will be used in a test.
<code>igmp_version</code>	Sets the version of IGMP used by the client.
<code>general_query_response_mode</code>	If <code>true</code> , the video client responds to General Query messages.
<code>unsolicited_response_mode</code>	If <code>true</code> , the video client automatically sends full IGMP membership messages at regular intervals without waiting for a query message.
<code>immediate_response</code>	If <code>true</code> , the video client will immediately respond to the Query by sending a Report.
<code>group_specific_query_response_mode</code>	If enabled, the client responds to group-specific Query messages.
<code>suppress_reports</code>	(IGMPv3 only) If <code>true</code> , the client allows its IGMPv3 MemberRecord to be "suppressed" by a membership report for version 2.
<code>report_frequency</code>	If <code>unsolicited_response_mode</code> is <code>true</code> , this option specifies the frequency (in seconds) at which unsolicited messages are generated.
<code>mld_version</code>	Version of the Multicast Listener Discovery (MLD) protocol used to listen for IPv6 multicast addresses. You can select MLDv1 or MLDv2. The <code>ip_version</code> has to be "IPv6" for MLD.
<code>client_mode</code>	Specifies the type of client: Video or IPTV.

Profiles

This command determines the channel switching behavior of the video client. Refer to `Profiles` for a full description of this command. The most significant options of this command are:

Option	Description
<code>name</code>	Sets the name of the profile table.
<code>percentage</code>	Percentage of video client that the profile will be applied to.
<code>duration_min</code>	Minimum length of time, in seconds, that users of this profile will view a channel.
<code>duration_max</code>	Maximum length of time, in seconds, that users of this profile will view a channel.

Option	Description
channel_switch_delay_min	Minimum length of time, in milliseconds, that users of this profile will pause before viewing a new channel.
channel_switch_delay_max	Maximum length of time, in milliseconds, that users of this profile will pause before viewing a new channel.

Stats

Configures the statistics that IxLoad gathers for the client's video streams. Refer to `Stats` for a full description of this command.

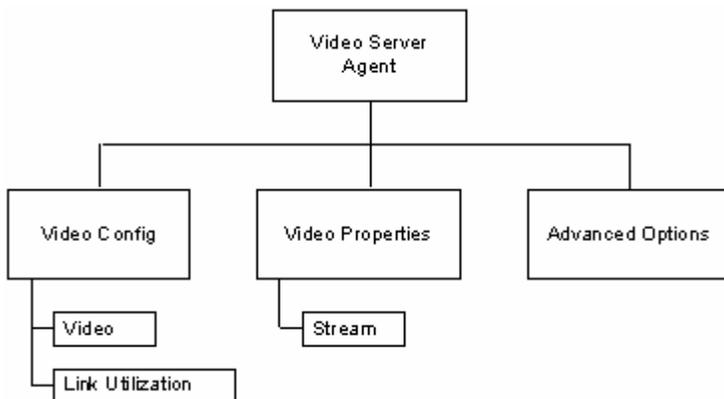
IPTV Options

Describes the options that are specific to the video client in IPTV mode. Refer to `IPTV Options` for a full description of this command. The most significant options of this command are:

Option	Description
iptv_switch_delay	If <code>iptv_switch_mode</code> is "2" then specify the fixed length of time here.
iptv_switch_mode	Selects how the IPTV client switches from the D server stream to the A server stream.

Video Server Commands

The diagram below shows the structure of the video server API.



Video Server Agent

The Video Server Agent command defines a simulated server offering video-on-demand (VOD) or real-time streaming video. Refer to `Video Server Agent` for a full description of this command. The most significant options of this command are listed below.

Option	Description
protocol	Protocol used by the agent.
type	Type of agent.
enable	Enables the use of this agent.
name	The name associated with this object, which must be set at object creation time.

Video Properties

This command adds a video stream. Refer to `Video Properties` for a full description of this command. The most significant options of this command are:

Option	Description
stream	This is a list of type <code>Stream</code> . The elements in this list comprise the list of streams available on the Video server.
name	Name of the video stream.
type	Type of the video stream.
stream_count	Number of instances of this stream that will be streamed out or hosted on the server.
starting_multicast_group_address	Specifies the address of the first multicast group that the channel will be available on.
addr_incr	Specifies the amount of increase in each multicast group address for the streams.
starting_dest_port	Specifies the first port number that the channel will be available on (multicast only).
dest_port_incr	Specifies the amount of increase in each port number for the streams (multicast only).
duration	If the stream <code>type</code> is VoD, this parameter specifies the duration of the video stream.



Stream

This command configures a video stream. Refer to `Stream` for a full description of this command. The most significant options of this command are:

Option	Description
name	Name of the video stream .
transport	Type of transport to be used.
content	Type of payload carried by stream .
filename	Name and path of the video file.
ip_bit_rate	Bit rate of the generated stream.
mpeg_type	MPEG version of video stream.
type	Type of the video stream .
stream_count	Specifies the number of instances of this stream that will be streamed out. or hosted by the server.
starting_multicast_group_addr	Specifies the address of the first multicast group that the channel will be available on.
addr_incr	Specifies the amount of increase in each multicast group address for the streams.
starting_dest_port	First port number that the channel will be available on (multionly).
dest_port_incr	Amount of increase in each port number for the streams (mulonly).
duration	Duration of the video stream.
tsperudp	Helps to configure the number of transport stream (TS) packpackaged into a single UDP packet.
enable_d_server_tos	Enables the Type of Service (ToS) bits.
d_server_tos_or_dscp	If <code>enable_d_server_tos</code> is set to 1, you can set the Type of Service (ToS) bits that will be set in this stream from the A server, D Server and V Server.

Advanced Options

This command sets the Video server agent's global configuration options. Refer to `Advanced Options` for a full description of this command. The most significant options of this command are:

Option	Description
listen_port	Port that RTSP server listens on for new connections.
enableEsm	Enables the use of ESM.
enableTosRTSP	Enables the setting of the TOS (Type of Service) bits in the IP header of the RTSP control packets.

Option	Description
enableTosData	Enables the setting of the TOS (Type of Service) bits in the IP header of the RTSP data packets.
esm	The ESM value.
type_of_service_for_rtsp	IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for RTSP control packets.
type_of_service_for_data	IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for RTSP data packets.
enable_d_server_tos	Enables the Type of Service (ToS) bits.
d_server_tos_or_dscp	If <code>enable_d_server_tos</code> is set to 1, you can set the Type of Service (ToS) bits that will be set in this stream from the A server, D Server and V Server.

Video Client Agent

Video Client Agent - configure an IPTV or video client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
option...]
$Activity_IPTV_VideoClient1 agent.config
```

DESCRIPTION

The Video Client Agent command defines a simulated user viewing video clips from a video-on-demand (VOD) server or real-time streaming video from a broadcast-type video source. A video client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `append` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`loopValue`

If this option is enabled (1), then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (0), then the client will progress through the command list only once, and then go idle. (Default = 0).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS



Note: For some of the per-stream and Video Quality monitoring statistics shown in the GUI, various scaling factors are applied to make the values easier to read. When you retrieve these statistics from the Tcl API, the values returned may be different from those shown in the GUI. The following statistics are affected:

- MDI-DF, MDI-DF-AVG, MDI-DF-MIN, and MDI-DF-MAX are in nanosec(ns) when retrieved from the Tcl API. In the GUI, they are displayed in milliseconds (ms).
 - Stream Bit Rate is returned in bits per second (bps) when retrieved from the Tcl API. In the GUI, it is displayed in kilobits per second (kbps).
 - When retrieved from the Tcl API, MOS_V, Degradation (Loss), Degradation (Discard), and Degradation (Video Codec) are scaled up by 256 compared to GUI. For example, the MOS score is displayed on a 0 - 5 scale in the GUI but is returned as a value in the range 0 - 1280 when retrieved from the Tcl API. The Degradation statistics are displayed in the GUI as a percentage. When retrieving them from the Tcl API, divide the returned value by 256 to get the percentage.
 - When retrieved from the Tcl API, VSTQ is scaled by 2 compared to the value in the GUI. Divide the returned value by 2 to get the actual value.
-

EXAMPLE

```

set Traffic1_Network1 [::IxLoad new ixNetTraffic]

#####

# Activity IPTV_VideoClient1 of NetTraffic Traffic1@Network1

#####

set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem \
    -protocolAndType                "Video Client" ]

#####

# Timeline1 for activities IPTV_VideoClient1

#####

set Timeline1 [::IxLoad new ixTimeline]

$Timeline1 config \
    -rampUpValue                    1 \
    -rampUpType                      0 \
    -offlineTime                    0 \
    -rampDownTime                   20 \
    -standbyTime                     0 \

```



```
-iterations 1 \  
-rampUpInterval 1 \  
-sustainTime 20 \  
-timelineType 0 \  
-name "Timeline1"  
$Activity_IPTV_VideoClient1 config \  
-enable true \  
-name "IPTV_VideoClient1" \  
-enableConstraint false \  
-userObjectiveValue 100 \  
-constraintValue 100 \  
-userObjectiveType "simulatedUsers" \  
-timeline $Timeline1  
$Activity_IPTV_VideoClient1 agent.config \  
-enable true \  
-name "IPTV_VideoClient1"  
$Activity_IPTV_VideoClient1 agent.pm.signalling.config \  
-general_query_response_mode true \  
-unsolicited_response_mode false \  
-report_frequency 60 \  
-igmp_version "IGMP v3" \  
-mld_version "MLD v2" \  
-router_alert true \  
-group_specific_query_response_mode true \  
-enable_custom false \  
-suppress_reports true \  
-ip_version "IPv4" \  
-immediate_response false \  

```

```
-client_mode                                0

$Activity_IPTV_VideoClient1 agent.pm.signalling.profile_table.clear

$Activity_IPTV_VideoClient1 agent.pm.signalling.profile_table.appendItem \

  -id                                         "ProfileTable" \
  -name                                       "Fast Switching" \
  -num_profiles                              1 \
  -channel_switch_delay_max                  0 \
  -duration_max                              30 \
  -duration_min                              10 \
  -percentage                                50.0 \
  -channel_switch_delay_min                  0

$Activity_IPTV_VideoClient1 agent.pm.signalling.profile_table.appendItem \

  -id                                         "ProfileTable" \
  -name                                       "Slow Switching" \
  -num_profiles                              1 \
  -channel_switch_delay_max                  0 \
  -duration_max                              300 \
  -duration_min                              100 \
  -percentage                                50.0 \
  -channel_switch_delay_min                  0

$Activity_IPTV_VideoClient1 agent.pm.stats.config \

  -MinDelay                                  20 \
  -MaxDelay                                  80 \
  -enableFrameStats                          false \
  -qualityLimit                              0 \
  -IgnoreLoss                                false \
  -frameLimit                                0 \
  -JBEMode                                   0 \
```



```
-enableVQmonStats                false \  
-totalLimit                       0 \  
-updateInterval                   2000 \  
-NomDelay                          20 \  
-bitrateLimit                     0  
  
$Activity_IPTV_VideoClient1 agent.pm.iptv_options.config \  
-iptv_switch_delay                1 \  
-iptv_switch_mode                 0  
  
$Activity_IPTV_VideoClient1 agent.pm.advanced.config \  
-vlan_priority                    0 \  
-type_of_service_for_rtsp         "Best Effort (0x0)" \  
-rtsp_header                      "Real Player" \  
-enableTosRTSP                   false \  
-implicitLoopCheck               true \  
-enableEsm                       false \  
-users_allowed                   1 \  
-esm                              1460 \  
-enableVlanPriority               false \  
-transport                        1  
  
$Activity_IPTV_VideoClient1 agent.pm.advanced.header_values.clear  
  
$Activity_IPTV_VideoClient1 agent.pm.advanced.header_values.appendItem \  
-id                               "Header" \  
-name                             "User-Agent" \  
-value                             "RealMedia Player  
(HelixDNAClient)"  
  
$Activity_IPTV_VideoClient1 agent.pm.ipHistory.clear  
  
$Activity_IPTV_VideoClient1 agent.pm.channelSrcHistory.clear  
  
$Activity_IPTV_VideoClient1 agent.pm.channelSrcHistory.appendItem \  
-id                               "channelSrc" \  

```

```
-name "ANY"

$Activity_IPTV_VideoClient1 agent.pm.UrlHistory.clear

$Activity_IPTV_VideoClient1 agent.pm.predefined_tos.clear

$Activity_IPTV_VideoClient1 agent.pm.predefined_tos.appendItem \
  -id "TypeOfService" \
  -tos_val "Best Effort (0x0)"

$Activity_IPTV_VideoClient1 agent.pm.predefined_tos.appendItem \
  -id "TypeOfService" \
  -tos_val "Class 1 (0x20)"

$Activity_IPTV_VideoClient1 agent.pm.predefined_tos.appendItem \
  -id "TypeOfService" \
  -tos_val "Class 2 (0x40)"

$Activity_IPTV_VideoClient1 agent.pm.predefined_tos.appendItem \
  -id "TypeOfService" \
  -tos_val "Class 3 (0x60)"

$Activity_IPTV_VideoClient1 agent.pm.predefined_tos.appendItem \
  -id "TypeOfService" \
  -tos_val "Class 4 (0x80)"

$Activity_IPTV_VideoClient1 agent.pm.predefined_tos.appendItem \
  -id "TypeOfService" \
  -tos_val "Express Forwarding (0xA0)"

$Activity_IPTV_VideoClient1 agent.pm.predefined_tos.appendItem \
  -id "TypeOfService" \
  -tos_val "Control (0xC0)"

$Activity_IPTV_VideoClient1 agent.pm.commands.clear

$Activity_IPTV_VideoClient1 agent.pm.commands.appendItem \
  -id "JoinCommand" \
  -destination_server_activity "Traffic2_IPTV_VideoServer1:0" \
```



```
-group_address_step           "0.0.0.1" \  
-channel_switch_mode         "Concurrent" \  
-start_group_address_sym     "" \  
-sigma                       1 \  
-start_group_address         "" \  
-channel_switch_delay_max    0 \  
-mu                          1 \  
-varLambda                   1 \  
-duration_max                10 \  
-duration_min                10 \  
-watch_count                 1 \  
-group_address_count         1 \  
-source_address              "ANY" \  
-concurrent_channels         1 \  
-channel_switch_delay_min    0
```

SEE ALSO

`ixNetTraffic`

Commands

Commands—Creates the list of Video commands that the client will send to a Video server.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
option...]
$Activity_IPTV_VideoClient1 agent.pm.commands.appendItem
```

DESCRIPTION

A command is added to the Commands object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

Video command to be executed. One of the following:

Command	Description
ICCCCommand	The Instant Channel Change (ICCCCommand) plays IPTV streams for a fixed duration and then switches to new streams.
JoinCommand	The JOIN command sends an IGMP JOIN message to one or more IGMP servers in order to play their broadcast channels. The client can join multiple multicast groups in sequence or at random intervals. After joining a multicast group, the client plays each channel for a specific duration. After the duration has expired, the client sends the IGMP LEAVE command for that channel. The client plays all the channels specified in the JOIN command, then it moves on to the next command in the command list. Once the test enters the rampdown phase, the client does not join any new channels.
PlayCommand	The PLAY command plays the VoD video stream from a video server. The PLAY command performs the following RTSP commands in order: DESCRIBE SETUP PLAY

PlayMediaCommand	<p>The PLAYMEDIA command supports symbolic destination for Video Server in Video Client. The <code>Media / URL</code> in PLAY command and <code>start_group_address</code> for JOIN command are resources that get populated when the server activity is selected.</p> <p>The server activity can be <code>None</code>, when IxLoad video client is run against an external video server. The PLAYMEDIA command performs the following RTSP commands, in order:</p> <p>DESCRIBE</p> <p>SETUP</p> <p>PLAY</p>
PlayStaticCommand	<p>PlayStatic command plays a video stream whose description is sourced from the PlayStatic command itself. PlayStatic is intended for use with RTSP servers that do not implement the RTSP Describe command, which is normally the source of a video stream's description.</p>
KeepAliveCommand	<p>KeepAliveCommand periodically sends an empty RTSP GET_PARAMETER command to the server so that the server does not assume that the client is inactive and then tears down the connection.</p> <p>Although you can add a KeepAliveCommand to any position in a command list, IxLoad will only send a KeepAliveCommand if a stream has been setup and is active. Typically, KeepAliveCommnds should only be added after PLAY, PAUSE and similar commands. KeepAliveCommand can only be used for VoD (unicast) streams.</p>
PauseCommand	<p>The PAUSE command sends an RTSP PAUSE command to pause playback of the current VoD video stream. To resume playback, use the RESUME command.</p>
SeekCommand	<p>The SEEK command jumps to a location in the media stream and plays from that location. The SEEK command must be preceded by a PAUSE command.</p> <p>The IxLoad video server only supports the SEEK command for transport streams (MPEG-2 TS files). It does not support the SEEK command for MPEG-4, H.264, or VC1 streams.</p>
ResumeCommand	<p>The RESUME command sends an RTSP PLAY command to resume playback of a paused VoD video stream.</p>
StopCommand	<p>The STOP command sends an RTSP STOP command to stop playback of the current VoD video stream.</p>
ThinkCommand	<p>The {THINK} command causes the client to become inactive.</p>

PassiveCommand	<p>The PassiveCommand passively monitors unicast or multicast audio and video streams and records statistics for them.</p> <p>When used in conjunction with an AFM module, this command enables you to use the AFM module as a tap, and gather statistics such as MDI or video quality metrics at various points within a network. Statistics are recorded both globally and on a per-stream basis. Quality metrics are recorded for both video and audio streams.</p>
LoopBeginCommand	<p>The {Loop Begin} command is an IxLoad command that you can add to the Command List to cause the commands between it and the {Loop End} to be executed a specified number of times.</p>
LoopEndCommand	<p>{Loop End} ends the list of commands that will be executed by the preceding {Loop Begin} command.</p>
RTSP Commands	<p>In addition to the high-level commands that simplify testing video, the IxLoad IPTV/Video client also allows you to configure and send individual RTSP commands. This enables you to test video using low-level RTSP commands, and take advantage of the quality metrics other statistics that are available in the IxLoad IPTV/Video client.</p>
DescribeCommand	<p>Retrieves the description of a presentation or media object identified by the URL in the <code>media</code> option. The server responds with a description of the requested resource.</p>
RTSPSetupCommand	<p>Specifies the transport mechanism to be used for the streamed media. A client can issue a SETUP request for a stream that is already playing to change transport parameters, if the server allows it. Specify the transport mechanism in the <code>arguments</code> option.</p>
RTSPPlayCommand	<p>Tells the server to start playback using the mechanism specified by a previous SETUP command. Specify the stream in the <code>media</code> option, and the playback duration in the <code>arguments</code> option.</p>
RTSPPauseCommand	<p>Causes the stream playback to be temporarily halted. If you specify a stream in the <code>media</code> option, only playback of that stream is halted. If you do not specify a stream, all streams are paused.</p>
RTSPSetParamCommand	<p>This method requests to set the value of a parameter for a stream specified by the URL. Specify the name of this parameter in the <code>arg</code> option. IxLoad Video Server does not support RTSP SET_PARAMETER command.</p>
RTSPGetParamCommand	<p>Retrieves the current value of a parameter from the server. If you issue the GET_PARAMETER with no arguments, it functions as a keep-alive to prevent the server from closing the connection when long presentations are playing. IxLoad Video Server does not support RTSP GET_PARAMETER command.</p>
RTSPTeardownCommand	<p>Stops the stream delivery for the URL listed in the <code>media</code> option, freeing the resources associated with it. After issuing the TEARDOWN command, the RTSP session identifier associated with the session is no longer valid.</p>

Arguments for `id = ICCCommand` (IPTV mode only)

The Instant Channel Change (ICC) command plays IPTV streams for a fixed duration and then switches to new streams. You can only use the ICC command in Activities running over IPv4 networks.

`destination_server_activity`

Video server hosting the media that the client will play.

- **IxLoad server:** If you are using an IxLoad video server, specify the server address.
- **External server:** If you are using an external video server, specify `None`.

Default = `None`.

`group_address_step`

Specifies the amount of increase in the channel number (A server address). See the description of the `group_address_count` for more information.

`channel_switch_mode`

Specifies the order in which the client joins the multicast groups in the Channel List to view the channels.

sequential: The client plays the channels in the Channel List one after the other, in order based on their address, starting with the `start_group_address`. After the Channel Watch Duration expires, the client sends an IGMP LEAVE for the channel being viewed. The client waits for the duration specified by Channel Switch Delay duration before joining the next group to view the next channel.

poisson: The client plays the channel in an order that follows a Poisson distribution. Configure the `watch_count`, then set the `varLambda` value for the Poisson distribution.

normal: The client plays the channel in an order that follows a Normal distribution. Configure the `watch_count`, then set the `mu` and `sigma` values for the Normal distribution.

unique: Each user starts from a different channel, and plays each channel in numerical order. There are no configuration options for a Unique sequence. The number of channels played is automatically set to the same value as the `Count` parameter.

custom: The client plays the channels following an existing profile, but in a sequence that you specify.

Default = `"sequential"`.

`start_group_address_sym`

The address of the video server hosting the media that the client will play.

`sigma`

In a Normal distribution, `m` (`mu`) is the location parameter and `s` (`sigma`) is the scale parameter. In IxLoad, `mu` is the mean average channel number that the distribution will be clustered around. As

channel numbers increase or decrease away from the mu value, they are less likely to be watched. Sigma determines the width of the distribution, the number of channels that may be watched.

`start_group_address`

Specifies the first group address.

`channel_switch_delay_max`

If you want the client to pause before switching to the next channel, specify the maximum length of the delay here.

`da_switchover_delay`

If you want the client to pause before switching to the next channel, specify the length of the delay here. You can specify a fixed-length delay (same delay before playing every channel) or a random-length delay (different delay before playing every channel).

`serverIP`

IP address of the D server.

`mu`

In a Normal distribution, m (mu) is the location parameter and s (sigma) is the scale parameter. In IxLoad, mu is the mean average channel number that the distribution will be clustered around. As channel numbers increase or decrease away from the mu value, they are less likely to be watched. Sigma determines the width of the distribution, the number of channels that may be watched.

`varLambda`

A Poisson distribution models the number of events that occur within a given time interval. In a Poisson distribution, λ (lambda) is the shape parameter, which indicates the average number of events in the given time interval. When used for IxLoad, the lambda value is the mean average channel number that the distribution will be clustered around. The bell-curved shape of the distribution ensures that the most-watched channels will be those closest to the mean (the lambda), with channels less likely to be watched as channel numbers move away from the lambda value.

`urls`

IPTV (multicast) streams to play from the D server.

You can enter sequence generators in this field to generate URLs for more than one stream.

The number of D server URLs must match the A server Channel Count.

`duration_max`



Maximum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

`duration_min`

Minimum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

`watch_count`

Number of channels that will be viewed as a part of this Join command.

If you set the `channel_switch_mode` to Normal or Poisson, you can configure the value here. For the other distribution options, this option is read-only and automatically set to the same value as the Channel Count parameter.

`group_address_count`

Specifies the number of additional channels, if you want the client to play more than one channel (stream).

`source_address`

Configures the source address (the IP address of the A server), if the client uses IGMP v3 and you want to send a source-specific JOIN to a multicast group.

If you specify ANY, the client does not specify a particular source address.

Default = "ANY".

`concurrent_channels`

Specifies the number of channels that each client plays at one time. You can specify up to four channels to play at one time.

Default = 1.

`channel_switch_delay_min`

If you want the client to pause before switching to the next channel, specify the minimum length of the delay here.

Arguments for `id = JoinCommand`

`start_group_address`

IP address of the first multicast group that the client will join.

group_address_count

Number of multicast groups that the client will join. Minimum = "1," Maximum = "1,000." (Default = "1").

group_address_step

If the client will join more than one multicast group, enter the amount of increase in the multicast group address. Minimum = "1." (Default = "1").

source_address

If the client uses IGMP v3 and you want the JOIN request to specify a source for the video stream, configure the source address in this field. If you specify ANY, the client does not specify a particular source address. (Default = "ANY")

duration_max

Maximum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

duration_min

Minimum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

concurrent_channels

If channel_switch_mode is set to Concurrent, this parameter specifies the number of channels that the client plays at one time. Minimum = "1," Maximum = "5." (Default = "1").

channel_switch_mode

Order in which the client joins the multicast groups in the Channel List to play the channels. The choices are:

Mode	Description
sequential	The client plays the channels in the Channel List one after the other, in order based on their address, starting with the Starting Group Address. After the Channel Watch Duration expires, the client sends an IGMP LEAVE for the channel being watched. The client waits for the duration specified by Channel Switch Delay duration before joining the next group to play the next channel.
random	The client plays the channels in the Channel List randomly.
concurrent	(default) The client plays the channels in the Channel List in order, based on their address. Specify the number of channels that it can play at any one time in the Concurrent Channels field.



poisson	The client plays the channel in an order that follows a Poisson distribution. For Poisson distribution, the <code>channel_switch_mode</code> is set to "Poisson". New attributes used are: <code>watch_count</code> and <code>varLambda</code> .
normal	The client plays the channel in an order that follows a Normal distribution. For Normal distribution, the <code>channel_switch_mode</code> is set to "Normal". New attributes used are: <code>mu</code> , <code>sigma</code> and <code>watch_count</code> .
unique	Each user starts from a different channel, and plays each channel in numerical order. There are no configuration options for a Unique sequence. The number of channels played is automatically set to the same value as the <code>Count</code> parameter.
custom	The client plays the channels following an existing profile, but in a sequence that you specify.

`channel_switch_delay_min`

Minimum length of the time, in milliseconds, that the client will pause before playing the next channel on the server. Minimum = "0," Maximum = "2,147,483,647." (Default = "0").

`channel_switch_delay_max`

Maximum length of the time, in milliseconds, that the client will pause before playing the next channel on the server. Minimum = "0," Maximum = "2,147,483,647." (Default = "0").

Arguments for `id = PlayCommand`

`serverIP`

Video server that hosts the video stream to be played.

`media`

Video stream to be played. You can include sequence generators in this field to automatically generate unique requests from simulated users. For information on how to use sequence generators, see the section on Using Automatic Sequence Generators. For example:

```
-media "Stream\[1-\]"
```

`duration_max`

Maximum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

`duration_min`

Minimum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

Arguments for id = PlayMediaCommand

`symServerIP`

Video server that hosts the video stream to be played.

`media`

Video stream to be played. You can include sequence generators in this field to automatically generate unique requests from simulated users. For information on how to use sequence generators, see the section on Using Automatic Sequence Generators. For example:

```
-media                "Stream\[1-\]"
```

`duration`

Length of time (in seconds) to play the video stream. Minimum = "1," Maximum = "2,147,483."
(Default = "1").

Arguments for id = PlayStaticCommand

`symServerIP`

Video server that hosts the video stream to be played.

`media`

Video stream to be played. You can include sequence generators in this field to automatically generate unique requests from simulated users. For information on how to use sequence generators, see the section on Using Automatic Sequence Generators.

`duration`

Length of time (in seconds) to play the video stream. Minimum = "1," Maximum = "2,147,483."
(Default = "1").

`destination_server_activity`

Represents the symbolic destination of the server.

`serverIP`

Video server that hosts the video stream to be played.

Arguments for id = PlayMediaStaticCommand

cmdName

Name of the command added to the command list. Default = "PlayMediaStaticCommand *n*" where *n* is the command's position in the command list.

commandType

Command type. Default = "PlayMediaStaticCommand"

symServerIP

Video server that hosts the video stream to be played. Default = "None".

media

Video stream to be played. You can include sequence generators in this field to automatically generate unique requests from simulated users. For information on how to use sequence generators, see the section on Using Automatic Sequence Generators.

duration_max

Maximum length of time (in seconds) to play the video stream. Minimum = "1," Maximum = "2,147,483."
(Default = "1").

duration_min

Minimum length of time (in seconds) to play the video stream. Minimum = "1," Maximum = "2,147,483."
(Default = "1").

seekTo

Reserved. Default = -1

serverIP

Video server that hosts the video stream to be played. This can be an IP address or a symbolic destination (IxLoad server). Default = ""(none)

Arguments for id = KeepAliveCommand

count

Number of {KeepAlive} messages to be sent.

`min_freq`

The minimum time, in milliseconds, that can elapse before the client sends the next {KeepAlive} message.

`max_freq`

The maximum time, in milliseconds, that can elapse before the client sends the next {KeepAlive} message.

Arguments for id = PauseCommand

None.

Arguments for id = SeekCommand

`duration_max`

Maximum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

`duration_min`

Minimum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

Arguments for id = ResumeCommand

`seekTo`

Number of seconds, measured from the start of the stream, to jump to and start playing from. Minimum = "1," Maximum = "2,147,483." (Default = "1").

`duration_max`

Maximum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

`duration_min`

Minimum length of time, in seconds, that users of this profile will view a channel. Minimum = "1," Maximum = "2,147,483." (Default = "10").

Arguments for id = StopCommand

None.

Arguments for id = ThinkCommand

minimumInterval

Minimum length of the time, in milliseconds, that the client will pause before playing the next channel on the server. Minimum = "1,000," Maximum = "2,147,483,647." (Default = "1,000").

maximumInterval

Maximum length of the time, in milliseconds, that the client will pause before playing the next channel on the server. Minimum = "1000," Maximum = "2,147,483,647." (Default = "1,000").

Arguments for id = PassiveCommand

enableUnicast

Enables an Unicast or Multicast stream that can be monitored. (Default = "0"). If you enter 1 that is Unicast, then a new rule needs to be configured. The rule is explained below.

Configuring Rule for Stream

```
$Activity_IPTV_VideoClient1 agent.pm.commands(0).rule.appendItem\  
-id "Rule" \  
-clock_rate 90000 \  
-codec "H264" \  
-value "10000-65535" \  
-rtp_pt 96
```

id

The name of the rule.

clock_rate

Specifies the stream's bit rate. (Default = "90000").

codec

Indicates the codec used on the stream. (Default = "H264").

value

Indicates the port range used by the stream. (Default = "10000-65535").

rtp_pt

Sets the RTP Payload type to a default value based on the `codec` value. The values are:

Codec	Default RTP Payload Type value
MPEG-TS	33
H264	96 (Default)
MPEG4 Part 2	97
VC1	98

Arguments for id = LoopBeginCommand

LoopCount

Number of times to iterate. Value '0' treated as infinity. Minimum = "0," Maxi= "2,147,483,647."
(Default = "5").

Arguments for id = LoopEndCommand

None.

Arguments for id = DescribeCommand

destination_server_activity

The Video server that the client will send the media URL described in `media`. The media URL identifies the set of stream to be controlled. Specify the destination as follows:

- If the destination is a real RTSP server, enter the server's host name or IP address. By default, the request will be sent to port 554. If the server is listening on a different port, specify the port number after the host name or IP address as follows: `server:port`.
- If the destination is an IxLoad RTSP server Activity, select the Activity.

(Default = "None").

serverIP

The IP address of the server.

media

The presentation URL sent to the server. The presentation URL identifies the stream to be controlled. Media names may only contain letters, numbers, and the special symbols ':', ',', '_', '/' and '-'. (Default = "None").

Arguments for id = RTSPPlayCommand

duration_max

Maximum length of time, in seconds, that users of this profile will view a channel.



duration_min

Minimum length of time, in seconds, that users of this profile will view a channel.

Arguments for id = RTSPSetParamCommand

content

Specifies the value of the content.

contentType

Specifies the parameter of the content.

Arguments for id = RTSPGetParamCommand

content

Specifies the value of the content.

contentType

Specifies the parameter of the content.

EXAMPLE

```
$Activity_IPTV_VideoClient1 agent.pm.commands.appendItem \  
-id "JoinCommand" \  
-destination_server_activity "Traffic2_IPTV_VideoServer1:0" \  
-group_address_step "0.0.0.1" \  
-channel_switch_mode "Concurrent" \  
-start_group_address_sym "" \  
-sigma 1 \  
-start_group_address "" \  
-channel_switch_delay_max 0 \  
-mu 1 \  
-varLambda 1 \  
-duration_max 10 \  
-duration_min 10 \  

```

```
-watch_count                1 \  
-group_address_count        1 \  
-source_address             "ANY" \  
-concurrent_channels        1 \  
-channel_switch_delay_min   0  
  
$Activity_IPTV_VideoClient1 agent.pm.commands.appendItem \  
-id                         "PassiveCommand" \  
-enableUnicast              0  
  
$Activity_IPTV_VideoClient1 agent.pm.commands.appendItem \  
-id                         "DescribeCommand" \  
-destination_server_activity "Traffic2_IPTV_VideoServer1:554"  
\  
-serverIP                   "198.18.0.101" \  
-media                       "Stream0"  
  
$Activity_IPTV_VideoClient1 agent.pm.commands.appendItem \  
-id                         "RTSPSetupCommand"  
  
$Activity_IPTV_VideoClient1 agent.pm.commands.appendItem \  
-id                         "RTSPPlayCommand" \  
-duration                   20 \  
-seekTo                     -1  
  
$Activity_IPTV_VideoClient1 agent.pm.commands.appendItem \  
-id                         "RTSPTeardownCommand"
```

SEE ALSO

Video Client Agent



Advanced

Advanced—Sets the Video client agent's global configuration options for unicast traffic.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
option...]
$Activity_IPTV_VideoClient1 agent.pm.advanced.config
```

DESCRIPTION

A Video client's advanced configuration options are set by modifying the options of the `pm.advanced` option of the `Video Client Agent` object using its `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`enableEsm`

If `true`, the use of the `esm` option is enabled. (Default = `false`).

`esm`

If `enableEsm` is `true`, this option specifies the TCP Maximum Segment Size in the `MSS (TX)` field. Otherwise, the TCP Maximum Segment Size is 1,460 bytes. (Default = 1,460).

`transport`

Transport protocol used to send the video stream. It applies only to `VoD`.

Value	Description
0	RTP over UDP
1	(default) UDP

`enableTosRTSP`

Enables the setting of the TOS (Type of Service) bits in the IP header of the RTSP packets.

Value	Description
0	(default) TOS bits not enabled.
1	TOS bits enabled.

type_of_service_for_rtsp

If `enableTosRTSP` is true, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

Value	Description
"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control

rtsp_header

Type of header used to identify the video player simulated by the Video client agent. The choices are:

Value	Description
Windows Media Player	Windows Media Player 9.0
Real Player	(default) Real Networks RealPlayer
Quick Time	Apple Quick Time 6.5
Custom	Custom player. Use the options to configure the headers that will identify this client.

header_values

List of headers included with RTSP requests that the client sends to the server. sent to the `rtsp_header` is set to `Custom`, use this option to define the capabilities of the custom video client. This list is of type `Header`; items are added to the list via the `appendItem` subcommand. Each element of the list must be of the form "name: value" without any spaces in the key. (Default = None).

disableStreamStats

Disables collection of stream-related statistics to reduce memory usage. Values = 1 (True), 0 = False (Default).

max_tracks_per_stream

Maximum number of tracks (RTP streams) that the client should expect in each RTSP stream. Values = Min="1", Max="500", Default="2".



`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = `false`).

`vlanPriority`

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.

`enableCustomSetup`

This enables or disables the entry of parameters specified in the `Transport:` line of the RTSP SETUP message. You can use these parameters to set or enable addiRTSP transport options on the server. Default = `false`.

`customSetup`

If `enableCustomSETUPtransportParam` is `false`, then the `Transport:` line contains the following data, which is mandatory for RTSP:

Transport protocol, connection type (unicast or multicast), and client IP port range used for the transport protocol. For example:

```
RTP/AVP;unicast;client_port=35246-35247
```

If `enableCustomSETUPtransportParam` is `true`, then IxLoad appends a semi-colon (;) to the mandatory data on `Transport:` line, and then appends the custom data in the field.

For example, if you specify the string `mode=PLAY`, the `Transport:` line will contain the following string:

```
RTP/AVP;unicast;client_port=35246-35247;mode=PLAY
```

`enable_custom_protocol`

If `true`, a user-defined name is used to identify a protocol instead of the default. Specify the name using the `custom_protocol_name` option. Default = `false`.

`custom_protocol_name`

If `enable_custom_protocol` is `true`, this option is the name used to identify a protocol instead of the standard name. Default = "MP2T".

`enable_custom_profile`

If `true`, a user-defined name is used to identify a profile instead of the default. Specify the name using the `custom_profile_name` option. Default = `false`.

`custom_profile_name`

If `enable_custom_profile` is `true`, this option is the name used to identify an A/V sync profile instead of the standard name. Default = "H2221".

`rtspProxyEnable`

Enables use of an RTSP RTSP proxy.

`rtspProxyIp`

If `enableRtspProxy` is `true`, specify the RTSP proxy IP address.

`rtspProxyPort`

If `enableRtspProxy` is `true`, specify the RTSP proxy port number.

`followRtspRedirects`

If enabled, the client follows RTSP redirect responses from the server. Default = `false`.

`rtcp_enable`

If `True`, the RTCP port number is included in the SDP description. Values = 1 (`True`), 0 (`False` (default)).

`enable_async_tearardown`

If `True`, playback is stopped when the client receives a request header that contains a specific text sub-string. Values = 1 (`True`), 0 (`False` (default)).

`async_tearardown_hdr_val`

If `enable_async_tearardown` is `True`, this option specifies the header sub-string that will stop playback. Default = "".

`enable_graceful_rampdown`

If `True`, the test is stopped by moving to the Ramp-down phase and sessions are torn down gracefully. If `False`, traffic is stopped as soon as possible, which may leave sessions up on the DUT. Default = `"false"`.

`enable_hwacc`

If `True`, hardware acceleration is used. Default = `"false"`.

EXAMPLE

```
$Activity_IPTV_VideoClient1 agent.pm.advanced.config \  
-followRtspRedirect           true \  
-vlan_priority                0 \  
-type_of_service_for_rtsp    "Best Effort (0x0)" \  
-rtsp_header                  "Real Player" \  
-enableTosRTSP                false \  
-implicitLoopCheck           true \  
-rtspProxyEnable              true \  
-CustomSetup                  "mode=PLAY" \  
-enableCustomSetup            true \  
-enableEsm                    false \  
-users_allowed                1 \  
-rtspProxyIp                  "0.0.0.0" \  
-rtspProxyPort                "554" \  
-esm                           1460 \  
-enableVlanPriority            false \  
-transport                     1
```

SEE ALSO

Video Client Agent, Header

Header

Header—Creates a list of RTSP headers to define a Video client as a custom video player.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
option...]
$Activity_IPTV_VideoClient1 agent.pm.advanced.header_values.appendItem
```

DESCRIPTION

If the Advanced option `rtsp_header` is set to `Custom`, use `Header` to create the name = value pairs that will form the header that the Video client agent sends to the server.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

name

Name of the header. RFC 2326 defines the RTSP headers. (Default = "").

value

Value for header. (Default = "").

EXAMPLE

```
$Activity_IPTV_VideoClient1 agent.pm.advanced.header_values.appendItem
  -name "Cache-Control" \
  -value "no-cache"
```

SEE ALSO

Advanced



Signaling

Signaling—Configures the multicast signaling options.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
option...]
$Activity_IPTV_VideoClient1 agent.pm.signalling.config
```

DESCRIPTION

A Video client's Signaling options are set by modifying the options of the `pm.Signalling.config` option of the `Video Client Agent` object.

SUBCOMMANDS

None.

OPTIONS

`enable_custom`

If enabled, the custom client profiles that have been configured will be used in a test. The duration and `channel_switch_mode` configured for individual JOIN commands (`Arguments for id = JoinCommand` on page 18-35) will not apply. (Default = "0").

`igmp_version`

Sets the version of IGMP used by the client. The choices are:

Value	Description
"IGMPv1"	IGMP version 1. Note: IGMP v1 requires IPv4 (see the <code>ip_version</code> option)
"IGMPv2"	IGMP version 2.
"IGMPv3"	(default) IGMP version 3.

`ip_version`

Sets the IP version used for multicast addresses. If multicast addresses are in IPv4 format, and you can select the `igmp_version`. If multicast addresses are in IPv6 format, and you can select the `mld_version`.

`general_query_response_mode`

If `true`, the video client responds to General Query messages.

Value	Description
0	Client does not respond to General Query messages.
1	(default) Client responds to General Query messages.

`unsolicited_response_mode`

If `true`, the video client automatically sends full IGMP membership messages at regular intervals without waiting for a query message. In the Report Interval Field, specify the frequency, in seconds, at which unsolicited messages are generated.

Value	Description
0	(default) Client does not send unsolicited IGMP membership messages.
1	Client sends unsolicited IGMP membership messages.

`immediate_response`

If `true`, the video client will ignore the value specified in the Maximum Response Delay in the Membership Query message, assume that the Delay is always zero (0) seconds, and immediately respond to the Query by sending a Report.

Value	Description
0	(default) Client does not immediately respond to a query with a report.
1	Client immediately responds to a query with a report.

`group_specific_query_response_mode`

If enabled, the client responds to group-specific Query messages. A group-specific Query message is sent by a multicast router so it can learn about the multireception state of one multicast address, for each of the neighboring interfaces, for example, when a member leaves a group.

Value	Description
0	(default) Client does not respond to group-specific queries.
1	Client responds to group-specific queries.

`mld_version`

Version of the Multicast Listener Discovery (MLD) protocol used to listen for IPv6 multicast addresses. You can select MLDv1 or MLDv2.

The `ip_version` has to be "IPv6" for MLD.

`suppress_reports`



(IGMPv3 only) If `true`, the client allows its IGMPv3 Membership Record to be “suppressed” by a membership report for version 2. The suppression will only be for group reports received from another port.

Value	Description
0	Client does not allow its membership record to be suppressed.
1	(default) Client allows its membership record to be suppressed.

`report_frequency`

If `unsolicited_response_mode` is `true`, this option specifies the frequency (in seconds) at which unsolicited messages are generated. (Default = "30").

`parallel_multicast_vod`

If `true`, simulated users can watch a VoD stream and one or more multicast streams simultaneously. Values = 1 (True), 0 (False, default)

`client_mode`

Specifies whether the client is a video client (0) or IPTV client (1). Default = 0.

EXAMPLE

```
$Activity_IPTV_VideoClient1 agent.pm.signalling.config \  
-general_query_response_mode      true \  
-unsolicited_response_mode        false \  
-report_frequency                  60 \  
-igmp_version                       "IGMP v3" \  
-mld_version                        "MLD v2" \  
-router_alert                       true \  
-group_specific_query_response_mode true \  
-enable_custom                     false \  
-suppress_reports                  true \  
-ip_version                         "IPv4" \  
-immediate_response                false \  
-client_mode                        0
```

SEE ALSO

Video Client Agent



Profiles

Profiles—Determines the channel switching behavior of the video client.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
option...]
$Activity_IPTV_VideoClient1 agent.pm.signalling.profile_table.appendItem
```

DESCRIPTION

If custom profiles is enabled, the channel watch `duration` (the length of time a channel is viewed) and the `channel_switch_mode` (how quickly the simulated user switches from to a different channel) for all the users' JOIN commands are controlled by the profiles.

A Video client's Profile table are set by modifying the options of the `pm.Signaloption` of the `Video Client Agent` object using its `appendItem`.

OPTIONS

`name`

This is the name of the profile table.

If enabled, the custom client profiles that have been configured will be used in a test. The duration and `channel_switch_mode` configured for individual JOIN commands (`Arguments for id = JoinCommand` on page 18-35) will not apply. (Default = "0").

`num_profiles`

This indicates the number of profiles to be added with the same parameters.

`percentage`

Percentage of video clients that the profile will be applied to. The percentages of all profiles must add up to 100.

The profile table is populated by default with a couple of profiles. If you want to declare a custom profile, you need to clear the table with the following com

```
$cInt_traffic agentList(0).pm.signalling.profile_table.clear
```

If you donot clear the table before you start adding profiles, you will get an exception saying you have too many profiles which add up to over 100%.

`duration_min`

Minimum length of time, in seconds, that users of this profile will view a channel (play a file). Minimum = "1," Maximum = "2,147,483." (Default = "1").

duration_max

Maximum length of time, in seconds, that users of this profile will view a chan(play a file). Minimum = "1," Maximum = "2,147,483." (Default = "1").

channel_switch_delay_min

Minimum length of time, in milliseconds, that users of this profile will pause before viewing a new channel (requesting a new file). Minimum = "0," Maximum = "2,147,483,647." (Default = "0").

channel_switch_delay_max

Maximum length of time, in milliseconds, that users of this profile will pause before viewing a new channel (requesting a new file). Minimum = "0," Maximum = "2,147,483,647." (Default = "0").

EXAMPLE

```
$Activity_IPTV_VideoClient1 agent.pm.signalling.profile_table.appendItem \  
-id "ProfileTable" \  
-name "Fast Switching" \  
-num_profiles 1 \  
-channel_switch_delay_max 0 \  
-duration_max 30 \  
-duration_min 10 \  
-percentage 50.0 \  
-channel_switch_delay_min 0  
$Activity_IPTV_VideoClient1 agent.pm.signalling.profile_table.appendItem \  
-id "ProfileTable" \  
-name "Slow Switching" \  
-num_profiles 1 \  
-channel_switch_delay_max 0 \  
-duration_max 300 \  
-duration_min 100 \  
-percentage 50.0 \  
-channel_switch_delay_min 0
```



SEE ALSO

Video Client Agent, Signaling

Channel View

Channel View Table—Describes the channel view configuration options.

DESCRIPTION

Describes the options that are specific to the channel view table in custom option for channel_switch_mode for IPTV and multicast.

SYNOPSIS

```
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem]
$Activity_IPTV_VideoClient1 agent.pm.commands(0).channelviewTable.appendItem
```

Options

view_sequence

Mentions the sequence in which the channel is viewed.

view_sequence

Indicates the name of the channel view table.

EXAMPLE

```
$Activity_IPTV_VideoClient1 agent.pm.commands(0).channelviewTable.appendItem
\  
-id                "ChannelViewTable" \  
-view_sequence     "0-8,9" \  
-name              "Fast Switching"  
  
$Activity_IPTV_VideoClient1 agent.pm.commands(0).channelviewTable.appendItem
\  
-id                "ChannelViewTable" \  
-view_sequence     "0-8,9" \  
-name              "Slow Switching"
```



IPTV Options

IPTV Options—Describes the options that are specific to the video client in IPTV mode.

SYNOPSIS

```
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_IPTV_VideoClient1 agent.pm.iptv_options.config
```

DESCRIPTION

Describes the options that are specific to the video client in IPTV mode. IPTV Options are configured with the `agent.pm.iptv_options.config` option of activity list of the Video Client Agent.

Options

`iptv_switch_delay`

If `iptv_switch_mode` is "2" then specify the fixed length of time here. minimum = "1", maximum = "60", default = "1".

`iptv_switch_mode`

Selects how the IPTV client switches from the D server stream to the A server stream. The choices are:

Value	Description
"0" (Default)	Stop receiving D server stream when first A server packet is received
"1"	Receive D server stream for its entire duration
"2"	Stop receiving D server streams after receiving A server stream for certain duration

EXAMPLE

```
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_IPTV_VideoClient1 agent.pm.iptv_options.config \
-iptv_switch_delay                1 \
-iptv_switch_mode                 2
```




Stats

Stats—Configures the statistics that IxLoad gathers for the client's video streams.

DESCRIPTION

Stats are configured with the `agent.pm.stats.config` option of activity list of the Video Client Agent.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_IPTV_VideoClient1 agent.pm.stats.config
```

Options

`PerStreamEntriesPerUser`

Number of streams displayed for each user in the per-Stream view of the statistics. Default = "4", Min = "1", Max = "4".

`updateInterval`

Frequency, in milliseconds, at which IxLoad gathers the Quality Metrics statistics. Default="2000", Min = "2000", Max = "100000".

`nominalDelay`

Length of time that packets are held in playout buffer before being played. Default="2", Min = "1", Max = "100000".

`bufferSize`

Maximum number of packets that can be stored in the playout buffer at any instance in time. Default="65535", Min = "1", Max = "65535".

`enableVuserMonitor`

Enables monitoring of a virtual user. Default = false.

`vuserId`

ID of the virtual user that you want to monitor. Min="1" Max="2147483647" Default="1".

`enableVQmonStats`

If enabled, IxLoad applies the values in the Quality Metrics fields to the video streams received by the client and computes the Quality Metrics statistics.

`updateInterval`

Frequency, in milliseconds, at which IxLoad gathers the statistics related to the quality metrics. Default="2000", Min = "2000", Max = "100000".

`MinDelay`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

`MaxDelay`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

`enableFrameStats`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

`NomDelay`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

`IgnoreLoss`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

`JBEMode`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

`NomDelay`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

`totalLimit`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

`frameLimit`

This parameter is no longer used in IxLoad 4.20 and subsequent releases.

EXAMPLE

```
$Activity_IPTV_VideoClient1 agent.pm.stats.config \  
-MinDelay 5 \  
-PerStreamEntriesPerUser 4 \  
-MaxDelay 80 \  
-enableFrameStats false \  
-NomDelay 20 \  
-qualityLimit 0 \  
-IgnoreLoss false \  
-frameLimit 0 \  
-JBEMode 0 \  
-enableVQmonStats false \  
-vuserId 1 \  
-enableVuserMonitor false \  
-totalLimit 0 \  
-updateInterval 2000 \  
-bufferSize 65535 \  
-bitrateLimit 0 \  
-nominalDelay 2
```

Video Server Agent

Video Server Agent - create a video server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IPTV_VideoServer1 agent.config
```

DESCRIPTION

A video server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time .

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
#####
# Activity IPTV_VideoServer1 of NetTraffic Traffic2@Network2
#####
set Activity_IPTV_VideoServer1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType "Video Server" ]
```



```
set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]

$Activity_IPTV_VideoServer1 config \

    -enable                true \

    -name                  "IPTV_VideoServer1" \

    -timeline              $_Match_Longest_

$Activity_IPTV_VideoServer1 agent.config \

    -enable                true \

    -name                  "IPTV_VideoServer1"

$Activity_IPTV_VideoServer1 agent.pm.videoConfig.config \

    -serverMode            0

$Activity_IPTV_VideoServer1 agent.pm.videoConfig.videoList.clear

$Activity_IPTV_VideoServer1 agent.pm.videoConfig.videoList.appendItem \

    -id                    "Video" \

    -dest_port_incr        0 \

    -addr_incr             "0.0.0.1" \

    -name                  "Stream0" \

    -stream_count          10 \

    -                       "" \

    -duration              10 \

    -IP_type               "" \

    -type                  "VoD" \

    -starting_dest_port    1234

$Activity_IPTV_VideoServer1 agent.pm.advancedOptions.config \

    -enableEsm             false \

    -type_of_service_for_rtsp "Best Effort (0x0)" \

    -enableVlanPriority_for_rtsp true \

    -listen_port           554 \

    -enableTosRTSP         false \
```

```
-enableTosData                false \  
-link_speed                    1000 \  
-type_of_service_for_data     "Best Effort (0x0)" \  
-esm                          1460 \  
-vlan_priority_rtsp           2  
  
$Activity_IPTV_VideoServer1 agent.pm.videoProp.stream.clear  
$Activity_IPTV_VideoServer1 agent.pm.videoProp.stream.appendItem \  
-id                            "Stream" \  
-mpeg4_contains_hint_track     "" \  
-mpeg4_profile                 "" \  
-num_frames                    0 \  
-fileButton                    false \  
-struct_c                      "" \  
-mpeg4_trackID                 0 \  
-ip_bit_rate                   3.75 \  
-cbr                           0 \  
-tsperudp                      7 \  
-h264_contains_hint_track     "" \  
-duration                      10 \  
-transport                     1 \  
-dest_port_incr                0 \  
-addr_incr                     "0.0.0.1" \  
-d_server_tos_or_dscp         "Best Effort (0x0)" \  
-h264_trackID                 0 \  
-tos_or_dscp                   "Best Effort (0x0)" \  
-hor_size                      0 \  
-filename                      "" \  
-content                       "Synthetic Payload" \  

```



```
-same_source_ip           false \  
-hrd_buffer               0 \  
-h264_packetization_mode 1 \  
-hrd_rate                 0 \  
-type                    "VoD" \  
-enable_d_server_tos     false \  
-profile                  0 \  
-starting_dest_port      1234 \  
-duration_in_packets     0 \  
-file_duration            0.0 \  
-min_frame_size          0 \  
-                          "" \  
-frame_rate              0.0 \  
-max_ip_bit_rate         0.0 \  
-file_duration_in_rtp_clock 0.0 \  
-h264_ignore_hint_track  false \  
-h264_requires_fragmentation "" \  
-mpeg4_level             "" \  
-enable_tos              false \  
-mpeg_type               "MPEG2 Transport Stream" \  
-name                    "Stream0" \  
-vert_size               0 \  
-level                   0 \  
-stream_count            10 \  
-max_packet_rate         0.0 \  
-max_frame_size          0 \  
-mpeg4_ignore_hint_track false \  
-max_allowed_requests_d_server 1 \  

```

```
-h264_level          "" \  
-h264_profile        ""  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_rtsp.clear  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_rtsp.appendItem \  
-id                  "TypeOfServiceForRtsp" \  
-tos_val_for_rtsp    "Best Effort (0x0)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_rtsp.appendItem \  
-id                  "TypeOfServiceForRtsp" \  
-tos_val_for_rtsp    "Class 1 (0x20)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_rtsp.appendItem \  
-id                  "TypeOfServiceForRtsp" \  
-tos_val_for_rtsp    "Class 2 (0x40)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_rtsp.appendItem \  
-id                  "TypeOfServiceForRtsp" \  
-tos_val_for_rtsp    "Class 3 (0x60)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_rtsp.appendItem \  
-id                  "TypeOfServiceForRtsp" \  
-tos_val_for_rtsp    "Class 4 (0x80)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_rtsp.appendItem \  
-id                  "TypeOfServiceForRtsp" \  
-tos_val_for_rtsp    "Express Forwarding (0xA0)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_rtsp.appendItem \  
-id                  "TypeOfServiceForRtsp" \  
-tos_val_for_rtsp    "Control (0xC0)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_data.clear  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_data.appendItem \  
-id                  "TypeOfServiceForData" \  
-tos_val_for_data    "Best Effort (0x0)"
```



```
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_data.appendItem \  
-id "TypeOfServiceForData" \  
-tos_val_for_data "Class 1 (0x20)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_data.appendItem \  
-id "TypeOfServiceForData" \  
-tos_val_for_data "Class 2 (0x40)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_data.appendItem \  
-id "TypeOfServiceForData" \  
-tos_val_for_data "Class 3 (0x60)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_data.appendItem \  
-id "TypeOfServiceForData" \  
-tos_val_for_data "Class 4 (0x80)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_data.appendItem \  
-id "TypeOfServiceForData" \  
-tos_val_for_data "Express Forwarding (0xA0)"  
$Activity_IPTV_VideoServer1 agent.pm.predefined_tos_for_data.appendItem \  
-id "TypeOfServiceForData" \  
-tos_val_for_data "Control (0xC0)"
```

SEE ALSO

`ixNetTraffic`

Video Properties

Video Properties—Adds a video stream.

SYNOPSIS

```
set serverTraffic [::IxLoad new ixServerTraffic options]
$serverTraffic agentList.appendItem options...
$serverTraffic agentList(0).videoProp.appendItem options...
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoServer1 [$Traffic2_Network2 activityList.appendItem
options...]
```

DESCRIPTION

A `videoProp` object is added to the `Video Client Agent` object using the `appen` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`name`

Name of the video stream. (Default = "Stream0").

`type`

Type of the video stream. The choices for video mode are:

Value	Description
Multicast	Broadcast-type real-time video stream.
VoD	Video-on-Demand stream.



Note: If a stream uses a payload file containing MPEG-4 Part-2 video or H264 video, the `type` must be `VoD`; it cannot be `Multicast`.

The choices for IPTV mode are:



Value	Description
AD Server	<p>An A (Acquisition) server packages RTP streams into multicast UDP packets and streams them onto the distribution network.</p> <p>A D (Distribution) server caches a certain amount of the multicast video data being streamed over the network. When a user changes a channel, the D server sends a short unicast burst of the new channel's video traffic for the user to view while the system switches the user from the previous channel's multicast group to the new chan's group.</p>
V Server	A V server provides Video-on-Demand service to an IPTV client.

`stream`

This is a list of type `Stream`. The elements in this list comprise the list of streams available on the Video server. (Default = {}).

`stream_count`

If the video or IPTV A Server type is Multicast, this parameter specifies the numof instances of this stream that will be streamed out. specifies the first Multicast Group Address.

If the video or D Server type is VoD, this parameter specifies how many instances of the stream that the server hosts.

Minimum = "1," Maximum = "1,000." (Default = "1").

`set payloadfile`

This option specifies the name of the video file that will be streamed by the IxLoad Video Server or IPTV Server. IxLoad Video Server can stream H264 and MPEG4 encoded video track, in a video file, provided the file is in MPEG-4 file format.

`starting_multicast_group_addr`

For a Multicast channel, this field specifies the address of the first multicast group that the channel will be available on.

`addr_incr`

If more than one instance of the Broadcast channel will be streamed out (`stream_count` is greater than 1), this parameter specifies the amount of increase in each multicast group address for the streams. Minimum = "1," Maximum = "2,147,483,647." (Default = "1").

`starting_dest_port`

For a Multicast channel, this field specifies the first port number that the channel will be available on. Minimum = "0," Maximum = "65,535." (Default = "0").

dest_port_incr

If more than one instance of the Multicast channel will be streamed out (`stream_count` is greater than 1), this parameter specifies the amount of increase in each port number for the streams. Minimum = "0," Maximum = "2,147,483,647." (Default = "0").

duration

If the stream type is VoD or D Server, this parameter specifies the duration of the video stream. Minimum = "0," Maximum = "2,147,483." (Default = "0").

EXAMPLE

```
set payloadfile "D:/MPEG4/Cloud-vs11-withaudio(3.75Mbps).ts.MP4"

puts $payloadfile

puts "Before adding Stream1."

$svr_traffic agentList(0).pm.videoProp.stream.appendItem \

    -name                "Stream1" \
    -content              "Real Payload" \
    -filename             $payloadfile \
    -ip_bit_rate          "3.5000" \
    -type                 "VoD" \
    -stream_count         "2" \
    -duration             "100"

puts "After adding Stream1."

#-----

#ipv6 example

#-----

$svr_traffic agentList(0).pm.videoConfig.videoList(0).config \

    -name                "Stream0" \
    -type                 "Multicast" \
    - "FF04::13" \
    -stream_count         1 \
    -addr_incr            "0::1"
```



SEE ALSO

Video Client Agent

Stream

Stream—Configures a video stream. This command provides the same functions as the Edit Stream window in the user interface. `Video Properties` and `Stream` set many of the same parameters; whichever command is used most recently takes precedence.

SYNOPSIS

```
set serverTraffic [::IxLoad new ixServerTraffic options]
$serverTraffic agentList.appendItem options...
$serverTraffic agentList(0).videoProp.appendItem options...
    -stream      "myStreams"
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoServer1 [$Traffic2_Network2 activityList.appendItem
options...]
```

DESCRIPTION

A `Stream` object is a part of a `Video Properties` object, which is a member of a `Video Server Agent` object. Its options are configured as per the `ixConfig` sub-commands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`name`

Name of the video stream. (Default = "").

`transport`

Transport protocol used to send the video stream to the client. This is independent of `VoD` or `Multicast` type.

Video	Description
0	RTP over UDP.
1	(default) UDP.

`content`

Type of payload carried by stream. The choices are:



Value	Description
Synthetic Payload	The payload is simulated MPEG 2 compressed video generated by the IxLoad video server. The file for the synthetic payload contains a valid MPEG2 TS header but does not contain any MPEG2 PES packets. Specify the bit rate for the simulated video data in the Bit Rate field. For synthetic payloads, the IxLoad Video client can record packet latencies. See the list of statistics.
Real Payload	The payload is an actual MPEG 2 compressed video. Specify the file containing the video data in the <code>filename</code> parameter. The file must be a valid MPEG2 SPTS, MPEG2 MPTS, MPEG4 Part2, MPEG4 Part10 (also known as H.264), or VC1 file.
Dummy Payload	The payload consists of random non-video data.

`filename`

If `content` is set to `Real Payload`, specify the name and path of the video file.

Each load module type supports a different maximum file size. The combined total of all the video files to be played cannot exceed the limits below:

Load Module	Maximum Video File Size(per port)
CPM1000T8	250MB
ALM (512MB)	40MB
ALM (1GB)	80MB
STXS4-256	20MB
STXS4-128	10MB
10GE-LSM	40MB divided by the number of streams (40MB / # of streams)

`ip_bit_rate`

If `content` is set to `Synthetic Payload`, specify the bit rate of the generated stream. Maximum = "20.0000." (Default = "3.750000").

`mpeg_type`

MPEG version of video stream. (Default = "MPEG2"). Possible options are:

- MPEG2 SPTS
- MPEG2 MPTS
- MPEG4 Part2

MPEG2 SPTS

MPEG4 Part10 (also known as H.264)

VC1

`type`

Type of the video stream . The choices for video mode are:

Value	Description
Multicast	Broadcast-type real-time video stream.
VoD	Video-on-Demand stream.



Note: If a stream uses a payload file that contains MPEG-4 Part-2 video or H264 video, the `type` must `VoD`; it cannot be `Multicast`.

The choices for IPTV mode are:

Value	Description
AD Server	<p>An A (Acquisition) server packages RTP streams into multicast UDP packets and streams them onto the distribution network.</p> <p>A D (Distribution) server caches a certain amount of the multicast video data being streamed over the network. When a user changes a channel, the D server sends a short unicast burst of the new channel's video traffic for the user to view while the system switches the user from the previous channel's multicast group to the new chan's group.</p>
V Server	A v server provides Video-on-Demand service to an IPTV client.

`stream_count`

If the video or IPTV A Server `type` is `Multicast`, this parameter specifies the numof instances of this stream that will be streamed out. specifies the first MultiGroup Address.

If the video or IPTV D Server `type` is `VoD`, this parameter specifies how many instances of the stream that the server hosts.

Minimum = "1," Maximum = "1,000." (Default = "10").

`starting_multicast_group_addr`



For a Multicast channel, this field specifies the address of the first multicast group that the channel will be available on.

`addr_incr`

If more than one instance of the Broadcast channel will be streamed out (`stream_count` is greater than 1), this parameter specifies the amount of increase in each multicast group address for the streams. Minimum = "1," Maximum = "2,147,483,647." (Default = "1").

`starting_dest_port`

For a Multicast channel, this field specifies the first port number that the channel will be available on. Minimum = "0," Maximum = "65,535." (Default = "1,234").

`dest_port_incr`

If more than one instance of the Multicast channel will be streamed out (`stream_count` is greater than 1), this parameter specifies the amount of increase in each port number for the streams. Minimum = "0," Maximum = "2,147,483,647." (Default = "0").

`duration`

If the stream `type` is VoD or D Server, this parameter specifies the duration of the video stream. Minimum = "0," Maximum = "2,147,483." (Default = "10").

`tsperudp`

Helps to configure the number of transport stream (TS) packets packaged into a single UDP packet. Minimum = "1," Maximum = "7." (Default = "7").

`new_pid`

The PMT PID column lists the Program Map Table (PMT) PIDs in the stream. The Program Map Table defines the set of PIDs associated with a program, such as the audio PID, the video PID, and so on.

`is_modified_pid`

This option helps to edit the Elementary Stream (ES) PIDs associated with a PMT PID.

`enable_d_server_tos`

This enables the Type of Service (ToS) bits. Default = 0.

`d_server_tos_or_dscp`

If `enable_d_server_tos` is set to 1, you can set the Type of Service (ToS) bits that will be set in this stream from the A server, D Server and V Server. The value that is set here take precedence over the setting on the Advanced Options configuraThe available choices are:

Value	Description
Best Effort (0x0) (Default)	Routine service.
Class 1 (0x20)	Priority service, Assured Forwarding class 1
Class 2 (0x40)	Immediate service, Assured Forwarding class 2
Class 3 (0x60)	Flash, Assured Forwarding class 3
Class 4 (0x80)	Flash-override, Assured Forwarding class 4
Express Forwarding (0xA0)	Critical-ecp
Control (0xC0)	Internet-control



Note: This field only sets the ToS type for the multicast (data plane) traffic; the ToS type for IGMP packets (the control plane traffic) will remain set to 0xC0.

`d_server_bit_rate`

If you are running the A and D server on different ports, configure the bit rate for the D server's stream with this option. The bit rate for the D server stream must be greater than the bit rate from the A server stream.

`max_allowed_requests_d_server`

Maximum number of requests for each URL that the D server accepts at one time.

EXAMPLE

```
set serverTraffic [::IxLoad new ixServerTraffic options]
$serverTraffic agentList.appendItem ...
$serverTraffic agentList(0).videoProp.stream.appendItem\
    -name      "testvid" \
    -content   "Real Payload" \
    -filename  $payloadfile \
    -type      VoD
    -tsperudp  "7"
```



```
$svr_traffic agentList(0).pm.videoProp.stream(1).program(0).config \  
-new_pid          "200" \  
-is_modified_pid  "1"      #Changing the PMTPID value.  
  
$svr_traffic agentList(0).pm.videoProp.stream(1).program(0).es_info(0).config \  
\br/>#Changing the ES info row 0.  
-new_pid          "400"  
  
$svr_traffic agentList(0).pm.videoProp.stream(1).program(0).es_info(1).config \  
\br/>#Changing the ES info row 1.  
-new_pid          "600"
```

SEE ALSO

Video Properties, Video Server Agent

LOD_THY_TrackList

Configures the number and packet size of packets in the RTP streams in a server stream that uses a Dummy payload. The trackList object is a standard list object.

`-id`

Name of list. Default = "Track"

`-max_packet_size`

Maximum size, in bytes, of the packets in each track (RTP stream). Default = 200.

`-numInstances`

Number of tracks (RTP streams) included in the RTSP stream. Default = 1.

Example

```
$Activity_IPTV_VideoServer1 agent.pm.videoProp.stream(0).trackList.appendItem \  
-id "Track" \  
-max_packet_size 200 \  
-numInstances 1
```



Advanced Options

Advanced Options—Sets the Video server agent's global configuration options.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IPTV_VideoServer1 agent.pm.advancedOptions.config
```

DESCRIPTION

A Video server's advanced configuration options are set by modifying the options of the `pm.advancedOptions.confog` option of the `Video Server Agent` object.

SUBCOMMANDS

None.

OPTIONS

`listen_port`

Port that RTSP server listens on for new connections. Minimum = "1," Maximum = "65,535." (Default = "554").

`enableEsm`

If true, the use of the ESM option is enabled. (Default = false).

`enableTosRTSP`

Enables the setting of the TOS (Type of Service) bits in the header of the RTSP control packets.

Value	Description
0	(default) TOS bits not enabled.
1	TOS bits enabled.

`enableTosData`

Enables the setting of the TOS (Type of Service) bits in the header of the RTSP data packets.

Value	Description
0	(default) TOS bits not enabled.
1	TOS bits enabled.

esm

If `enableEsm` is true, the ESM value to negotiate with. (Default = 1,460).

`type_of_service_for_rtsp`

If `enableTosRTSP` is true, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "Best Effort 0x0"). If you want to specify the standard choices that are in the GUI, you can use a string representation. To specify any of the other 255 TOS values, specify the decimal value. The default choices are:

Value	Description
"Best Effort (0x0)"	(Default) routine priority
"Class 1 (0x20)"	Priority service, Assured Forwarding class 1
"Class 2 (0x40)"	Immediate service, Assured Forwarding class 2
"Class 3 (0x60)"	Flash, Assured Forwarding class 3
"Class 4 (0x80)"	Flash-override, Assured Forwarding class 4
"Express Forwarding (0xA0)"	Critical-ecp
"Control (0xC0)"	Internet-control

`type_of_service_for_data`

If `enableTosData` is true, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes for RTSP data packets. See `type_of_service_for_rtsp` for the list of choices (Default = "Best Effort (0x0)").

`enableVlanPriority_for_rtsp`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If true, `IxLoad` sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = false).

`vlanPriority`

When `enableVlanPriority` is true, this option accepts the `vlan` priority value.

`enable_d_server_tos`

This enables (1) the Type of Service (ToS) bits. Default = 0.



d_server_tos_or_dscp

If enable_d_server_tos is set to 1, you can set the Type of Service (ToS) bits that will be set in this stream from the A server, D Server and V Server. The value set here can be over-ridden by the value that is set for d_server_tos_or_dscp in Stream configuration. The available choices are:

Value	Description
Best Effort (0x0) (Default)	Routine service.
Class 1 (0x20)	Priority service, Assured Forwarding class 1
Class 2 (0x40)	Immediate service, Assured Forwarding class 2
Class 3 (0x60)	Flash, Assured Forwarding class 3
Class 4 (0x80)	Flash-override, Assured Forwarding class 4
Express Forwarding (0xA0)	Critical-ecp
Control (0xC0)	Internet-control



Note: This field only sets the ToS type for the multicast (data plane) traffic; the ToS type for IGMP packets (the control plane traffic) will remain set to 0xC0.

enable_hwacc

If True, hardware acceleration is used. Default = "false".

EXAMPLE

```
$Activity_IPTV_VideoServer1 agent.pm.advancedOptions.config \  
-enableEsm false \  
-type_of_service_for_rtsp "Best Effort (0x0)" \  
-enableVlanPriority_for_rtsp true \  
-listen_port 554 \  
-enableTosRTSP false \  
-enableTosData false \  
-link_speed 1000 \  
-type_of_service_for_data "Best Effort (0x0)" \  
-esm 1460 \  

```

-vlan_priority_rtsp

2

SEE ALSO

Video Server Agent



Video Config

Video Config—Contains the list of video streams hosted by the IxLoad IPTV AD and V Servers.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_IPTV_VideoServer1 [$Traffic2_Network2 activityList.appendItem
options...]
$Activity_IPTV_VideoServer1 agent.pm.videoConfig.config
```

DESCRIPTION

A Video server's configuration options are set by modifying the options of the `pm.videoConfig.config` option of the `Video Server Agent` object.

SUBCOMMANDS

None.

OPTIONS

`a_port_ip`

IP address of the A server port. (Default = "")

`iptv_multiport_enable`

Indicates whether traffic from A and D server originates from the same ixia port or multiple ports. (Default = "0").

`serverMode`

Sets the server mode to Video or IPTV. (Default = "0").

EXAMPLE

```
$Activity_IPTV_VideoServer1 agent.pm.videoConfig.config \  
-a_server_ip                "10.0.2.6" \  
-iptv_multiport_enable      1 \  
-serverMode                  1
```

SEE ALSO

Advanced Options

IPTV / Video Statistics

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<>version>\Results\<>repository>_<test
configuration>_<yyy><mm><dd>_<hh><mm><ss>
```

For the IPTV / Video client statistics, see IPTV / Video Client Statistics (on page 1299).

For the IPTV / Video server statistics, see IPTV / Video Server Statistics (on page 1328).

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).



Note: If the client is receiving a large number of streams (for example, about 820 1Mbps streams on an ALM1G client port), keep the page size (the number of rows) of the per-stream statistics view small. Having large page sizes (large numbers of rows per page) causes the statistics to not refresh correctly and causes paging (moving from one page to another) to take a long time.

IPTV / Video Client Statistics

This section describes the statistics for IPTV and Video clients .

Global Stream Statistics

The table below lists the IxLoad IPTV / Video client global stream statistics.

Statistic	Description
VoD Streams Playback Successful	Number of RTP streams played with no lost packets.
VoD Streams With Errors	Number of RTP streams played in which one or more packets were lost.
Frame Stats Disabled	Deprecated

Quality Metrics Disabled	<p>Initially, this statistic displays no value.</p> <p>If the received data rate exceeds the cut-off threshold, IxLoad stops computing the Quality Metrics, and this statistic will display "YES".</p> <p>The value will remain YES until the end of the iteration. Once the Quality Metrics computation is disabled during a run, it remains disabled throughout the remainder of the run.</p> <p>Prior to starting the next run (or the next iteration of the same test), this statistic will be cleared and IxLoad will again begin computing the Quality Metrics. It will continue to compute the metrics as long as the bit rate remains below the cut-off threshold.</p> <p>Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i>.</p>
Total Bytes Rcvd	Total number of bytes received by the client.
Total packets Rcvd	Total number of packets received by the client.
Total Loss	Total number MPEG2-TS packets lost.
Unexpected UDP Packets Received	Number of UDP packets received packets during a time when no channels are active.
Overload Packets Dropped	Number of RTP packets dropped because a port did not have enough computing power to process them.
Total RTP Packets Lost	Total number of RTP packets lost while using RTP over UDP transport.
Total Out Of Order RTP Packets Rcvd	Total number of RTP packets received in the wrong order while using RTP over UDP transport.
Total Duplicate RTP Packets	Total number of duplicate RTP packets received.
Global Jitter	<p>Average variation in arrival times of packets on all streams.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
RTCP Avg Packet Size	Average outbound RTCP packet size.
RTCP Avg Packet Transmission Time	Amount of time between the most recent two consecutive RTCP packets sent.
RTCP Packets Sent	Number of RTCP packets sent.
RTP Lost Sequence: One Packet	Number of instances in which 1 packet was lost.
RTP Lost Sequence: Two or Three Packets	Number of instances in which 2 or 3 consecutive packets were lost.
RTP Lost Sequence: Four or Five Packets	Number of instances in which 4 or 5 consecutive packets were lost.
RTP Lost Sequence: Six to Ten Packets	Number of instances in which 6-10 consecutive packets were lost.

RTP Lost Sequence: Eleven or More Packets	Number of instances in which 11 or more consecutive packets were lost.
RTP Maximum Lost Sequence	Maximum gap between the sequence numbers of RTP packets received on a stream. This statistic represents the maximum burst loss that has occurred in the network.
Jitter less than 50 us	Number of packets received with 0 to 50 microseconds of jitter.
Jitter between 50 - 100 us	Number of packets received with 50 to 100 microseconds of jitter.
Jitter between 100 - 500 us	Number of packets received with 100 -500 microseconds of jitter.
Jitter between 500 us - 2 ms	Number of packets received with 500 microseconds to 2 milliseconds of jitter.
Jitter between 2 - 5 ms	Number of packets received with 2 to 5 milliseconds of jitter.
Jitter between 5 - 10 ms	Number of packets received with 5 to 10 milliseconds of jitter.
Jitter greater than 10 ms	Number of packets received with more than 10 milliseconds of jitter.
Packet Size between 0 - 100 bytes	Number of packets received that were between 100 and 200 bytes in size.
Packet Size between 100 - 200 bytes	Number of packets received that were between 100 and 200 bytes in size.
Packet Size between 200 - 400 bytes	Number of packets received that were between 200 and 400 bytes in size.
Packet Size between 400 - 600 bytes	Number of packets received that were between 400 and 600 bytes in size.
Packet Size between 600 - 1000 bytes	Number of packets received that were between 600 and 1000 bytes in size.
Packet Size greater than 1000 bytes	Number of packets received that were larger than 1000 bytes.
Inter Packet Arrival Time between 0 - 2 ms	Number of packets that arrived less than 2 milliseconds after the preceding packet was received.
Inter Packet Arrival Time between 2 - 5 ms	Number of packets that arrived between 2 and 5 milliseconds after the preceding packet was received.
Inter Packet Arrival Time between 5 - 10 ms	Number of packets that arrived between 5 and 10 milliseconds after the preceding packet was received.
Inter Packet Arrival Time between 10 - 25 ms	Number of packets that arrived between 10 and 25 milliseconds after the preceding packet was received.
Inter Packet Arrival Time between 25 - 50 ms	Number of packets that arrived between 25 and 50 milliseconds after the preceding packet was received.
Inter Packet Arrival Time between 50 - 100 ms	Number of packets that arrived between 50 and 100 milliseconds after the preceding packet was received.



Inter Packet Arrival Time between 100 - 200 ms	Number of packets that arrived between 100 and 200 milliseconds after the preceding packet was received.
Inter Packet Arrival Time between 200 - 500 ms	Number of packets that arrived between 200 and 500 milliseconds after the preceding packet was received.
Inter Packet Arrival Time greater than 500 ms	Number of packets that arrived more than 500 milliseconds after the preceding packet was received.
Note: The following packet latency statistics are only available for streams from an IxLoad Video server with synthetic payloads.	
Packet Latency between 0 - 2 ms	Number of UDP packets that required between 0 and 2 milliseconds to travel from the server to the client.
Packet Latency between 2 - 5 ms	Number of UDP packets that required between 2 and 5 milliseconds to travel from the server to the client.
Packet Latency between 5 - 10 ms	Number of UDP packets that required between 5 and 10 milliseconds to travel from the server to the client.
Packet Latency between 10 - 25 ms	Number of UDP packets that required between 10 and 25 milliseconds to travel from the server to the client.
Packet Latency between 25 - 50 ms	Number of UDP packets that required between 25 and 50 milliseconds to travel from the server to the client.
Packet Latency between 50 - 100 ms	Number of UDP packets that required between 50 and 100 milliseconds to travel from the server to the client.
Packet Latency between 100 - 200 ms	Number of UDP packets that required between 100 and 200 milliseconds to travel from the server to the client.
Packet Latency between 200 - 500 ms	Number of UDP packets that required between 200 and 500 milliseconds to travel from the server to the client.
Packet Latency greater than 500 ms	Number of UDP packets that more than 500 milliseconds to travel from the server to the client.
Avg Packet Latency	<p>Average amount of time required for a packet to travel from the server to the client.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
Max Packet Latency	<p>Maximum amount of time required for a packet to travel from the server to the client.</p> <p>For Dummy payloads, this statistic is updated only if the packets are sent over RTP transport.</p> <p>For Synthetic payloads, this statistic is updated for packets sent over UDP or RTP transport, but only if Hardware Acceleration is disabled.</p>

Per-Stream Statistics

The table below lists the IxLoad IPTV / Video client statistics that are available on a per-stream basis.



Note: In Video mode, the per-stream statistics view displays as:
Video Client Per Stream.

In IPTV mode, the per-stream statistics view displays as :
Video Client IPTV Per Stream.

Statistic	Description
Active	Indicates whether the stream is active or not: 0 = inactive 1 = active Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
Stream Name	Name of stream. Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
Flow ID	Number identifying the flow used by the stream. A flow consists of the packets flowing between a source IP:port and a destination IP:port. Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
Transport	Type of transport used on the stream. Note for Tcl API users: 0 = UDP 1 = RTP/UDP For this statistic, use the Aggregation Type <i>kString</i> .
Video Codec	Video codec used on the stream. Note for Tcl API users: For this statistic, use the Aggregation Type <i>kString</i> .
Stream Bit Rate	Bit rate used on stream.
Bytes	Number of bytes received on the stream.
Packets	Number of packets received on the stream.
Loss	Number of packets lost on the stream.
Maximum Lost Sequence	Maximum gap between the sequence numbers of RTP packets received on the stream. This statistic represents the maximum burst loss that has occurred on the stream.
MDI-DF	Media Delivery Index - Delay Factor (MDI-DF) experienced on stream.



MIN MDI-DF	Smallest MDI Delay Factor experienced on stream. Note: When retrieved from the Tcl API, this statistic is returned in units of nanoseconds (ns).
MAX MDI-DF	Largest MDI Delay Factor experienced on stream. Note: When retrieved from the Tcl API, this statistic is returned in units of nanoseconds (ns).
AVG-MDI-DF	Average MDI Delay Factor experienced on stream. Note: When retrieved from the Tcl API, this statistic is returned in units of nanoseconds (ns).
MDI-MLR	Media Delivery Index - Media Loss Rate experienced on stream.
Jitter	Current instantaneous jitter.
Inter Pkt Arrival Time	Amount of time between received packets.
Min Inter Pkt Arrival Time	Smallest amount of time between received packets, in milliseconds.
Max Inter Pkt Arrival Time	Largest amount of time between received packets, in milliseconds.
Packet Latency (ns)	Average packet latency on the stream.
Min Packet Latency (ns)	Smallest packet latency on the stream.
Max Packet Latency (ns)	Longest packet latency on the stream.
Play Latency (ms)	Amount of time, in milliseconds, elapsed between the time the IPTV client sent an IGMP JOIN (to play a multicast channel on an AD server) or RTSP PLAY (to play a VoD channel on a V server) and the time it received the first byte of data.
Join Latency (ms)	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP JOIN (broadcast channel) or RTSP PLAY (VoD channel) and the time it received the first byte of data.
Leave Latency (ms)	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP LEAVE (broadcast channel) or RTSP PAUSE (VoD channel) and the time it received the last byte of data. Leave latency has a maximum timeout of 10 seconds; if the client continues to receive data 10 seconds after it has sent the Leave command, the latency is measured as 10 seconds.
Channel Switch Latency	Amount of time elapsed between the time the client sent an IGMP LEAVE to change to a new channel, and the time it received the first byte of the new stream.
Channel Gap Duration	When changing channels, this statistic measures the amount of time elapsed between the time the client received the last byte of the old stream and the time it received the first byte of the new stream.
Channel Overlap Duration	When changing channels, this statistic measures the amount of time that the client was simultaneously receiving both the old and new streams.

Control Sent	<p>Indicates the type of control command that has most recently been sent:</p> <p>0 = LEAVE or PAUSE/TEARDOWN sent</p> <p>1 = JOIN or PLAY sent</p>
Data Rcvd	<p>Indicates whether or not data is being received:</p> <p>0 = no data received</p> <p>1 = data received</p>
RTP Packets Lost	Number of RTP packets lost.
RTP Packets Out of Order	Number of RTP packets received out of order.
RTP Packets Duplicated	Number of duplicate RTP packets received.
ICC Unicast Bytes (IPTV mode only)	Number of unicast bytes received by the client.
ICC Unicast Packets (IPTV mode only)	Number of unicast packets received by the client.
ICC Multicast Bytes (IPTV mode only)	Number of multicast bytes received by the client.
ICC Multicast Packets (IPTV mode only)	Number of multicast packets received by the client.
ICC Packets Lost or Dup Due To Switch (IPTV mode only)	<p>Number of packets lost or duplicated due to switching from a unicast (D server) stream to a multicast (A server) stream.</p> <p>If packets were lost, this statistic is displayed with a minus-sign (-). For example, -100 indicates that 100 packets were lost.</p> <p>If packets were duplicated, this statistic is displayed with a plus-sign (+). For example, +100 indicates that 100 packets were duplicated.</p>
Unicast-Multicast Switch Latency (ms)	<p>Time elapsed switching from a VOD (unicast) stream to a broadcast (multicast) stream.</p> <p>This statistic is only returned for the ICC command for MSIPTV emulation.</p>
RTCP Packets Sent	<p>Number of RTCP packets sent.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>



Video Client Data Conditional Statistics

The table below lists the IxLoad IPTV / Video client data conditional statistics.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
UDP Bytes Received	All	Number of UDP bytes received.
UDP Packets Received	All	Number of UDP packets received.
MPEG2 TS Loss	All	Number of MPEG-2 Transport Stream packets lost. This statistic differs from the <i>Total TS Loss</i> statistic in that Total TS Loss measures the total MPEG2 TS packet lost, while this statistic measures only the loss on for the IP address, VLAN, or user.
MDI-DF (ms)	User	Media Delivery Index - Delay Factor (MDI-DF) experienced on stream.
MDI-MLR	User	Media Delivery Index - Media Loss Rate experienced on stream.
Avg MDI-DF (ms)	All	Average MDI Delay Factor experienced on stream. Note: When retrieved from the Tcl API, this statistic is returned in units of nanoseconds (ns).
Min MDI-DF (ms)	All	Smallest MDI Delay Factor experienced on stream. Note: When retrieved from the Tcl API, this statistic is returned in units of nanoseconds (ns).
Max MDI-DF (ms)	All	Largest MDI Delay Factor experienced on stream. Note: When retrieved from the Tcl API, this statistic is returned in units of nanoseconds (ns).
Stream Bit Rate (Kbps)	User	Bit rate used on stream.
Avg Stream Bit Rate (Kbps)	All	Average bit rate calculated for the stream.
Received Bit Rate (Kbps)	All	Actual bit rate of received stream.

RTP Clock Rate	User	Clock rate used for RTP connection.
RTP SSRC	User	Value of the SSRC field in RTP packets in the stream.
Video PID	User	Package Identifier used on video stream
Audio PID	User	Package Identifier used on audio stream.
RTP Packets Lost	All	Total number of RTP packets lost while using RTP over UDP transport.
RTP Packets Out of Order	All	Total number of RTP packets received in the wrong order while using RTP over UDP transport.
RTP Packets Duplicate	All	Number of duplicate RTP packets received.
Join Latency (ms)	User	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP JOIN (broadcast channel) or RTSP PLAY (VoD channel) and the time it received the first byte of data.
Leave Latency (ms)	User	Amount of time, in milliseconds, elapsed between the time the client sent an IGMP LEAVE (broadcast channel) or RTSP PAUSE (VoD channel) and the time it received the last byte of data. Leave latency has a maximum timeout of 10 seconds; if the client continues to receive data 10 seconds after it has sent the Leave command, the latency is measured as 10 seconds.
Channel Switch Latency (ms)	User	Average time elapsed between the time the client changed to a new channel and the time it received the first byte of the new stream.
Channel Overlap Duration (ms)	User	When changing channels, this statistic measures the amount of time that the client was simultaneously receiving both the old and new streams.
Channel Gap Duration (ms)	User	When changing channels, this statistic measures the amount of time elapsed between the time the client received the last byte of the old stream and the time it received the first byte of the new stream.



Avg Inter Pkt Arrival Time (us)	All	Amount of time between received packets. This statistic differs from the <i>Inter-Packet Arrival Time</i> statistic in that this statistic is an average, while <i>Inter-Packet Arrival Time</i> is an instantaneous measure.
Min Inter Pkt Arrival Time (us)	All	Smallest amount of time between received packets, in milliseconds.
Max Inter Pkt Arrival Time (us)	All	Largest amount of time between received packets, in milliseconds.
Avg One Way Delay (us)	All	
Min One Way Delay (us)	All	
Max One Way Delay (us)	All	
Avg Packet Latency (us)	All	Average packet latency on the stream.
Avg Jitter (us)	All	Current instantaneous jitter.
Min Jitter (us)	All	
Max Jitter (us)	All	
Transport	User	Type of transport used on the stream. Note for Tcl API users: 0 = UDP 1 = RTP/UDP For this statistic, use the Aggregation Type <i>kString</i> .

Multicast and VoD Global Statistics

The table below lists the global IxLoad IPTV / Video client multicast and VoD statistics.

Statistic	Description
Active Multicast Channels	Number of multicast channels joined across all users.
Multicast Channels Requested	Number of multicast channels that the client requested.
Multicast Requests Successful	Number of multicast requests for which the client received a successful response.
Multicast Requests Failed	Number of multicast requests for which the client received a failure response.
VoD Streams Played	Total number of VoD streams played by the client.
VoD Streams Playback Successful	Number of VoD streams played to completion by the client.

VoD Streams Played Failed	Number of VoD streams that could not be played.
---------------------------	---

Multicast and VoD Conditional Statistics

The table below lists the IxLoad IPTV / Video client multicast and VoD statistics that are available in Conditional view.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
Active Multicast Channels	All	Number of multicast channels joined across all users.
Multicast Channels Requested	All	Number of multicast channels that the client requested.
Multicast Requests Successful	All	Number of multicast requests for which the client received a successful response.
Multicast Requests Failed	All	Number of multicast requests for which the client received a failure response.
VoD Streams Played	All	Total number of VoD streams played by the client.
VoD Streams Playback Successful	All	Number of VoD streams played to completion by the client.
VoD Streams Played Failed	All	Number of VoD streams that could not be played.



IGMP and MLD Conditional Statistics

The table below lists the IxLoad IPTV / Video client IGMP and MLD that are available in Conditional view.

The Conditional Views column indicates the Conditional views in which a statistic is available:

IP: per-IP view
User: per-User view
VLAN: per-VLAN view
All: all views

Statistic	Conditional Views	Description
IGMP Statistics		
IGMPv1 Reports Sent	All	Number of IGMP version 1 Report messages sent.
IGMPv2 Reports Sent	All	Number of IGMP version 2 Report messages sent.
IGMPv2 Leaves Sent	All	Number of IGMP version 2 Leave messages sent.
IGMPv3 Reports Sent	All	Number of IGMP version 3 Report messages sent.
IGMP General Query Received	All	Number of IGMP General Query messages received.
IGMP Group Query Received	All	Number of IGMP Group Query messages received.
IGMPv3 Group Source Query Received	All	Number of IGMP version 3 Group Source Query messages received.
IGMPv1 Reports Received	All	Number of IGMP version 1 Report messages received.
IGMPv2 Reports Received	All	Number of IGMP version 2 Report messages received.
IGMPv3 Reports Received	All	Number of IGMP version 3 Report messages received.
MLD Statistics		
MLDv1 General Query Received	All	Number of MLD version 1 General Query messages received.
MLDv2 General Query Received	All	Number of MLD version 2 General Query messages received.
MLDv1 Group Query Received	All	Number of MLD version 1 Group Query messages received.
MLDv2 Group Query Received	All	Number of MLD version 2 Group Query messages received.
MLDv2 Group Source Query Received	All	Number of MLD version 2 Group Source Query messages received.

MLDv1 Reports Sent	All	Number of MLD version 1 Report messages received.
MLDv2 Reports Sent	All	Number of MLD version 2 Report messages sent.
MLDv1 Leave Sent	All	Number of MLD version 1 Leave messages sent.
MLDv1 Reports Received	All	Number of MLD version 1 Report messages received.
MLDv2 Reports Received	All	Number of MLD version 2 Report messages received.

IPTV Global Statistics

The table below lists the IxLoad IPTV / Video client statistics for IPTV clients.



Note: IGMP and MLD are not applicable to VoD, so in a VoD test, no IGMP or MLD statistics are displayed.

Statistic	Description
Active D Server Channels	Number of streams currently playing on the D server.
Active V Server Channels	Number of streams currently playing on the V server.
D Server Channels Requested	Number of streams requested from the D server.
D Server Requests Successful	Number of requests to the D server that were successful.
D Server Requests Failed	Combined total of control and data requests to the D server that failed.
D Server Requests Failed (Control)	Number of control plane requests to the D server that failed.
D Server Requests Failed (Data)	Number of data plane requests to the D server that failed.
V Server Channels Requested	Number of streams requested from the V server.
V Server Requests Successful	Number of requests to the V server that were successful.
V Server Requests Failed	Combined total of control and data requests to the V server that failed.
V Server Requests Failed (Control)	Number of control plane requests to the V server that failed.
V Server Requests Failed (Data)	Number of data plane requests to the V server that failed.
IGMP Queries Rcvd	Number of IGMP Query messages received by the client.
IGMP Reports Sent	Number of IGMP Report messages sent by the client for all users.
IGMP Leaves Sent	Number of IGMP Leave messages sent by the client.
MLD Queries Rcvd	Number of MLD Report messages received.
MLD Reports Sent	Number of MLD Report messages sent.



MLD Leaves Sent	Number of MLD Leave messages sent.
Join Latency	Average time elapsed between the time the client sent an IGMP Join message and the time it received a response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
Leave Latency	Average time elapsed between the time the client sent an IGMP Leave message and the time it received a response. This statistic is valid only for IGMPv2. For IGMPv3, Leaves for multicast groups are sent by sending an IGMP report with the modified group list. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
Channel Switch Latency	Average time elapsed between the time the client changed to a new channel and the time it received the first byte of the new stream. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RTSP Bytes Sent	Number of bytes sent by the RTSP client, including the payload and all headers.
RTSP Bytes Received	Number of bytes received in RTSP messages.
RTSP Packets Sent	Number of RTSP packets sent by the client.
RTSP Packets Received	Number of RTSP packets received by the client.
RTSP Concurrent Sessions	Number of concurrent RTSP sessions maintained.
RTSP Connection Rate	Rate at which the client established RTSP connections.
RTSP Transactions	Number of RTSP transactions completed.
RTSP Transaction Rate	Rate at which the client completed RTSP transactions.
RTSP Connections	Number of RTSP connections established by the client.
RTSP Setup Latency (ms)	Amount of time elapsed, in milliseconds, between a client sending a request to establish an RTSP connection and receiving the first byte of the response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RTSP Teardown Latency (ms)	Amount of time elapsed, in milliseconds, between a client sending a request to end an RTSP connection and receiving the first byte of the response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RTSP Play Latency (0 ms - 10 ms)	Number of instances in which 0 to 10 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.

RTSP Play Latency (10 ms - 50 ms)	Number of instances in which 10 to 50 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (50 ms - 100 ms)	Number of instances in which 50 to 100 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (100 ms - 300 ms)	Number of instances in which 100 to 300 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (300 ms - 1 s)	Number of instances in which 300 to 1000 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (Greater Than 1s)	Number of instances in which more than one second elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Presentations Active	Number of RTSP presentations available.
RTSP Presentations Playing	Number of RTSP presentations playing.
RTSP Presentations Paused	Number of RTSP presentations paused.
RTSP Presentations Requested	Number of RTSP presentations requested by the client.
RTSP Presentation Requests Successful	Number of presentation requests sent by the client for which it received a successful response.
RTSP Presentation Requests Failed	Number of presentation requests sent by the client that failed.
RTSP SET PARAMETER Sent	Number of RTSP SET PARAMETER messages sent.
RTSP GET PARAMETER Sent	Number of RTSP GET PARAMETER messages sent.
RTSP DESCRIBE Sent	Number of RTSP DESCRIBE messages sent.
RTSP SETUP Sent	Number of RTSP SETUP messages sent.
RTSP PLAY Sent	Number of RTSP PAUSE commands sent.
RTSP PAUSE Sent	Number of RTSP PAUSE commands sent.
RTSP TEARDOWN Sent	Number of RTSP TEARDOWN commands sent.
RTSP DESCRIBE Successful	Number of RTSP DESCRIBE commands for which a successful response was received.
RTSP SETUP Successful	Number of RTSP SETUP commands for which a successful response was received.
RTSP SET PARAMETER Successful	Number of RTSP SET PARAMETER commands for which a successful response was received.
RTSP GET PARAMETER Successful	Number of RTSP GET PARAMETER commands for which a successful response was received.
RTSP PLAY Successful	Number of RTSP PLAY commands for which a successful response was received.
RTSP PAUSE Successful	Number of RTSP PAUSE commands for which a successful response was received.
RTSP TEARDOWN Successful	Number of RTSP TEARDOWN commands for which a successful response was received.



RTSP DESCRIBE Failed	Number of RTSP DESCRIBE commands that failed.
RTSP SETUP Failed	Number of RTSP SETUP commands that failed.
RTSP SET PARAMETER Failed	Number of SET_PARAMETER replies received with a code other than OK (200).
RTSP GET PARAMETER Failed	Number of RTSP GET PARAMETER commands that failed.
RTSP PLAY Failed	Number of RTSP PLAY commands that failed.
RTSP PAUSE Failed	Number of RTSP PAUSE commands that failed.
RTSP TEARDOWN Failed	Number of RTSP TEARDOWN commands that failed.
Average Play latency	Average amount of time elapsed between the time the client sent a Play command and then time it received the first byte of the video stream. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
Average Pause latency	Average amount of time elapsed between the time the client sent a Pause command and the time it stopped receiving data from the video stream. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
Video Simulated Users	Number of users simulated by the client.

IPTV Conditional Statistics

The table below lists the IxLoad IPTV / Video client statistics for IPTV clients.



Note: IGMP and MLD are not applicable to VoD, so in a VoD test, no IGMP or MLD statistics are displayed.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
Active D Server Channels	All	Number of streams currently playing on the D server.
Active V Server Channels	All	Number of streams currently playing on the V server.
D Server Channels Requested	All	Number of streams requested from the D server.

D Server Requests Successful	All	Number of requests to the D server that were successful.
D Server Requests Failed	All	Combined total of control and data requests to the D server that failed.
D Server Requests Failed (Control)	All	Number of control plane requests to the D server that failed.
D Server Requests Failed (Data)	All	Number of data plane requests to the D server that failed.
V Server Channels Requested	All	Number of streams requested from the V server.
V Server Requests Successful	All	Number of requests to the V server that were successful.
V Server Requests Failed	All	Combined total of control and data requests to the V server that failed.
V Server Requests Failed (Control)	All	Number of control plane requests to the V server that failed.
V Server Requests Failed (Data)	All	Number of data plane requests to the V server that failed.

RTSP Conditional Statistics

The table below lists the IxLoad IPTV / Video client statistics for IPTV clients.



Note: IGMP and MLD are not applicable to VoD, so in a VoD test, no IGMP or MLD statistics are displayed.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
RTSP Bytes Sent	All	Number of bytes sent by the RTSP client, including the payload and all headers.
RTSP Bytes Received	All	Number of bytes received in RTSP messages.
RTSP Packets Sent	All	Number of RTSP packets sent by the client.
RTSP Packets Received	All	Number of RTSP packets received by the client.
RTSP Concurrent Sessions	All	Number of concurrent RTSP sessions maintained.
RTSP Connection Rate	All	Rate at which the client established RTSP connections.



RTSP Transactions	All	Number of RTSP transactions completed.
RTSP Transaction Rate	All	Rate at which the client completed RTSP transactions.
RTSP Connections	All	Number of RTSP connections established by the client.
RTSP Setup Latency (ms)	All	Amount of time elapsed, in milliseconds, between a client sending a request to establish an RTSP connection and receiving the first byte of the response.
RTSP Teardown Latency (ms)	All	Amount of time elapsed, in milliseconds, between a client sending a request to end an RTSP connection and receiving the first byte of the response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the kWeightedAverage aggregation type.
RTSP Presentations Active	All	Number of RTSP presentations available.
RTSP Presentations Playing	All	Number of RTSP presentations playing.
RTSP Presentations Paused	All	Number of RTSP presentations paused.
RTSP Presentations Requested	All	Number of RTSP presentations requested by the client.
RTSP Presentation Requests Successful	All	Number of presentation requests sent by the client for which it received a successful response.
RTSP Presentation Requests Failed	All	Number of presentation requests sent by the client that failed.
RTSP DESCRIBE Sent	All	Number of RTSP DESCRIBE messages sent.
RTSP SETUP Sent	All	Number of RTSP SETUP messages sent.
RTSP SET PARAMETER Sent	All	Number of RTSP SET PARAMETER messages sent.
RTSP GET PARAMETER Sent	All	Number of RTSP GET PARAMETER messages sent.
RTSP PLAY Sent	All	Number of RTSP PLAY messages sent.
RTSP PAUSE Sent	All	Number of RTSP PAUSE commands sent.
RTSP TEARDOWN Sent	All	Number of RTSP TEARDOWN commands sent.
RTSP DESCRIBE Successful	All	Number of RTSP DESCRIBE commands for which a successful response was received.
RTSP SETUP Successful	All	Number of RTSP SETUP commands for which a successful response was received.
RTSP SET PARAMETER Successful	All	Number of RTSP SET PARAMETER commands for which a successful response was received.
RTSP GET PARAMETER Successful	All	Number of RTSP GET PARAMETER commands for which a successful response was received.

RTSP PLAY Successful	All	Number of RTSP PLAY commands for which a successful response was received.
RTSP PAUSE Successful	All	Number of RTSP PAUSE commands for which a successful response was received.
RTSP TEARDOWN Successful	All	Number of RTSP TEARDOWN commands for which a successful response was received.
RTSP DESCRIBE Failed	All	Number of RTSP DESCRIBE commands that failed.
RTSP SETUP Failed	All	Number of RTSP SETUP commands that failed.
RTSP SET PARAMETER Failed	All	Number of SET_PARAMETER replies received with a code other than OK (200).
RTSP GET PARAMETER Failed	All	Number of RTSP GET PARAMETER commands that failed.
RTSP PLAY Failed	All	Number of RTSP PLAY commands that failed.
RTSP PAUSE Failed	All	Number of RTSP PAUSE commands that failed.
RTSP TEARDOWN Failed	All	Number of RTSP TEARDOWN commands that failed.
Average Play latency (ms)	All	Average amount of time elapsed between the time the client sent a Play command and the time it received the first byte of the video stream.
Average Pause latency (ms)	All	Average amount of time elapsed between the time the client sent a Pause command and the time it stopped receiving data from the video stream.

Video Quality Statistics

This section describes the video quality (TVQM VQmon/HD) statistics.

Global Video Quality Statistics

The table below lists the IxLoad IPTV / Video global video quality statistics.

These statistics measure the overall video and audio quality of all the currently active streams.



Note: Relative and Absolute MOS scores are described in Mean Opinion Score (MOS).

Statistic	Description
I Frames Rcvd	The number of video I-frames received.
P Frames Rcvd	The number of video P-frames received.
B Frames Rcvd	The number of video B-frames received.
I Frames Impaired	Number of I-frames impaired due to packet loss or discards.

P Frames Impaired	Number of P-frames impaired due to packet loss and/or discards. This does not include frames impaired due to error propagation through temporal reference.
B Frames Impaired	Number of B-frames impaired due to packet loss and/or discards. This does not include frames impaired due to error propagation through temporal reference.
Avg Curr Abs MOS V	Absolute MOS for all the currently active video streams, averaged from stream start to the current time.
Avg Curr Rel MOS V	Relative MOS for all the currently active video streams, averaged from stream start to the current time.
Avg Curr MOS AV	Audio/video (multimedia) MOS for all the currently active streams, averaged from stream start to the current time.
Avg Curr MOS A	Absolute audio MOS, averaged from stream start to the current time.
Avg Comp Abs MOS V	Absolute MOS for all completed video streams, averaged across all streams.
Avg Comp Rel MOS V	Relative MOS for all completed video streams, averaged across all streams.
Avg Comp MOS AV	Audio/video (multimedia) MOS for all completed streams, averaged across all streams.
Avg Comp MOS A	Audio MOS for all completed streams, averaged across all streams.
Avg Interval Abs MOS V	Absolute MOS, averaged over the most recent statistics Update Interval (you can configure the Update Interval is on the video client Statistics Options tab).
Avg Interval Rel MOS V	Relative MOS averaged over the most recent statistics Update Interval (you can configure the Update Interval on the video client Statistics Options tab).
Avg Video Bw	The average video bandwidth, in bits/second, excluding transport packet header overhead and error correction/retransmission.
I Frame Avg Video Bw	The average bandwidth of I-frame video content transmitted, in bits/second, for all currently active streams.
P Frame Avg Video Bw	The average bandwidth of P-frame video content transmitted, in bits/second, for all currently active streams.
B Frame Avg Video Bw	The average bandwidth of B-frame video content transmitted, in bits/second, for all currently active streams.
Scene Avg Detail Level	The average amount of detail in the currently active streams, expressed on a scale of 0 (little detail) to 100 (maximum detail).
Scene Avg Panning Level	The average amount of panning in the currently active streams, expressed on a scale of 0 (no panning) to 100 (continuous panning).
Scene Avg Motion Level	The average amount of motion in the currently active streams, expressed on a scale of 0 (no motion) to 100 (continuous motion).

Per-stream Video Quality Statistics

The table below lists the IxLoad IPTV / Video client per-stream video quality statistics.

These statistics measure the video and audio quality of a single stream.



Note: Relative and Absolute MOS scores are described in Mean Opinion Score (MOS).

Statistic	Description
TVQM Avg Video Bw	The average video bandwidth, in bits/second, excluding transport packet header overhead and error correction/retransmission.
TVQM Peak Video Bw	The peak video bandwidth, in bits/second, measured during a one second window, excluding transport packet header overhead and error correction/retransmission.
TVQM Packets Received	Number of stream video transport packets received properly for playout.
TVQM Packets Discarded	Number of stream video transport packets discarded.
TVQM Frame Rate	The video frame rate, in frames per one thousand seconds – e.g. 29,970 equals 29.97 frames per second.
TVQM Avg Abs MOSV	The average absolute video stream MOS over the stream duration.
TVQM Avg Rel MOSV	The average relative video stream MOS over the stream duration.
TVQM Avg MOSA	Absolute audio MOS, averaged from stream start to the current time.
TVQM Avg MOSAV	The average audio/video stream MOS over the stream duration.
TVQM Int Avg Abs MOSV	Absolute MOS, averaged over the most recent statistics Update Interval (you can configure the Update Interval is on the video client Statistics Options tab).
TVQM Int Avg Rel MOSV	Relative MOS averaged over the most recent statistics Update Interval (you can configure the Update Interval on the video client Statistics Options tab).
TVQM I Frames Rcvd	The number of video I-frames received.
TVQM P Frames Rcvd	The number of video P-frames received.
TVQM B Frames Rcvd	The number of video B-frames received.
TVQM I Frames Impaired	Number of I-frames impaired due to packet loss or discards.



TVQM P Frames Impaired	Number of P-frames impaired due to packet loss and/or discards. This does not include frames impaired due to error propagation through temporal reference.
TVQM B Frames Impaired	Number of B-frames impaired due to packet loss and/or discards. This does not include frames impaired due to error propagation through temporal reference.
TVQM Detail Level	The instantaneous amount of detail, expressed on a scale of 0 (little detail) to 100 (maximum detail).
TVQM Panning Level	The instantaneous amount of panning, expressed on a scale of 0 (no panning) to 100 (continuous panning).
TVQM Motion Level	The instantaneous amount of motion, expressed on a scale of 0 (no motion) to 100 (continuous motion).
TVQM Inter I Frame Gap	The average gap, in frames, between I frames (excluding the I-frames)
TVQM PPDV	The stream transport Packet-to-Packet Delay Variation (RFC3550), in milliseconds.

Packet Transport Conditional Statistics

The table below lists the IxLoad IPTV / Video client video quality packet transport statistics.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
JB Packets Rcvd	User	The number of stream transport packets received.
JB Packets Lost	User	The number of stream transport packets lost in the network.
JB Packets Discarded	User	The number of stream transport packets discarded by the endpoint due to late arrival.

Video Description Conditional Statistics

The table below lists the IxLoad IPTV / Video client video quality video description statistics.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
Codec Type	User	The video CODEC type for the video stream.
GOP Structure	User	The GOP structure expressed as a series of 'I', 'B', 'P' characters describing the frame type series in the structure.
Avg GOP Length	User	The average GOP length, in frames.
Avg Inter I Frame Gap (Frames)	User	The average gap, in frames, between I frames (excluding the I-frames)
Frame Rate	User	The video frame rate, in frames per one thousand seconds – e.g. 29,970 equals 29.97 frames per second.

Video Perceptual Quality Conditional Statistics

The table below lists the IxLoad IPTV / Video client video perceptual quality statistics.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
Avg Absolute MOS V	User	The average absolute video stream MOS over the stream duration.
Avg Relative MOS V	User	The average relative video stream MOS over the stream duration.
Avg MOS A	User	The average audio stream MOS over the stream duration.



Avg MOS AV	User	The average audio/video stream MOS over the stream duration.
Interval Absolute MOS V	User	The absolute stream instantaneous video MOS sampled at the end of the interval configured on the video client Statistics Options tab.
Interval Relative MOS V	User	The relative stream instantaneous video MOS sampled at the end of the interval configured on the video client Statistics Options tab.
EPSNR (ATIS)	User	The Estimated Peak Signal to Noise Ratio (PSNR) calculated according to ATIS specifications.

Video Frame Conditional Statistics

The table below lists the IxLoad IPTV / Video client video quality video frame statistics.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
I Frames Rcvd	User	The number of video I-frames received.
I Frames Impaired	User	Number of I-frames impaired due to packet loss or discards.
P Frames Rcvd	User	The number of video P-frames received.
P Frames Impaired	User	Number of P-frames impaired due to packet loss and/or discards. This does not include frames impaired due to error propagation through temporal reference.
B Frames Rcvd	User	The number of video B-frames received.
B Frames Impaired	User	Number of B-frames impaired due to packet loss and/or discards. This does not include frames impaired due to error propagation through temporal reference.
SI Frames Rcvd	User	The number of video SI-frames received.
SI Frames Impaired	User	The number of video SI-frames impaired by packet loss or discard.
SP Frames Rcvd	User	The number of video SP-frames received.
SP Frames Impaired	User	The number of video SP-frames impaired by packet loss or discard.
I Frame Pkts Rcvd	User	The number of transport packets received containing video I-frame information.
I Frame Pkts Lost	User	The number of transport packets lost containing video I-frame information.
I Frame Pkts Discarded	User	The number of transport packets discarded due to late arrival containing video I-frame information.
P Frame Pkts Rcvd	User	The number of transport packets received containing video P-frame information.
P Frame Pkts Lost	User	The number of transport packets lost containing video P-frame information.



P Frame Pkts Discarded	User	The number of transport packets discarded due to late arrival containing video P-frame information.
B Frame Pkts Rcvd	User	The number of transport packets received containing video B-frame information.
B Frame Pkts Lost	User	The number of transport packets lost containing video B-frame information.
B Frame Pkts Discarded	User	The number of transport packets discarded due to late arrival containing video B-frame information.

Bandwidth Conditional Statistics

The table below lists the IxLoad IPTV / Video client video quality bandwidth statistics. These statistics pertain to the distribution of I, B, P, SI and SP video frame and audio frame bandwidth consumption.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
Avg Video Bw (Kbps)	User	The average video bandwidth, in bits/second, measured during a one second window, excluding transport packet header overhead and error correction/retransmission.
Peak Video Bw (Kbps)	All	The peak video bandwidth, in bits/second, measured during a one second window, excluding transport packet header overhead and error correction/retransmission.
Avg Audio Bw (Kbps)	User	The average audio bandwidth, in bits/second, measured during a one second window, excluding transport packet header overhead and error correction/retransmission.
Peak Audio Bw (Kbps)	All	The peak audio bandwidth, in bits/second, measured during a one second window, excluding transport packet header overhead and error correction/retransmission.
I Frame Avg Video Bw (Kbps)	User	The average bandwidth of I-frame transport packets received, in bits/second.
I Frame Peak Video Bw (Kbps)	All	The maximum bandwidth of I-frame transport packets received, in bits/second.
P Frame Avg Video Bw (Kbps)	User	The average bandwidth of P-frame transport packets received, in bits/second.
P Frame Peak Video Bw (Kbps)	All	The maximum bandwidth of P-frame transport packets received, in bits/second.
B Frame Avg Video Bw (Kbps)	User	The average bandwidth of B-frame transport packets received, in bits/second.
B Frame Peak Video Bw (Kbps)	All	The maximum bandwidth of B-frame transport packets received, in bits/second.



Frame Jitter Conditional Statistics

The table below lists the IxLoad IPTV / Video client video quality video jitter statistics. These statistics contain the video frame jitter and transmission delay statistics.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
User: per-User view
VLAN: per-VLAN view
All: all views

Statistic	Conditional Views	Description
Frame Inter Arrival Jitter (ms)	User	The average frame inter-arrival jitter, in milliseconds. The inter-arrival jitter is computed relative to the expected arrival time based on the frame rate.
I Frame Inter Arrival Jitter (ms)	User	The average I-frame inter-arrival jitter, in milliseconds. The inter-arrival jitter is computed relative to the expected arrival time based on the frame rate.

Packet Jitter Conditional Statistics

The table below lists the IxLoad IPTV / Video client video quality packet jitter statistics. These statistics provide a variety of statistics about the transport packet jitter experienced throughout the duration of the stream.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
User: per-User view
VLAN: per-VLAN view
All: all views

Statistic	Conditional Views	Description
PPDV (ms)	User	The stream transport Packet-to-Packet Delay Variation (RFC3550), in milliseconds.
Max PPDV (ms)	All	The maximum stream transport Packet-to-Packet Delay Variation (RFC3550), in milliseconds.



Scene Analysis Conditional Statistics

The table below lists the IxLoad IPTV / Video client TVQM Scene Analysis statistics. These statistics describe the scene types and content detected within the video stream.

The Conditional Views column indicates the conditional views in which a statistic is available:

IP: per-IP view
User: per-User view
VLAN: per-VLAN view
All: all views

Statistic	Conditional Views	Description
Scene Detail Level	User	The instantaneous amount of detail, expressed on a scale of 0 (little detail) to 100 (maximum detail).
Scene Motion Level	User	The instantaneous amount of motion, expressed on a scale of 0 (no motion) to 100 (continuous motion).
Scene Panning Level	User	The instantaneous amount of panning, expressed on a scale of 0 (no panning) to 100 (continuous panning).

IPTV / Video Server Statistics



Note: The video servers do not have a Ramp Down period; they stream for the duration of the test and then stop as quickly as possible at the end of the test. Therefore, the statistics may show the server bit rates still above 0 (zero) shortly after the end of the test.

The table below lists the IxLoad Video server statistics.

The Conditional Views column indicates the views in which a statistic is available:

IP: per-IP view
User: per-User view
VLAN: per-VLAN view
All: all views

Statistic	Conditional Views	Description
Total Streams Playing		Total number of video streams playing on the video server.

No of Multicast Streams Playing		Number of multicast (broadcast-type) streams playing.
No of VoD Streams Active	IP, VLAN	Number of video-on-demand streams active.
No of VoD Streams Playing	IP, VLAN	Number of video-on-demand streams playing.
No of VoD Streams Paused	IP, VLAN	Number of video-on-demand streams currently paused.
No of Multicast Streams Played	IP, VLAN	Number of multicast (broadcast-type) streams played.
No of VoD Streams Played	IP, VLAN	Number of video-on-demand streams played.
Total Streaming Bit Rate		Aggregate bit rate of all video streams playing on the server.
Multicast Streams Bit Rate		Bit rate of multicast (broadcast-type) video streams playing on the server.
VoD Streams Bit Rate	IP, VLAN	Bit rate of video-on-demand video streams playing on the server.
No of IPTV D Server Requests Received	IP, VLAN	Number of requests received by the D server.
No of IPTV V Server Requests Received	IP, VLAN	Number of requests received by the V server.
No of IPTV D Server Requests Successful	IP, VLAN	Number of requests received by the D server that were successful.
No of IPTV V Server Requests Successful	IP, VLAN	Number of requests received by the V server that were successful.
No of IPTV D Server Requests Failed	IP, VLAN	Total number of requests received by the D server that failed for all reasons.
No of IPTV V Server Requests Failed	IP, VLAN	Total number of requests received by the V server that failed for all reasons.
No of IPTV D Server Requests Failed for Bandwidth	IP, VLAN	Number of requests received by the D server that failed because not enough bandwidth was available on the server.
No of IPTV V Server Requests Failed for Bandwidth	IP, VLAN	Number of requests received by the V server that failed because not enough bandwidth was available on the server.
No of IPTV D Server Requests Failed for Port Overload	IP, VLAN	Number of requests received by the D server that failed because the Ixia port that the server was running on was oversubscribed.
No of IPTV V Server Requests Failed for Port Overload	IP, VLAN	Number of requests received by the V server that failed because the Ixia port that the server was running on was oversubscribed.
No of IPTV D Server Requests Failed for Other Reasons	IP, VLAN	Number of requests received by the D server that failed for reasons other than lack of bandwidth or port overload.
No of IPTV V Server Requests Failed for Other Reasons	IP, VLAN	Number of requests received by the V server that failed for reasons other than lack of bandwidth or port overload.



No of IPTV Active A Server Streams Playing	--	Number of streams available on the A server that are currently playing.
No of IPTV Active D Server Streams Playing	IP, VLAN	Number of streams available on the D server that are currently playing.
No of IPTV Active V Server Streams	IP, VLAN	Number of streams available on the V server.
No of IPTV Active V Server Streams Playing	IP, VLAN	Number of streams on the V server that are currently playing.
No of IPTV Active V Server Streams Paused	IP, VLAN	Number of streams on the V server that are currently paused.
A Server Streams Bit Rate	--	Combined bit rate of all streams currently playing on the A server.
D Server Streams Bit Rate	IP, VLAN	Combined bit rate of all streams currently playing on the D server.
V Server Streams Bit Rate	IP, VLAN	Combined bit rate of all streams currently playing on the V server.
IPTV Total Streaming Bit Rate	--	Combined bit rate of all streams currently playing on the A, D, and V servers.
RTSP Presentations Received	IP, VLAN	Number of RTSP Presentation requests received by the servers.
RTSP Presentations Successful	IP, VLAN	Number of RTSP Presentation requests that succeeded.
RTSP Presentations Failed	IP, VLAN	Number of RTSP Presentation requests that failed.
RTSP Bytes Sent	IP, VLAN	Number of RTSP-related bytes (commands and responses) sent by the server.
RTSP Bytes Received	IP, VLAN	Number of RTSP-related bytes (commands and responses) received by the server.
RTSP Packets Sent	IP, VLAN	Number of RTSP packets sent by the server.
RTSP Packets Received	IP, VLAN	Number of RTSP packets received the server.
RTSP Play Latency (ms)	All	Average amount of time elapsed, in milliseconds, between the time the server received a PLAY request and the time it transmitted the first byte of the video stream. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RTSP Commands Received	IP, VLAN	Total number of RTSP commands of all types received by the server.
RTSP DESCRIBE Received	IP, VLAN	Total number of RTSP DESCRIBE commands received by the server.
RTSP SETUP Received	IP, VLAN	Total number of RTSP SETUP commands received by the server.

RTSP PLAY Received	IP, VLAN	Total number of RTSP PLAY commands received by the server.
RTSP PAUSE Received	IP, VLAN	Total number of RTSP PAUSE commands received by the server.
RTSP TEARDOWN Received	IP, VLAN	Total number of RTSP TEARDOWN commands received by the server.
RTSP Response Codes Sent (2xx)	IP, VLAN	Number of 200-range (Success) responses sent. A 200-range response indicates that the action was successfully received, understood, and accepted.
RTSP Response Codes Sent (3xx)	IP, VLAN	Number of 300-range (Redirection) responses sent. A 300-range response indicates that further action must be taken in order to complete the request.
RTSP Response Codes Sent (4xx)	IP, VLAN	Number of 400-range (Client Error) responses sent. A 400-range response indicates that the request contains bad syntax or cannot be fulfilled.
RTSP Response Codes Sent (5xx)	IP, VLAN	Number of 500-range (Server Error) responses sent. A 500-range response indicates that the server failed to fulfill an apparently valid request.
RTSP Response Codes Sent (6xx-1xxx)	IP, VLAN	Number of 600- to 1000-range responses sent.
Total Bytes Sent	--	Total bytes sent by the server.
Total Packets Sent	--	Total packets sent by the server.
Tx Jitter (ns)	--	Variation in packet transmission times, in nanoseconds.
Tx Packets Dropped	--	Number of packets dropped before transmission.



LDAP

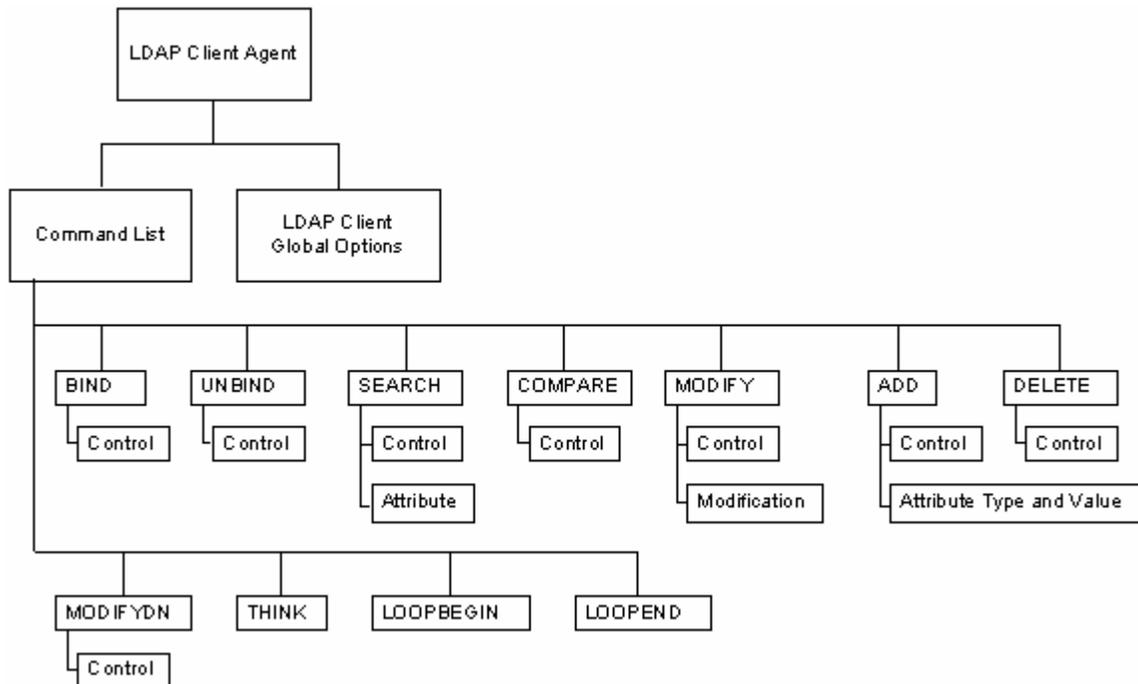
This section describes the LDAP Tcl API objects.

In this chapter:

Overview	1333
LDAP Client Agent	1337
LDAP Statistics	1360

Overview

LDAP protocol commands are organized as shown in the figure below.



Objectives

The objectives (userObjective) you can set for LDAP are listed below. Test objectives are set in the ixTimeline object.

- connectionRate
- transactionRate
- simulatedUsers
- concurrentConnections
- throughputMBps
- throughputKBps

LDAP Client Commands

This section lists the LDAP client commands.



LDAP Client Agent

The LDAP Client Agent command defines a simulated user performing LDAP requests against one or more LDAP servers. Refer to `LDAP Client Agent` for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>enable</code>	Enables the use of this client agent.
<code>name</code>	The name associated with this object, which must be set at object creation time .
<code>protocol</code>	Protocol used by the client agent.
<code>type</code>	Defines the agent as either a client or server.

Command List

This command defines the list of commands that the client sends to the server. Refer to `Command List` for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>id</code>	LDAP command to be executed.

Global Options

The LDAP client Global Options control network level operation of the client. Refer to `Global Options` for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>version</code>	Version of the LDAP protocol used for all client sessions.
<code>persistentConnection</code>	If <code>true</code> , the client opens a new TCP connection for each command sent.
<code>maxRequestsPerConn</code>	Maximum number of requests sent on each connection.
<code>maxConcurrentConnPerUser</code>	Maximum number of concurrent connections per user.
<code>followReferral</code>	If <code>true</code> ("On"), and the client receives a response that is a referral to another server, it redirects the request to the referred server.
<code>commandTimeout</code>	Time (in ms) to wait for a response before aborting.
<code>mustBind</code>	If <code>true</code> , the client sends an implicit BIND on every new connection that it establishes.

Control

Configures a control to be included in a list associated with a command. Refer to `Control` for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>controlType</code>	LDAP OID of the control associated with the command.
<code>criticality</code>	If <code>true</code> , the control is critical.
<code>controlValue</code>	Value for control.

Modification

A modification to be included in the `modificationList` of the `MODIFY` command. Refer to `Modification` for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>operation</code>	Type of modification to be performed.
<code>type</code>	Attribute to be modified.
<code>valueList</code>	List of values for the operation.

Attribute

An attribute to be included in the `searchAttributeList` of the `SEARCH` command. Refer to `Attribute` for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>attrib</code>	Attribute

Attribute Type and Values

An attribute and one or more values to be included in the `attribDescValueList` of the `ADD` command. Refer to `Attribute Type and Values` for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>type</code>	Attribute.
<code>valueList</code>	List of values for the attribute.



LDAP Client Agent

LDAP Client Agent - create an LDAP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_LDAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_LDAPClient1 agent.config
```

DESCRIPTION

An LDAP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`loopValue`

If this option is enabled (1), then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (0), then the client will progress through the command list only once, and then go idle. (Default = 0).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
```



```
#####  
# Activity LDAPClient1 of NetTraffic Traffic1@Network1  
#####  
set Activity_LDAPClient1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "ldap Client" ]  
#####  
# Timeline1 for activities LDAPClient1  
#####  
set Timeline1 [::IxLoad new ixTimeline]  
$Timeline1 config \  
    -rampUpValue                      1 \  
    -rampUpType                        0 \  
    -offlineTime                      0 \  
    -rampDownTime                    20 \  
    -standbyTime                      0 \  
    -iterations                       1 \  
    -rampUpInterval                   1 \  
    -sustainTime                      20 \  
    -timelineType                     0 \  
    -name                              "Timeline1"  
$Activity_LDAPClient1 config \  
    -enable                            true \  
    -name                              "LDAPClient1" \  
    -enableConstraint                  false \  
    -userObjectiveValue                100 \  
    -constraintValue                   100 \  
    -userObjectiveType                 "simulatedUsers" \  
    -timeline                          $Timeline1
```

```
$Activity_LDAPClient1 agent.config \  
    -enable true \  
    -name "LDAPClient1"  
$Activity_LDAPClient1 agent.pm.DistinguishedName.DN.clear  
$Activity_LDAPClient1 agent.pm.DistinguishedName.DNList.clear  
$Activity_LDAPClient1 agent.pm.globalOptions.config \  
    -initiateCloseFromClient true \  
    -commandTimeout 10000 \  
    -mustBind 1 \  
    -persistentConnection 1 \  
    -maxRequestsPerConn 10000 \  
    -authType "CLEARTEXT Password" \  
    -version 3 \  
    -maxConcurrentConnPerUser 10 \  
    -followReferral 1 \  
    -password "ixia" \  
    -implicitLoopCheck true \  
    -name "c=US,o=IXIA"  
$Activity_LDAPClient1 agent.pm.AddressHistory.clear  
$Activity_LDAPClient1 agent.pm.cmdList.clear  
$Activity_LDAPClient1 agent.pm.cmdList.appendItem \  
    -id "BIND" \  
    -authType "CLEARTEXT Password" \  
    -password "ixia" \  
    -name "c=US,o=IXIA" \  
    -serverAddr "1.2.3.4"  
$Activity_LDAPClient1 agent.pm.cmdList(0).bindControls.clear  
$Activity_LDAPClient1 agent.pm.cmdList(0).bindControls.appendItem \  

```



-id	"Control" \
-controlValue	"2" \
-controlType	"1" \
-criticality	1

SEE ALSO

`ixNetTraffic`

Command List

Command List—Creates the list of LDAP commands that the client will send to an LDAP server.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_LDAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_LDAPClient1 agent.pm.cmdList.appendItem
```

DESCRIPTION

A command is added to the `Command List` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

LDAP command to be executed. One of the following:

Command	Description
BIND	Exchanges authentication information between the client and server.
UNBIND	Terminates an LDAP session. After transmitting an UNBIND command, the client assumes that the session is terminated. There is no response for an UNBIND command. When the server receives an UNBIND, it assumes that the client has terminated the session and all outstanding requests may be discarded.
SEARCH	Requests that the server search its directory for information requested by the client. A SEARCH command can be used to read attributes from a single entry, from entries immediately below a particular entry in the directory tree, or a whole subtree of entries. The SEARCH option includes an Attribute List. You can use this list to add attributes to an LDAPv3 SEARCH command. For a list of the attributes, refer to RFC 2256.



COMPARE	Allows a client to ask the server whether the named entry has an attribute/value pair. The COMPARE command allows the server to keep certain attribute/value pairs secret (that is, not exposed for general search access) while still allowing the client limited use of them. For example, some servers might use this feature for password although it is insecure for the client to pass clear-text passwords in the COMPARE operation itself.
MODIFY	Requests that the server edit an entry on behalf of the client.
ADD	Requests that the server add an entry to the directory.
DELETE	Requests that the server delete a leaf entry from the directory.
MODIFYDN	Allows a client to change a distinguished name (DN) entry or to move a subtree of entries to a new location.
THINK	Causes the client to become inactive. THINK is an internal IxLoad command intended to assist your testing; it is not a command defined in the LDAP protocol. If you specify identical values for the minimum and maximum intervals, the client will be inactive for a fixed length of time. If you specify different values for the minimum and maximum intervals, IxLoad will select a value within the range and cause the client to be inactive for that length of time.
LOOPBEGIN	An IxLoad command that you can add to the Command List to cause the commands between it and the {Loop End} to be executed a specified number of times. For example, in a Command List that contains the following commands: {Loop Begin} BIND SEARCH UNBIND {Loop End} The BIND, SEARCH, and UNBIND commands would be executed for the Number of Iterations specified for the {Loop Begin} command.
LOOPEND	Ends the list of commands that will be executed by the preceding {Loop Begin} command.

Arguments for id = BIND

serverAddr

IP address or host name of the LDAP server that the client will bind to. To specify the port number, add the suffix ":<port number>" to the address or host name. If you do not specify a port number, IxLoad sends the request to the default LDAP port, 389. (Default = "1.2.3.4").

name

Name of directory object that the client will bind as. (Default = "c = US,o = IXIA").

authType

Authentication method. The choices for `authType` are:

<code>authType</code>	Description
"ANONYMOUS"	No authentication. Anonymous authentication is most often used for public read-only directories.
"CLEARTEXT Password"	(Default) Authentication is by user name and password, transmitted unencrypted.
"DIGEST-MD5"	Authentication is by user name and password, transmitted as a SASL MD5 digest. This method provides client authentication with protection against passive eavesdropattacks, but does not provide protection against active intermediary attacks.

`password`

Password of the user who wishes to bind. (Default = "ixia").

`bindControls`

List of optional controls to extend the functionality of the BIND command. See `Control` for a description of how to define a control.

Arguments for `id = UNBIND`

`serverAddr`

IP address or host name of the LDAP server that the client will bind to. To spec the port number, add the suffix ":< port number>" to the address or host name . If you do not specify a port number, IxLoad sends the request to the default LDAP port, 389. (Default = "1.2.3.4").

`unbindControls`

List of optional controls to extend the functionality of the UNBIND command. See `Control` for a description of how to define a control.

Arguments for `id = SEARCH`

`serverAddr`

IP address or host name of the LDAP server that the client will bind to. To spec the port number, add the suffix ":< port number>" to the address or host name . If you do not specify a port number, IxLoad sends the request to the default LDAP port, 389. (Default = "1.2.3.4").

`baseObject`

LDAP DN is the root of the subtree to be searched. (Default = "c = US,o = IXIA").



scope

Restricts the search to specific portions of the LDAP directory. The choices are:

Value	Description
0	Base object: Only the DN specified in the Base Object field is searched.
1	Single Level: All fields at the level specified in the Base Object field are searched.
2	(default) Whole subtree: All fields at the level specified in the Base Object field and below it are searched.

derefAliases

Indicates how aliases are to be handled. In LDAP, one entry may point to another object in the namespace. This is called an `alias` entry, and it contains the DN of the object that it points to. If you look up an object using the alias, the alias is de-referenced so that what is returned is the object pointed to by the alias's DN. The choices are:

Value	Description
0	(default) Never dereference aliases.
1	Dereference aliases after performing name resolution.
2	Dereference during name resolution.
3	Always dereference aliases.

sizeLimit

Maximum number of entries to be returned. Minimum = "0," Maximum = "2,147,483,647." (Default = "10").

timeLimit

Maximum time allowed for search, in seconds. Minimum = "0," Maximum = "2,147,483,647." (Default = "5").

typesOnly

Determines whether the contents of the search results contain attributes and value only attributes: The choices are:

Value	Description
0	Both type and value.
1	(default) Only attribute type.

filter

Search filter. RFC 2254 defines the filter representation. `minLength = "5" Default = "(objectClass=*)"`.

`searchAttributeList`

List of attributes for the search. `searchAttributeList` is of type `ixConfigSequenceContainer`; items are added to the list via the `appendItem` sub-command. See the following example:

```
$attribList searchAttribList.appendItem \  
    -attrib          cn
```

See `Attribute` for a description of how to configure an attribute.

`searchControls`

List of optional controls to extend the functionality of the `SEARCH` command. See `Control` for a description of how to define a control.

Arguments for `id = COMPARE`

`serverAddr`

IP address or host name of the LDAP server that the client will bind to. To specify the port number, add the suffix “:< port number>” to the address or host name. If you do not specify a port number, `IxLoad` sends the request to the default LDAP port, 389. (Default = “1.2.3.4”).

`entry`

Name of the entry to be compared. (Default = “c = US,o = IXIA”).

`attributeDesc`

Attribute that is the object of the comparison.

`assertionValue`

Attribute value that is the object of the comparison.

`compareControls`

List of optional controls to extend the functionality of the `COMPARE` command. See `Control` for a description of how to define a control.

Arguments for `id = MODIFY`

`serverAddr`



IP address or host name of the LDAP server that the client will bind to. To specthe port number, add the suffix “:< port number>” to the address or host name. If you do not specify a port number, IxLoad sends the request to the default LDAP port, 389. (Default = "1.2.3.4").

directoryObject

Directory Object to be modified. (Default = "c = US,o = IXIA").

modificationList

The list of modifications to be performed. See `Modification` for the description of a modification.

modifyControls

List of optional controls to extend the functionality of the `MODIFY` command. See `Control` for a description of how to define a control.

Arguments for id = ADD

serverAddr

IP address or host name of the LDAP server that the client will bind to. To specthe port number, add the suffix “:< port number>” to the address or host name. If you do not specify a port number, IxLoad sends the request to the default LDAP port, 389. (Default = "1.2.3.4").

entry

Name of the entry to be compared. (Default = "c = US,o = IXIA").

attribDescValueList

List of attributes and values to be added. See `Attribute Type and Values` for the description of adding an attribute type and values.

addControls

List of optional controls to extend the functionality of the `ADD` command. See `Control` for a description of how to define a control.

Arguments for id = DELETE

serverAddr

IP address or host name of the LDAP server that the client will bind to. To specthe port number, add the suffix “:< port number>” to the address or host name . If you do not specify a port number, IxLoad sends the request to the default LDAP port, 389. (Default = "1.2.3.4").

entry

Name of the entry to be compared. (Default = "c = US,o = IXIA").

deleteControls

List of optional controls to extend the functionality of the DELETE command. See `Control` for a description of how to define a control.

Arguments for id = MODIFYDN

serverAddr

IP address or host name of the LDAP server that the client will bind to. To specify the port number, add the suffix ":< port number>" to the address or host name. If you do not specify a port number, IxLoad sends the request to the default LDAP port, 389. (Default = "1.2.3.4").

entry

Name of the entry to be compared. (Default = "c = US,o = IXIA").

newRDN

Relative Distinguished Name (RDN) that will form the leftmost component of the new name of the entry. (Default = "c = US,o = IXIA").

deleteoldrdn

Indicates whether the old RDN attribute values are to be deleted. The choices are:

Value	Description
0	(default) False
1	True

newSuperiorPresent

Indicates whether a new superior DN is to be added. Specify the DN in the `newS` parameter.

Value	Description
0	(default) False
1	True

newSuperior



If `newSuperiorPresent` is true, this is the DN of the entry that becomes the immediate superior of the new entry. If `newSuperiorPresent` is false, this parameter has no effect. (Default = "c = US,o = IXIA").

`modifydnControls`

List optional of controls to extend the functionality of the MODIFYDN comSee `Control` for a description of how to define a control.

Arguments for id = THINK

`minimumInterval`

Minimum length of time to pause. Minimum = "1,000," Maximum = "2,147,483,647." (Default = "1,000").

`maximumInterval`

Maximum length of time to pause. Minimum = "1000," Maximum = "2,147,483,647." (Default = "1,000").

Arguments for id = LOOPBEGIN

`iterations`

Number of times to iterate. Value 0 (zero) is treated as infinity. Minimum = "0" Maximum = "2,147,483,647." (Default = "5").

Arguments for id = LOOPEND

None.

EXAMPLE

```
$Activity_LDAPClient1 agent.pm.cmdList.appendItem \  
-id "BIND" \  
-authType "CLEARTEXT Password" \  
-password "ixia" \  
-name "c=US,o=IXIA" \  
-serverAddr "1.2.3.4" \  
$Activity_LDAPClient1 agent.pm.cmdList(0).bindControls.clear
```

SEE ALSO

LDAP Client Agent



Global Options

Global Options - configure an LDAP client's global options

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_LDAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_LDAPClient1 agent.pm.globalOptions.config
```

DESCRIPTION

An LDAP client's global options are set by modifying the options of the `pm.Glooption` of the `LDAP Client Agent` object using its `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`version`

Version of the LDAP protocol used for all client sessions. IxLoad supports ver2 and 3. (Default = 3).

`persistentConnection`

If true, the client opens a new TCP connection for each command sent. The choices are:

Value	Description
0	False. If "0" is specified, then the client will not reuse the TCP connection to send a request. It will close the existing connection and open a new connection for sendeach request.
1	(default) True If "1" is specified, then the client will use the existing TCP connection, if one exists. If the TCP connection does not exist, then a new TCP connection will be estab

`maxRequestsPerConn`

Maximum number of requests that can be sent on each connection. On exceeding this limit, the existing connection will be closed and a new one will be opened to send the next request. This parameter is effective only when `persistentConnec` is "1." Minimum = 1, Maximum = 2,147,483,647. (Default = 100).

`maxConcurrentConnPerUser`

Maximum number of concurrent connections per user. Minimum = 1, Maximum = 2,147,483,647.
(Default = 10").

followReferral

If true ("On"), and the client receives a response that is a referral to another server, it redirects the request to the referred server. The choices are:

Value	Description
0	Off
1	(default) On

commandTimeout

Time (in ms) to wait for a response before aborting. Minimum = 1, Maximum = 2,147,483,647.
(Default = 10,000).

mustBind

If true, the client sends an implicit BIND on every new connection that it establishes. The choices are:

Value	Description
0	False.
1	(default) True. If set to "1," the client sends an implicit BIND as the first Protocol Data Unit on every new connection that it establishes, provided that a user-configBIND is not the next command to be sent.

Arguments for id = mustBind

name

Name of directory object that the client will bind as. (Default = "c = US,o = IXIA").

authType

Authentication method. The choices for authType are:

Value	Description
"ANONYMOUS"	No authentication. Anonymous authentication is most often used for public read-only directories.
"CLEARTEXT Password"	(Default) Authentication is by user name and password, transmitted unencrypted.



"DIGEST-MD5"	Authentication is by user name and password, transmitted as a SASL MD5 digest. This method provides client authentication with protection against passive eavesdropattacks, but does not provide protection against active intermediary attacks.
--------------	--

password

Password of the user who wishes to bind. (Default = "ixia").

EXAMPLE

```
$Activity_LDAPClient1 agent.pm.globalOptions.config \  
-initiateCloseFromClient           true \  
-commandTimeout                    10000 \  
-mustBind                           1 \  
-persistentConnection              1 \  
-maxRequestsPerConn                10000 \  
-authType                          "CLEARTEXT Password" \  
-version                            3 \  
-maxConcurrentConnPerUser          10 \  
-followReferral                     1 \  
-password                           "ixia" \  
-implicitLoopCheck                  true \  
-name                               "c=US,o=IXIA"
```

SEE ALSO

LDAP Client Agent

Control

Control—An optional control to be included with a command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_LDAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_LDAPClient1 agent.pm.cmdList(0).bindControls.appendItem
```

DESCRIPTION

Configures a control to be included in a list associated with a command. A conlist is of type `ixConfigSequenceContainer`; items are added to the list via the `appendItem` subcommand.

SUBCOMMANDS

None.

OPTIONS

`controlType`

LDAP OID of the control associated with the command.

`criticality`

If `true`, the control is critical. The choices are:

Value	Description
0	False.
1	(default) True.

`controlValue`

Value for control.

EXAMPLE

```
$Activity_LDAPClient1 agent.pm.cmdList(0).bindControls.appendItem \  
-id "Control" \  
-controlValue "2" \  
-controlType "1" \  
-criticality 1
```

SEE ALSO

Command List

Modification

Modification—Configures a modification by the MODIFY command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_LDAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_LDAPClient1 agent.pm.cmdList(1).modificationList.appendItem
```

DESCRIPTION

A modification to be included in the `modificationList` of the MODIFY command. The `modificationList` is of type `ixConfigSequenceContainer`; items are added to the list via the `appendItem` subcommand.

SUBCOMMANDS

None.

OPTIONS

`operation`

Type of modification to be performed. The choices are:

Value	Description
0	Add.
1	Delete.
2	Replace.

`type`

Attribute to be modified. RFC 2256 describes the LDAP attributes.

`valueList`

List of values for the operation. Use semicolons (;) to separate multiple values. See the following example: `"value1; value."` (Default = {}).

EXAMPLE

```
$Activity_LDAPClient1 agent.pm.cmdList.appendItem \  
-id "MODIFY" \  
-serverAddr "1.2.3.4" \  
-valueList "value1; value."
```



```
-directoryObject                "c=US,o=IXIA"

$Activity_LDAPClient1 agent.pm.cmdList(1).modificationList.clear

$Activity_LDAPClient1 agent.pm.cmdList(1).modificationList.appendItem \

  -id                            "Modification" \
  -operation                       0 \
  -type                            "1" \
  -valueList                       "1;2;3"

$Activity_LDAPClient1 agent.pm.cmdList(1).modifyControls.clear

$Activity_LDAPClient1 agent.pm.cmdList(1).modifyControls.appendItem \

  -id                            "Control" \
  -controlValue                     "2" \
  -controlType                       "1" \
  -criticality                       1
```

SEE ALSO

Command List

Attribute

Attribute—Configures an attribute for the SEARCH command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_LDAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_LDAPClient1 agent.pm.cmdList(2).searchAttributeList.appendItem
```

DESCRIPTION

An attribute to be included in the `searchAttributeList` of the SEARCH command. The `searchAttributeList` is of type `ixConfigSequenceContainer`; items are added to the list via the `appendItem` subcommand.

SUBCOMMANDS

None.

OPTIONS

`attrib`

Attribute. RFC 2256 describes the LDAP attributes.

EXAMPLE

```
$Activity_LDAPClient1 agent.pm.cmdList.appendItem \  
-id "SEARCH" \  
-typesOnly 1 \  
-filter "(objectClass=*)" \  
-baseObject "c=US,o=IXIA" \  
-derefAliases 0 \  
-timeLimit 5 \  
-serverAddr "1.2.3.4" \  
-sizeLimit 10 \  
-scope 2  
  
$Activity_LDAPClient1 agent.pm.cmdList(2).searchControls.clear  
$Activity_LDAPClient1 agent.pm.cmdList(2).searchControls.appendItem \  

```



```
-id "Control" \  
-controlValue "2" \  
-controlType "1" \  
-criticality 1  
  
$Activity_LDAPClient1 agent.pm.cmdList(2).searchAttributeList.clear  
$Activity_LDAPClient1 agent.pm.cmdList(2).searchAttributeList.appendItem \  
-id "Attribute" \  
-attrib "authorityRevocationList"
```

SEE ALSO

Command List

Attribute Type and Values

Attribute Type and Values—Configures an attribute and values for the ADD command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_LDAPClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_LDAPClient1 agent.pm.cmdList(3).attribDescValueList.appendItem
```

DESCRIPTION

An attribute and one or more values to be included in the `attribDescValueList` of the ADD command. The `attribDescValueList` is of type `ixConfigSequenceContainer`; items are added to the list via the `appendItem` sub-command.

SUBCOMMANDS

None.

OPTIONS

`type`

Type of attribute. RFC 2256 describes the LDAP attributes.

`valueList`

List of values for the attribute. Use semi-colons (;) to separate multiple values. See the following example: "value1; value2". (Default = {}).

EXAMPLE

```
$Activity_LDAPClient1 agent.pm.cmdList.appendItem \  
-id "ADD" \  
-entry "c=US,o=IXIA" \  
-serverAddr "1.2.3.4" \  
$Activity_LDAPClient1 agent.pm.cmdList(3).attribDescValueList.clear \  
$Activity_LDAPClient1 agent.pm.cmdList(3).attribDescValueList.appendItem \  
-id "AttributeTypeAndValues" \  
-type "1" \  
-valueList "1;2;3"
```

SEE ALSO

Command List

LDAP Statistics

The table below describes the LDAP statistics.

Statistic	Description
LDAP Total Requests Sent	Total number of requests of all types sent. This statistic is the total of Total number of responses received + Total number of requests aborted.
LDAP Total Requests Sent per sec	Rate at which client sent LDAP requests.
LDAP Total Responses Received	Total number of LDAP responses of all types received.
LDAP Total Responses Received per sec	Rate at which client received LDAP responses.
LDAP Total Requests Aborted	Total number of requests of all types for which no response was received within the time limit.
LDAP BIND Requests Sent	Total number of BIND requests sent. This statistic is the total of Total number of BIND responses received + Total number of BIND requests aborted.
LDAP BIND Responses Received	Total number of BIND responses of all types received.
LDAP BIND Requests Aborted	Total number of BIND requests for which no response was received within the time limit.
LDAP BIND Success Responses	Total number of successful responses to BIND commands received by the client.
LDAP BIND Failure Responses	Total number of failure responses to BIND commands received by the client.
LDAP BIND Responses With Referrals	Total number of responses to BIND commands that contained referrals.
LDAP UNBIND Requests	Total number of UNBIND requests sent by the client.
LDAP SEARCH Requests Sent	Total number of SEARCH requests sent by the client.

LDAP SearchResultDone Responses Received	<p>Total number of responses received indicating that the search was completed.</p> <p>This statistic is a total of = Total number of SearchResultDone with success + Total number of SearchResultDone with failure + Total number of SearchResultDone with sizelimit exceeded error + Total number of SearchResultDone with timelimit exceeded error + Total number of SearchResultDone with referral.</p>
LDAP Search Requests Aborted	Total number of Search requests for which no response was received within the time limit specified on the Global tab.
LDAP SearchResultDone Responses With Success	Total number of successful searches completed.
LDAP SearchResultDone Responses With Failure	Total number of failed searches completed.
LDAP SearchResultDone Responses With Sizelimit Exceeded Error	Total number of searches completed whose results exceeded the size limit.
LDAP SearchResultDone Responses With Timelimit Exceeded Error	Total number of searches completed that exceeded the time limit specified on the SEARCH form.
LDAP SearchResultDone Responses With Referrals	Total number of searches completed that contained referrals.
LDAP SearchResult Entries Received	Total number of entries received in response to searches. One search may return zero, one, or more than one entries.
LDAP SearchResult Entries Received per sec	Total number of referrals received in response to searches. One search may return zero, one, or more than one referrals.
LDAP ADD Requests Sent	<p>Total number of Add requests sent by the client.</p> <p>This statistic is a total of Total number of Add responses received + Total number of Add requests aborted.</p>
LDAP ADD Responses Received	Total number of Add responses received by the client.
LDAP ADD Requests Aborted	Total number of Add requests for which no response was received within the time limit.
LDAP ADD Success Responses	Total number of responses received indicating that an Add request succeeded.
LDAP ADD Failure Responses	Total number of responses received indicating that an Add request failed.
LDAP ADD Responses With Referrals	Total number of responses to Add requests that contained a referral.



LDAP MODIFY Requests Sent	Total number of Modify requests sent by the client. This statistic is the total of: Total number of Modify responses received + Total number of Modify requests aborted.
LDAP MODIFY Responses Received	Total number of responses to Modify requests received by the client.
LDAP MODIFY Requests Aborted	Total number of Modify requests for which no response was received within the time limit.
LDAP MODIFY Success Responses	Total number of responses received indicating that a Modify request succeeded.
LDAP MODIFY Failure Responses	Total number of responses received indicating that a Modify request failed.
LDAP MODIFY Responses With Referrals	Total number of responses to Modify requests that contained a referral.
LDAP DELETE Requests Sent	Total number of Delete requests sent by the client. This statistic is the total of: Total number of Delete responses received + Total number of Delete requests aborted.
LDAP DELETE Responses Received	Total number of Delete responses received by the client.
LDAP DELETE Requests Aborted	Total number of Delete requests for which no response was received within the time limit.
LDAP DELETE Success Responses	Total number of responses received indicating that a Delete request succeeded.
LDAP DELETE Failure Responses	Total number of responses received indicating that a Delete request failed.
LDAP DELETE Responses With Referrals	Total number of responses to Delete requests that contained a referral.
LDAP MODIFYDN Requests Sent	Total number of ModifyDN requests sent by the client. This statistic is the total of: Total number of ModifyDN responses received + Total number of ModifyDN requests aborted.
LDAP MODIFYDN Responses Received	Total number of ModifyDN responses received by the client.
LDAP MODIFYDN Requests Aborted	Total number of ModifyDN requests for which no response was received within the time limit.
LDAP MODIFYDN Success Responses	Total number of responses received indicating that a ModifyDN request succeeded.
LDAP MODIFYDN Failure Responses	Total number of responses received indicating that a ModifyDN request failed.

LDAP MODIFYDN Responses With Referrals	Total number of responses to ModifyDN requests that contained a referral.
LDAP COMPARE Requests Sent	Total number of Compare requests sent by the client. This statistic is the total of: Total number of Compare responses received + Total number of Compare requests aborted.
LDAP COMPARE Responses Received	Total number of Compare responses received by the client.
LDAP COMPARE Requests Aborted	Total number of Compare requests for which no response was received within the time limit.
LDAP COMPARE Responses With Result TRUE	Total number of responses indicating that the string in the Compare request existed in the directory.
LDAP COMPARE Responses With Result FALSE	Total number of responses indicating that the string in the Compare request did not exist in the directory.
LDAP COMPARE Failure Responses	Total number of responses received indicating that a Compare request failed.
LDAP COMPARE Responses With Referrals	Total number of responses to Compare requests that contained a referral.
LDAP Notice Of Disconnection Received	Total number of Notices of Disconnection received by the client.
LDAP BIND Response Time	Average time elapsed between sending a Bind request and receiving a complete response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
LDAP SEARCH Response Time	Average time elapsed between sending a Search request and receiving a SearchResultDone response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
LDAP ADD Response Time	Average time elapsed between sending an Add request and receiving a complete response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.

LDAP MODIFY Response Time	<p>Average time elapsed between sending a Modify request and receiving a complete response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
LDAP DELETE Response Time	<p>Average time elapsed between sending a Delete request and receiving a complete response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
LDAP MODIFYDN Response Time	<p>Average time elapsed between sending a ModifyDN request and receiving a complete response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
LDAP COMPARE Response Time	<p>Average time elapsed between sending a Compare request and receiving a complete response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
LDAP Avg Time To Receive First Byte Of Response	<p>Average time required to receive the first byte of a SEARCH response. The time is averaged because a SEARCH command may return multiple responses.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
LDAP Avg Time To Receive Last Byte Of Response	<p>Average time required to receive the last byte of a SEARCH response. The time is averaged because a SEARCH command may return multiple responses.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
LDAP Simulated Users	Number of users simulated during the test.

LDAP Connections Established	Total number of LDAP connections established. An LDAP connection occurs when an LDAP client successfully connects to an LDAP server.
LDAP Connections Established per sec	Rate at which the client established LDAP connections
LDAP Active Connections	Number of simultaneous LDAP connections established.
LDAP Total Transactions	Total number of LDAP transactions completed by the client. An LDAP transaction occurs when an LDAP client sends a request to an LDAP server and receives a response, either of success or failure.
LDAP Transactions per sec	Rate at which the client completed LDAP transactions.
LDAP Concurrent Sessions	Number of simultaneous LDAP sessions in progress. An LDAP session occurs when an LDAP client successfully connects to an LDAP server.
LDAP Total Bytes Transmitted	Total number of bytes sent by the client in LDAP requests. This statistic counts only the bytes in the payload portion of the LDAP packets.
LDAP Total Bytes Received	Total number of bytes received by the client in LDAP responses.
LDAP Total Bytes Transmitted per sec	Rate at which the client transmitted bytes in LDAP requests.
LDAP Total Bytes Received per sec	Rate at which the client received bytes in LDAP responses.
LDAP Total Bytes Sent and Received	Combined total of bytes sent and received in LDAP requests and responses.
LDAP Throughput	Total throughput over the LDAP connections, in bytes per second.

19

Peer-to-Peer Application

This section describes the Peer-to-Peer Application Tcl API objects.

In this chapter:

Overview	1368
Peer-to-Peer Application Agent	1370
Peer-to-peer Global Statistics	1378

Overview

The IxLoad Peer-to-Peer Application API consists of the Peer-to-Peer ApplicaAgent and its commands, shown in Figure 31-1.

Figure 31-1. Peer-to-Peer ApplicationReplay API Structure

Objectives

The objectives (userObjective) you can set for Peer-to-Peer are listed below. Test objectives are set in the ixTimeline object.

- simulatedUsers
- peerCount (displays as “Initiator Peer Count” in the GUI)
- connectionRate
- concurrentConnections
- throughputMBps
- throughputKBps
- transactionRate

Peer-to-Peer Application Agent Commands

This sections the Peer-to-Peer Application commands.

Peer-to-Peer Application Agent

The Peer-to-peer (P2P) Application Agent simulates traffic in a peer-to-peer file-sharing network using several of the common file-sharing protocols, such as Bitand eDonkey. A P2P Application agent can be configured as an initiator, a responder, or both.

Refer to *Peer-to-Peer Application Agent* on page 31-4 for a full description of this command.

The most significant options of this command are:

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time .



Peer-to-Peer Application Agent

Peer-to-Peer Application Agent - create a peer-to-peer agent

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_P2PApplicationPeer1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_P2PApplicationPeer1 agent.config
```

DESCRIPTION

An Activity_P2PApplicationPeer1 agent is added to the activityList object. The activityList object is added to the ixNetTraffic object using the appendItem subfrom the ixConfigSequenceContainer command.

SUBCOMMANDS

The options for this command are configured and read using the standard config, cget, and getOptions subcommands defined in the ixConfig command.

OPTIONS

enable

Enables the use of this client agent. (Default = true).

name

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity Activity_P2PApplicationPeer1 of NetTraffic Traffic1@Network1
#####
set Activity_P2PApplicationPeer1 [$Traffic1_Network1 activityList.appendItem
\
    -protocolAndType                "p2papp Peer" ]
```

```
$Activity_P2PApplicationPeer1 agent.config \  
-enable true \  
-name "P2PApplicationPeer1"
```

SEE ALSO

`ixNetTraffic`



FlowDefinition

FlowDefinition—Defines a list of of P2P flows.

SYNOPSIS

```
set ServerTraffic1_ServerNetwork1 [::IxLoad new ixNetTraffic]
set Activity_P2PApplicationPeer1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_P2PApplicationPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the list of protocol flows using the appendItem subcommand from the ixConfigSequenceContainer command.

SUBCOMMANDS

None

OPTIONS

remotePeer

P2P activity that is the destination of traffic sent by this peer, and the origin of traffic received by it.

responderPort

Port number that responding peer listens on (Default=10000).

EXAMPLE

```
Activity_P2PApplicationPeer1 agent.pm.protocolFlows.clear
$Activity_P2PApplicationPeer1 agent.pm.protocolFlows.appendItem \
    -id                "InbuiltFlow" \
    -remotePeer        "Traffic2_P2PApplicationPeer2" \
    -subType           "Bittorrent" \
    -flowType          "Simple Bidirectional" \
    -responderPort     10000
```

SEE ALSO

InbuiltFlow



InbuiltFlow

InbuiltFlow —Defines the parameters of an inbuilt P2P flow.

SYNOPSIS

```
$Activity_P2PApplicationPeer1 agent.pm.protocolFlows.appendItem
```

DESCRIPTION

An option is added to the `ProtocolFlows` list of using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None

OPTIONS

`subType`

The peer-to-peer protocol type:

- Bittorrent
- e-donkey

`flowType`

A peer-to-peer flow type defined as one of the following:

- Simple Download (Bittorent)
- Simple Upload (Bittorent)
- Simple Bidirectional (Bittorent)
- Bidirectional to Download (Bittorent)
- Bidirectional to Long Download (Bittorent)
- Bidirectional to Upload (Bittorent)
- Bidirectional to Long Upload (Bittorent)
- Download to Upload (Bittorent)
- Download to Long Upload (Bittorent)
- Download to Bidirectional (Bittorent)
- Download to Long Bidirectional (Bittorent)
- SimpleControlFlow-1 (e-donkey)
- SimpleControlFlow-2 (e-donkey)
- Simple Download (e-donkey)
- Download to Upload (e-donkey)

- Download to Long Upload (e-donkey)
- Download to Bidirectional (e-donkey)
- Download to Long Bidirectional (e-donkey)
- Simple Upload (e-donkey)
- Simple Upload to Download (e-donkey)
- Upload to Bidirectional (e-donkey)
- Simple Bidirectional (e-donkey)
- Unknown Direction-1, Unknown Direction-2, Unknown Direction-3 (e-donkey)
- Queued (e-donkey)
- Queued Callback (e-donkey)
- Queued Callback to Download (e-donkey)
- Queued Callback to Bidirectional (e-donkey)

dataSegments

A list of DataSegment objects.

EXAMPLE

```
Activity_AppReplayPeer1 agent.pm.protocolFlows.clear
$Activity_P2PApplicationPeer1 agent.pm.protocolFlows.appendItem \
    -id          "InbuiltFlow" \
    -remotePeer  "Traffic2_P2PApplicationPeer2" \
    -subType     "Bittorrent" \
    -flowType    "Simple Bidirectional" \
    -responderPort 10000
```

SEE ALSO

FlowDefinition



DataSegment

DataSegment—Defines an inbuilt flow's data.

SYNOPSIS

```
$Activity_P2PApplicationPeer1  
agent.pm.responder.responderFlows(0).dataSegments.clear  
$Activity_P2PApplicationPeer1  
agent.pm.responder.responderFlows(0).dataSegments.appendItem
```

DESCRIPTION

An option is added to the options list using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None

OPTIONS

`dataDirection`

The flow data direction (read-only), evaluated on the basis of flow type:

Value	Description
0	Upload
1	Download
2	Bidirectional

`bidirectionMode`

If the `dataDirection` parameter has been selected as `Bidirectional`, this parameter can be set as:

- Parallel
- Upload first
- Download first

`uploadSizeRange`

The size of payload to be uploaded in case of an upload or bidirectional flow type. When the flow type is download, this parameter cannot be set.

`uploadDataUnit`

The units for the upload data size. One of the following:

- KB

- MB
- GB

downloadSizeRange

The size of payload to be downloaded in case of a download or bidirectional flow type. When the flow type is upload, this parameter cannot be set.

downloadDataUnit

The units for the download data size. One of the following:

- KB
- MB
- GB

EXAMPLE

```
$Activity_P2PApplicationPeer1
agent.pm.responder.responderFlows(0).dataSegments.clear
```

```
$Activity_P2PApplicationPeer1
agent.pm.responder.responderFlows(0).dataSegments.appendItem \

    -id                    "DataSegment" \
    -uploadSizeRange       "N/A" \
    -downloadSizeRange     "10000-1000000000" \
    -downloadDataUnit      "Bytes" \
    -uploadDataUnit        "N/A" \
    -bidirectionMode       "N/A" \
    -dataDirection         1
```

SEE ALSO

Peer-to-peer Global Statistics

The following table describes the Peer-to-peer statistics.

Statistic	Description
Test Objective Statistics	
P2P Application Initiator Peer Count	Number of P2P initiators created.
P2P Application Responder Peer Count	Number of P2P responders created.
P2P Application Concurrent Sessions	Number of concurrent sessions established between peers.
P2P Application Connection Rate	Rate (in connections per second) at which P2P peers connected to each other.
P2P Application Transaction Rate	<p>Rate (in transactions per second) at which P2P peers completed transactions.</p> <p>For P2P peers, transactions consist of exchanges of request-response control byte codes.</p> <p>A transaction begins when an initiator sends the first control byte code, and ends when the responder sends a control byte code in response.</p> <p>If a response requires multiple control byte codes, the transaction ends when the responder sends the final byte code.</p>
P2P Application Initiator Total Bytes Sent/sec	Rate at which the initiators sent data.
P2P Application Initiator Total Bytes Received/sec	Rate at which the initiators received data.
P2P Application Initiator Total Throughput	Combined rate at which the initiators sent and received data.
P2P Application Responder Total Bytes Sent/sec	Rate at which the responders sent data.
P2P Application Responder Total Bytes Received/sec	Rate at which the responders received data.
P2P Application Responder Total Throughput	Combined rate at which the responders sent and received data.
Total Connection Statistics	
P2P Application Connection Requests Sent	Number of connection requests sent by the initiators to the responders.
P2P Application Connection Requests Successful	Number of connection attempts that succeeded.
P2P Application Connection Requests Failed	Number of connection attempts that failed.
P2P Application Active Connections	Number of connections currently active.
P2P Application Connection Requests Received	Number of connection requests received by the responders.

P2P Application Connections Accepted	<p>Number of connections accepted by the responders.</p> <p>This statistic measures the number of successful connections from the point of view of the responder.</p>
P2P Application Connections Failed	<p>Number of connections that were established but then closed because they would have exceeded the maximum number of connections that the responder could support.</p> <p>The maximum number of connections that the responder can accept is calculated based on the test configuration and depends on the resources available on the load module, such as memory.</p>
Total Transaction Statistics	
P2P Application Total Transactions Initiated	Total number of P2P transactions initiated.
P2P Application Total Transactions Successful	Total number of P2P transactions that succeeded.
Total Flow Statistics	
P2P Application Total Flow Initiated	Total number of control and data flows initiated.
P2P Application Total Active Flow	Total number of control and data flows active.
P2P Application Total Flow Succeeded	Total number of flows of control and data that completed successfully.
P2P Application Total Flow Failed	Total number of control and data flows that failed for any reason.
P2P Application Total Flow Failed Error	Total number of control and data flows that failed due to a network error.
P2P Application Total Flow Failed Timeout	Total number of control and data flows that failed due to a timeout.
P2P Application Total Flow Failed Mismatch	Total number of control and data flows that failed because the data sent did not match the data that was expected.
P2P Application Total Flow Aborted	Number of P2P sessions that ended abnormally.
Initiator Total Bytes Statistics	
P2P Application Initiator Total Bytes Sent	Total number of bytes sent by the initiators.
P2P Application Initiator Total Bytes Received	Total number of bytes received by the initiators.
P2P Application Initiator Total Bytes Sent and Received	Combined total of bytes sent and received by the initiators.
Responder Total Bytes Statistics	
P2P Application Responder Total Bytes Sent	Total number of bytes sent by the responders.
P2P Application Responder Total Bytes Received	Total number of bytes received by the responders.
P2P Application Responder Total Bytes Sent and Received	Combined total number of bytes sent and received by the responders.

Control Tx/Rx Statistics	
P2P Application Control Segment Transmission Initiated	Number of control flows established. A control flow is the series of messages exchanged between peers before beginning the data flow. Control flows can also sometimes occur between data flows.
P2P Application Control Segment Transmission Succeeded	Number of control flows that succeeded (Initiator side).
P2P Application Control Segment Transmission Failed	Total number of control flows that failed for any reason (Initiator side).
P2P Application Control Segment Transmission Failed (Error)	Number of control flows that failed due to a network error (Initiator side).
P2P Application Control Segment Transmission Failed (Timeout)	Number of control flows that failed due to a timeout (Initiator side).
P2P Application Control Segment Reception Initiated	Number of control flows that the responders are receiving.
P2P Application Control Segment Reception Succeeded	Number of control flows that completed successfully (Responder side).
P2P Application Control Segment Reception Failed	Number of control flows that failed to complete for any reason (Responder side).
P2P Application Control Segment Reception Failed (Error)	Number of control flows that failed to complete due to a network error (Responder side).
P2P Application Control Segment Reception Failed (Timeout)	Number of control flows that failed to complete due to a timeout (Responder side).
P2P Application Control Segment Reception Failed (Mismatch)	Number of control flows that failed to complete because the data sent did not match the data that was expected (Responder side).
Data Tx/Rx Statistics	
P2P Application Data Segment Transmission Initiated	Number of data flows currently active. A data flow is the stream of related payload data sent from an initiator or a responder. For example, if, in the Data Definition table, a Simple Upload flow is selected and the Upload Data Size is 4096 bytes, then the transmission of 4096 bytes of data from initiator to responder constitutes one successful data flow.
P2P Application Data Segment Transmission Succeeded	Number of data flows that completed successfully (Initiator side).
P2P Application Data Segment Transmission Failed	Number of data flows that failed (Initiator side).
P2P Application Data Segment Transmission Failed (Error)	Number of data flows that failed due to a network error (Initiator side).
P2P Application Total Data Segment Transmission Failed (Timeout)	Number data flows that failed due to a timeout (Initiator side).

P2P Application Data Segment Reception Initiated	Number of data flows that the responders are receiving.
P2P Application Data Segment Reception Succeeded	Number of data flows that completed successfully (Responder side).
P2P Application Data Segment Reception Failed	Number of data flows that failed for any reason (Responder side).
P2P Application Data Flow Reception Failed (Error)	Number of data flows that failed due to a network error (Responder side).
P2P Application Data Segment Reception Failed (Timeout)	Number of data flows that failed due to a timeout (Responder side).

20

Packet Monitor

This section describes the Packet Monitor Tcl API objects.



In this chapter:

Overview	1383
PacketMonitor Server Agent	1385
PluginProperties	1387
PacketFilter	1389
ProtocolFilter	1395
PortRange	1412

Overview

The IxLoad PacketMonitor API consists of the PacketMonitor Server Agent and its commands, organized as shown in the figure below.

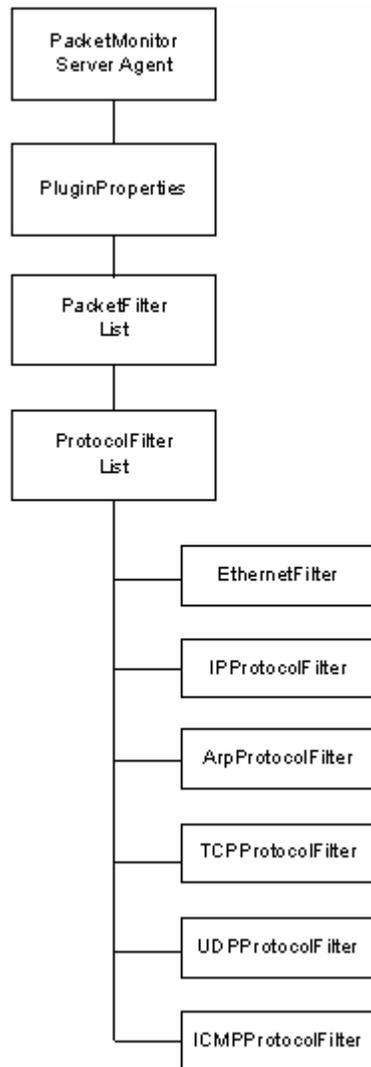


Figure 29-1. PacketMonitor Server API Structure



Packet Monitor Server Agent

The PacketMonitor Server Agent command defines a server that identifies packets in a DDoS attack that the DUT has failed to capture and stop. Refer to `PacketMonitor Server Agent` on page 29-3 for a full description of this command.

The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time .

PacketMonitor Server Agent

PacketMonitor Server Agent - create a Packet Monitor server

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_PacketMonitorServer1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_PacketMonitorServer1 agent.config
```

DESCRIPTION

A PacketMonitor server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this peer agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity_PacketMonitorServer1 of NetTraffic Traffic1@Network1
#####
set Activity_PacketMonitorServer1 [$Traffic1_Network1 activityList.appendItem
\
    -protocolAndType                "packetmonitor Server" ]
```



```
$Activity_PacketMonitorServer1 agent.config \  
-enable          true \  
-name            "PacketMonitorServer1"
```

SEE ALSO

`ixNetTraffic`

PluginProperties

PluginProperties—Configures the list of parameters for a PacketMonitor Server Agent.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_PacketMonitorServer1 [$Traffic1_Network1
activityList.appenoptions...]
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.config
```

DESCRIPTION

This object configures the properties of a PacketMonitor Server Agent.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`allPacketFilters`

A list of `PacketFilter` objects.

`filterSettingsOption`

The method used to configure the PacketMonitor filters, which can be either of the following:

- `Automatic (0)`: Packet Monitor automatically configures itself to monitor the attacks in the selected DDoS client. In this mode, all the filter settings are read-only; you cannot change them.
- `Manual (1)`: Packet Monitor automatically creates the filters needed to monitor the DDoS client's attacks (as in Automatic mode), but in this mode you can edit the filter settings.

`reorderFilter`

If the value of this parameter is 1 (Default=1), the configured filters are reorganized such that a more specific filter is ranked higher than a less specific filter. A filter (for example, `filter1`) is said to be more specific than another filter (for example `filter2`) if any packet that matches `filter1` also matches `filter2`, but not all packets matching `filter2` also match `filter1`.

`DdosClient_ip`

The IP address of the DDoS client that is the origin of the attack packets.

EXAMPLE

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.config \  
-filterSettingsOption          0 \  
-DdosClient_ip                 "None"
```

SEE ALSO

PacketMonitor Server Agent

PacketFilter

PacketFilter—Defines the packet filter parameters.

SYNOPSIS

```
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters.clear
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters.appendItem \
  -id                "PacketFilter" \
  -filterName        "NewSynFloodFilter2" \
  -comments          "Syn Flood Filter"
```

DESCRIPTION

A packet filter is added to the list of PacketFilter objects using the `appendItem` subcommand.

SUBCOMMANDS

None

OPTIONS

`filterName, comments`

The name of the packet filter and the filter description.

`type`

A predefined packetfilter, which can be either of the following:

- SynFloodFilter
- RstFloodFilter
- XMasTreeFilter
- TCPScanFilter
- PingSweepFilter
- PingOfDeathFilter
- UnreachableHostFilter
- SmurfFilter
- EvasiveUDPFilter
- UDPFloodFilter
- UDPScanFilter
- TearDropFilter
- LandFilter

- ArpFloodFilter

When specifying a predefined packetfilter, the packetfilter object is initialized with the default setting pertaining to the filter type, all protocol filters get appended by default and all protocol-related settings, such as the TCP flags for SynFlood Filters, get configured. For predefined packetfilters you need to configure only the IP addresses and the ports, as shown in Example 2 of the `EXAMPLES` section.

`disableProtocolFilters`

If it is enabled (1), Packet Monitor does not use the protocol filters to filter packets, and only signature settings are matched. (Default = 0).



Note: For a PacketFilter object at least one of the protocol filters or the signature settings must be configured.

`protocolFilters`

A list of protocol filters that Packet Monitor has imported from the DDoS client.

`indentifySignature`

If this option is enabled (1), the signature – a string in the payload – is matched. A match occurs if the payload contains this string, beginning at the location specified by the `offset` option (Default = 0).

`signatureMatchType`

The method used to select payloads to be inspected for signatures, which can be either of the following:

- `Must (0)`: A packet is selected only if the length of its payload is greater than or equal to the `size` value.
- `May (1)`: A packet is selected only if the length of its payload is greater than the value of the `size` value.

`offset`

The location (in bytes) in the payload where the signature string begins.

`size`

Length of the signature, in bytes.

`view_type`

The view type, which can be hexadecimal (`view_type=0`) or `ascii` (`view_type=1`). The default is "1".

`hexValue`

The hexadecimal signature string (Default = 49 78 4c 6f 61 64) that is searched for, having a string length between 0 and 2048.

ascValue

The hexadecimal signature string (Default = "IxLoad") that is searched for, having a string length between 0 and 2048.

enablePayloadSize
Check

If this option is selected, the Packet Monitor agent filters out payloads within a certain size range for protocols that include payloads, such as TCP, UDP, and ICMP.

payloadSizeLimits

The payload size range.



Note: You can use the following format to specify the size range, in bytes:

minimum-maximum

The maximum size must be larger than the total of the size and offset fields (size + offset).

EXAMPLES

Example 1

```
$Activity_PacketMonitorServer1 \  
agent.pm.pluginProperties.allPacketFilters.clear  
  
$Activity_PacketMonitorServer1 \  
agent.pm.pluginProperties.allPacketFilters.appendItem \  
-id "PacketFilter" \  
-filterName "NewSynFloodFilter2" \  
-comments "Syn Flood Filter"  
# -enablePayloadSizeCheck 0  
# -view_type 1  
# -ascValue "IxLoad"  
# -signatureMatchType 0  
# -indentifySignature 0  
# -hexValue "49 78 4c 6f 61 64"
```



```
# -offset 0
# -payloadSizeLimits ""
# -size 0
# -disableProtocolFilters 0
```

Example 2

```
$Activity_PacketMonitorServer1 \
agent.pm.pluginProperties.allPacketFilters.appendItem \

  -id "PacketFilter"

$Activity_PacketMonitorServer1 \
agent.pm.pluginProperties.allPacketFilters(0).config \

  -type "SynFloodFilter"

$Activity_PacketMonitorServer1 \
agent.pm.pluginProperties.allPacketFilters(0).config \

  -enablePayloadSizeCheck 0 \
  -view_type 1 \
  -filterName "SynFloodAttack1@DDoSClient1@Traffic1" \
  -ascValue "IxLoad" \
  -comments "Syn Flood Filter-Imported from Ddos Client \
'DDoSClient1@Traffic1'" \
  -signatureMatchType 0 \
  -indentifySignature 0 \
  -hexValue "49 78 4c 6f 61 64" \
  -offset 0 \
  -payloadSizeLimits "1-1" \
  -size 0 \
  -disableProtocolFilters 0
```

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(1). \srcIpRange  
s.clear
```

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(1). \srcIpRange  
s.appendItem \
```

```
-id                "IPRange" \  
-count            8 \  
-stepSize         "0.0.0.1" \  
-firstIp          "198.18.0.1"
```

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(1). \dstIpRange  
s.clear
```

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(1). \dstIpRange  
s.appendItem \
```

```
-id                "IPRange" \  
-count            8 \  
-stepSize         "0.0.0.1" \  
-firstIp          "198.18.0.101"
```

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(2).srcPortRange  
s.clear
```

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(2).srcPortRange  
s.appendItem \
```

```
-id                "PortRanges" \  
-count            1025 \  
-stepSize         1 \  
-firstIp          "198.18.0.101"
```



```
-firstPort 1024
```

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(2).dstPortRange  
s.clear
```

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(2).dstPortRange  
s.appendItem \
```

```
-id "PortRanges" \  
-count 1025 \  
-stepSize 1 \  
-firstPort 1024
```

SEE ALSO

PacketMonitor Server Agent

ProtocolFilter

ProtocolFilter—Defines a protocol filter to be applied to the incoming stream.

SYNOPSIS

```
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.clear
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem
```

DESCRIPTION

The `ProtocolFilter` command configures a filter applied to the incoming packet stream. A protocol filter is added to a `PacketFilter` object using the `appendItem` subcommand.



Note: When adding a particular protocol filter, all lower layer protocol filters need to be added as well. For example, adding an ICMP protocol filter requires that Ethernet and IP protocol filters, with the next layer protocol values properly set, be added as well.

SUBCOMMANDS

None

OPTIONS

None.

EXAMPLE

```
$Activity_PacketMonitorServer1 \
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters. \
appendItem \

  -id                               "EthernetFilter" \

  -ethernetProtocol                 1
```



EthernetFilter

EthernetFilter—Defines an Ethernet protocol filter applied to the incoming packet stream.

SYNOPSIS

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem \  
-id "EthernetFilter"
```

DESCRIPTION

An `EthernetFilter` is an object derived from a `ProtocolFilter` object.

SUBCOMMANDS

None

OPTIONS

`ethernetProtocol`

The next higher-layer protocol inside the packet. A match occurs when the next higher-layer protocol in a packet matches this choice, which can be any of the following:

Value	Description
0	ARP
1	IP
2 (default)	ALL

`srcMacRanges`

The source MAC address range to be filtered.

`dstMacRanges`

The destination MAC address range to be filtered.

EXAMPLE

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters.clear
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters.appendItem \  
-id "EthernetFilter" \  
-ethernetProtocol 1
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters(0).srcMacRanges.clear
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters(0).dstMacRanges.clear
```

SEE ALSO



IPProtocolFilter

IpProtocol Filter—Defines an IP protocol filter applied to the incoming packet stream.

SYNOPSIS

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem \  
-id "IpProtocolFilter"
```

DESCRIPTION

An `IPProtocolFilter` is an object derived from a `ProtocolFilter` object.

SUBCOMMANDS

None

OPTIONS

`useServerNetConfig`

If enabled (1), the Packet Monitor agent uses the destination IP addresses configin the NetTraffic that it runs on (Default=0).

`srcIpRanges`

List of source IP addresses to be filtered. A match occurs when a packet's source IP address falls within this range.

`dstIpRanges`

List of destination IP addresses to be filtered. A match occurs when a packet's destination IP address falls within this range.

`ipProtocol`

The next higher-layer protocol inside the packet. A match occurs when the next higher-layer protocol in a packet matches this choice. For example, when the value corresponds to TCP, a match occurs when a packet is a TCP-over-IP (TCP/IP) packet.

Value	Description
0	Next higher-layer protocol must be TCP (TCP/IP).
1	Next higher-layer protocol must be UDP (UDP/IP).
2	Next higher-layer protocol must be ICMP (ICMP/IP).

Value	Description
3	Next higher-layer protocol can be TCP, UDP, or ICMP.

dropFragmentedPackets

If enabled (1), fragmented IP packets are handled as follows:

- If the packet is the first packet among the fragmented packet series and its contents matches the configured IP options, then its other settings (TCP/UDP/ICMP and signature) are evaluated. If the other settings match, the packet is dropped, otherwise it is allowed.
- If the packet is not the first packet among the fragmented packet series and its contents matches the configured IP options, the packet is dropped immedi

The default value of this parameter is 0.



Note: The filter configuration above the IP layer is applied only to the first packet of the fragmented packet sequence and the fragmented packets are not reassembled.

EXAMPLE

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).
protocolFilters.appendItem \

    -id                "IpProtocolFilter" \

    -useServerNetConfig false \

    -dropFragmentedPackets false \

    -ipProtocol        0
```

SEE ALSO



ArpProtocolFilter

ArpProtocol Filter—Defines an ARP protocol filter applied to the incoming packet stream.

SYNOPSIS

```
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(2).protocolFilters.appendItem \
    -id "ArpProtocolFilter"
```

DESCRIPTION

An `ArpProtocolFilter` is an object derived from a `ProtocolFilter` object.

SUBCOMMANDS

None

OPTIONS

`useServerNetConfig`

If enabled (1), the Packet Monitor agent uses the destination IP addresses configin the NetTraffic that it runs on (Default=0).

`srcIpRanges`

List of source IP addresses to be filtered. A match occurs when a packet's source IP address falls within this range.

`dstIpRanges`

List of destination IP addresses to be filtered. A match occurs when a packet's destination IP address falls within this range.

`srcMacRanges`

List of source MAC address range to be filtered. (Default = 0).

`dstMacRanges`

List of destination MAC address range to be filtered. (Default = 0).

`operationCode`

The Arp operation code (opcode), in decimal format, contained in the ARP packet. A match occurs when a packet contains this opcode. The ARP opcodes are:

1 Request

2 Reply

3 Request Reverse

4 Reply Reverse

EXAMPLE

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters.appendItem \  

```

```
    -id                                "ArpProtocolFilter"
```

```
# -useServerNetConfig                  false
```

```
# -operationCode                       1
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters(1).srcMacRanges.clear
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters(1).dstMacRanges.clear
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters(1).srcIpRanges.clear
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters(1).dstIpRanges.clear
```

SEE ALSO



TCPProtocol Filter

TcpProtocol Filter—Defines a TCP protocol filter to be applied to the incoming stream.

SYNOPSIS

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem
```

DESCRIPTION

A `TcpProtocolFilter` is an object derived from a `ProtocolFilter` object.

SUBCOMMANDS

None

OPTIONS

`srcPortRanges`

List of source port ranges to be filtered. A match occurs when a packet's source port falls within this range.

`dstPortRanges`

List of destination port ranges to be filtered. A match occurs when a packet's desport falls within this range.

`synFlag, rstFlag, finFlag, ackFlag, pshFlag, urgFlag`

These parameters represent the basic TCP flags. To set filtering based on these parameters, a `Flag` parameter needs to be enabled (true), while the the `op` parameter defines the value for which filtering is done.

`Op` type parameters have a default value of "0" (set).

EXAMPLE

```
$Activity_PacketMonitorServer1 \  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem \  
  
-id                "TcpProtocolFilter" \  
  
-urg_op            0 \  
  
-finFlag           false \  
  
-syn_op            0 \  
  
-psh_op            0 \  

```

-ack_op	0 \
-synFlag	true \
-fin_op	0 \
-rstFlag	false \
-urgFlag	false \
-pshFlag	false \
-rst_op	0 \
-ackFlag	false

SEE ALSO



UdpProtocol Filter

UdpProtocol Filter—Defines an UDP protocol filter to be applied to the incomstream.

SYNOPSIS

```
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem \
    -id                                "UdpProtocolFilter"
```

DESCRIPTION

An `UdpProtocolFilter` is an object derived from a `ProtocolFilter` object.

SUBCOMMANDS

None

OPTIONS

`srcPortRanges`

Source ports to be filtered. A match occurs when a packet's source port falls within this range.

`dstPortRanges`

Destination ports to be filtered. A match occurs when a packet's destination port falls within this range.

EXAMPLE

```
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem \
    -id                                "UdpProtocolFilter"

$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters(2).srcPortRange
s.clear

$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).
protocolFilters(2).srcPortRanges.appendItem
    -id                                "PortRanges"
# -count                                1000
# -stepSize                             1
# -firstPort                             2048
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters(2).dstPortRanges.clear
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(0).  
protocolFilters(2).dstPortRanges.appendItem
```

```
    -id                                "PortRanges"  
# -count                               1000  
# -stepSize                             1  
# -firstPort                             2048
```

SEE ALSO



IcmpProtocol Filter

ICMP Protocol Filter—Defines ICMP protocol filter to be applied to the incomstream.

SYNOPSIS

```
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem \
    -id "IcmpProtocolFilter"
```

DESCRIPTION

An `IcmpProtocolFilter` is an object derived from a `ProtocolFilter` object.

SUBCOMMANDS

None

OPTIONS

`icmpType`

ICMP message type to be filtered. A match occurs when an ICMP packet's `mestype` matches this selection. Only the message types defined in RFC 792 and RFC 950 are supported:

Value	Description
0	echo reply (ping)
3	destination unreachable
4	source quench
5	redirect (route change)
8	echo request (ping)
11	time exceeded
12	parameter problem
13	timestamp request
14	timestamp reply
17	address mask request
18	address mask reply

`icmpCode`

Code that further defines ICMP message type. A match occurs when an ICMP packet's `code` matches this selection.

EXAMPLE

```
$Activity_PacketMonitorServer1 \  
agent.pm.pluginProperties.allPacketFilters(0).protocolFilters.appendItem \  
-id "IcmpProtocolFilter" \  
-icmpCode "0" \  
-icmpType 12
```

SEE ALSO



MACAddressRange

MACAddressRange—Defines a protocol filter MAC address range.

SYNOPSIS

```
Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(1).protocolFilters(0).srcMacRanges
.appendItem
```

DESCRIPTION

A MAC address range is added to the ProtocolFilter object using appendItem subcommand.

SUBCOMMANDS

None

OPTIONS

firstMacAddress

The first MAC address in the range (Default="00:00:00:00:00:AA").

stepSize

The step size (Default="00:00:00:00:00:01").

count

The number of MAC addresses (Default = "100").

EXAMPLE

```
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(1).protocolFilters(0).srcMacRanges
.clear
```

```
$Activity_PacketMonitorServer1
agent.pm.pluginProperties.allPacketFilters(1).protocolFilters(0).srcMacRanges
.appendItem
```

```
-id                "MACAddressRange"
# -count           100
# -stepSize        "00:00:00:00:00:01"
# -firstMacAddress "00:00:00:00:00:AA"
```

SEE ALSO



IPAddressRange

IP Address Range—Defines a protocol filter IP address range.

SYNOPSIS

```
$Activity_PacketMonitorServer1  
agent.pm.pluginProperties.allPacketFilters(1).protocolFilters(1).srcIpRanges.  
appendItem
```

DESCRIPTION

An IP address range is added to the `ProtocolFilter` object using its `appendItem` subcommand.

SUBCOMMANDS

None

OPTIONS

`firstIp`

The first IP address in the range (Default="1.1.1.1").

`stepSize`

The step size (Default="0.0.0.1").

`count`

The number of IP addresses (Default="100").

EXAMPLE

```
$Activity_PacketMonitorServer1 \  
agent.pm.pluginProperties.allPacketFilters(1).protocolFilters(1).srcIpRanges.  
clear
```

```
$Activity_PacketMonitorServer1 \  
agent.pm.pluginProperties.allPacketFilters(1).protocolFilters(1).srcIpRanges.  
appendItem
```

```
-id                "IPRange"  
# -count           100  
# -stepSize        "0.0.0.1"  
# -firstIp         "1.1.1.1"
```

SEE ALSO

PortRange

PortRange—Defines a protocol filter port range.

SYNOPSIS

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(1).  
protocolFilters(2).srcPortRanges.appendItem
```

DESCRIPTION

A port range is added to the ProtocolFilter object using its appendItem subcommand.

SUBCOMMANDS

None

OPTIONS

firstPort

The first port in the range (Default = "2048").

stepSize

The step size (Default="1").

count

The number of ports (Default="1000").

EXAMPLE

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(1).  
protocolFilters(2).srcPortRanges.clear
```

```
$Activity_PacketMonitorServer1 agent.pm.pluginProperties.allPacketFilters(1).  
protocolFilters(2).srcPortRanges.appendItem
```

```
    -id                "PortRanges"  
# -count              1000  
# -stepSize           1  
# -firstPort          2048
```

SEE ALSO

21

POP3

This section describes the POP3 Tcl API objects.



In this chapter:

Overview	1415
POP3 Client Agent	1418
POP3 Server Agent	1426
Using Auto-Generated Strings	1432

Overview

POP3 protocol commands are organized as:

- POP3 Client Agent
 - Pop3Command
- POP3 Server Agent
 - MailBoxItem

An additional discussion item is included:

- `Using Auto-Generated Strings`—which pertains to several commands.

Objectives

The objectives (userObjective) you can set for POP3 are listed below. Test objecare set in the ixTimeline object.

- connectionRate
- transactionRate
- simulatedUsers
- concurrentConnections
- throughputMBps
- throughputKBps

POP3 Client Agent

The POP3 Client Agent defines a simulated user performing POP3 requests against one or more POP3 servers. Refer to `POP3 Client Agent` for a full descripof this command. The important options of this command are listed below.

Option	Usage
enable	Enables the use of the POP3 client agent.
name	The name associated with the client agent.
commandList	A list of commands to be sent to the server. Each list member is of type <code>Pop3Command</code> .
commandTimeout	Client timeout value.



Pop3Command

Each client command is a single step in the interaction. Refer to `Pop3Command` for a full description of this command. The important options of this command are listed below.

Subcommand	Usage
<code>checkConfig</code>	Checks the configuration of the action.

Option	Usage
<code>command arguments</code>	The POP3 command, with optional arguments, to be executed.

POP3 Server Agent

The POP3 Server Agent defines the operation of the POP3 server. Refer to `POP3 Server Agent` for a full description of this command. The important options of this command are listed below.

Option	Usage
<code>enable</code>	Enables the use of this server agent.
<code>name</code>	The name associated with the server agent.
<code>concurrentSessionLimit</code>	The maximum number of concurrent sessions that the server will allow.
<code>Server_Listening_Port</code>	Port that the POP3 server listens on for new connections.
<code>mailbox</code>	The contents of a user's mailbox, to be returned to the POP3 user upon request. A list, each of whose elements are of type <code>MailBoxItem</code> .

MailBoxItem

Each `MailBoxItem` is a mail item that a POP3 user will retrieve from a server. Refer to `MailBoxItem` for a full description of this command. The important options of this command are listed below.

Option	Usage
<code>count</code>	The number of messages in <code>mailMessage</code> to be returned.
<code>mailMessage</code>	A reference to a mail message, of type <code>MailMessage</code> . <code>MailMessage</code> is a command shared by the SMTP and POP3 protocols.



POP3 Client Agent

POP3 Client Agent - configure a POP3 client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_POP3Client1 [$Traffic1_Network1 activityList.appendItem
$Activity_POP3Client1 agent.config options...
```

DESCRIPTION

A POP3 client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`commandList`

This is a list of type `ixConfigSequenceContainer` used to hold objects of type `Pop3Command`. The elements in this list describe the operations to be performed by the server. (Default = {}).

`commandTimeout`

Amount of time allowed to an POP3 command to complete. If the command does not complete within the allowed time, `IxLoad` closes the POP3 client's connect to the POP3 server. (Default = 120).

`enable`

Enables the use of this client agent. (Default = true).

`ipPreference`

This option indicates the order by which the POP3 client will use the subnets, if there is a mixture of IPv4 and IPv6 subnets in the network. The values are: `IpPreferenceV4`, `IpPreferenceV6`, `IpPreferenceV4Any`, `IpPreferenceV6Any`.

`loopValue`

If this option is enabled (1) then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (0), then the client will progress through the command list only once, and then go idle. (Default = 0).

name

The name associated with this object, which must be set at object creation time .

enableVlanPriority

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If true, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = false).

vlanPriority

When `enableVlanPriority` is true, this option accepts the `vlan` priority value.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity POP3Client1 of NetTraffic Traffic1@Network1
#####
set Activity_POP3Client1 [$Traffic1_Network1 activityList.appendItem \
    -protocolAndType          "POP3 Client" ]
#####
# Timeline1 for activities POP3Client1
#####
set Timeline1 [::IxLoad new ixTimeline]
$Timeline1 config \
    -rampUpValue              1 \
    -rampUpType                0 \
    -offlineTime              0 \
```



```
-rampDownTime                20 \  
-standbyTime                  0 \  
-iterations                    1 \  
-rampUpInterval                1 \  
-sustainTime                   20 \  
-timelineType                  0 \  
-name                          "Timeline1"  
  
$Activity_POP3Client1 config \  
-enable                        true \  
-name                          "POP3Client1" \  
-enableConstraint              false \  
-userObjectiveValue            100 \  
-constraintValue                100 \  
-userObjectiveType              "simulatedUsers" \  
-timeline                      $Timeline1  
  
$Activity_POP3Client1 agent.config \  
-commandTimeout                120 \  
-enable                        true \  
-ipPreference                   0 \  
-name                          "POP3Client1" \  
-vlanPriority                    0 \  
-enableVlanPriority              false \  
-loopValue                      true  
  
$Activity_POP3Client1 agent.commandList.clear  
set my_Pop3Command [::IxLoad new Pop3Command]  
$my_Pop3Command config \  
-command                       "USER" \  
-arguments                      "username"
```

```
$Activity_POP3Client1 agent.commandList.appendItem -object $my_Pop3Command  
  
set my_Pop3Command1 [::IxLoad new Pop3Command]  
  
$my_Pop3Command1 config \  
    -command                "PASSWORD" \  
    -arguments              "password"  
  
$Activity_POP3Client1 agent.commandList.appendItem -object $my_Pop3Command1  
  
set my_Pop3Command2 [::IxLoad new Pop3Command]  
  
$my_Pop3Command2 config \  
    -command                "{Get}" \  
    -arguments              "None"  
  
$Activity_POP3Client1 agent.commandList.appendItem -object $my_Pop3Command2
```

SEE ALSO

ixNetTraffic, Pop3Command



Pop3Command

Pop3Command—Specifies the contents of an POP3 client command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_POP3Client1 [$Traffic1_Network1 activityList.appendItem
set my_Pop3Command [::IxLoad new Pop3Command]
$Activity_POP3Client1 agent.commandList.appendItem -object $my_Pop3Command
```

DESCRIPTION

An `POP3Command` object is added to the `commandList` option of the `POP3 Client Agent` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

arguments

Optional arguments related to the POP3 command to be executed. One of:

Command	Usage
"USER"	The user name that this client will use to login to the POP3 server. You can include variables in the user name; see <i>Using Auto-Generated Strings</i> .
"PASSWORD"	The password used to login to the POP3 server. You can use variin this field to generate multiple passwords. See <i>Using Auto-Generated Strings</i> .
"APOP"	The shared secret to be used to connect to a POP3 server that uses the APOP command to authenticate clients.
"OPEN"	The name and port of the server, of the form: "<IP address>:<TCP Port Number>"
"STAT"	N/A
"DELE"	The ID number of the message to be deleted.
"NOOP"	N/A
"RSET"	N/A
"LIST"	A single message ID or an empty string ("") for all IDs.
"UIDL"	A single message ID or an empty string ("") for all IDs.

Command	Usage
"RETR"	A message ID.
"TOP"	This can be either: "<Message ID>" or "<Message ID>:<Number of Lines>."
"QUIT"	N/A
"{Get}"	(Default) The name and port of the server, of the form: "<symbolic/IP address>:<TCP Port Number>"
"{Think}"	The number of seconds to wait. Default is 1 second.

command

The POP3 command to be executed. One of:

Command	Usage
"USER"	The user name that this client will use to login to the POP3 server is specified in the <code>argument</code> option. You can include variables in the user name; see <i>Using Auto-Generated Strings</i> .
"PASSWORD"	The password used to login to the POP3 server is specified in the <code>argument</code> option. You can use variables in this field to generate mulpasswords. See <i>Using Auto-Generated Strings</i> .
"APOP"	The shared secret to be used to connect to a POP3 server that uses the APOP command to authenticate clients. APOP (Authenticated POP) is a method of authenticating a POP3 session that does not require a cleartext password to be sent. The shared secret is a string known only to the POP3 client and server, and is part of the authentication process. You can use variables in this field to generate multiple shared secrets. See <i>Using Auto-Generated Strings</i> .
"OPEN"	Opens the TCP connection.
"STAT"	Similar to the LIST command in that it causes the server to return the number of messages in the mail drop along with the total space occupied (in octets) by those messages. Unlike the RETR or LIST commands, STAT cannot be used to display messages.
"DELE"	Identifies a message to be deleted by passing its ID numbers in the <code>argument</code> option. The LIST command returns message IDs.
"NOOP"	A null or NO OPERATION. A POP3 server's response to a NOOP comis to do nothing.
"RSET"	Resets the state of messages marked for deletion.
"LIST"	Lists the number of stored messages and their combined size, in octets. You can also use the result of this command to obtain the size of a single message; include the message's number as the conof the <code>argument</code> option.

Command	Usage
"UIDL"	Returns the Unique ID Listing for a message. If the contents of <code>argument</code> is empty, a numerical listing of all messages and their associated UIDLs is returned. If the <code>arguments</code> option contains a specific UIDL, then the contents of the message is returned.
"RETR"	Returns the full text of the specified message, and marks that mesas read. Passes the message number returned by the LIST command in the <code>argument</code> option to identify the message to be retrieved.
"TOP"	Displays a message's header and the specified number of lines, counted from the top. This command takes two arguments: the mesnumber and the number of lines to display. The server returns the message headers followed by a blank line and then the specified number of lines from the message.
"QUIT"	Ends the POP3 session and deletes any messages marked for dele
"{Get}"	(Default) An IxLoad command that retrieves all waiting messages for the user, then logs out. {Get} is a single command that performs the same function as multiple POP3 commands. However, {Get} is not a standard POP3 command. It is included in IxLoad for your conveto make configuring POP3 clients easier.
"{Think}"	An amount of time to wait before issuing the next command.
"{LoopBegin}"	An IxLoad command that you can add to the Command List to cause the commands between it and the {Loop End} to be executed a specified number of times.
"{LoopEnd}"	Ends the list of commands that will be executed by the preceding {Loop Begin} command.

EXAMPLE

```

set my_Pop3Command [::IxLoad new Pop3Command]

$my_Pop3Command config \
    -command                "USER" \
    -arguments              "username"

$Activity_POP3Client1 agent.commandList.appendItem -object $my_Pop3Command

set my_Pop3Command1 [::IxLoad new Pop3Command]

$my_Pop3Command1 config \
    -command                "PASSWORD" \
    -arguments              "password"

$Activity_POP3Client1 agent.commandList.appendItem -object $my_Pop3Command1

```

```
set my_Pop3Command2 [::IxLoad new Pop3Command]

$my_Pop3Command2 config \

    -command                "{Get}" \
    -arguments              "None"

$Activity_POP3Client1 agent.commandList.appendItem -object $my_Pop3Command2
```

SEE ALSO

POP3 Client Agent, Using Auto-Generated Strings

POP3 Server Agent

POP3 Server Agent - create a POP3 server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_POP3Server1 [$Traffic2_Network2 activityList.appendItem
$Activity_POP3Server1 agent.config options...
```

DESCRIPTION

A POP3 server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`concurrentSessionLimit`

The maximum number of concurrent sessions to be supported by the agent. (Default = 1,000).

`enable`

Enables the use of this action. (Default = true).

`mailBox`

This is a list of type `MailBoxItem`. The elements in this list are the messages that a POP3 user will receive when he queries the mailbox. (Default = {}).

`mailMessageList`

This is a list of type `MailMessage`. The elements in this list contain the messages to be returned to a POP3 client. (Default = {}).

`name`

The name associated with this object, which must be set at object creation time.

serverlisteningport

Port that the POP3 server listens on. To specify multiple ports, separate the port numbers with commas (.). You can specify up to 50 listening ports. (Default = 110).

enableVlanPriority

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If true, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in vlanPriority. (Default = false).

vlanPriority

When enableVlanPriority is true, this option accepts the vlan priority value.

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]

#####

# Activity POP3Server1 of NetTraffic Traffic2@Network2

#####

set Activity_POP3Server1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType          "POP3 Server" ]

set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]

$Activity_POP3Server1 config \
    -enable                   true \
    -name                     "POP3Server1" \
    -timeline                 $_Match_Longest_

$Activity_POP3Server1 agent.config \
    -Server_Listening_Port   110 \
    -enable                   true \
    -name                     "POP3Server1" \
    -vlanPriority             0 \
```



```
-concurrentSessionLimit          1000 \  
-enableVlanPriority               true  
  
$Activity_POP3Server1 agent.mailMessageList.clear  
  
set Simple [::IxLoad new MailMessage]  
  
$Simple config \  
-bodySizeType                    0 \  
-name                             "Simple" \  
-fileNameAsBody                  "" \  
-description                      "100 bytes plain text body" \  
-textContentAsBody               "" \  
-bodySizeRandomMax                4096 \  
-bodySizeFixed                   100 \  
-mimeTypeAndEncode                0 \  
-bodySizeRandomMin                1 \  
-bodyDataType                     0 \  
-useFileAsBody                   true \  
-bodyFormat                       0  
  
$Simple headerList.clear  
  
set From [::IxLoad new MailHeader]  
  
$From config \  
-name                             "From" \  
-value                             "fromName@company.com"  
  
$Simple headerList.appendItem -object $From  
  
set To [::IxLoad new MailHeader]  
  
$To config \  
-name                             "To" \  
-value                             "toName@company.com"  
  
$Simple headerList.appendItem -object $To
```

```
set Subject [::IxLoad new MailHeader]

$Subject config \

    -name                "Subject" \

    -value                "sample subject"

$Simple headerList.appendItem -object $Subject

$Simple attachmentList.clear

$Activity_POP3Server1 agent.mailMessageList.appendItem -object $Simple
```

SEE ALSO

`ixNetTraffic`, `MailBoxItem`, `MailMessage`



MailBoxItem

MailBoxItem—Specifies the contents of a mail box.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_POP3Server1 [$Traffic2_Network2 activityList.appendItem
$Activity_POP3Server1 agent.mailBox.appendItem -object $my_MailBoxItem
set my_MailBoxItem [::IxLoad new MailBoxItem]
```

DESCRIPTION

A MailBoxItem object is added to the mailBox option of the POP3 Server Agent object using the appendItem subcommand from the ixConfigSequenceContainer command.

SUBCOMMANDS

The options for this command are configured and read using the standard config, cget, and getOptions subcommands defined in the ixConfig command.

OPTIONS

count

The number of copies of the mail message detained in mailMessage to be returned. (Default = 10).

SUB-OBJECTS

mailMessage

An object of type MailMessage which contains the message to be returned to the POP3 user. (Default = {}).

EXAMPLE

```
set my_MailBoxItem [::IxLoad new MailBoxItem]

$my_MailBoxItem config \

    -count                10 \

    -mailMessage          $Simple1

$Activity_POP3Server1 agent.mailBox.appendItem -object $my_MailBoxItem
```

SEE ALSO

POP3 Server Agent, MailMessage



Using Auto-Generated Strings

In some of the fields in the POP3 and SMTP client and server Activities, you can include variables that will cause IxLoad to automatically generate multiple values for the field. For example, the POP3 Username and Password fields both support the inclusion of variables.

You can use the following types of variables:

Fixed Numeric Range

This is a variable of the format `[xxx-yyy]`. The first value in the variable, `xxx`, defines the initial value of the range and the number of digits in the range (3, in the case of `xxx`). The second value defines the maximum value in the range.

For example, if you specify the following user name: `user_[001-100]`, Ixload will generate the user names `user_001`, `user_002`, `user_003`, and so on up to `user_100`, which will be the final unique user name generated. If more user names are required (because there are more users than user names), IxLoad returns to the initial values of the range and repeats the generation process.

Open-Ended Numeric Range

This is a variable of the format `[xx-]`. The value in the variable determines the initial value of the range and the number of digits in the range (2, in the case of `xx`). The maximum value of the range will be reached when all digits are 9.

For example, if you specify the following user name: `user_[01-]`, Ixload will generate the user names `user_01`, `user_02`, `user_03`, and so on up to `user_99`, which will be the final unique user name generated. If more user names are required (because there are more users than user names), IxLoad returns to the initial value of the range and repeats the generation process.

Fixed Alphabetic Range

This is a variable of the format `[aaa-zzz]`. The first value in the variable, `aaa`, defines the initial value of the range and the number of letters in the range (3, in the case of `aaa`). The second value defines the final value in the range.

For example, if you specify the following user name: `user_[aaa-bbb]`, Ixload will generate the user names `user_aaa`, `user_aab`, `user_abb`, `user_aba`, `user_baa`, `user_bba`, and `user_bbb`, which will be the final unique user name generated. If more user names are required (because there are more users than user names), IxLoad returns to the initial values of the range and repeats the generation process.

IxLoad treats lowercase and uppercase as different values. For example, the all-lowercase range `[aaa-bbb]` does not generate values of `AAA`, `BBB`, or any other uppercase letters.

Open-Ended Alphabetic Range

This is a variable of the format `[aa-]`. The value in the variable determines the initial value of the range and the number of letters in the range (2, in the case of `aa`). The maximum value of the range will be reached when all digits are 9.

For example, if you specify the following user name: user_[aa-], Ixload will generate the user names user_aa, user_ab, user_ac, and so on up to user_zz, which will be the final unique user name generated. If more user names are required (because there are more users than user names), IxLoad returns to the initial value of the range and repeats the generation process.

IxLoad treats lowercase and uppercase as different values. For example, the all-lowercase range [aa-] does not generate values of AA, BB, or any other upperletters.

POP3 Statistics

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<>version>\Results\<>repository>_<test configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

If you review your statistics and find many instances of POP3 Client statistics and server statistics that should match but do not, that may be an indication that the Ramp Down Time is too short. When the Ramp Down Time expires, IxLoad terminates any users that are still running. If those users still have work in progress (such as transferring data) when IxLoad terminates them, the work will not be completed and the effect will be that statistics that should match may not.

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

For the POP3 statistics, see the following:

POP3 Client Statistics (on page 1433)

POP3 Server Statistics (on page 1436)

POP3 Client Statistics

The table below lists the statistics that IxLoad reports for POP3 clients. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

For TCP statistics, see Appendix D, TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Description
POP3 Simulated Users	Number of simulated POP3 users.



POP3 Concurrent Connections	Maximum number of concurrent POP3 connections maintained during the test.
POP3 Connections	Number of POP3 sessions established by the clients.
POP3 Transactions	Number of POP3 transactions completed by the clients.
POP3 Bytes	Number of POP3 bytes transmitted and received by the clients.
POP3 Sessions Requested	Number of POP3 sessions attempted by the clients.
POP3 Sessions Established	Number of POP3 sessions established by the clients.
POP3 Sessions Failed	Number of attempts to establish POP3 sessions that failed.
POP3 Mails Received	Number of mail messages retrieved by the clients using POP3.
POP3 Authentication Sent	Number of POP3 authentication messages sent by the clients.
POP3 Authentication Ok	Number of authentication messages which resulted in the servers allowing access.
POP3 Authentication Failed	Number of authentication messages which resulted in the servers denying access.
POP3 LIST Sent	Number of POP3 LIST commands sent.
POP3 LIST Ok	Number of POP3 LIST commands that received a positive response. If an argument was given with the command, the POP3 server issues a response with a line containing information for the message specified by the argument. If no argument was given, the POP3 server issues a multi-line response.
POP3 LIST Failed	Number of POP3 LIST commands that did not receive a positive response.
POP3 STAT Sent	Number of POP3 STAT messages sent.
POP3 STAT Ok	Number of POP3 STAT commands that received a positive response. A positive response to this command consists of +OK followed by a single space, the number of messages in the maildrop, a single space, and the size of the maildrop in octets.
POP3 STAT Failed	Number of POP3 STAT commands that did not receive a positive response.
POP3 RETR Sent	Number of POP3 RETR messages sent.
POP3 RETR Ok	Number of POP3 RETR messages that received a positive response. A positive response to this command consists of an initial +OK followed by the message corresponding to the given message-number.

POP3 RETR Failed	Number of POP3 RETR commands that did not receive a positive response.
POP3 DELE Sent	Number of POP3 DELE commands sent.
POP3 DELE Ok	Number of POP3 DELE messages that received a positive response. In a positive response to this message, the POP3 server marks the message as deleted.
POP3 DELE Failed	Number of POP3 DELE commands that did not receive a positive response.
POP3 UIDL Sent	Number of POP3 UIDL commands sent.
POP3 UIDL Ok	Number of POP3 UIDL messages that received a positive response. If an argument was given, a positive response to this command consists of a line containing information for the message passed as the argument. If no argument was given, a positive response consists of an initial +OK followed by multiple lines, each line containing information for one message in the maildrop.
POP3 UIDL Failed	Number of POP3 UIDL commands that did not receive a positive response.
POP3 RSET Sent	Number of POP3 RSET messages sent.
POP3 RSET Ok	Number of POP3 RSET messages that received a positive response.
POP3 RSET Failed	Number of POP3 RSET commands that did not receive a positive response.
POP3 NOOP Sent	Number of POP3 NOOP messages sent.
POP3 NOOP Ok	Number of POP3 NOOP messages that received a positive response.
POP3 NOOP Failed	Number of POP3 NOOP commands that did not receive a positive response.
POP3 TOP Sent	Number of POP3 TOP messages sent.
POP3 TOP Ok	Number of POP3 TOP messages that received a positive response. A positive response consists of the initial +OK followed by the headers of the message, the blank line separating the headers from the body, and then the number of lines indicated message's body.
POP3 TOP Failed	Number of POP3 TOP messages that did not receive a positive response.
POP3 QUIT Sent	Number of POP3 QUIT messages sent.
POP3 QUIT Ok	Number of POP3 QUIT messages that received a positive response.
POP3 QUIT Failed	Number of POP3 QUIT messages that did not receive a positive response.

POP3 Total Bytes Sent	Total number of POP3-related bytes (commands, responses, and messages) sent by the clients.
POP3 Total Bytes Received	Total number of POP3-related bytes (commands, responses, and messages) received by the clients.
POP3 Mail Bytes Received	Total number of bytes contained in the mail messages retrieved using POP3.
POP3 Sessions Active	Total number of POP3 sessions in progress.
POP3 Connection Rate	Rate at which the POP3 client established connections to the server.
POP3 Transaction Rate	Rate at which the POP3 client completed transactions.
POP3 Concurrent Connections	Number of POP3 connections active at the same time.
POP3 Simulated Users	Number of simulated POP3 users.
POP3 Throughput	Rate at which the client sent and received POP3 data.



Note: If the average table and bar graphs do not contain any data for the clients, that is an indication that they did not reach the Sustained (SU) run state. This can be caused by the following:

1. Stopping a test during the Ramp-Up phase.
2. Configuring a large number of page requests for the client agent so that not all the users configured for the client can attain the SU state within the allotted time.
3. Configuring a value for the statistics interval (Statistics tab) which is much larger than the SU time.

POP3 Server Statistics

The table below lists the statistics that IxLoad reports for POP3 servers. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

For TCP statistics, see Appendix D, TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Description
POP3 Session Requests Received	Number of requests to establish POP3 sessions received by the servers.
POP3 Session Requests Successful	Number of POP3 sessions established by the servers.
POP3 Session Requests Failed	Number of POP3 sessions requested by the clients that the servers failed to establish.
POP3 Total Mails Sent	Total number of mail messages sent by the servers.

POP3 Total Attachments Sent	Total number of attachments sent by the servers.
POP3 Total Mails With Attachments Sent	Total number of messages sent that included one or more attachments.
POP3 USER Ccmds Received	Number of POP3 USER commands received.
POP3 PASS Ccmds Received	Number of POP3 PASS commands received.
POP3 APOP Ccmds Received	Number of POP3 APOP commands received.
POP3 LIST Ccmds Received	Number of POP3 LIST commands received.
POP3 STAT Ccmds Received	Number of POP3 STAT commands received.
POP3 RETR Ccmds Received	Number of POP3 RETR commands received.
POP3 DELE Ccmds Received	Number of POP3 DELE commands received.
POP3 UIDL Ccmds Received	Number of POP3 UIDL commands received.
POP3 RSET Ccmds Received	Number of POP3 RSET commands received.
POP3 NOOP Ccmds Received	Number of POP3 NOOP commands received.
POP3 TOP Ccmds Received	Number of POP3 TOP commands received.
POP3 QUIT Ccmds Received	Number of POP3 QUIT commands received.
POP3 Total Bytes Sent	Total number of POP3-related bytes (commands, responses, and messages) sent by the servers.
POP3 Total Bytes Received	Total number of POP3-related bytes (commands, responses, and messages) received by the servers.



22

QuickHTTP

This section describes the QuickHTTP Tcl API objects.

In this chapter:

Overview	1439
QuickHTTP Client Agent	1442
QuickHTTP Server Agent	1453
QuickHTTP Statistics	1459

Overview

QuickHTTP protocol commands are organized as:

- QuickHTTP Client Agent
 - QuickHTTP Client HTTP Options
 - QuickHTTP Client Advanced Options
 - QuickHTTP Client Commands
- QuickHTTP Server Agent
 - QuickHTTP Server HTTP Options
 - QuickHTTP Server Advanced Options

Objectives

The objectives (userObjective) you can set for QuickHTTP are listed below. Test objectives are set in the ixTimeline object.

- connectionRate
- transactionRate
- simulatedUsers
- concurrentConnections
- throughputMBps
- throughputKBps

QuickHTTP Client Agent

The QuickHTTP Client Agent defines a simulated user performing HTTP requests against one or more HTTP clients, either full HTTP or QuickHTTP cliRefer to `QuickHTTP Client Agent` for a full description of this command. The important options of this command are:

Option	Usage
enable	Enables the use of the QuickHTTP client agent.
name	The name associated with the client agent.

QuickHTTP Client HTTP Options

The QuickHTTP HTTP client options define basic HTTP protocol usage. Refer to `QuickHTTP Client HTTP` for a full description of this command. The important options of this command are:

Option	Usage
browserEmulation	The type of browser (IE vs. Netscape) that will be emulated.



Option	Usage
keepAlive maxTransactionsPerConnection concurrentConnectionsPerUser limitTransactionsPerConnection	Controls persistent connection operation.
additionalHeaders	Additional headers to be used with each GET request.

QuickHTTP Client Advanced Options

The QuickHTTP client advanced options define additional connection options. Refer to `QuickHTTP Client Advanced` for a full description of this command. The important options of this command are:

Option	Usage
enableESM esm	Controls the use of the effective send MSS.

QuickHTTP Client Commands

Each client command is a single GET command to be transmitted. Refer to `QuickHTTP Client Command` for a full description of this command. The important options of this command are:

Option	Usage
destination	The name/address of the HTTP or QuickHTTP client.
pageObject	The page at the destination to be retrieved.

QuickHTTP Server Agent

The QuickHTTP Server Agent defines the operation of the QuickHTTP server. Refer to `QuickHTTP Server Agent` for a full description of this command. The important options of this command are:

Option	Usage
enable	Enables the use of the HTTP server agent.
name	The name associated with the server agent.

QuickHTTP Server HTTP Options

The QuickHTTP Server HTTP options define basic HTTP protocol usage. See `QuickHTTP Server HTTP` for a description of this command.

Option	Usage
<code>listenPort</code>	The port number that the server will listen on.
<code>enableRequestTimeout</code>	Enables or disables the <code>requestTimeout</code> option.
<code>requestTimeout</code>	If enabled, the server enforces the timeout value as configured through this option.

QuickHTTP Server Advanced Options

The QuickHTTP Server advanced options define additional connection options. Refer to `QuickHTTP Server Advanced` for a full description of this command. The important options of this command are:

Option	Usage
<code>enableESM</code> <code>esm</code>	Controls the use of the effective send MSS.



QuickHTTP Client Agent

QuickHTTP Client Agent - configure a QuickHTTP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_QuickHTTPClient1 [$Traffic1_Network1 activityList.appendItem \
$Activity_QuickHTTPClient1 agent.config options...
```

DESCRIPTION

An QuickHTTP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this action. (Default = true).

`name`

The name associated with this object, which must be set at object creation.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]

#-----
# Activity QuickHTTPClient1 of NetTraffic Traffic1@Network1
#-----

set Activity_QuickHTTPClient1 [$Traffic1_Network1 activityList.appendItem \
```

```
-protocolAndType                                "QuickHTTP Client" ]

#-----
# Timeline1 for activities QuickHTTPClient1
#-----

set Timeline1 [::IxLoad new ixTimeline]

$Timeline1 config \
    -rampUpValue                                1 \
    -rampUpType                                  0 \
    -offlineTime                                 0 \
    -rampDownTime                                20 \
    -standbyTime                                 0 \
    -iterations                                  1 \
    -rampUpInterval                              1 \
    -sustainTime                                 20 \
    -timelineType                                0 \
    -name                                         "Timeline1"

$Activity_QuickHTTPClient1 config \
    -enable                                      true \
    -name                                         "QuickHTTPClient1" \
    -enableConstraint                             false \
    -userObjectiveValue                           100 \
    -constraintValue                              100 \
    -userObjectiveType                            "simulatedUsers" \
    -timeline                                     $Timeline1

$Activity_QuickHTTPClient1 agent.config \
```



```
-enable true \  
-name "QuickHTTPClient1"
```

```
$Activity_QuickHTTPClient1 agent.pm.http.config \  
\  
-limitTransactionsPerConnection "upto" \  
-standardVersion "HTTP/1.1" \  
-browserEmulation "custom" \  
-additionalHeaders "" \  
-concurrentConnectionsPerUser 3 \  
-maxTransactionsPerConnection 1 \  
-keepAlive false
```

```
$Activity_QuickHTTPClient1 agent.pm.advanced.config \  
\  
-noClose false \  
-esm 1460 \  
-enableESM false \  
-closeWithRST false \  
-threeWayClose true
```

```
$Activity_QuickHTTPClient1 agent.pm.commands.clear
```

```
$Activity_QuickHTTPClient1 agent.pm.commands.appendItem \  
\  
-id "GET" \  
-destination "10.205.10.19" \  
-page "None"
```

SEE ALSO

QuickHTTP Client HTTP, QuickHTTP Client Command



QuickHTTP Client HTTP

QuickHTTP Client HTTP - configure a QuickHTTP client's HTTP options

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_QuickHTTPClient1 [$Traffic1_Network1 activityList.appendItem \
$Activity_QuickHTTPClient1 agent.pm.http.config options...
```

DESCRIPTION

A QuickHTTP client's HTTP options are set by modifying the options of the `pm.http` option of the `QuickHTTP Client Agent` object using its `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`additionalHeaders`

This string, consisting of a set of headers will be transmitted as part of a client request following the automatically generated `Host` header line. (Default = "").

`browserEmulation`

The type of browser that the client will emulate. One of:

Option	Usage
::HTTP_Client(kBrowserTypeNone) or "<Custom>"	(Default) No browser is emulated by the client. The headers may be entered in the <code>headerList</code> option.
::HTTP_Client(kBrowserTypeIE5) "Microsoft IE 5.x"	Microsoft Internet Explorer 5.x browser is emulated by the client.
::HTTP_Client(kBrowserTypeMozilla) "Mozilla"	Netscape, Mozilla and Firefox browsers are emulated by the client.
::HTTP_Client(kBrowserTypeIE6) "Microsoft IE 6.x"	Microsoft Internet Explorer 6.x browser is emulated by the client.
::HTTP_Client(kBrowserTypeFirefox) "Firefox"	The Firefox browser is emulated by the client.

`concurrentConnectionsPerUser`

This value determines the maximum number of connections that a single user can have open at any given time. For example, clients may open multiple connections when their command list contains URLs for multiple clients.

The value for this parameter has an effect on the total number of users that can be configured; increasing the number of concurrent connections decreases the number of users that can be configured. Setting this parameter to 1 allows the maximum number of users to be created.

ixLoad enforces these limits for clients. For clients, the limits are the same but not enforced. (Default = 3).

keepAlive

If this option is set to `true`, the client adds the `Connection: Keep-Alive` header to its request. Each request from a client creates a new socket connection to the client. The client reads from that socket connection to get the response. If `keepAlive` is not set, the client closes the connection. If the client needs to make a new request, it will establish a new connection.

If the client sends the Keep-Alive header, the client keeps the connection open. When the client sends another request, it uses the same connection. This will continue until either the client or the server decides that the session is over, and one of them closes the connection. (Default = false).

limitTransactionsPerConnection

This option controls how the number of transactions per connection is handled. One of:

Option	Usage
"max"	The maximum number of connections that the port can support will be used.
"up to"	(Default) The value set in the <code>maxTransactionsPerConnection</code> will be used.

maxTransactionsPerConnection

This option enables you to control the number of transactions that can occur during a single connection.

ixLoad limits the number of transactions that can occur during a single TCP connection. If a user reaches the maximum number of transactions and needs to communicate with the client, it will close the connection and open a new one. (Default = 1).

standardVersion

The HTTP protocol version to be used. One of:

Option	Usage
"HTTP/1.0"	HTTP version 1.0.
"HTTP/1.1"	(Default). HTTP version 1.1.

EXAMPLE

```
$Activity_QuickHTTPClient1 agent.pm.http.config \
```



-limitTransactionsPerConnection	"upto" \
-standardVersion	"HTTP/1.1" \
-browserEmulation	"custom" \
-additionalHeaders	" " \
-concurrentConnectionsPerUser	3 \
-maxTransactionsPerConnection	1 \
-keepAlive	false

SEE ALSO

QuickHTTP Client Agent, QuickHTTP Client Command

QuickHTTP Client Advanced

QuickHTTP Client Advanced - configure a QuickHTTP client's advanced options

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_QuickHTTPClient1 [$Traffic1_Network1 activityList.appendItem \.
$Activity_QuickHTTPClient1 agent.pm.advanced.config options...
```

DESCRIPTION

A QuickHTTP client's advanced options are set by modifying the options of the `pm.advanced` option of the `QuickHTTP Client Agent` object using its `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`closeWithRST`

If `true`, this option causes the QuickHTTP client to terminate the connection by sending a TCP RST (Reset) segment instead of a FIN.

`enableESM`

If `true`, the use of the `esm` option is enabled. (Default = `false`).

`esm`

If `enableESM` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size as 1,460 bytes. (Default = 1,460).

`noclose`

This options if enabled does not close the conenction even if all the transactions for that connection are finished. The server can close the connections, for examin case of timeout.

Note: `noClose` is not usable with quickHTTP 1.0. The server closes the connection in this case, and not the client.

`threeWayClose`

If `true`, the connections are closed using three way handshake. This field accepts boolean value. If it is not enabled then the traditional four way close is done.

EXAMPLE

```
$Activity_QuickHTTPClient1 agent.pm.advanced.config \  
-noClose                false \  
-esm                    1460 \  
-enableESM              false \  
-closeWithRST           false \  
-threeWayClose          true
```

SEE ALSO

QuickHTTP Client Agent, QuickHTTP Client HTTP, QuickHTTP Client Command

QuickHTTP Client Command

QuickHTTP Commands - configure the list of commands a QuickHTTP client will execute

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]

set Activity_QuickHTTPClient1 [$Traffic1_Network1 activityList.appendItem \
$Activity_QuickHTTPClient1 agent.pm.commands.appendItem options...
```

DESCRIPTION

A QuickHTTP client command is added to the `pm.commands` option of the `QuickHTTP Client Agent` object using its `appendItem`.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

None.

OPTIONS

`id`

Selects the QuickHTTP command to be used. One of:

Option	Usage
"GET"	(Default) Retrieves the page specified in the <code>page</code> option.
"THINK"	Pauses the execution of the command list. THINK/{Think} command starts, only after the previous command is completed.

Arguments for `id = GET` Command

`destination`

Either the IP address of a real HTTP client or the value of the `-name` option of an HTTP Client Agent. If the HTTP client listens on a port other than the standard (80), enter a colon after the IP address and then enter the port number. If an HTTP Client Agent is used, the port number should agree with the port number associated with the Client Agent. If you are testing an SLB with a virtual IP address (VIP), enter its address here. See the following example:



192.168.0.1:80

page

- This option contains a page stored on the HTTP client specified in the Destination field. Three formats are available:
- /#n identifies a target that contains n bytes of HTTP data. For example, /#1 is 1 byte of HTTP data. In addition to the default sizes listed, you can cause the HTTP Client agent to generate a customized target by specifying the size using the same convention used for the default sizes. For example, to specify a target of 16 bytes, use /#16.
- /nk.htm identifies a target that is an HTML page that is n kilobytes in size. For example, /4k.htm is an 4096-byte HTML page.

Arguments for id = THINK Command

duration

Specifies the duration of the pause in milliseconds. Maximum = "60,000", Mini = "1", (Default = "1000").

EXAMPLE

```
#-----
# Add actions to this client agent
#-----
$Activity_QuickHTTPClient1 agent.pm.commands.appendItem \
    -id                "GET" \
    -destination       "10.205.10.19" \
    -page              "None"
```

SEE ALSO

QuickHTTP Client Agent, QuickHTTP Client HTTP

QuickHTTP Server Agent

Quick HTTP Server Agent - configure a QuickHTTP server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_QuickHTTPServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_QuickHTTPServer1 agent.config options...
```

DESCRIPTION

An QuickHTTP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

The set of Web pages available through the server is described in the `webPageListoption`, which references response headers held in the `resoption` and cookies held in the `cookList`.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this action. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]

#####

# Activity QuickHTTPServer1 of NetTraffic Traffic2@Network2
```



```
#####

set Activity_QuickHTTPServer1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType                "QuickHTTP Server" ]

set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]

$Activity_QuickHTTPServer1 config \
    -enable                true \
    -name                  "QuickHTTPServer1" \
    -timeline              $_Match_Longest_

$Activity_QuickHTTPServer1 agent.config \
    -enable                true \
    -name                  "QuickHTTPServer1"

$Activity_QuickHTTPServer1 agent.pm.http.config \
    -requestTimeout        300000 \
    -enableRequestTimeout true \
    -listenPort            80

$Activity_QuickHTTPServer1 agent.pm.advanced.config \
    -esm                   1460 \
    -enableESM             true \
    -closeWithRST          true \
    -threeWayClose         true
```

SEE ALSO

QuickHTTP Server HTTP



QuickHTTP Server HTTP

QuickHTTP Server HTTP - configure a QuickHTTP server's HTTP options

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_QuickHTTPServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_QuickHTTPServer1 agent.pm.http.config options...
```

DESCRIPTION

A QuickHTTP server's HTTP options are set by modifying the options of the `pm.http` option of the QuickHTTP Server Agent object using its `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`listenPort`

The port number to which the QuickHTTP server will respond to requests. (Default = 80).

`enableRequestTimeout`

This enables the `requestTimeout` option. (Default = 0).

`requestTimeout`

If enabled, the server enforces the timeout value configured in the `requestTimeout` option. (Default = 1000).

EXAMPLE

```
$Activity_QuickHTTPServer1 agent.pm.http.config \  
    -requestTimeout          300000 \  
    -enableRequestTimeout    true \  
    -listenPort              80
```

SEE ALSO

QuickHTTP Server Agent



QuickHTTP Server Advanced

QuickHTTP Server Advanced - configure a QuickHTTP server's advanced options

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_QuickHTTPServer1 [$Traffic2_Network2 activityList.appendItem \
$Activity_QuickHTTPServer1 agent.pm.advanced.config options...
```

DESCRIPTION

A QuickHTTP server's advanced options are set by modifying the options of the `pm.advanced` option of the `QuickHTTP Server Agent` object using its `append`.

SUBCOMMANDS

None.

OPTIONS

`closeWithRST`

If `true`, this option causes the QuickHTTP server to terminate the connection by sending a TCP RST (Reset) segment instead of a FIN.

`enableESM`

If `true`, the use of the `esm` option is enabled. (Default = `false`).

`esm`

If `enableESM` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size as 1,460 bytes. (Default = 1,460).

`threeWayClose`

If `true`, the connections are closed using three way handshake. This field accepts boolean value. If it is `false`, then the connections are closed using a four-way handshake using this option.

EXAMPLE

```
$Activity_QuickHTTPServer1 agent.pm.advanced.config \
    -esm 1460 \
    -enableESM true \
    -closeWithRST true \
```

-threeWayClose

true

SEE ALSO

QuickHTTP Server Agent, QuickHTTP Server HTTP

QuickHTTP Statistics

For QuickHTTP statistics, see the following sections:

QuickHTTP Client Statistics (see "[HTTP/QuickHTTP Client Statistics](#)" on page 1459)

QuickHTTP Server Statistics (see "[HTTP/QuickHTTP Server Statistics](#)" on page 1463)

HTTP/QuickHTTP Client Statistics

The table below lists the statistics IxLoad reports for HTTP clients. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test configuration>_<yyyy><mm><dd>_<hh><mm><ss>

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Description
HTTP Simulated Users	Number of users to be simulated during the test.
Physical Rx Drops	Number of incoming packets dropped due to buffer overflow.
Physical Tx Drops	Number of outgoing packets dropped due to buffer overflow. Typically, this is caused by stopping a large test or configuring a Ramp Down time that is too short.
HTTP Concurrent Connections	Number of concurrent connections maintained during the test.
HTTP Connections	Total number of connections established by the clients.
HTTP Transactions	Total number of transactions completed by the clients.
HTTP Requests Sent	Number of HTTP requests sent by the clients. The statistics show the number of requests for each URL (page).



HTTP Requests Successful	Number of positive HTTP responses (2xx- and 3xx-range responses) received by the clients. The statistics show the number of requests for each URL (page).
HTTP Requests Failed	Number of HTTP requests that failed for any reason. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Write)	Number of HTTP requests that failed due to a socket write error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Read)	Number of HTTP requests that failed due to a socket read error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Bad Header)	Number of HTTP requests that failed due to a defective HTTP header. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (4xx)	Number of 4xx-range responses received by the clients in response to an HTTP request. The statistics show the number of requests for each URL (page). 408 responses are counted separately by the HTTP Session Timeout (408) statistic and may or may not also be included in the HTTP Requests Failed (4xx) count. See the description of HTTP Session Timeout (408) for more information.
HTTP Requests Failed (400)	Bad Request. Number of requests that failed due to a syntax error in the URL. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (401)	Unauthorized. Number of requests that failed due to because the server did not receive the correct user name or password from the browser. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (403)	Forbidden. Number of requests that failed due to because the name or password supplied by the browser are incorrect. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (404)	Not Found. Number of requests that failed because requested object is not stored on the server on the path supplied. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (407)	Proxy Authentication Required. Number of requests that failed because access to the URL requires authentication with a proxy server.
HTTP Requests Failed (408)	Timeout. Number of requests that failed due to communications between the client and server taking too long. The statistics show the number of requests for each URL (page).

HTTP Requests Failed (4xx other)	Number of HTTP requests that failed for reasons other than a Bad Request (400), Unauthorized (401), Forbidden (403), Not Found (404), or Timeout (408) error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (5xx)	Number of HTTP requests that failed due to lack of resources on the server (HTTP 500-series errors). This statistic is only incremented if the client had issued a request to the server before receiving the 5xx response. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (505)	HTTP Version not Supported. Number of requests that failed because the server does not support the HTTP version used by the client. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (5xx other)	Number of requests that failed for reasons other than an HTTP version mis-match (505). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Timeout)	Number of HTTP requests that failed because the clients did not receive a response within 600 seconds. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Aborted)	Number of HTTP requests that ended prematurely due to events outside HTTP or TCP. For example, if any HTTP requests are pending when the Ramp-Down period ends, those requests are aborted by IxLoad. The statistics show the number of requests for each URL (page).
HTTP Session Timeouts (408)	<p>Number of HTTP 408 responses received. This statistic includes all 408 responses received regardless of whether they were received for a pending HTTP request or not.</p> <p>IxLoad counts 408 responses differently depending on whether or not a client has a pending HTTP request:</p> <p>If a client has an HTTP request pending and it receives a 408 response, IxLoad increments the HTTP Received 408, HTTP Requests Failed (4xx), and HTTP Requests Failed statistics.</p> <p>If a client does not have an HTTP request pending and it receives a 408 response, IxLoad only increments the HTTP Received 408 statistic.</p>
HTTP Sessions Rejected (503)	Service Unavailable. Number of HTTP sessions that could not be established due to lack of resources on the server.
HTTP Transactions Active	Number of HTTP transactions transferring HTTP commands or data.
HTTP Users Active	Number of HTTP users simulated.

HTTP Bytes Sent	<p>Number of HTTP bytes transmitted by the clients.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic (increased by retransmits) or less than this statistic (decreased by broken or reset connections).</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not (HTTP only).</p>
HTTP Bytes Received	<p>Number of HTTP bytes received by the clients.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic due to increases caused by retransmits.</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not (HTTP only).</p>
HTTP Bytes	<p>Amount of HTTP data sent and received by the clients, in bytes.</p>
HTTP Connect Time (ms)	<p>Average time elapsed, in microseconds, between the time the client sends a SYN packet and the time it receives the SYN/ACK.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
HTTP Time To First Byte (ms)	<p>Average time elapsed, in microseconds, before clients received the first byte of an HTTP response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
HTTP Time To Last Byte (ms)	<p>Average time elapsed, in microseconds, before clients received the last byte of an HTTP response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
HTTP TCP Connection Lifetime (ms)	<p>Time elapsed between the time the client sent the first SYN to start the TCP connection handshake and the time it sent the last FIN or ACK to end the TCP connection.</p> <p>This statistic measures the total duration of a TCP connection: handshake duration + data transfer duration + close duration.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>

HTTP/QuickHTTP Server Statistics

The table below describes the statistics that IxLoad records for the QuickHTTP servers. Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<version>\Results\<repository>_<test
configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

For TCP statistics, TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Description
HTTP Requests Received	Number of HTTP requests received by the servers. The statistics show the number of requests for each URL (page).
HTTP Requests Successful	Number of complete and positive HTTP responses (2xx- and 3xx-range responses) sent to the clients. The statistics show the number of requests for each URL (page).
HTTP Requests Failed	Number of HTTP requests from the clients that failed for any reason. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (404)	Number of HTTP requests that failed due to missing files (error 404). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (50x)	Number of HTTP requests that failed due to lack of resources (500-series errors). The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Write Error)	Number of HTTP requests that failed due to a socket write error. The statistics show the number of requests for each URL (page).
HTTP Requests Failed (Other)	Number of HTTP requests that failed for the following reasons: <ul style="list-style-type: none"> ▪ A request was received before the response to the previous request was fully transmitted. ▪ The client closed the connection before the response to the previous request was fully transmitted. ▪ The client reset the connection before the response to the previous request was fully transmitted. ▪ The client violated the TCP specification before the response to the previous request was fully transmitted.



HTTP Sessions Rejected (503)	Service Unavailable. Number of HTTP sessions that could not be established due to lack of resources on the server.
HTTP Sessions Timeouts (408)	<p>Number of HTTP 408 responses sent. This statistic includes all 408 responses sent regardless of whether they were received for a pending HTTP request or not.</p> <p>IxLoad counts 408 responses differently depending on whether or not a client has a pending HTTP request:</p> <p>If a client has an HTTP request pending and it receives a 408 response, IxLoad increments the HTTP Received 408, HTTP Requests Failed (4xx), and HTTP Requests Failed statistics.</p> <p>If a client does not have an HTTP request pending and it receives a 408 response, IxLoad only increments the HTTP Received 408 sta</p>
HTTP Transactions Active	Number of HTTP transactions transferring HTTP commands or data.
HTTP Bytes Received	<p>Number of HTTP bytes received by the servers.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic due to increases caused by retransmits.</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not (HTTP only).</p>
HTTP Bytes Sent	<p>Number of HTTP bytes sent by the servers.</p> <p>If you probe the network link with a sniffer, this statistic is not the same as the total amount of TCP payload that appears on the link. The total amount of TCP payload can be greater than this statistic (increased by retransmits) or less than this statistic (decreased by broken or reset connections).</p> <p>SSL-encrypted payload data is included in this statistic but SSL handshake overhead is not (HTTP only).</p>
Physical Rx Drops	Number of incoming packets dropped due to buffer overflow.
Physical Tx Drops	<p>Number of outgoing packets dropped due to buffer overflow.</p> <p>Typically, this is caused by stopping a large test or configuring a Ramp Down time that is too short.</p>



QuickTCP

This section describes the QuickTCP Tcl API objects.

In this chapter:

Overview	1466
QuickTCP Client Agent	1468
QuickTCP Server Agent	1476
QuickTCP Statistics	1479

Overview

QuickTCP protocol commands are organized as a simple structure as follows:

- QuickTCP Client Agent
 - QuickTCP Command
 - QuickTCP Client Advanced
- QuickTCP Server Agent
 - QuickTCP Server TCP

Objectives

The objectives (userObjective) you can set for QuickTCP are listed below. Test objectives are set in the ixTimeline object.

- connectionRate
- simulatedUsers
- concurrentConnections

QuickTCP Client Agent

The QuickTCP Client Agent defines a simulated user performing QuickTCP concommands. Refer to `QuickTCP Client Agent` for a full description of this command. The important options of this command are shown below.

Option	Usage
enable	Enables the use of the QuickTCP client agent.
name	The name associated with the client agent.

QuickTCP Command

Each client command is a single attack. Refer to `QuickTCP Command` for a full description of this command. The important options of this command are shown below.

Option	Usage
id	The type of command. Only <code>Connect</code> is currently supported.
destination	The TCP connect destination.
payload	QuickTCP client supports sending out and receiving payload data.

QuickTCP Server Agent

The QuickTCP Server Agent defines the operation of the QuickTCP server. Refer to `QuickTCP Server Agent` for a full description of this command. The important options of this command are shown below.

Option	Usage
enable	Enables the use of the HTTP server agent.
name	The name associated with the server agent.

QuickTCP Server HTTP Options

The QuickTCP Server HTTP options define basic HTTP protocol usage. Refer to `QuickTCP Server TCP` for a full description of this command. The important options of this command are shown below.

Option	Usage
listenPort	The port number that the server will listen on.
payload	The response of the QuickTCP server, to the payload, it receives from the client.



QuickTCP Client Agent

QuickTCP Client Agent - configure a QuickTCP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_QuickTCPClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_QuickTCPClient1 agent.config
```

DESCRIPTION

A QuickTCP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

The statistics published by this agent are a combination of statistics unique to this agent and common statistics available from other protocols. The TCP statistics unique to this agent are listed below. See *Network and Timeline Statistics* on page A-1 for a list of the common TCP statistics.

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity QuickTCPClient1 of NetTraffic Traffic1@Network1
#####
```

```
set Activity_QuickTCPClient1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "QuickTCP Client" ]  
  
#####  
# Timeline1 for activities QuickTCPClient1  
#####  
set Timeline1 [::IxLoad new ixTimeline]  
$Timeline1 config \  
    -rampUpValue                    1 \  
    -rampUpType                      0 \  
    -offlineTime                    0 \  
    -rampDownTime                   20 \  
    -standbyTime                    0 \  
    -iterations                     1 \  
    -rampUpInterval                 1 \  
    -sustainTime                    20 \  
    -timelineType                   0 \  
    -name                           "Timeline1"  
  
$Activity_QuickTCPClient1 config \  
    -enable                         true \  
    -name                           "QuickTCPClient1" \  
    -enableConstraint               false \  
    -userObjectiveValue             100 \  
    -constraintValue                100 \  
    -userObjectiveType              "simulatedUsers" \  
    -timeline                       $Timeline1  
  
$Activity_QuickTCPClient1 agent.config \  
    -enable                         true \  
    -name                           "QuickTCPClient1"
```



```
$Activity_QuickTCPClient1 agent.pm.advanced.config \  
-noClose                                false \  
-requestTimeout                          300000 \  
-closeWithRST                            false \  
-threeWayClose                            true  
  
$Activity_QuickTCPClient1 agent.pm.commands.clear  
  
$Activity_QuickTCPClient1 agent.pm.commands.appendItem \  
-id                                       "Connect" \  
-fixedPayload                            32 \  
-destination                             "198.18.0.101" \  
-minPayload                               1 \  
-payload                                  0 \  
-maxPayload                               32
```

SEE ALSO

QuickTCP Command, ixNetTraffic

QuickTCP Command

QuickTCP Command - configure a command that the QuickTCP client will execute

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_QuickTCPClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_QuickTCPClient1 agent.pm.commands.appendItem
```

DESCRIPTION

A QuickTCP client command is added to the `pm.commands` option of the QuickTCP Client Agent object using its `appendItem`.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

None.

OPTIONS

`id`

The name of the type of command.

QTCP command to be executed. One of the following:

TCP SessionCommand	The {TCP Session} command establishes a connection to the TCP server, transfers a payload to or from it (depending on the how the Quick TCP server is configured), and then closes the connection. A single {TCP Session} command is equivalent to using the separate commands Open, Payload, and Close. This command has two parameters: Destination and Payload.
OpenCommand	The Open command establishes a connection to the TCP server. This command has one parameter: Destination.
CloseCommand	The Close command closes the most recently estabconnection to the TCP server. This command has no parameters.
PayloadCommand	The Payload command transfers a TCP payload to or from the Quick TCP server (depending on the how the Quick TCP server is configured) connected to in the most recent Open command. This command has one parameter: Payload.
ThinkCommand	The {Think} command pauses the execution of the command list. This command has one parameter: Duration.



`destination`

Indicates the TCP server the QuickTCP client will connect to. (Default = "").

`duration`

Specifies the duration of the pause (Think time), in milliseconds.

`payload`

The QuickTCP client can send a payload with fixed (1), or random (2) data.



Note: If the server receives a smaller payload than the size it is configured to respond with, it will add random data to pad the payload. That is, it will send more data than it actually received.

`fixedPayload`

This accepts integer value. The payload size is fixed in every packet.

`minPayload`

For random payload, the payload size varies between a minimum and maximum packet size. The `minPayload` option specifies the minimum size of the payload.

`maxPayload`

This option indicates the maximum payload size in a random payload packet.

`closeWithRST`

If true, the QuickTCP client terminates TCP connections by sending RST instead of FINs.

`requestTimeout`

The amount of time, in milliseconds, the client should wait for the server to return a payload. (Min =1, Max =300,000).

`threeWayClose`

This specifies whether connections are terminated by an exchange of only three TCP packets. If disabled, the regular TCP termination sequence is used, which implies four packets.



Note: This is related to the side doing the passive close.

EXAMPLE

```
$Activity_QuickTCPClient1 agent.pm.commands.appendItem \  
-id "Connect" \  
-fixedPayload 32 \  
-destination "198.18.0.101" \  
-minPayload 1 \  
-payload 0 \  
-maxPayload 32
```

SEE ALSO

QuickTCP Client Agent



QuickTCP Client Advanced

QuickTCP Client Advanced Options - configure the QuickTCP client's advanced options

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_QuickTCPClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_QuickTCPClient1 agent.pm.advanced.config
```

DESCRIPTION

A QuickHTTP client's advanced options are set by modifying the options of the `pm.advanced` option of the `QuickTCP Client Agent` object using its `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`closeWithRST`

This option causes the client to terminate the connection by sending a TCP RST (Reset) segment instead of a FIN.

`noclose`

This options if enabled does not close the conenction even if all the transactions for that connection are finished. The server can close the connections, for examin case of timeout.

`threeWayClose`

If true, the connections are closed using three way handshake. This field accepts boolean value. If false, the connections are closed using a four-way handshake using this option.

EXAMPLE

```
$Activity_QuickTCPClient1 agent.pm.advanced.config \  
-noClose                               false \  
-requestTimeout                         300000 \  
-closeWithRST                           false \  
-threeWayClose                           true
```

SEE ALSO

`QuickTCP Client Agent`



QuickTCP Server Agent

QuickTCP Server Agent - configure a QuickTCP server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_QuickTCPServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_QuickTCPServer1 agent.config
```

DESCRIPTION

A QuickTCP server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

The set of Web pages available through the server is described in the `webPageLisoption`, which references response headers held in the `resoption` and cookies held in the `cookieList`.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this action. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
#####
# Activity QuickTCPServer1 of NetTraffic Traffic2@Network2
```

```
#####  
  
set Activity_QuickTCPServer1 [$Traffic2_Network2 activityList.appendItem \  
    -protocolAndType                "QuickTCP Server" ]  
  
set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]  
  
$Activity_QuickTCPServer1 config \  
    -enable                          true \  
    -name                            "QuickTCPServer1" \  
    -timeline                        $_Match_Longest_  
  
$Activity_QuickTCPServer1 agent.config \  
    -enable                          true \  
    -name                            "QuickTCPServer1"  
  
$Activity_QuickTCPServer1 agent.pm.advanced.config \  
    -threeWayClose                   true  
  
$Activity_QuickTCPServer1 agent.pm.tcp.config \  
    -fixedPayload                    32 \  
    -minPayload                      1 \  
    -payload                         0 \  
    -listenPort                      5000 \  
    -maxPayload                      32
```

SEE ALSO

QuickTCP Server TCP



QuickTCP Server TCP

QuickTCP Server TCP Options - set the QuickTCP server's TCP options

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_QuickTCPServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_QuickTCPServer1 agent.pm.tcp.config
```

DESCRIPTION

A QuickTCP server's TCP options are set by modifying the options of the `pm.tcp` option of the QuickTCP Server Agent object using its `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`listenPort`

The port number to which the QuickTCP server will respond to requests. (Default = 5,000).

`payload`

Specifies the response of the server to the payload received from the QuickTCP client.

0 = Do not send any payload.

1 = Echo everything that is received.

2 = Echo a fixed amount of the payload received.

3 = Echo a random amount of the payload received.

`minPayload`

For random payload, the payload size varies between a minimum and maximum packet size. The `minPayload` option specifies the minimum size of the payload.

`maxPayload`

This option indicates the maximum payload size in a random payload packet.

`threeWayClose`

This enables (1) or disables (0) a three-way close instead of the normal four-way TCP connection termination sequence.

EXAMPLE

```
$Activity_QuickTCPServer1 agent.pm.tcp.config \
    -fixedPayload          32 \
    -minPayload            1 \
    -payload               0 \
    -listenPort            5000 \
    -maxPayload            32
```

SEE ALSO

QuickTCP Server Agent

QuickTCP Statistics

The statistics published by this agent are a combination of statistics unique to this agent and statistics common to other protocols. The TCP statistics unique to this agent are listed in the table below. See TCP, Run State, and Curve Segment Statistics (see ["IP, TCP, Run State, and Curve Segment Statistics"](#) on page 2096) for a list of the common TCP statistics.

Statistic	Description
TCP Simulated Users	Number of simulated users.
Concurrent Connections	Number of connections established at the same time.
Connection Rate	Rate (per second) at which connections are established.
Network statistics	
Physical Rx Drops	Number of incoming packets dropped due to buffer overflow.
Physical Tx Drops	Number of outgoing packets dropped due to buffer overflow. Typically, this is caused by stopping a large test or configuring a Ramp Down time that is too short.

24

Radius

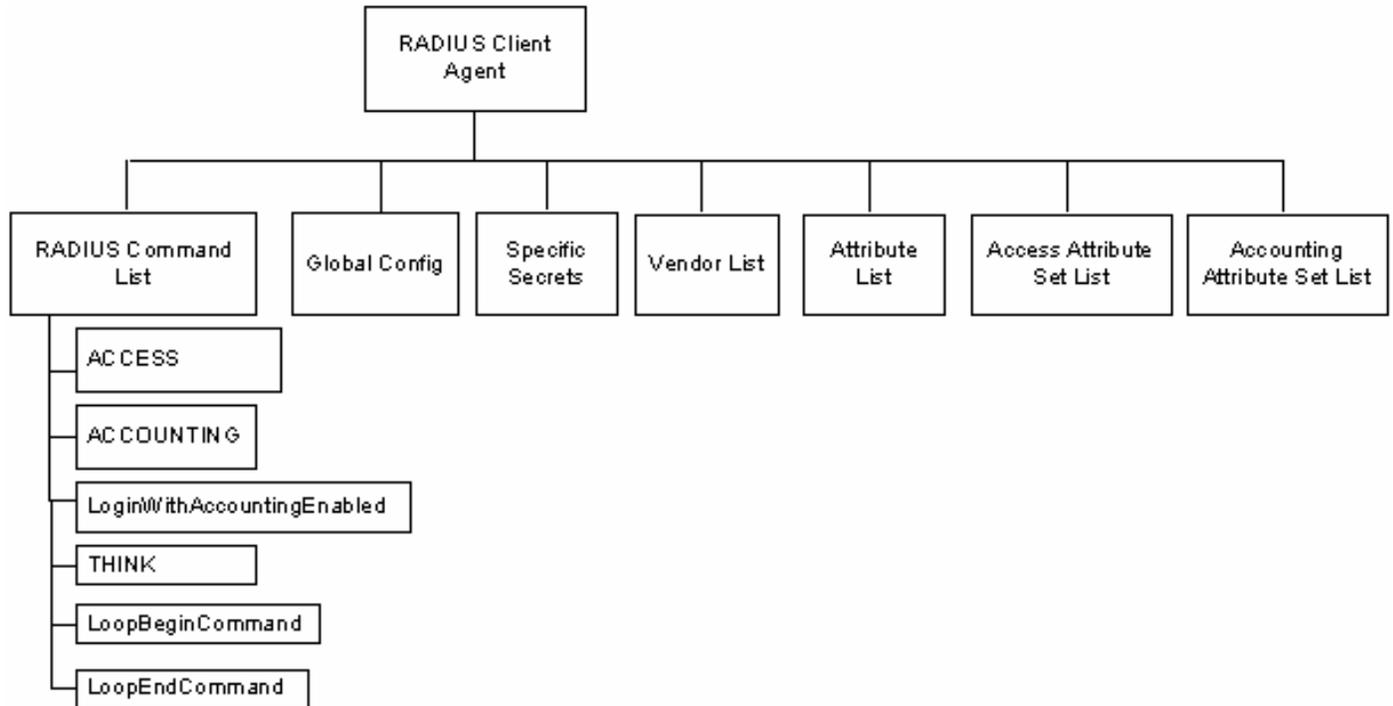
This section describes the Radius Tcl API objects.

In this chapter:

Overview	1482
Radius Client Agent	1486
RADIUS Client Statistics	1505

Overview

The IxLoad Radius API consists of a client agent and its commands, structured as shown below.



Objectives

The objectives (userObjective) you can set for Radius are listed below. Test objectives are set in the ixTimeline object.

- transactionRate
 - simulatedUsers
 - concurrentSessions
-

Radius Client Agent

The Radius Client Agent simulates user requests for access by sending user names and passwords to a RADIUS server, and recording the responses returned by the server. Refer to `Radius Client Agent` on page 25-5 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>protocolAndType</code>	Protocol used by the client agent. Defines the agent as either a client or server.

Radius Command List

The Radius Command List creates the list of Radius commands that the client will send to a Radius server. Refer to `Radius Command List` on page 25-11 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>id</code>	Command that client will send.

Global Config

The Global Config contains the parameters that define the way the IxLoad RADIUS client performs overall. Refer to `Global Config` on page 25-16 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>defaultAuthentica</code>	The UDP port on the RADIUS server to which the IxLoad client sends Access-Requests.
<code>defaultAccounting</code>	The UDP port on the RADIUS server to which the IxLoad client sends Accounting-Requests.
<code>authenticationRe</code>	Number of times the IxLoad RADIUS client will re-send an unacAccess-Request. If the RADIUS server does not respond to an Access-Request within the Response Timeout period, the client resends the Access-Request.
<code>responseTimeouot</code>	Elapsed time, in seconds, allowed for the server to respond to a clirequest.
<code>defaultSharedSe</code>	Secret used if no server-specific secret is configured. To configure server-specific secrets, see <code>Creating and Editing Server-Specific Shared Secrets</code> on page 23-13.



Option	Description
send_ACCOUNTING_REQUESTS_when_ACCESS_REQUEST_are_pending	Enabled: If enabled, the IxLoad client requests accounting data even if requests for authentication (Access-Requests) are still pending. Disabled: If disabled, the IxLoad client does not send Accounting-data if any Access-Requests are pending.
maxPendingRe	Maximum number of pending requests per client that the IxLoad client maintains with the RADIUS server.

Specific Secrets

To configure secrets to be used with specific servers. Refer to *Specific Secrets* on page 25-18 for a full description of this command. The options supported are listed below.

Option	Description
sharedSecretList	The list of shared secrets to be used with specific servers.

Vendor List

The Vendors tab contains the predefined vendors and their vendor-codes that the IxLoad client uses. You cannot modify or delete the predefined vendors and codes, but you can add additional vendors and codes. Refer to *Vendor List* on page 25-19 for a full description of this command. Attribute List

The Attributes list contains the predefined Attributes, their values, and the vendor that originally specified them. All the predefined Attributes are standard RADIUS Attributes; there are no vendor-specific Attributes in the list. Refer to *Attribute List* on page 25-20 for a full description of this command.

Access Attribute Set List

Access Attribute sets are groups of Access Attributes that are included in RADIUS messages. Refer to *AccessAttribSetList* on page 25-22 for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	This represents the name of the ACCESS attribute set.

Accounting Attribute Set List

Accounting Attribute sets are groups of Accounting Attributes that are included in RADIUS messages. Refer to `AcctngAttribSetList` on page 25-23 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>acctngAttribVal</code>	This represents the name of the ACCOUNTING attribute set.



Radius Client Agent

Radius Client Agent - create a Radius client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_RADIUSClient1 agent.config
```

DESCRIPTION

The Radius Client Agent simulates user requests for access by sending user names and passwords to a RADIUS server, and recording the responses returned by the server. A Radius client agent is added to the `activityList` option of the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` submay be used to modify the `activityList`.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#-----
# Activity RADIUSClient1 of NetTraffic Traffic1@Network1
#-----
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem \
```

```
-protocolAndType           "radius Client" ]  
$Activity_RADIUSClient1 agent.config \  
-enable                    true \  
-name                      "RADIUSClient1"
```

SEE ALSO

`ixNetTraffic`



Radius Command List

Radius Command List—Creates the list of Radius commands that the client will send to a Radius server.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_RADIUSClient1 agent.pm.cmdList.appendItem options...
```

DESCRIPTION

A command is added to the Radius Command List object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command (see the example).

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

Radius command to be executed. One of the following:

Command	Description
ACCESS	Sends an ACCESS request to a RADIUS server. An ACCESS request is a query to determine whether a user should be allowed access to a specific NAS. The ACCESS request also can include a request for special services that the user may require.
ACCOUNTING	Sends an ACCOUNTING request to a RADIUS server. An ACCOUNTING request is a query for obtaining information that is used to provide accounting for a service provided to a user.
LoginWithAccountin	This is a combination of an ACCESS and ACCOUNTING comThis command simulates a scenario in which a user logs in to a NAS port and the NAS has accounting enabled for that user.
THINK	Causes the client to become idle. {Think} is an internal IxLoad command intended to assist your testing; it is not a command defined in the RADIUS protocol. If you specify identical values for the minimum and maximum times, the client will be idle for a fixed length of time. If you specify different values for the minimum and maximum times, IxLoad will select a value within the range and cause the client to be idle for that length of time.

Command	Description
LoopBeginCommand	An IxLoad command that you can add to the Command List to cause the commands between it and the {LoopEndCommand} to be executed a specified number of times.
LoopEndCommand	Ends the list of commands that will be executed by the preced{Loop Begin} command.

Arguments for id = ACCESS

serverAddr

The IP address or symbolic destination (DUT configuration) of the RADIUS server to which the IxLoad client sends the ACCESS request packet. To specify a port, enter colon (:) then the port number after the IP address. For example: 192.168.100.1:1813. (Default = "198.18.0.100").

authenticationMethod

Method used to establish (and in the case of EAP-MD5, encrypt) the authenticacredentials of the simulated supplicants. Depending on the method you select, IxLoad enables and disables various Credentials fields.

The choices are:

Value	Description
PAP	(minimum) Password Authentication Protocol.
CHAP	Challenge Handshake Authentication Protocol. For CHAP the challenge that is normally generated by the authenticator/RAS is internally generated by IxLoad.
EAP-MD5	(default) Extensible Authentication Protocol, with MD5 encryption.
MS-CHAP	Microsoft CHAP, version 1
MS-CHAPv2	(maximum) Microsoft CHAP, version 2

userName

User name of the supplicant included in ACCESS request. maximum = 256.

password

Password for the supplicant. maximum = 128.

eapMD5Identity

If the authenticationMethod is EAP-MD5, this is the identity of the supplicant. maximum = 256.



attributeSetName

Attributes sent with the ACCESS request. See *AccessAttribSetList* on page 25-22. maximum = 256.

Arguments for id = ACCOUNTING

serverAddr

The IP address or symbolic destination (DUT configuration) of the RADIUS server to which the IxLoad client sends the ACCOUNTING request packet. To specify a port, enter colon (:) then the port number after the IP address. For example: 192.168.100.1:1813. (Default = "198.18.0.100")

userName

User name included in ACCOUNTING request. maximum = 256

You can use sequence generators in this field to create a range of unique user names. See Appendix W, *Using Automatic Sequence Generators*.

acctSessionId

Numeric identifier of the call for which the ACCOUNTING request is being sent. maximum = 256.

acctStatusType

Type of information that the ACCOUNTING request obtains. The values are:

Value	Description
1	Start (start time of call)
2	Stop (end time of call)
3	Interim-Update
7	Accounting-On
8	Accounting-Off
9-14	Reserved for Tunnel Accounting
15	Reserved for Failed

attributeSetName

Attributes sent with the ACCOUNTING request. See *AcctngAttribSetList* on page 25-23. maximum = 256.

Arguments for id = LoginWithAccountingEnabled

serverAddr

The IP address or symbolic destination (DUT configuration) of the RADIUS server to which the IxLoad client sends the Accounting-Request packet. To specify port, enter colon (:) then the port number after the IP address. For example: 192.168.100.1:1813. (Default = "198.18.0.100")

authenticationPort

The UDP port on the RADIUS server to which the IxLoad client sends ACCESS requests. minimum = "1", maximum = "65535", default = "1812"

accountingPort

The UDP port on the RADIUS server to which the IxLoad client sends ACCOUNTING requests. minimum = "1" maximum = "65535" default = "1813"

authenticationMethod

Method used to establish (and in the case of EAP-MD5, encrypt) the authentic credentials of the simulated supplicants. Depending on the method you select, IxLoad enables and disables various Credentials fields.

The choices are:

Value	Description
PAP	(minimum) Password Authentication Protocol.
CHAP	Challenge Handshake Authentication Protocol. For CHAP the challenge that is normally generated by the authenticator/RAS is internally generated by IxLoad.
EAP-MD5	(default) Extensible Authentication Protocol, with MD5 encryption.
MS-CHAP	Microsoft CHAP, version 1
MS-CHAPv2	(maximum) Microsoft CHAP, version 2

userName

User name of the supplicant included in ACCESS request. maximum = 256.

You can insert sequence generators into this field to create unique entries autoFor information on how to use sequence generators, see [Using Automatic Sequence Generators](#) on page A-1.

password

Password for the supplicant. maximum = 128.

You can insert sequence generators into this field to create unique entries autoFor information on how to use sequence generators, see [Using Automatic Sequence Generators](#) on page A-1.



eapMD5Identity

If the authenticationMethod is EAP-MD5, this is the identity of the supplicant. maximum = 256

attributeSetName

Attributes sent with the ACCESS request. See AccessAttribSetList on page 25-22.

learnFramedIp

After the client receives an ACCESS-ACCEPT, the first ACCOUNTING-START request that it sends may include the attribute Framed-IP, and a value for it. This parameter determines the source of the value for the Framed-IP attribute. If this option is enabled:

- If the ACCESS-ACCEPT contains a Framed-IP attribute and a value, the cli uses the value from the ACCESS-ACCEPT.
- If the attribute set includes a Framed-IP attribute and value, the client ignores the value in the attribute set and uses the value from the ACCESS-ACCEPT. •If the ACCESS-ACCEPT does not contain a Framed-IP attribute but the attribute set does, the client uses the value from the attribute set.
- If neither the ACCESS-ACCEPT nor the attribute set contains a Framed-IP attribute, then this option is ignored and the ACCOUNTING-START does not contain a Framed-IP attribute.

Default = false.

Arguments for id = THINK

minimumInterval

Minimum length of time that the user will remain inactive for. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

maximumInterval

Maximum length of time that the user will remain inactive for. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

Arguments for id = LoopBeginCommand

loopCount

Number of times to repeat the enclosed commands. '0' treated as infinity. Mini= "0," maximum = "2,147,483,647." (Default = "5").

Arguments for id = LoopEndCommand

None.

EXAMPLE

```
$Activity_RADIUSClient1 agent.pm.cmdList.appendItem \  
-id                "ACCESS" \  
-userName          "ixia" \  
-authenticationMethod 0 \  
-attributeSetName  "ACCESS-REQUEST-Attribute-Set-1" \  
-eapMD5Identity    "" \  
-serverAddr        "198.18.0.100" \  
-password          "ixia"
```

SEE ALSO

Radius Client Agent



Global Config

Global Config—Configures the parameters that define the way the IxLoad RADIUS client performs overall.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_RADIUSClient1 agent.pm.globalConfig.config options...
```

DESCRIPTION

To configure the Global Config options, use the `appendItem` command on the `pm.optionSetManager` component of the Radius Client Agent. Note the use of the `'pm.'` component. See the following example:

```
$Activity_RADIUSClient1 agent.pm.globalConfig.config
```

SUBCOMMANDS

None.

OPTIONS

`defaultAuthenticationPort`

The UDP port on the RADIUS server to which the IxLoad client sends Access-Requests. minimum = "1", maximum = "65535", default= "1812".

`defaultAccountingPort`

The UDP port on the RADIUS server to which the IxLoad client sends Accountminimum = "1", maximum = "65535", default = "1813".

`authenticationRetryCount`

Number of times the IxLoad RADIUS client will re-send an unacknowledged Access-Request.

If the RADIUS server does not respond to an Access-Request within the Response Timeout period, the client resends the Access-Request. minimum = "0", maximum = "65535", default = "3".

`responseTimeouot`

Elapsed time, in seconds, allowed for the server to respond to a client request. minimum = "1", maximum = "65535", default = "5".

`defaultSharedSecret`

Secret used if no server-specific secret is configured. See `Specific Secrets` on page 25-18.
`minimum = "1", maximum = "256", default = "ixia".`

`send_ACCOUNTING_REQUESTS_when_ACCESS_REQUEST_are_pending`

If enabled (1), the IxLoad client requests accountdata even if requests for authentication (Access-Requests) are still pending. If disabled (0), the IxLoad client does not send accounting data if any Access-Requests are pending. Default = "1".

`maxPendingRequestPerClient`

Maximum number of pending requests per client that the IxLoad client maintains with the RADIUS server. `minimum = "1", maximum = "64000", default = "100".`

EXAMPLE

```
$Activity_RADIUSClient1 agent.pm.globalConfig.config \  
-defaultAccountingPort          1813 \  
-defaultAuthenticationPort      1812 \  
-defaultSharedSecret            "ixia" \  
-authenticationRetryCount       3 \  
-accountingRetryCount           3 \  
-responseTimeout                5 \  
-send_ACCOUNTING_REQUESTS_when_ACCESS_REQUESTS_are_pending true \  
-maxConcurrentSessions          100 \  
-implicitLoopCheck              true
```

SEE ALSO

Radius Client Agent



Specific Secrets

Specific Secrets—Configures secrets to be used with specific servers.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_RADIUSClient1 agent.pm.specificSecrets.config options...
```

DESCRIPTION

To configure Specific Secrets, use the `appendItem` command on the `pm.optioncomponent` of the Radius Client Agent. Note the use of the `'pm.'` component.

SUBCOMMANDS

None.

OPTIONS

`sharedSecretList`

The list of shared secrets to be used with specific servers.

`clientIdRange`

This corresponds to the IP addresses configured in the network portion of the RADIUS client's NetTraffic. maximum = 256.

`serverIp`

IP address of the server to which the secret applies. minimum = "7" maximum = "19" default = "198.18.0.100".

`sharedSecret`

The shared secret is entered in this field. minimum = "1" maximum = "256" default = "ixia".

EXAMPLE

```
$Activity_RADIUSClient1 agent.pm.specificSecrets.sharedSecretList.appendItem \
-id          "ClientServerSecrets" \
-clientIdRange "1-5" \
```

```
-secretListString      ("\"198.18.0.101\", \"ixia\"), (\"198.18.0.102\",  
\"ixia\")"  
  
$Activity_RADIUSClient1  
agent.pm.specificSecrets.sharedSecretList(0).serverSecretList.appendItem \  
  
    -id                "ServerSecrets" \  
  
    -serverIP          "198.18.0.101" \  
  
    -sharedSecret      "ixia"
```

SEE ALSO

Radius Client Agent



Vendor List

Vendor List—contains the predefined vendors and their vendor-codes that the IxLoad client uses.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_RADIUSClient1 agent.pm.vendorList.v_list.appendItem \ options...
```

DESCRIPTION

To configure a Vendor List, use the `appendItem` command on the `pm.optionSetComponent` of the Radius Client Agent. Note the use of the `'pm.'` com

SUBCOMMANDS

None.

OPTIONS

`vendorName`

Name of the vendor. maximum = 256

`vendorId`

This is the vendor code. maximum = 8

EXAMPLE

```
$Activity_RADIUSClient1 agent.pm.vendorList.v_list.appendItem \
  -id "Vendor" \
  -vendorId "NA" \
  -vendorName "IETF RADIUS STANDARD" \
  -isPredefined true
```

SEE ALSO

Radius Client Agent

Attribute List

Attribute List—contains the predefined Attributes, their values, and the vendors that originally specified them.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_RADIUSClient1 agent.pm.attributeList.a_list.appendItem \ options...
```

DESCRIPTION

To configure an Attribute List, use the `appendItem` command on the `agent.pm` component of the Radius Client Agent. Note the use of the 'pm.' component.

SUBCOMMANDS

None.

OPTIONS

`attributeName`

Name of the attribute. maximum = "256"

`attributeCode`

The attribute code. minimum = "0" maximum ="255"

`vendorName`

Name of the vendor. maximum = "256"

`vendorId`

This represents the vendor id. maximum = "8"

`valueType`

This represents the data type of the attribute value. minimum = "0", maximum = "7"

The choices are:

Value	Description
0	Integer (1 octet)



1	Integer (2 octets)
2	Integer (3 octets)
3	Integer (4 octets)
4	String
5	IPv4 Address
6	MTU
7	Hexadecimal

relevance

This represents the request type with which the attribute can be used. minimum = "0" maximum = "2"

The choices are:

Value	Description
0	Both Authentication And Accounting
1	Authentication Only
2	Accounting Only

EXAMPLE

```
$Activity_RADIUSClient1 agent.pm.attributeList.a_list.appendItem \  
-id "Attribute" \  
-attributeCode 40 \  
-attributeName "Acct-Status-Type" \  
-valueType 3 \  
-relevance 2 \  
-vendorName "IETF RADIUS STANDARD" \  
-isPredefined true
```

SEE ALSO

Radius Client Agent



AccessAttribSetList

AccessAttribSetList—Configures the list of Access Attribute Sets.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem
options...]
Activity_RADIUSClient1 agent.pm.accessAttribSetList.accessAttribVal\
options...
```

DESCRIPTION

To configure an `AccessAttribSetList`, use the `appendItem` command on the `agent.pm` component of the `Radius Client Agent`. Note the use of the `'pm.'` com

SUBCOMMANDS

None.

OPTIONS

`attributeValueSetName`

This represents the name of the ACCESS attribute set.

`refCount`

The numerical order of the attribute set.

EXAMPLE

```
$Activity_RADIUSClient1
agent.pm.accessAttribSetList.accessAttributeValueSetList.appendItem \
-id                          "AttributeValueSet" \
-attributeValueSetName      "ACCESS-REQUEST-Attribute-Set-1" \
-refCount                    1
```

SEE ALSO

`Radius Client Agent`



AcctngAttribSetList

Accounting Attribute Set List—Configures the list of Accounting Attribute Sets.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RADIUSClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_RADIUSClient1
agent.pm.acctngAttribSetList.acctngAttribValueSetList.appendItem \ options...
```

DESCRIPTION

To configure an Accounting Attribute Set List, use the `appendItem` command on the `agent.pm` component of the `Radius Client Agent`. Note the use of the `'pm.'` component.

SUBCOMMANDS

None.

OPTIONS

`attributeValueSetName`

This represents the name of the ACCOUNTING attribute set.

`refCount`

The numerical order of the attribute set.

EXAMPLE

```
$Activity_RADIUSClient1
agent.pm.acctngAttribSetList.acctngAttribValueSetList.appendItem \
-id "AttributeValueSet" \
-attributeValueSetName "ACCOUNTING-REQUEST-Attribute-Set-1" \
-refCount 1
```

SEE ALSO

`Radius Client Agent`

RADIUS Client Statistics

The table below describes the RADIUS client statistics.

The Conditional Views column indicates the views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
Test Objective Related Statistics		
RADIUS Simulated Users	--	Number of RADIUS clients (NAS, RAS, or other RADIUS-enabled devices) simulated by the IxLoad RADIUS client.
RADIUS Transaction Rate	All	Average rate at which the client completed RADIUS transactions.
RADIUS Total Transactions	All	Total number of RADIUS transactions completed.
RADIUS Sessions Alive	All	Total number of RADIUS sessions online.
RADIUS Logged In Supplicants	All	Total number of simulated supplicants authenticated by the IxLoad RADIUS client.
Authentication Statistics		
RADIUS Authentications Attempted	All	Number of authentications attempted using RADIUS.
RADIUS Authentications Succeeded	All	Number of authentications that succeeded.
RADIUS Authentications Failed	All	Number of authentications that failed.
Access-Request and -Response Statistics		
RADIUS Total Access Requests Sent	All	Number of Access-Request packets sent by the client.
RADIUS Total Access Accept Received	All	Number of Access-Accept packets received by the client.
RADIUS Total Access Reject Received	All	Number of Access-Reject packets received by the client.
RADIUS Total Access Challenge Received	All	Number of Access-Challenge packets received by the client.
RADIUS Total Access Request Timeouts	All	Number of Access-Request packets sent for which no response was received within the timeout period.



RADIUS Total Invalid Replies To Access Requests	All	Number of invalid responses to Access-Request packets received by the client.
RADIUS Total Access Request Errors	All	Total number of errors that occurred either while sending an Access Request or afterwards. This statistic mainly counts socket-level errors and does not include timeouts and invalid responses, which are counted by other statistics.
RADIUS Total Access Request Aborted	All	Number of Access-Requests that were aborted.
Successful Authentications Statistics		
RADIUS Total Authentications Succeeded Using PAP	All	Number of successful PAP authentications.
RADIUS Total Authentications Succeeded Using CHAP-MD5	All	Number of successful CHAP-MD5 authentications.
RADIUS Total Authentications Succeeded Using EAP-MD5	All	Number of successful EAP-MD5 authentications.
RADIUS Total Authentications Succeeded Using MS-CHAPv1	All	Number of successful MS-CHAPv1 authentications.
RADIUS Total Authentications Succeeded Using MS-CHAPv2	All	Number of successful MS-CHAPv2 authentications.
Failed Authentications Statistics		
RADIUS Total Authentications Failed Using PAP	All	Number of failed PAP authentications.
RADIUS Total Authentications Failed Using CHAPMD5	All	Number of failed CHAP-MD5 authentications.
RADIUS Total Authentications Failed Using EAPMD5	All	Number of failed EAP-MD5 authentications.
RADIUS Total Authentications Failed Using MSCHAPv1	All	Number of failed MS-CHAPv1 authentications.
RADIUS Total Authentications Failed Using MSCHAPv2	All	Number of failed MS-CHAPv2 authentications.
Accounting Requests Statistics		

RADIUS Total Accounting Requests Sent	All	Total number of Accounting-Request packets sent by the client.
RADIUS Total Accounting Start Requests Sent	All	Number of Accounting-Request-Start packets sent by the client.
RADIUS Total Accounting Stop Requests Sent	All	Number of Accounting-Request-Stop packets sent by the client.
RADIUS Total Accounting Responses Received	All	Number of Accounting-Response packets received by the client.
RADIUS Total Timeouts for Accounting Requests	All	Total number of Accounting-Request packets for which no response was received within the timeout period.
RADIUS Total Timeouts for Accounting Start Requests	All	Number of Accounting-Request-Start packets for which no response was received within the timeout period.
RADIUS Total Timeouts for Accounting Stop Requests	All	Number of Accounting-Request-Stop packets for which no response was received within the timeout period.
RADIUS Total Invalid Replies To Accounting Requests	All	Total number of invalid replies to Accounting-Request packets received by the client.
RADIUS Total Accounting Request Errors	All	Total number of timeouts and invalid responses to Accounting-Request packets.
RADIUS Total Accounting Request Aborted	All	Total number of Accounting-Requests that were aborted.
Request / Response Statistics		
RADIUS Total Requests Sent	All	Total number of Access-Requests and Accounting-Requests sent by the client.
RADIUS Total Responses Received	All	Total number of responses to Access-Requests and Accounting-Requests received by the client.
RADIUS Total Responses To Accounting Stop	All	Total number of responses to Accounting-Request-Stop packets received by the client.
Request / Response Rate Statistics		
RADIUS Requests Sent Per Second	All	Rate at which the client sent Access-Request or Accounting-Request packets.
RADIUS Responses Received Per Second	All	Rate at which the client received responses to Access-Request or Accounting-Request packets.
RADIUS Access Requests Sent Per Second	All	Rate at which the client sent Access-Request packets.
RADIUS Total Accounting Requests Sent Per Second	All	Rate at which the client sent Accounting-Request packets.
RADIUS Total Accounting Start Requests Sent Per Second	All	Rate at which the client sent Accounting-Request-Start packets.



RADIUS Total Accounting Stop Requests Sent Per Second	All	Rate at which the client sent Accounting-Request-Stop packets.
RADIUS Total Session Teardowns	All	Number of RADIUS sessions torn down.
RADIUS Session Teardown Rate	All	Rate at which the client tore down RADIUS sessions.
Throughput Statistics		
RADIUS Total Bytes Sent	All	Total number of RADIUS bytes (headers+payload) sent.
RADIUS Total Bytes Received	All	Total number of RADIUS bytes (headers+payload) received.
RADIUS Total Bytes Sent and Received	All	Combined total of RADIUS bytes (headers+payload) sent and received.
RADIUS UDP Packets Sent	All	Number of UDP packets sent with RADIUS payloads.
RADIUS UDP Packets Received	All	Number of UDP packets received with RADIUS payloads.
RADIUS Bytes Sent per sec	All	Rate at which the client sent RADIUS data, in bytes per second.
RADIUS Bytes Received per sec	All	Rate at which the client received RADIUS data, in bytes per second.
RADIUS UDP Packets Sent per sec	All	Rate at which the client sent UDP packets with RADIUS payloads, in bytes per second.
RADIUS UDP Packets Received per sec	All	Rate at which the client received UDP packets with RADIUS payloads, in bytes per second.
Retransmission Statistics		
RADIUS Total Retransmissions For Access Requests	All	Total number of Access-Requests that had to be retransmitted.
RADIUS Total Retransmissions For Accounting Requests	All	Total number of Accounting-Requests that had to be retransmitted.
RADIUS Total Retransmissions For Accounting Start Requests	All	Total number of Accounting-Request-Start packets that had to be retransmitted.
RADIUS Total Retransmissions For Accounting Stop Requests	All	Total number of Accounting-Request-Stop packets that had to be retransmitted.
Response Time Statistics		

RADIUS Average Time To Receive Access Response	All	Average time elapsed between the time the client sent an Access-Request and the time it received any type of response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RADIUS Average Time To Receive Access Accept Response	All	Average time elapsed between the time the client sent an Access-Request and the time it received an Access-Accept in response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RADIUS Average Time To Receive Access Reject Response	All	Average time elapsed between the time the client sent an Access-Request and the time it received an Access-Reject in response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RADIUS Average Time To Receive Accounting Response	All	Average time elapsed between the time the client sent an Accounting-Request and the time it received an Accounting-Response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
Access Accept Latency statistics		
RADIUS Total Access Accept Responses With Latency Between 0 to 2 millisecc	--	Number of Access Accept responses received with latencies between 0 and 2 milliseconds.
RADIUS Total Access Accept Responses With Latency Between 2 to 5 millisecc	--	Number of Access Accept responses received with latencies between 2 and 5 milliseconds.
RADIUS Total Access Accept Responses With Latency Between 5 to 10 millisecc	--	Number of Access Accept responses received with latencies between 5 and 10 milliseconds.
RADIUS Total Access Accept Responses With Latency Between 10 to 50 millisecc	--	Number of Access Accept responses received with latencies between 10 and 50 milliseconds.
RADIUS Total Access Accept Responses With Latency Between 50 to 100 millisecc	--	Number of Access Accept responses received with latencies between 50 and 100 milliseconds.
RADIUS Total Access Accept Responses With Latency Between 100 to 500 millisecc	--	Number of Access Accept responses received with latencies between 100 and 500 milliseconds.



RADIUS Total Access Accept Response With Latency Greater Than 500 millisec	--	Number of Access Accept responses received with latencies over 500 milliseconds.
Access Reject Latency statistics		
RADIUS Total Access Reject Responses With Latency Between 0 to 2 millisec	--	Number of Access Reject responses received with latencies between 0 and 2 milliseconds.
RADIUS Total Access Reject Responses With Latency Between 2 to 5 millisec	--	Number of Access Reject responses received with latencies between 2 and 5 milliseconds.
RADIUS Total Access Reject Responses With Latency Between 5 to 10 millisec	--	Number of Access Reject responses received with latencies between 5 and 10 milliseconds.
RADIUS Total Access Reject Responses With Latency Between 10 to 50 millisec	--	Number of Access Reject responses received with latencies between 10 and 50 milliseconds.
RADIUS Total Access Reject Responses With Latency Between 50 to 100 millisec	--	Number of Access Reject responses received with latencies between 50 and 100 milliseconds.
RADIUS Total Access Reject Responses With Latency Between 100 to 500 millisec	--	Number of Access Reject responses received with latencies between 100 and 500 milliseconds.
RADIUS Total Access Reject Response With Latency Greater Than 500 millisec	--	Number of Access Reject responses received with latencies over 500 milliseconds.
Accounting Response Latency statistics		
RADIUS Total Accounting Responses With Latency Between 0 to 2 millisec	--	Number of Access Response responses received with latencies between 0 and 2 milliseconds.
RADIUS Total Accounting Responses With Latency Between 2 to 5 millisec	--	Number of Access Response responses received with latencies between 2 and 5 milliseconds.
RADIUS Total Accounting Responses With Latency Between 5 to 10 millisec	--	Number of Access Response responses received with latencies between 5 and 10 milliseconds.
RADIUS Total Accounting Responses With Latency Between 10 to 50 millisec	--	Number of Access Response responses received with latencies between 10 and 50 milliseconds.

RADIUS Total Accounting Responses With Latency Between 50 to 100 millisecc	--	Number of Access Response responses received with latencies between 50 and 100 milliseconds.
RADIUS Total Accounting Responses With Latency Between 100 to 500 millisecc	--	Number of Access Response responses received with latencies between 100 and 500 milliseconds.
RADIUS Total Accounting Response With Latency Greater Than 500 millisecc	--	Number of Access Response responses received with latencies over 500 milliseconds.

25

RTSP

This section describes the RTSP Tcl API objects.



In this chapter:

Overview	1513
RTSP Client Agent	1517
RTSP Server Agent	1535
RTSP Statistics	1545

Overview

RTSP protocol commands are organized as:

- RTSP Client Agent
- RtspCommand
- RtspHeaders
- RtspHeader
- RTSP Server Agent
- PresentationItem
- Content
- Stream

Objectives

The objectives (userObjective) you can set for RTSP are listed below. Test objecare set in the ixTimeline object.

- connectionRate
- transactionRate
- simulatedUsers
- concurrentConnections

RTSP Client Agent

The RTSP Client Agent defines a simulated user performing RTSP requests against one or more RTSP servers. Refer to `RTSP Client Agent` for a full descriptof this command. The important options of this command are listed below.

Option	Usage
enable	Enables the use of the RTSP client agent.
name	The name associated with the client agent.
rtspTransport	The RTSP transport mechanism that the client will request.
commandList	A list of RTSP commands that the client will transmit, with arguments. Each list element is of type <code>RtspCommand</code> .
rtspHeaders	A list of RTSP headers that the client will transmit with each command. Each list element is of type <code>RtspHeaders</code> .
commandTimeou t	The client command timeout.



RtspCommand

Each client command is a single step in the interaction. Refer to `RtspCommand` for a full description of this command. The important options of this command are listed below.

Option	Usage
command arguments	The RTSP command, with optional arguments, to be executed.
destination	The name/address of the RTSP server.
media	The URL of the media object to be controlled.

RtspHeaders

The `RtspHeaders` command specifies a client emulation and includes a list of `name=value` header pairs. Refer to `RtspHeaders` for a full description of this command. The important options of this command are listed below.

Option	Usage
clientEmulation	Indicates the type of RTSP client to emulate.
list	A list of individual RTSP headers. Each list item is of type <code>RtspHeader</code> .

RtspHeader

Each `RtspHeader` item represents a single `name=value` header pair. Refer to `RtspHeader` for a full description of this command. The important options of this command are listed below.

Option	Usage
name	The <code>name</code> part of the pair.
value	The <code>value</code> part of the pair.

RTSP Server Agent

The RTSP Server Agent defines the operation of the RTSP server. Refer to `RTSP Server Agent` for a full description of this command. The important options of this command are listed below.

Option	Usage
<code>enable</code>	Enables the use of the server agent.
<code>name</code>	The name associated with the server agent.
<code>port</code>	The port number that the server will respond on.
<code>serverEmulation</code>	The type of RTSP server that the server agent will emulate.
<code>presentationList</code>	The set of media presentations that the server will respond for. Each item is of type <code>PresentationItem</code> .
<code>contentList</code>	A list of contents that are used in the <code>presentationList</code> . Each item is of type <code>Content</code> .
<code>commandTimeout</code>	Response timeout value.

PresentationItem

The `PresentationItem` is a specification of a media presentation offered by the server. Refer to `PresentationItem` for a full description of this command. The important options of this command are listed below.

Option	Usage
<code>path</code>	The URL of the media file.
<code>content</code>	The name of an item in the RTSP Server Agent's <code>contentList</code> .
<code>duration</code>	The length of the media presentation.

Content

The `Content` object is a named set of media streams. Refer to `Content` for a full description of this command. The important options of this command are listed below.

Option	Usage
<code>name</code>	The name of the content.
<code>streamList</code>	A list of streams that compose the content. Each list item is of type <code>Stream</code> .



Stream

The `Stream` object is a single media stream object. Refer to `Stream` for a full description of this command. The important options of this command are listed below.

Option	Usage
<code>name</code>	The name of the stream.
<code>clockRate</code>	The sampling rate.
<code>dataRate</code>	The data transmission rate.
<code>packetization</code>	The time between packets.

RTSP Client Agent

RTSP Client Agent - create an RTSP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RTSPClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_RTSPClient1 agent.config options...
```

DESCRIPTION

An RTSP client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`commandList`

This is a list of type `RtspCommand` used to hold RTSP commands. The elements in this list describe the commands to be executed by the client. (Default = {}).

`commandTimeout`

The amount of time allowed for each command to complete, in seconds. (Default = 60).

`enable`

If `true`, this agent will be used. (Default = `true`).

`enableEsm`

If `true`, the use of the `esm` option is enabled. (Default = `false`).

`enableTos`

Enables the setting of the TOS (Type of Service) bits in the header of the RTSP packets. Use the `tos` option to specify the TOS bit setting.



- 0 (default) TOS bits not enabled.
- 1 TOS bits enabled.

esm

If `enableEsm` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size as 1,460 bytes. (Default = 1,460).

enableVlanPriority

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, `IxLoad` sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = false).

vlanPriority

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.

name

The name associated with this object, which must be set at object creation time.

rtpTransport

The RTSP mechanism to be requested by the client. One of:

Option	Usage
<code>\$.RTSP_Client(kRtpTransportTcp)</code>	TCP.
<code>\$.RTSP_Client(kRtpTransportUdp)</code>	(Default) UDP.

tos

If `enableTos` is `true`, this option specifies the IP Precedence / TOS (Type of Serbit setting and Assured Forwarding classes. (Default="0"). The choices are:

- 0 (Default) (0x000) routine
- 32 (0x0020) priority service, Assured Forwarding class 1
- 64 (0x0040) immediate service, Assured Forwarding class 2
- 96 (0x0060) flash, Assured Forwarding class 3
- 128 (0x0080) flash-override, Assured Forwarding class 4
- 160 (0x00A0) critical-ecp
- 192 (0x00C0) internet-control

`enableCustomSETUPtransportParam`

This enables or disables the entry of parameters specified in the Transport: line of the RTSP SETUP message. You can use these parameters to set or enable addiRTSP transport options on the server. Default = false.

`customSETUPtransportParam`

If `enableCustomSETUPtransportParam` is false, then the Transport: line contains the following data, which is mandatory for RTSP:

Transport protocol, connection type (unicast or multicast), and client IP port range used for the transport protocol. For example:

```
RTP/AVP;unicast;client_port=35246-35247
```

If `enableCustomSETUPtransportParam` is true, then IxLoad appends a semi-colon (;) to the mandatory data on Transport: line, and then appends the custom data in the field.

For example, if you specify the string `mode=PLAY`, the Transport: line will contain the following string:

```
RTP/AVP;unicast;client_port=35246-35247;mode=PLAY
```

`enableSETUPargs`

If enabled, you can specify the IP address, Media and arguments (which compose the presentation to setup (such as "audio" or "audio, video") for the SETUP command.

Normally, these parameters are specified in the DESCRIBE command. However, some servers do not support the DESCRIBE command.

If no arguments are specified, IxLoad sets up the URL.

`followRtspRedirects`

If enabled, the client follows RTSP redirect responses from the server.

`useSameRtpPort`

If enabled, all RTP streams for one presentation use the same UDP port number.

RTP audio and video streams are usually sent over different UDP ports. However, some Windows RTP servers send both streams over the same port. If this is the case with your server, enable this option.

`enableRtspProxy`

If enabled, you can enter the Rtsp proxy server address.

`rtspProxy`



If `enableRtspProxy` is true, then you can enter the Rtsp proxy ip and port address.

SUB-OBJECTS

`rtspHeaders`

This is an object of type `RtspHeaders`, which holds information about the type of client emulation desired as well as a list of RTSP headers to be supplied by the client for each request. (Default = default object of type `RtspHeaders`).

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]

#####
# Activity RTSPClient1 of NetTraffic Traffic1@Network1
#####

set Activity_RTSPClient1 [$Traffic1_Network1 activityList.appendItem \
    -protocolAndType          "RTSP Client" ]

#####
# Timelinel for activities RTSPClient1
#####

set Timelinel [::IxLoad new ixTimeline]

$Timelinel config \
    -rampUpValue          1 \
    -rampUpType           0 \
    -offlineTime          0 \
    -rampDownTime        20 \
    -standbyTime          0 \
    -iterations           1 \
    -rampUpInterval       1 \
```

```
-sustainTime                20 \  
-timelineType                0 \  
-name                        "Timeline1"  
  
$Activity_RTSPClient1 config \  
-enable                      1 \  
-name                        "RTSPClient1" \  
-enableConstraint            false \  
-userObjectiveValue          100 \  
-constraintValue              100 \  
-userObjectiveType            "simulatedUsers" \  
-timeline                    $Timeline1  
  
set my_RtspHeaders [::IxLoad new RtspHeaders]  
$my_RtspHeaders config \  
-clientEmulation             1  
  
$my_RtspHeaders list.clear  
set User_Agent [::IxLoad new RtspHeader]  
$User_Agent config \  
-name                        "User-Agent" \  
-value                        "QTS (qtver=6.5)"  
  
$my_RtspHeaders list.appendItem -object $User_Agent  
  
$Activity_RTSPClient1 agent.config \  
-enableTos                    0 \  
-loopValue                    true \  
-commandTimeout               60 \  
-enable                       1 \  
-name                          "RTSPClient1" \  
-setEnableCustomSETUPtransportParam true \  
-tos                           0 \  

```



```
-vlanPriority 0 \  
-customSETUPtransportParam "mode=PLAY" \  
-followRtspRedirects 0 \  
-enableRtspProxy 0 \  
-enableSETUPargs true \  
-rtpTransport 3 \  
-enableEsm 0 \  
-rtspProxy "0.0.0.0:554" \  
-useSameRtpPort 0 \  
-esm 1460 \  
-enableVlanPriority 0 \  
-enableCustomSETUPtransportParam true \  
-rtspHeaders $my_RtspHeaders  
  
$Activity_RTSPClient1 agent.urlList.clear  
$Activity_RTSPClient1 agent.setParamOptionList.clear  
$Activity_RTSPClient1 agent.commandList.clear  
set my_RtspCommand [::IxLoad new RtspCommand]  
$my_RtspCommand config \  
-media "/test1.mp3" \  
-destination "Traffic2_RTSPServer1" \  
-command "{PlayMedia}" \  
-arguments "PLAY_TILL_END"  
  
$Activity_RTSPClient1 agent.commandList.appendItem -object $my_RtspCommand  
$Activity_RTSPClient1 agent.getParamOptionList.clear
```

SEE ALSO

`ixNetTraffic`, `RtspCommand`, `RtspHeaders`



RtspCommand

RTSP Command — Specifies an RSTP command to be executed.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RTSPClient1 [$Traffic1_Network1 activityList.appendItem
set my_RtspCommand [::IxLoad new RtspCommand]
$Activity_RTSPClient1 agent.commandList.appendItem -object $my_RtspCommand
```

DESCRIPTION

An RTSP command is added to the `commandList` option of the RTSP Client Agent object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

arguments

This option contains an argument that is used by the various commands defined in `command`. The type of the value depends on the command:

command option	Usage
"{PlayMedia}"	N/A.
"DESCRIBE"	N/A.
"SETUP"	The transport mechanism to be used. One of: "TCP" or "UDP."
"SET_PARAMETER"	Takes only one string value. This string argument appear as <code>name</code> in one of the entries of <code>setParamOptionList</code> .
"GET_PARAMETER"	
"PLAY"	The playback duration, in seconds.
"{KeepAlive}"	Keeps the client and server connection alive
"PAUSE"	N/A.
"{Think}"	The length of time to pause, in seconds.
"TEARDOWN"	N/A.

command

Selects the RTSP command to be used. One of:

Option	Usage
"{PlayMedia}"	(Default) An IxLoad command that plays the file listed in the <code>media</code> option. This command sets up the RTSP control connection, requests the URL from the server, then tears down the RTSP connection.
"DESCRIBE"	Retrieves the description of a presentation or media object identified by the URL in the <code>media</code> option. The server responds with a description of the requested resource.
"{KeepAlive}"	Periodically sends a short message (and empty <code>GET_PARAMETER</code> command) to the server so that the server does not assume that the client is inactive and then tears down the connection. Although you can add a <code>{KeepAlive}</code> to any position in a command list, it should typically be placed after a <code>PLAY</code> command.
"SETUP"	Specifies the transport mechanism to be used for the streamed media. A client can issue a <code>SETUP</code> request for a stream that is already playing to change transport parameters, if the server allows it. Specify the transport mechanism in the <code>arguments</code> option.
"SET_PARAMETER"	This method requests to set the value of a parameter for a presentation or stream specified by the URL. Specify the name of this parameter in the <code>arguments</code> option.
"PLAY"	Tells the server to start playback using the mechanism specified by a previous <code>SETUP</code> command. Specify the stream in the <code>media</code> option, and the playback duration in the <code>arguments</code> option.
"PAUSE"	Causes the stream playback to be temporarily halted. If you specify a stream in the <code>media</code> option, only playback of that stream is halted. If you do not specify a stream, all streams are paused.
"GET_PARAMETER"	Retrieves the current value of a parameter from the server. If you issue the <code>GET_PARAMETER</code> with no arguments, it functions as a keep-alive to prevent the server from closing the connection when long presentations are playing. The IxLoad RTSP client does not process responses to <code>GET_PARAMETER</code> commands.
"{Think}"	An IxLoad command that pauses execution of the command list. Specify the length of time to pause, in seconds, in the <code>arguments</code> option.
"TEARDOWN"	Stops the stream delivery for the URL listed in the <code>media</code> option, freeing the resources associated with it. After issuing the <code>TEAR</code> command, the RTSP session identifier associated with the session is no longer valid.
"{LoopBegin}"	An IxLoad command that you can add to the Command List to cause the commands between it and the <code>{Loop End}</code> to be executed a specified number of times.
"{LoopEnd}"	Ends the list of commands that will be executed by the preceding <code>{Loop Begin}</code> command.



destination

The RTSP server that the client will send the media URL described in `media` to. The media URL identifies the set of stream to be controlled. Specify the destination as follows:

- If the destination is a real RTSP server, specify the server's host name or IP address.
- If the destination is an IxLoad RTSP Server Agent, specify the name of the RTSP Server Agent.
- If the destination is the DUT, specify `DUT:n`—where DUT is the name of the DUT and n is the port number on that DUT.

(Default = "None").

media

The presentation URL sent to the server. The presentation URL identifies the stream to be controlled. Media names may only contain letters, numbers, and the special symbols `'.'`, `'.'`, `'_'`, `'/'` and `'-'`. (Default = "None").

In an RTSP test, you can use sequence generators in the media field of the following RTSP client commands:

DESCRIBE

{Playmedia}

EXAMPLE

```
set my_RtspCommand [::IxLoad new RtspCommand]

$my_RtspCommand config \

    -media                "/test1.mp3" \
    -destination          "Traffic2_RTSPServer1" \
    -command              "{PlayMedia}" \
    -arguments            "PLAY_TILL_END"

$Activity_RTSPClient1 agent.commandList.appendItem -object $my_RtspCommand
```

SEE ALSO

RTSP Client Agent

RtspHeaders

RtspHeaders—Specifies RTSP headers.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RTSPClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_RTSPClient1 agent.config
set my_RtspHeaders [::IxLoad new RtspHeaders]
```

DESCRIPTION

RtspHeaders is an option of the `RTSP Client Agent` object and is used to specify the client emulation and hold a list of individual RTSP headers. See the following example below.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`clientEmulation`

The RTSP client application that the client emulates. One of::

Option	Usage
<code>::\$RtspHeaders</code> (<code>kClientEmulationTypeCustom</code>)	If this option is selected, the conof the <code>list</code> option should be used to specify the client and its options.
<code>::\$RtspHeaders</code> (<code>kClientEmulationTypeQuicktime</code>)	(Default) Apple QuickTime version 6.5.
<code>::\$RtspHeaders</code> (<code>kClientEmulationTypeWindowsMediaPlayer</code>)	Microsoft Windows Media Player.
<code>::\$RtspHeaders</code> (<code>kClientEmulationTypeRealOne</code>)	Real Networks RealMedia Player.

`list`

This is a list of type `RtspHeader`. The elements in this list describe RTSP headers. (Default = `{}`).

EXAMPLE

See the example for `RtspHeader`.



SEE ALSO

RTSP Client Agent, RtspHeader

RtspHeader

RtspHeader—Specifies a single RTSP headers name = value pair.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RTSPClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_RTSPClient1 agent.config
set User_Agent [::IxLoad new RtspHeader]
$my_RtspHeaders list.appendItem -object $User_Agent
```

DESCRIPTION

An RTSP header is added to the `list` option of the `RtspHeaders` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

name

The name of the RTSP header item to configure. (Default = "").

value

The value of the RTSP header item to configure. (Default = "").

EXAMPLE

```
set my_RtspHeaders [::IxLoad new RtspHeaders]
$my_RtspHeaders config \
    -clientEmulation                1
$my_RtspHeaders list.clear
set User_Agent [::IxLoad new RtspHeader]
$User_Agent config \
    -name                          "User-Agent" \
    -value                          "QTS (qtver=6.5)"
$my_RtspHeaders list.appendItem -object $User_Agent
```



```
$Activity_RTSPClient1 agent.config \  
-enableTos 0 \  
-loopValue true \  
-commandTimeout 60 \  
-enable 1 \  
-name "RTSPClient1" \  
-vlanPriority 0 \  
-tos 0 \  
-followRtspRedirects 0 \  
-enableSETUPargs 0 \  
-rtpTransport 3 \  
-enableEsm 0 \  
-rtspProxy "0.0.0.0:554" \  
-useSameRtpPort 0 \  
-esm 1460 \  
-enableVlanPriority 0 \  
-enableRtspProxy 0 \  
-rtspHeaders $my_RtspHeaders
```

SEE ALSO

RtspHeaders

RtspsetParamOptionList

Specifies the properties of the SET_PARAMETER command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RTSPClient1 [$Traffic1_Network1 activityList.appendItem
set Option1 [::IxLoad new SetParamOption]
$Activity_RTSPClient1 agent.setParamOptionList.appendItem -object $Option1
```

DESCRIPTION

The SET_PARAMETER command is added to the `commandList` option of the RTSP Client Agent object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. The string specified in the `arguments` field of the SET_PARAMETER command appears as `name` in one of the entries of `setParamOptionList`.

SUBCOMMANDS

None

OPTIONS

`name`

Each user-specified `content` and `contentType` pair, has a name associated with it. This is by default `Option1`, `Option2`, and so on.

`content`

This is a user-specified string value specifying the content of the parameter. It is dependent on the server that the client is running on.

`contentType`

This is a user-specified string value specifying the content type of the parameter. It is dependent on the server that the client is running on.

EXAMPLE

```
set my_RtspCommand1 [::IxLoad new RtspCommand]

$my_RtspCommand1 config \

    -media                "None" \
    -destination          "None" \
    -command              "SET_PARAMETER" \
```



```
-arguments                                "Option1"

$Activity_RTSPClient1 agent.commandList.appendItem -object $my_RtspCommand1

$Activity_RTSPClient1 agent.setParamOptionList.clear

set Option1 [::IxLoad new SetParamOption]

$Option1 config \

    -content                                "12345" \

    -contentType                             "12" \

    -name                                    "Option1"

$Activity_RTSPClient1 agent.setParamOptionList.appendItem -object $Option1
```

SEE ALSO

RtspCommand, RTSP Client Agent

RtspgetParamOptionList

Specifies the properties of the GET_PARAMETER command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RTSPClient1 [$Traffic1_Network1 activityList.appendItem
set Option1 [::IxLoad new GetParamOption]
$Activity_RTSPClient1 agent.getParamOptionList.appendItem -object $Option1
```

DESCRIPTION

The GET_PARAMETER command is added to the `commandList` option of the RTSP Client Agent object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. The string specified in the `arguments` field of the GET_PARAMETER command appears as `name` in one of the entries of `getParamOptionList`

SUBCOMMANDS

None

OPTIONS

`name`

Each user-specified `content` and `contentType` pair, has a name associated with it. This is by default `Option1`, `Option2`, and so on.

`content`

This is a user-specified string value specifying the content of the parameter. It is dependent on the server that the client is running on.

`contentType`

This is a user-specified string value specifying the content type of the parameter. It is dependent on the server that the client is running on.

EXAMPLE

```
set my_RtspCommand1 [::IxLoad new RtspCommand]

$my_RtspCommand1 config \

    -media                "None" \
    -destination          "None" \
    -command              "GET_PARAMETER" \
```



```
-arguments                                "Option1"

$Activity_RTSPClient1 agent.commandList.appendItem -object $my_RtspCommand1

$Activity_RTSPClient1 agent.getParamOptionList.clear

set Option1 [::IxLoad new GetParamOption]

$Option1 config \

    -content                                "12345" \

    -contentType                             "12" \

    -name                                    "Option1"

$Activity_RTSPClient1 agent.getParamOptionList.appendItem -object $Option1
```

SEE ALSO

RtspCommand, RTSP Client Agent

RTSP Server Agent

RTSP Server Agent

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_RTSPServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_RTSPServer1 agent.config options...
```

DESCRIPTION

An RTSP server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`commandTimeout`

The amount of time, expressed in seconds, allowed for the RTSP client to respond to a message. If no response is received within this time, `IxLoad` closes the RTSP client's connection. (Default = 60).

`contentList`

This is a list of type `Content`. The elements in this list are the media types used in the `presentationList`. (Default = {}).

`enable`

Enables the use of this agent. (Default = true).

`enableEsm`

If true, the use of the `esm` option is enabled. (Default = false).

`enableTos`



Enables the setting of the TOS (Type of Service) bits in the header of the RTSP packets. Use the `tos` option to specify the TOS bit setting.

- 0 (default) TOS bits not enabled.
- 1 TOS bits enabled.

`esm`

If `enableEsm` is `true`, this option specifies the TCP Maximum Segment Size in the MSS (RX) field. Otherwise, the TCP Maximum Segment Size as 1,460 bytes. (`Default = 1,460`).

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If `true`, `ixLoad` sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (`Default = false`).

`vlanPriority`

When `enableVlanPriority` is `true`, this option accepts the `vlan` priority value.

`name`

The name associated with this object, which must be set at object creation time.

`port`

The port number to which the RTSP server will respond. To specify multiple ports, separate the port numbers with commas (,). You can specify up to 50 listenports. (`Default = 554`).

`presentationList`

This is a list of type `PresentationItem`. The elements in this list are the presentaavailable from the RTSP Server Agent. (`Default = {}`).

`serverEmulation`

The RTSP server application that the server emulates. One of:

Option	Usage
<code>\$.RTSP_Server (kServerEmulationCustom)</code>	If this option is selected, the conof the <code>list</code> option should be used to specify the client and its options.
<code>\$.RTSP_Server (kServerEmulationQuicktime)</code>	(Default) Apple QuickTime version 6.5.

Option	Usage
<code>\$.:RTSP_Server</code> (<code>kServerEmulationWindowsMediaPlayer</code>)	Microsoft Windows Media Player.
<code>\$.:RTSP_Server</code> (<code>kServerEmulationRealOne</code>)	Real Networks RealMedia Player.

tos

If `enableTos` is true, this option specifies the IP Precedence / TOS (Type of Service) bit setting and Assured Forwarding classes. (Default = "0"). The choices are:

- 0 (Default) (0x0000) routine
- 32 (0x0020) priority service, Assured Forwarding class 1
- 64 (0x0040) immediate service, Assured Forwarding class 2
- 96 (0x0060) flash, Assured Forwarding class 3
- 128 (0x0080) flash-override, Assured Forwarding class 4
- 160 (0x00A0) critical-ecp
- 192 (0x00C0) Internet-control

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]

#####
# Activity RTSPServer1 of NetTraffic Traffic2@Network2
#####

set Activity_RTSPServer1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType                "RTSP Server" ]

set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]

$Activity_RTSPServer1 config \
    -enable                          true \
    -name                             "RTSPServer1" \
    -timeline                         $_Match_Longest_

$Activity_RTSPServer1 agent.config \
```



```
-enableTos 0 \  
-commandTimeout 60 \  
-enable true \  
-serverEmulation 0 \  
-name "RTSPServer1" \  
-tos 0 \  
-rtspVlanPriority 0 \  
-enableEsm 0 \  
-rtspVlanPriority 0 \  
-esm 1460 \  
-enableRTSPVlanPriority 0 \  
-port 554 \  
-enableRTPVlanPriority 0  
  
$Activity_RTSPServer1 agent.presentationList.clear  
  
set MP3_128kbit [::IxLoad new Content]  
  
$MP3_128kbit config \  
    -name "MP3/128kbit"  
  
$MP3_128kbit streamList.clear  
  
set my_Stream [::IxLoad new Stream]  
  
$my_Stream config \  
    -clockRate "Audio MP3 (90000 Hz)" \  
    -dataRate 128.0 \  
    -packetization 20  
  
$MP3_128kbit streamList.appendItem -object $my_Stream  
  
set my_PresentationItem [::IxLoad new PresentationItem]  
  
$my_PresentationItem config \  
    -duration 30 \  
    -path "/test1.mp3" \  

```

```
-content                                $MP3_128kbit

$Activity_RTSPServer1 agent.presentationList.appendItem -object
$my_PresentationItem

$Activity_RTSPServer1 agent.contentList.clear

set Voice__1016_ [::IxLoad new Content]

$Voice__1016_ config \
    -name                                "Voice (1016)"
$Voice__1016_ streamList.clear

set my_Stream1 [::IxLoad new Stream]

$my_Stream1 config \
    -clockRate                            "Audio 8 bit (8000 Hz)" \
    -dataRate                              0.48 \
    -packetization                          200

$Voice__1016_ streamList.appendItem -object $my_Stream1

$Activity_RTSPServer1 agent.contentList.appendItem -object $Voice__1016_
```

SEE ALSO

`ixNetTraffic`, `Content`, `PresentationItem`



PresentationItem

PresentationItem—Specifies a presentation available from a server agent.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_RTSPServer1 [$Traffic2_Network2 activityList.appendItem
set my_PresentationItem [::IxLoad new PresentationItem]
$Activity_RTSPServer1 agent.presentationList.appendItem -object
$my_PresentationItem
```

DESCRIPTION

A `PresentationItem` is added to the `presentationList` option of the RTSP Server Agent object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`content`

A reference to an item in the `contentList` option of the RTSP Server Agent. This must match the name of a `Content` object in the `contentList`. (Default = "").

`duration`

The maximum length of time that a stream will play, in seconds. (Default = 30).

`path`

The URL of the media file. (Default = "/test1.mp3").

EXAMPLE

```
set my_PresentationItem [::IxLoad new PresentationItem]

$my_PresentationItem config \
    -duration                30 \
    -path                    "/test1.mp3" \
```

-content

\$MP3_128kbit

```
$Activity_RTSPServer1 agent.presentationList.appendItem -object  
$my_PresentationItem
```

SEE ALSO

RTSP Server Agent, Content



Content

Content — Specifies the streams that compose a presentation item.

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_RTSPServer1 [$Traffic2_Network2 activityList.appendItem
set MP3_128kbit [::IxLoad new Content]
$MP3_128kbit streamList.appendItem -object $my_Stream
$Activity_RTSPServer1 agent.presentationList.appendItem -object
$my_PresentationItem
```

DESCRIPTION

A `Content` object is a part of a `PresentationItem` object, which is a member of a `RTSP Server Agent` object. Its options are configured as per the `ixConfig` sub-commands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`name`

The name associated with the streams. (Default = "MP3/128kbit").

`streamList`

This is a list of type `Stream`. The elements in this list are the streams that coma presentation. (Default = {}).

EXAMPLE

```
set MP3_128kbit [::IxLoad new Content]

$MP3_128kbit config \
    -name "MP3/128kbit"

$MP3_128kbit streamList.clear

set my_Stream [::IxLoad new Stream]

$my_Stream config \
    -clockRate "Audio MP3 (90000 Hz)" \
    -dataRate 128.0 \
```

```
-packetization                20
$MP3_128kbit streamList.appendItem -object $my_Stream
set my_PresentationItem [::IxLoad new PresentationItem]
$my_PresentationItem config \
    -duration                  30 \
    -path                      "/test1.mp3" \
    -content                   $MP3_128kbit
$Activity_RTSPServer1 agent.presentationList.appendItem -object
$my_PresentationItem
```

SEE ALSO

PresentationItem, RTSP Server Agent, Stream



Stream

Stream—Specifies a stream used in a presentation item.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_RTSPClient1 [$Traffic1_Network1 activityList.appendItem
set my_Stream [::IxLoad new Stream]
set MP3_128kbit [::IxLoad new Content]
$MP3_128kbit streamList.appendItem -object $my_Stream
$Activity_RTSPServer1 agent.presentationList.appendItem -object
$my_PresentationItem
```

DESCRIPTION

A `Stream` object is a part of a `Content` object that is part of a `PresentationItem` object, which is a member of a `RTSP Server Agent` object. Its options are configas per the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`clockRate`

The rate at which a sound or moving image is sampled in order to represent it digitally, expressed in hertz. Note: An integer value must be used. (Default = 90,000). Some typical values are::

Usage	Rate
Audio MP3	90000 Hz
Audio 8 bit	8000 Hz
Audio 16 bit	16000 Hz
Video	90000 Hz

`dataRate`

The rate at which data is sent, expressed in kbps. (Default = 128).

`packetization`

The amount of time elapsed between packets, in milliseconds. (Default = 20).

EXAMPLE

```
set my_Stream [::IxLoad new Stream]
```

```
$my_Stream config \  
  
-clockRate                "Audio MP3 (90000 Hz)" \  
  
-dataRate                  128.0 \  
  
-packetization             20  
  
$MP3_128kbit streamList.appendItem -object $my_Stream
```

SEE ALSO

Content, RTSP Server Agent

RTSP Statistics

For the RTSP statistics, see the following:

RTSP Client Statistics (on page 1546)

RTSP Server Statistics (on page 1551)

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program  
Files\Ixia\IxLoad\Results\<<version>\<repository>_<testconfiguration>_<yyyy><mm><  
dd>_<hh><mm><ss>
```

If you review your statistics and find many instances of RTSP client statistics and server statistics that should match but do not, that may be an indication that the Ramp Down Time is too short. When the Ramp Down Time expires, IxLoad terminates any users that are still running. If those users still have work in progress (such as transferring data) when IxLoad terminates them, the work will not be completed and the effect will be that statistics that should match (such as Bytes Sent) may not.



RTSP Client Statistics

The table below describes the statistics available for RTSP clients.

The Conditional Views column indicates the views in which a statistic is available:

IP: per-IP view
User: per-User view
VLAN: per-VLAN view
All: all views

Statistic	Conditional Views	Description
RTSP Simulated Users	--	Number of simulated RTSP users.
RTSP Concurrent Sessions	--	Number of concurrent RTSP sessions maintained.
RTSP Connections	All	Number of RTSP connections established.
RTSP Connection Rate	All	Rate at which the client established RTSP connections.
RTSP Transactions	All	Number of RTSP transactions completed.
RTSP Transaction Rate	All	Rate at which the client completed RTSP transactions.
RTP Lost Packets	All	Number of RTP packets lost during transmission.
RTP Out Of Order Packets	All	Number of RTP packets received out of order.
RTP Concurrent Sessions	--	Number of concurrent RTP sessions established.
RTSP Presentations Active	--	Number of RTSP presentations playing or paused.
RTSP Presentations Playing	--	Number of presentations playing.
RTSP Presentations Paused	--	Number of presentations paused.
RTSP Presentation Requests Successful	--	Number of presentations requests that succeeded.
RTSP Presentation Requests Failed	--	Number of presentations requests that failed.
RTSP Presentations Playback Successful	--	Number of RTSP presentation requests that resulted in actual RTP data being received by the client. This statistic is incremented only once for each successful RTSP PLAY command, even if a PLAY results in multiple RTP streams being received (for example, a video and an audio stream).
RTSP DESCRIBE Sent	All	Number of RTSP DESCRIBE messages sent.
RTSP SETUP Sent	All	Number of RTSP SETUP messages sent.
RTSP SET PARAMETER Sent	All	Number of RTSP SET PARAMETER messages sent.

RTSP GET PARAMETER Sent	All	Number of RTSP GET PARAMETER messages sent.
RTSP PLAY Sent	All	Number of RTSP PLAY commands sent.
RTSP PAUSE Sent	All	Number of RTSP PAUSE commands sent.
RTSP TEARDOWN Sent	All	Number of RTSP TEARDOWN commands sent.
RTSP DESCRIBE Successful	All	Number of RTSP DESCRIBE commands for which a successful response was received.
RTSP SETUP Successful	All	Number of RTSP SETUP commands for which a successful response was received.
RTSP SET PARAMETER Successful	All	Number of SET_PARAMETER replies received with code OK (200).
RTSP GET PARAMETER Successful	All	Number of RTSP GET PARAMETER commands for which a successful response was received.
RTSP PLAY Successful	All	Number of RTSP PLAY commands for which a successful response was received.
RTSP PAUSE Successful	All	Number of RTSP PAUSE commands for which a successful response was received.
RTSP TEARDOWN Successful	All	Number of RTSP TEARDOWN commands for which a successful response was received.
RTSP DESCRIBE Failed	All	Number of RTSP DESCRIBE commands that failed.
RTSP SETUP Failed	All	Number of RTSP SETUP commands that failed.
RTSP SET PARAMETER Failed	All	Number of SET_PARAMETER replies received with a code other than OK (200).
RTSP GET PARAMETER Failed	All	Number of RTSP GET PARAMETER commands that failed.
RTSP PLAY Failed	All	Number of RTSP PLAY commands that failed.
RTSP PAUSE Failed	All	Number of RTSP PAUSE commands that failed.
RTSP TEARDOWN Failed	All	Number of RTSP TEARDOWN commands that failed.
RTSP Presentations Requested	All	Number of presentation requests sent.
RTSP Presentations Successful	All	Number of presentations received.
RTSP Presentations Failed	All	Number of presentations requested but not received.
RTSP Presentations Active	All	Number of presentations active.
RTSP Presentations Playing	All	Number of presentations playing.
RTSP Presentations Paused	All	Number of presentations paused.
RTP Packets Received	All	Number of RTP packets received.
RTP Bytes Received	All	Number of RTP bytes received.

RTSP Packets Sent	All	Number of RTSP packets received.
RTSP Packets Received	All	Number of RTSP packets received.
RTSP Bytes Sent	All	Number of RTSP bytes transmitted.
RTSP Bytes Received	All	Number of RTSP bytes received.
RTSP Setup Latency (ms)	All	Amount of time elapsed, in milliseconds, between a client sending a request to establish an RTSP connection and receiving the first byte of the response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RTSP Teardown Latency (ms)	All	Amount of time elapsed, in milliseconds, between a client sending a request to end an RTSP connection and receiving the first byte of the response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RTSP Play Latency (ms)	All	Amount of time elapsed, in milliseconds, between a client sending a PLAY command and receiving the first byte of the media stream. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
RTSP Play Latency (0 ms - 10 ms)	All	Number of instances in which 0 to 10 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (10 ms - 50 ms)	All	Number of instances in which 10 to 50 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (50 ms - 100 ms)	All	Number of instances in which 50 to 100 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (100 ms - 300 ms)	All	Number of instances in which 100 to 300 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (300 ms - 1s)	All	Number of instances in which 300 to 1000 milliseconds elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.
RTSP Play Latency (Greater Than 1s)	All	Number of instances in which more than one second elapsed between the time a client sent a PLAY command and the time it received the first byte of the media stream.

RTP Jitter (0 ms - 50 ms)	All	Number of RTP packets whose arrival times deviated from normal by 0 to 50 milliseconds.
RTP Jitter (50 ms - 100 ms)	All	Number of RTP packets whose arrival times deviated from normal by 50 to 100 milliseconds.
RTP Jitter (100 ms - 150 ms)	All	Number of RTP packets whose arrival times deviated from normal by 100 to 150 milliseconds.
RTP Jitter (150 ms - 200 ms)	All	Number of RTP packets whose arrival times deviated from normal by 150 to 200 milliseconds.
RTP Jitter (200 ms - 250 ms)	All	Number of RTP packets whose arrival times deviated from normal by 200 to 250 milliseconds.
RTP Jitter (250 ms - 300 ms)	All	Number of RTP packets whose arrival times deviated from normal by 250 to 300 milliseconds.
RTP Jitter (300 ms - 400 ms)	All	Number of RTP packets whose arrival times deviated from normal by 300 to 400 milliseconds.
RTP Jitter (400 ms - 500 ms)	All	Number of RTP packets whose arrival times deviated from normal by 400 to 500 milliseconds.
RTP Jitter (500 ms - 700 ms)	All	Number of RTP packets whose arrival times deviated from normal by 500 to 700 milliseconds.
RTP Jitter (700 ms - 1000 ms)	All	Number of RTP packets whose arrival times deviated from normal by 700 to 1000 milliseconds.
RTP Jitter (1 s - 3 s)	All	Number of RTP packets whose arrival times deviated from normal by 1 to 3 seconds.
RTP Jitter (Greater Than 3s)	All	Number of RTP packets whose arrival times deviated from normal by more than 3 seconds.
RTP Bandwidth Usage (0 - 30 KB/s)	All	Amount of time during which RTP bandwidth usage was between 0 and 30 kilobits per second.
RTP Bandwidth Usage (30 KB/s - 100 KB/s)	All	Amount of time during which RTP bandwidth usage was between 30 and 100 kilobits per second.
RTP Bandwidth Usage (100 KB/s - 300 KB/s)	All	Amount of time during which RTP bandwidth usage was between 100 and 300 kilobits per second.
RTP Bandwidth Usage (300 KB/s - 1 MB/s)	All	Amount of time during which RTP bandwidth usage was between 300 kilobits and 1 megabit per second.
RTP Bandwidth Usage (Greater Than 1 MB/s)	All	Amount of time during which RTP bandwidth usage exceeded 1 megabit per second.
RTP Packet Loss Distribution (0 Percent)	All	Amount of time during which 0 percent of packets were lost.
RTP Packet Loss Distribution (0 - 0.1 Percent)	All	Amount of time during which 0 to 0.1 percent of packets were lost.
RTP Packet Loss Distribution (0.1 - 0.5 Percent)	All	Amount of time during which 0.1 to 0.5 percent of packets were lost.
RTP Packet Loss Distribution (0.5 - 2 Percent)	All	Amount of time during which 0.5 to 2 percent of packets were lost.

RTP Packet Loss Distribution (2 - 5 Percent)	All	Amount of time during which 2 to 5 percent of packets were lost.
RTP Packet Loss Distribution (5 - 100 Percent)	All	Amount of time during which 5 to 100 percent of packets were lost.
OK Responses Received	All	Number of RTSP OK messages received. This statistic is only available in Conditional View.
Error Responses Received	All	Number of RTSP error messages received. This statistic is only available in Conditional View.



Note: If the average table and bar graphs do not contain any data for the clients, that is an indication that they did not reach the Sustained (SU) run state. This can be caused by the following:

1. Stopping a test during the Ramp-Up phase.
2. Configuring a large number of page requests for the client agent so that not all the users configured for the client can attain the SU state within the allotted time.
3. Configuring a value for the statistics interval (Statistics tab) which is much larger than the SU time.

Matching the TEARDOWN Statistics to Other Statistics

When you review the statistics from an RTSP test, you may find that the number of TEARDOWN commands does not match the numbers of other commands. The cause may be that the test entered the ramp down phase sooner than expected. For example:

Describe command: If an IxLoad RTSP client receives a response to a DESCRIBE command and then the test enters the ramp down phase, the test does not send a TEARDOWN command, because no session has been set up. In this case, the number of DESCRIBE and TEARDOWN commands will not match.

PLAY command: For the PLAY command, sending of TEARDOWN commands depend on whether the requested media stream plays to its end or not:

- If an IxLoad RTSP client receives a response to a PLAY command (the response being the requested media stream), the media stream plays to its end and then the test enters the Ramp Down phase. The test sends its own implicit TEARDOWN command immediately afterwards to allow the test to complete gracefully. In this case, the number PLAY commands should match the number of TEARDOWN commands.
- If the test enters the ramp down phase while the media stream is still playing, the test will not send a TEARDOWN command. In this case, the number of PLAY and TEARDOWN commands will not match, and the session will not be torn down gracefully. IxLoad will display a warning message.

All other commands: If an IxLoad RTSP client receives a response to a command other than DESCRIBE or PLAY and then the test enters the Ramp Down phase, the test sends its own implicit TEARDOWN command to allow the test to complete gracefully. In this case, the number of each command sent should match the number of TEARDOWN commands.

To cause the statistics for TEARDOWN to match those of other commands, you can either increase the test duration or select shorter media streams.

RTSP Server Statistics

The table below describes the statistics available for RTSP servers.

The Conditional Views column indicates the views in which a statistic is available:

IP: per-IP view
 User: per-User view
 VLAN: per-VLAN view
 All: all views

Statistic	Conditional Views	Description
RTSP Presentations Received	IP, VLAN	Number of presentation requests received by the servers.
RTSP Presentations Successful	IP, VLAN	Number of presentation requests that succeeded.
RTSP Presentations Failed	IP, VLAN	Number of presentation requests that failed.
RTSP Commands Received	IP, VLAN	Number of RTSP commands received.
RTSP DESCRIBE Received	IP, VLAN	Number of RTSP DESCRIBE commands received.



RTSP SETUP Received	IP, VLAN	Number of RTSP SETUP commands received.
RTSP PLAY Received	IP, VLAN	Number of RTSP PLAY commands received.
RTSP PAUSE Received	IP, VLAN	Number of RTSP PAUSE commands received.
RTSP TEARDOWN Received	IP, VLAN	Number of RTSP TEARDOWN commands received.
RTSP Response Codes Sent (2xx)	IP, VLAN	Number of 200-range (Success) responses sent. A 200-range response indicates that the action was successfully received, understood, and accepted.
RTSP Response Codes Sent (3xx)	IP, VLAN	Number of 300-range (Redirection) responses sent. A 300-range response indicates that further action must be taken in order to complete the request.
RTSP Response Codes Sent (4xx)	IP, VLAN	Number of 400-range (Client Error) responses sent. A 400-range response indicates that the request contains bad syntax or cannot be fulfilled.
RTSP Response Codes Sent (5xx)	IP, VLAN	Number of 500-range (Server Error) responses sent. A 500-range response indicates that the server failed to fulfill an apparently valid request.
RTSP Response Codes Sent (6xx-1xxx)	IP, VLAN	Number of 600- to 1000-range responses sent.
RTSP Packets Sent	IP, VLAN	Number of RTSP packets transmitted by the servers.
RTSP Packets Received	IP, VLAN	Number of RTSP packets received by the servers.
RTSP Bytes Sent	IP, VLAN	Number of RTSP-related bytes (commands and responses) transmitted by the servers.
RTSP Bytes Received	IP, VLAN	Number of RTSP-related bytes (commands and responses) received by the servers.
Total RTP Bytes Sent	IP, VLAN	Number of RTP bytes transmitted by the servers.
Total RTP Packets Sent	IP, VLAN	Number of RTP packets transmitted by the servers.
Total UDP Packets Sent	IP, VLAN	Number of UDP packets transmitted by the servers.

RTSP Play Latency (ms)	IP, VLAN	Average amount of time elapsed, in milliseconds, between the time a server received a PLAY request and the time it transmitted the first byte of the media stream. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
------------------------	----------	--

26

SMTP

This section describes the SMTP Tcl API objects.



In this chapter:

Overview	1554
SMTP Client Agent	1557
SMTP Server Agent	1570
SMTP Statistics	1573

Overview

SMTP protocol commands are organized as:

SMTP Client Agent

- SmtCommand
- MailMessage
 - Header
 - Attachment

SMTP Server Agent

Objectives

The objectives (userObjective) you can set for SMTP are listed below. Test objectives are set in the ixTimeline object.

- connectionRate
- transactionRate
- simulatedUsers
- concurrentConnections
- throughputMBps
- throughputKBps

SMTP Client Agent

The SMTP Client Agent defines a simulated user performing SMTP requests against one or more SMTP servers. Refer to SMTP Client Agent for a full description of this command. The important options of this command are listed below.

Option	Usage
enable	Enables the use of the SMTP client agent.
name	The name associated with the client agent.
helloType	The type of hello (HELO or EHLO) used.
commandList	A list of commands to be sent to the server. Each list member is of type SmtplibCommand.
mailMessageList	A list of mail messages used in various commands contained in the <i>commandList</i> . Each list member is of type MailMessage.
commandTimeout	Client timeout value.

SmtplibCommand

Each client command is a single step in the interaction. Refer to SmtplibCommand for a full description of this command. The important subcommands and options of this command are listed below.

Subcommand	Usage
checkConfig	Checks the configuration of the action.

Option	Usage
command arguments	The SMTP command, with optional arguments, to be executed.
destination	The name/address of the SMTP server.
mailMessage	A reference to a mail message in the SMTP Client Agent's <i>mailMessageList</i> .

MailMessage

The MailMessage object embodies a set of mail messages, complete with headers and attachments. Refer to MailMessage for a full description of this command. The important options of this command are listed below.

Option	Usage
name	The name associated with the mail message.



Option	Usage
bodyFormat	The type of contents for the body of the message: text or HTML.
bodySizeType bodySizeFixed bodySizeRandomMin bodySizeRandomMax	Controls the size of the body of the message.
headerList	A set of headers to accompany the mail message. Each member is of type Header.
attachmentList	A set of attachments to accompany the mail message. Each member is of type Attachment.

Header

The Header object embodies a single mail header for use with a mail message. Refer to Header for a full description of this command. The important options of this command are:

Option	Usage
name	An e-mail header item. For example, From or To.
value / data	The text for the e-mail header item. For example, "john@smith.org".

Attachment

The *Attachment* object embodies a set of mail attachments, which may be included with a mail message. Refer to Attachment for a full description of this command. The important options of this command are listed below.

Option	Usage
dataType	The type of contents for the body of the attachment: text or HTML.
type fileName sizeMin sizeMax	Controls whether the attachment is taken from a file or generated within a size range.
countMin countMax	Controls how many attachments of this type are attached to a mail message.

SMTP Server Agent

The SMTP Server Agent defines the operation of the SMTP server. The emulated SMTP Server Agent accepts all mail messages sent to it, so it has few options. Refer to SMTP Server Agent for a full description of this command. The impropertions of this command are listed below.

Option	Usage
enable	Enables the use of this server agent.
name	The name associated with the server agent.
concurrentSessionLimit	The maximum number of concurrent sessions that the server will allow.
Server_Listening_Port	Port that the SMTP server listens on for new connections.

SMTP Client Agent

SMTP Client Agent - create an SMTP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SMTPClient1 [$Traffic1_Network1 activityList.appendItem
$Activity_SMTPClient1 agent.config options...
```

DESCRIPTION

An SMTP client agent is added to the activityList object. The *activityList* object is added to the *ixNetTraffic* object using the *appendItem* subcommand from the *ixConfigSequenceContainer* command.

Each member of the list, however may be separately addressed and modified using the *ixConfig* subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard *config*, *cget*, and *getOptions* subcommands defined in the *ixConfig* command.

OPTIONS

commandList

This is a list of type *ixConfigSequenceContainer* used to hold objects of type *SmtplibCommand*. The elements in this list describe the commands to be executed by the agent. (*Default = {}*).

commandTimeout



Amount of time allowed for an SMTP command to complete. If the command does not complete within the allowed time, IxLoad closes the SMTP client's con to the SMTP server. (*Default = 120*).

`enable`

Enables the use of this agent. (*Default = true*).

`helloType`

Type of HELLO command used by this SMTP client. One of::

Option	Usage
<code>\$.::SMTP_Client (kHelloTypeEhlo)</code>	(<i>Default</i>) EHLO. The Enhanced SMTP (ESMTP) version of HELO. The server's response includes a list of the options that the server supports.
<code>\$.::SMTP_Client (kHelloTypeHelo)</code>	HELO. The sender-SMTP sends a HELO to the receiver-SMTP to identify itself and open a con. An argument sent with the command con the host name of the sender-SMTP.

`ipPreference`

This option indicates the order by which the POP3 client will use the subnets, if there is a mixture of IPv4 and IPv6 subnets in the network. The values are: `IpPreferenceV4`, `IpPreferenceV6`, `IpPreferenceV4Any`, `IpPreferenceV6Any`.

`loopValue`

If this option is enabled (*1*), then the client progresses through the command list repeatedly until the test's sustain time. If the option is disabled (*0*), then the client will progress through the command list only once, and then go idle. (*Default = 0*).

`mailMessageList`

This is a list of type `ixConfigSequenceContainer` used to hold objects of type `MailMessage`. The elements in this list are used as the contents of messages transmitted by the client. (*Default = {}*).

`name`

The name associated with this object, which must be set at object creation time.

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If *true*, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (*Default = false*).

`vlanPriority`

When `enableVlanPriority` is *true*, this option accepts the vlan priority value.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]

#####

# Activity SMTPClient1 of NetTraffic Traffic1@Network1

#####

set Activity_SMTPClient1 [$Traffic1_Network1 activityList.appendItem \
    -protocolAndType          "SMTP Client" ]

#####

# Timeline1 for activities SMTPClient1

#####

set Timeline1 [::IxLoad new ixTimeline]

$Timeline1 config \
    -rampUpValue              1 \
    -rampUpType                0 \
    -offlineTime              0 \
    -rampDownTime             20 \
    -standbyTime              0 \
    -iterations               1 \
    -rampUpInterval           1 \
    -sustainTime              20 \
    -timelineType             0 \
    -name                      "Timeline1"

$Activity_SMTPClient1 config \
    -enable                   1 \
    -name                      "SMTPClient1" \
```



```
-enableConstraint                false \  
-userObjectiveValue             100 \  
-constraintValue                100 \  
-userObjectiveType              "simulatedUsers" \  
-timeline                       $Timeline1  
$Activity_SMTPLClient1 agent.config \  
-loopValue                      true \  
-commandTimeout                 120 \  
-enable                         1 \  
-ipPreference                   2 \  
-name                           "SMTPLClient1" \  
-vlanPriority                    0 \  
-helloType                      0 \  
-enableVlanPriority              true  
$Activity_SMTPLClient1 agent.mailMessageList.clear  
set Simple [::IxLoad new MailMessage]  
$Simple config \  
-bodySizeType                   0 \  
-name                           "Simple" \  
-fileNameAsBody                 "" \  
-description                     "100 bytes plain text body" \  
-textContentAsBody              "" \  
-bodySizeRandomMax              4096 \  
-bodySizeFixed                  100 \  
-mimeTypeAndEncode               0 \  
-bodySizeRandomMin              1 \  
-bodyDataType                   0 \  
-useFileAsBody                  true \  

```

```
-bodyFormat                                0

$Simple headerList.clear

set From [::IxLoad new MailHeader]

$From config \

    -name                                    "From" \

    -value                                    "fromName@company.com"

$Simple headerList.appendItem -object $From

set To [::IxLoad new MailHeader]

$To config \

    -name                                    "To" \

    -value                                    "toName@company.com"

$Simple headerList.appendItem -object $To

set Subject [::IxLoad new MailHeader]

$Subject config \

    -name                                    "Subject" \

    -value                                    "sample subject"

$Simple headerList.appendItem -object $Subject

$Simple attachmentList.clear

$Activity_SMTPClient1 agent.mailMessageList.appendItem -object $Simple
```

SEE ALSO

ixNetTraffic, SmtplibCommand, MailMessage, Attachment, Header

SmtplibCommand

SmtplibCommand—Specifies the contents of an SMTP command.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]

set Activity_SMTPClient1 [$Traffic1_Network1 activityList.appendItem

set my_SmtplibCommand [::IxLoad new SmtplibCommand]

$Activity_SMTPClient1 agent.commandList.appendItem -object $my_SmtplibCommand
```

DESCRIPTION

An *SmtplibCommand* object is added to the *commandList* option of the SMTP Client Agent object using the *appendItem* subcommand from the *ixConfigSequenceContainer* command.

Each member of the list, however may be separately addressed and modified using the *ixConfig* subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard *config*, *cget*, and *getOptions* subcommands defined in the *ixConfig* command.

OPTIONS

arguments

Optional arguments related to the SMTP command to be executed. One of:

Option	Usage
"{Send} "	N/A.
"OPEN"	N/A.
"MAIL"	The number of copies of the selected mail message to transfer. (<i>Default = 10</i>).
"NOOP "	N/A.
"RSET"	N/A.
"{Think}"	The length of the pause, in seconds, in the <i>arguments</i> option. (<i>Default = 1</i>).
"QUIT"	N/A.

command

The SMTP command to be executed. One of:

Option	Usage
"{Send} "	(<i>Default</i>) An <i>IxLoad</i> command that opens a connection to the SMTP server, transfers all configured messages to it, then logs out. {Send} is a single command that performs the same function as multiple SMTP commands. However, {Send} is not a standard SMTP comIt is included in <i>IxLoad</i> for your convenience to make <i>configSMTP</i> clients easier.
"OPEN"	Opens a connection to the SMTP server.
"MAIL"	Initiates a transaction that transfers mail messages to an SMTP server. In the <i>arguments</i> option, specify the number of copies of the selected mail message to transfer.
"NOOP "	(NO OPERATION) specifies no action other than that the receiver send an OK reply.
"RSET"	Aborts the current mail transaction. Any stored sender, recipients, and mail data are discarded. A client can issue a RSET command at any time.

Option	Usage
"{Think}"	Pauses the mail transaction. Specify the length of the pause, in sec in the <i>arguments</i> option.
"QUIT"	Closes the transmission channel.
"{LoopBegin}"	An IxLoad command that you can add to the Command List to cause the commands between it and the {Loop End} to be executed a specified number of times.
"{LoopEnd}"	Ends the list of commands that will be executed by the preceding {Loop Begin} command.

`destination`

The SMTP server that the client will send the command to. Specify the destination as follows:

- If the destination is a real SMTP server, specify the server's host name or IP address.
- If the destination is an IxLoad SMTP Server Agent, specify the name of the SMTP Server Agent.
- If the destination is the DUT, specify DUT:n – where DUT is the name of the DUT and n is the port number on that DUT.

(Default = "198.18.1.1").

`mailMessage`

A reference to an instance of the *MailMessage* object. (Default = "").

EXAMPLE

```
set my_SmtpCommand [::IxLoad new SmtpCommand]

$my_SmtpCommand config \
    -destination           "Traffic2_SMTPServer1" \
    -command               "{Send}" \
    -arguments             "10" \
    -mailMessage           $Simple1

$Activity_SMTPClient1 agent.commandList.appendItem -object $my_SmtpCommand
```

SEE ALSO

SMTP Client Agent, MailMessage

MailMessage

MailMessage—Specifies the contents of a mail message.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SMTPClient1 [$Traffic1_Network1 activityList.appendItem
set Simple [::IxLoad new MailMessage]
$Activity_SMTPClient1 agent.mailMessageList.appendItem -object $Simple
```

DESCRIPTION

A MailMessage object is added to the mailMessageList option of the SMTP Client Agent object using the appendItem subcommand from the ixConfigSequenceContainer command.

Each member of the list, however may be separately addressed and modified using the ixConfig subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard config, cget, and getOptions subcommands defined in the ixConfig command.

OPTIONS

attachmentList

This is a list of type Attachment. The elements in this list are the attachments associated with the mail message. (Default = {}).

bodyFormat

The format of the body of the mail message. One of:

Option	Usage
<code>::MailMessage(kBodyFormatPlainText)</code>	(Default) The message body contains only ASCII characters and no formatting or disinformation.
<code>::MailMessage(kBodyFormatHtml)</code>	The message body contains HTML tags for formatting and display. An HTML message is identified by the MIME type <i>text/html</i> .
<code>::MailMessage(kBodyFormatRandom)</code>	Message bodies are a random mixture of plain and HTML format.

bodySizeFixed

If bodySizeType is set to `::MailMessage(kBodySizeTypeFixed)`, then this is the fixed size of the message body. (Default = 100).

bodySizeRandomMax

If bodySizeType is set to `::MailMessage(kBodySizeTypeRandom)`, then this is the maximum size of the message body. (Default = 4,096).

bodySizeRandomMin

If bodySizeType is set to `::$MailMessage(kBodySizeTypeRandom)`, then this is the minimum size of the message body. (Default = 1).

bodySizeType

The manner in which the body size is specified. One of::

Option	Usage
<code>::\$MailMessage(kBodySizeTypeFixed)</code>	(Default) The size of the message body is fixed at a single size. Enter the size in the <i>bodySize</i> option.
<code>::\$MailMessagev(kBodySizeTypeRandom)</code>	The size of the message body varies ranbetween a minimum and a maxisize. Enter the minimum and maximum sizes in the <i>boand booptions</i> .

custom_mail_body_use_real_file

This option accepts boolean value of 0 or 1. If zero is given, there is no need to specify a file name. You have to enter the mail message text in `custom_mail_body_content`. If 1 is given, a file name is specified in the `custom_mail_body_filename`.

custom_mail_body_encode

This option specifies the encoding option for the real file. For boolean value 0, `lxlloads` encodes the file using the default encoding. For already encoded files, you choose boolean value 1.

custom_mail_body_filename

This option specifies the absolute path for the real file. For example: `"c:\temp.txt"` \

custom_mail_body_content

This option accepts the mail message text. For example: `"abcd123."`

description

A short textual description for the mail message. (Default = `"100 bytes plain text body"`).

headerList

This is a list of type `Header`. The elements in this list are the headers associated with the mail message. (Default = an object with three items in the list: `"From:fromName@company.com," "To:toName@company.com," "Subject:sample subject"`).

name



The name associated with this object. (Default = "Simple").

mail_body_type

The mail body type can be generated or custom data. You cannot import files through Tcl so you can work only with default or custom data. (Default = 1).

EXAMPLE

```
set Simple [::IxLoad new MailMessage]

$Simple config \
    -bodySizeType                0 \
    -name                        "Simple" \
    -fileNameAsBody              "" \
    -description                  "100 bytes plain text body" \
    -textContentAsBody           "" \
    -bodySizeRandomMax           4096 \
    -bodySizeFixed                100 \
    -mimeTypeAndEncode            0 \
    -bodySizeRandomMin            1 \
    -bodyDataType                 0 \
    -useFileAsBody                true \
    -bodyFormat                    0

$Simple headerList.clear

set From [::IxLoad new MailHeader]

$From config \
    -name                        "From" \
    -value                        "fromName@company.com"

$Simple headerList.appendItem -object $From

set To [::IxLoad new MailHeader]

$To config \
    -name                        "To" \
    -value                        "toName@company.com"
```

```
$Simple headerList.appendItem -object $To

set Subject [::IxLoad new MailHeader]

$Subject config \

    -name                "Subject" \

    -value                "sample subject"

$Simple headerList.appendItem -object $Subject

$Simple attachmentList.clear

$Activity_SMTPClient1 agent.mailMessageList.appendItem -object $Simple
```

SEE ALSO

SMTP Client Agent, Attachment, Header

Header

Header—Specifies the contents of a mail message header.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SMTPClient1 [$Traffic1_Network1 activityList.appendItem
$AttachmentSmall attachmentList.appendItem -object $my_MailAttachment
$Activity_SMTPClient1 agent.mailMessageList.appendItem -object $Attach
set From4 [::IxLoad new MailHeader]
$AttachmentSmall headerList.appendItem -object $From4
```

DESCRIPTION

A **Header** object is added to the `headerList` option of a `MailMessage` object, which is list item in of the `mailMessageList` option of the `SMTP Client Agent` object. Three required header items are included by default:

- From
- To
- Subject

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

name

The e-mail header item. Example: From, To, Subject. (Default = From, To). The standard headers that IxLoad supports are:



Option	Usage
From	From
To	To
Subject	Subject
Cc	Carbon copy.
Bcc	Blind carbon copy.

In addition you can include your own custom headers by editing the header fields. You can enter any printable US ASCII characters into the fields, except the space () and the colon (:). The custom headers accepts MIME type headers also. MIME type headers start with "Content-"

value / data

The text which forms the header. (Default = "fromName@company.com").

EXAMPLE

```
set From [::IxLoad new MailHeader]

$From config \

    -name                "From" \
    -value                "fromName@company.com"

$Simple headerList.appendItem -object $From
```

SEE ALSO

MailMessage, SMTP Client Agent.

Attachment

Attachment—Specifies the contents of a mail attachment.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SMTPClient1 [$Traffic1_Network1 activityList.appendItem
set my_MailAttachment [::IxLoad new MailAttachment]
$AttachmentSmall attachmentList.appendItem -object $my_MailAttachment
$Activity_SMTPClient1 agent.mailMessageList.appendItem -object $Attach
```

DESCRIPTION

An Attachment object is added to the attachmentList option of a MailMessage object, which is list item in of the mailMessageList option of the SMTP Client Agent object.

SUBCOMMANDS

The options for this command are configured and read using the standard config, cget, and getOptions subcommands defined in the ixConfig command.

OPTIONS

countMax

The upper limit on the number of attachments attached to each message. IxLoad attaches a varying number of attachments of this type. (Default = 3).

countMin

The lower limit on the number of attachments attached to each message. IxLoad attaches a varying number of attachments of this type. (Default = 1).

dataType

If *type* is set to `::$Attachment(kGeneratedData)`, this field specifies the format of the generated data. You can select from the following formats:

Option	Usage
<code>::\$Attachment(kPlainText)</code> or "Plain Text"	(Default) IxLoad generates ASCII text for the attachment.
<code>::\$Attachment(kHtml)</code> or "HTML"	IxLoad generates text for the attachment that includes HTML tags.
<code>::\$Attachment(kRandom)</code> or "Random"	IxLoad randomly generates plain text for some instances of this attachment, and HTML for other instances.

fileName

If *type* is set to `::$Attachment(kExistingFile)`, this field specifies the file to be attached. You can specify any file on the local IxLoad client PC or accessible over your network. (Default = "<specify file>").

sizeMax

If *type* is set to `::$Attachment(kGeneratedData)`, this specifies the upper limit of the size of the attachment. IxLoad generates attachments that vary randomly between the minimum and maximum sizes. (Default = 4,096).

sizeMin

If *type* is set to `::$Attachment(kGeneratedData)`, this specifies the lower limit of the size of the attachment. IxLoad generates attachments that vary randomly between the minimum and maximum sizes. (Default = 1,024).

type

The type of data contained in the attachment. One of:



Option	Usage
<code>\$::Attachment(kGeneratedData)</code> or "Generated Data"	(Default) IxLoad automatically creates random data in the attachment. Use the <i>dataType</i> option to specify the format of the generated data.
<code>\$::Attachment(kExistingFile)</code> or "Existing File"	IxLoad attaches the file specified in the <i>fileName</i> option to the message.

EXAMPLE

```
set my_MailAttachment [::IxLoad new MailAttachment]

$my_MailAttachment config \

    -sizeMax                100 \
    -countMax               1 \
    -dataType               0 \
    -countMin               1 \
    -fileName               "<specify file>" \
    -attchStr               "" \
    -type                   0 \
    -sizeMin                100

$AttachmentSmall attachmentList.appendItem -object $my_MailAttachment

$Activity_SMTPClient1 agent.mailMessageList.appendItem -object
$AttachmentSmall
```

SEE ALSO

MailMessage, SMTP Client Agent

SMTP Server Agent

SMTP Server Agent - configure an SMTP server

SYNOPSIS

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
set Activity_SMTPServer1 [$Traffic2_Network2 activityList.appendItem
$Activity_SMTPServer1 agent.config options...
```

DESCRIPTION

An SMTP server agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`concurrentSessionLimit`

The maximum number of concurrent sessions to be supported by the agent. (Default = 1,000).

`enable`

Enables the use of this action. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

`Server_Listening_Port`

Port that the SMTP server listens on. To specify multiple ports, separate the port numbers with commas (.). You can specify up to 50 listening ports. (Default = 25).

`enableVlanPriority`

VLAN Priority can be set on a per-activity basis or on a per-network (NetTraffic) basis. This parameter sets the VLAN priority for the activity. An activity's VLAN Priority bit setting takes precedence over a network's Priority bit setting. If true, IxLoad sets the VLAN Priority bit in traffic from this activity. Configure the VLAN priority value in `vlanPriority`. (Default = false).

`vlanPriority`

When `enableVlanPriority` is true, this option accepts the vlan priority value.

STATISTICS

EXAMPLE

```
set Traffic2_Network2 [::IxLoad new ixNetTraffic]
```



```
#####  
# Activity SMTPServer1 of NetTraffic Traffic2@Network2  
#####  
set Activity_SMTPServer1 [$Traffic2_Network2 activityList.appendItem \  
    -protocolAndType                "SMTP Server" ]  
set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]  
$Activity_SMTPServer1 config \  
    -enable                          true \  
    -name                            "SMTPServer1" \  
    -timeline                        $_Match_Longest_  
$Activity_SMTPServer1 agent.config \  
    -Server_Listening_Port           "25" \  
    -enable                          true \  
    -name                            "SMTPServer1" \  
    -vlanPriority                     0 \  
    -concurrentSessionLimit          1000 \  
    -enableVlanPriority               false
```

SEE ALSO

`ixNetTraffic`

SMTP Statistics

Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program Files\Ixia\IxLoad\<<version>\Results\<<repository>_<test configuration>_<yyyy><mm><dd>_<hh><mm><ss>
```

If you review your statistics and find many instances of SMTP Client statistics and server statistics that should match but do not, that may be an indication that the Ramp Down Time is too short. When the Ramp Down Time expires, IxLoad terminates any users that are still running. If those users still have work in progress (such as transferring data) when IxLoad terminates them, the work will not be completed and the effect will be that statistics that should match may not.

For the SMTP statistics, see the following:

SMTP Client Statistics (on page 1573)

SMTP Server Statistics (on page 1575)

SMTP Client Statistics

The table below lists the statistics that IxLoad reports for SMTP clients.

Statistic	Description
SMTP Sessions Requested	Number of requests to establish SMTP sessions sent by the clients.
SMTP Sessions Established	Number of SMTP sessions established by the clients.
SMTP Sessions Failed	Number of attempts to establish SMTP sessions that failed.
SMTP Mails Sent	Number of mail messages sent by the clients using SMTP.
SMTP Messages Failed	Number of messages attempted to be sent using SMTP that failed.
SMTP Message Timeouts	Number of messages that could not be sent due to timeouts.
SMTP HELO Sent	Number of SMTP HELO commands sent.
SMTP HELO Ok	Number of SMTP HELO commands that received a positive response.
SMTP HELO Failed	Number of SMTP HELO commands that did not receive a positive response.
SMTP EHLO Sent	Number of SMTP EHLO commands sent.



SMTP EHLO Ok	Number of SMTP EHLO commands that received a positive response.
SMTP EHLO Failed	Number of SMTP EHLO commands that did not receive a positive response.
SMTP MAIL Sent	Number of SMTP MAIL commands sent.
SMTP MAIL Ok	Number of SMTP MAIL commands that received a positive response.
SMTP MAIL Failed	Number of SMTP MAIL commands that did not receive a positive response.
SMTP RCPT Sent	Number of SMTP RCPT commands sent.
SMTP RCPT Ok	Number of SMTP RCPT commands that received a positive response.
SMTP RCPT Failed	Number of SMTP RCPT commands that did not receive a positive response.
SMTP DATA Sent	Number of SMTP DATA commands sent.
SMTP DATA Ok	Number of SMTP DATA commands that received a positive response.
SMTP DATA Failed	Number of SMTP DATA commands that did not receive a positive response.
SMTP NOOP Sent	Number of SMTP NOOP commands sent.
SMTP NOOP Ok	Number of SMTP NOOP commands that received a positive response.
SMTP NOOP Failed	Number of SMTP NOOP commands that did not receive a positive response.
SMTP RSET Sent	Number of SMTP RSET commands sent.
SMTP RSET Ok	Number of SMTP RSET commands that received a positive response.
SMTP RSET Failed	Number of SMTP RSET commands that did not receive a positive response.
SMTP QUIT Sent	Number of SMTP QUIT commands sent.
SMTP QUIT Ok	Number of SMTP QUIT commands that received a positive response.
SMTP QUIT Failed	Number of SMTP QUIT commands that did not receive a positive response.
SMTP Total Bytes Sent	Total number of SMTP-related (commands, responses, and mail messages) bytes sent by the clients.
SMTP Total Bytes Received	Total number of SMTP-related (commands, responses, and mail messages) bytes received by the clients.
SMTP Total Attachments Sent	Total number of attachments sent by the clients.
SMTP Total Mails with Attachments Sent	Total number of messages sent that included one or more attachments.

SMTP Simulated Users	Number of simulated SMTP users.
SMTP Concurrent Connections	Number of concurrent SMTP connections maintained.
SMTP Connections	Number of SMTP connections established by the clients.
SMTP Transactions	Number of SMTP transactions completed by the clients.
SMTP Bytes	Number of SMTP-related bytes sent and received by the clients.
SMTP Connection Rate	Rate at which the SMTP clients established connections to servers.
SMTP Transaction Rate	Rate at which the SMTP clients completed SMTP transactions.
SMTP Throughput	Rate at which the SMTP clients sent and received SMTP data.



Note: If the average table and bar graphs do not contain any data for the clients, that is an indication that they did not reach the Sustained (SU) run state. This can be caused by the following:

1. Stopping a test during the Ramp-Up phase.
2. Configuring a large number of page requests for the client agent so that not all the users configured for the client can attain the SU state within the allotted time.
3. Configuring a value for the statistics interval (Statistics tab) which is much larger than the SU time.

SMTP Server Statistics

The table below lists the statistics that IxLoad reports for SMTP servers.

For TCP statistics, see Appendix D, TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistic	Description
SMTP Session Requests Received	Number of requests to establish SMTP sessions received by the servers.
SMTP Session Requests Successful	Number of SMTP sessions established by the servers.
SMTP Session Requests Failed	Number of requests to establish SMTP sessions that failed.
SMTP HELO Received	Number of SMTP HELO commands received.
SMTP EHLO Received	Number of SMTP EHLO commands received.



SMTP MAIL Received	Number of SMTP MAIL commands received.
SMTP RCPT Received	Number of SMTP RCPT commands received.
SMTP DATA Received	Number of SMTP DATA commands received.
SMTP NOOP Received	Number of SMTP NOOP commands received.
SMTP RSET Received	Number of SMTP RSET commands received.
SMTP QUIT Received	Number of SMTP QUIT commands received.
SMTP Mail Bytes Received	Number of bytes contained in SMTP mail messages received by the servers.
SMTP Mails Received	Number of mail messages received using SMTP. NOTE for API Users: There is a trailing space after the word 'Received' in the name of this statistic.
SMTP Total Bytes Sent	Number of SMTP-related bytes (commands, responses, and messages) sent.
SMTP Total Bytes Received	Number of SMTP-related bytes (commands, responses, and messages) received.



SSH

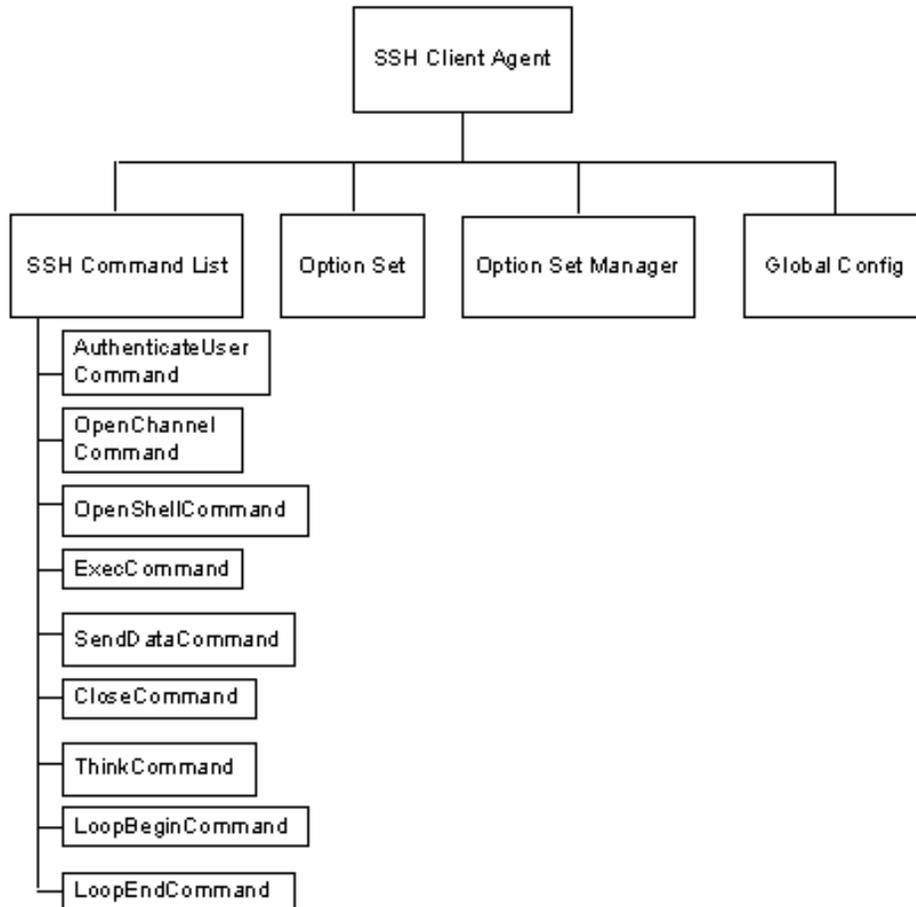
This section describes the SSH Tcl API objects.

In this chapter:

API Overview	1578
SSH Client Agent	1581
SSH Client Statistics	1597

API Overview

The IxLoad SSH API consists of a client agent and its commands. The structure of the API is shown below.



Objectives

The objectives (userObjective) you can set for SSH are listed below. Test objective set in the ixTimeline object.

- connectionRate
- transactionRate
- simulatedUsers
- concurrentConnections

SSH Client Agent

Secure Shell (SSH) is a protocol for securely logging into a remote host over an insecure network. Refer to `SSH Client Agent` on page 24-4 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>protocolAndType</code>	Protocol used by the client agent. Defines the agent as either a client or server.

SSH Command List

The SSH Command List creates the list of SSH commands that the client will send to a SSH server. Refer to `SSH Command List` on page 24-10 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>id</code>	Command that client will send.

Option Set

The Option Set object configures the list of SSH options that the SSH commands will use. Refer to `Option Set` on page 24-15 for a full description of this com

Option Set Manager

The Option Set Manager object configures the list of Option Sets. Refer to `Option Set Manager` on page 24-16 for a full description of this command.

Global Config

Configures the parameters that define the way the IxLoad SSH client performs overall. Refer to `Global Config` on page 24-18 for a full description of this com



SSH Client Agent

SSH Client Agent - create an SSH client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_SSHClient1 [$Traffic1_Network1 activityList.appendItem \
options...
$Activity_SSHClient1 agent.config
```

DESCRIPTION

A SSH client agent is added to the `activityList` option of the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` `comOther` `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. It is customary to set all the options of the client agent during the `appendItem` call.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time .

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#-----
# Activity SSHClient1 of NetTraffic Traffic1@Network1
#-----
```



```
set Activity_SSHClient1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "ssh Client" ]  
  
$Activity_SSHClient1 agent.config \  
    -enable                          true \  
    -name                             "SSHClient1"
```

SEE ALSO

`ixNetTraffic`

SSH Command List

SSH Command List—Creates the list of SSH commands that the client will send to a SSH server.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
$Activity_SSHClient1 agent.pm.commands.appendItem \ options...
```

DESCRIPTION

A command is added to the SSH Command List object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command (see the example below).

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

DHCP command to be executed. One of the following:

Command	Description
AuthenticateUserCom	AuthenticateUserCommand simulates a key-based SSH user authentication session establishment and termination. It sends the following messages or commands: Key-Exchange-Init (SSH_MSG_KEXINIT) New-Keys (SSH_MSG_NEWKEYS) Service-Request (SSH_MSG_SERVICE_REQUEST) User-Authentication-Request (SSH_MSG_USERAUTH_REQUEST) {Close}

Command	Description
OpenChannelCom	<p>OpenChannelCommand performs a key-based SSH user authentication, establishes an SSH session, opens a new channel, and then terminates the session. It is a combination of an AuthenticateUserCommand, plus an OpenChannelComIt sends the following messages or commands:</p> <p>Key-Exchange-Init (SSH_MSG_KEXINIT)</p> <p>New-Keys (SSH_MSG_NEWKEYS)</p> <p>Service-Request (SSH_MSG_SERVICE_REQUEST)</p> <p>User-Authentication-Request (SSH_MSG_USERAUTH_REQUEST)</p> <p>Channel-Open (SSH_MSG_CHANNEL_OPEN)</p>
OpenShellCommand	<p>An OpenShellCommand requests that the server open a new channel and establish a new shell process. It is a combination of an OpenChannelCommand, plus a ChannelRequestComwith requestType set to "shell". It sends the following messages or commands:</p> <p>{OpenChannel}</p> <p>ChannelRequestCommand (SSH_MSG_CHANNEL_REQUEST) with requestType = shell</p>
ExecCommand	<p>An ExecCommand executes a command on the SSH server. It is a combination of an OpenChannelCommand, plus a Chanwith requestType set to "exec". It sends the following messages or commands:</p> <p>{OpenChannel}</p> <p>Channel-Request (SSH_MSG_CHANNEL_REQUEST) with Request Type = exec</p>
SendDataCommand	<p>A SendDataCommand sends data over the channel estabby a previous OpenChannelCommand.</p> <p>The maximum amount of data that can be sent, depends on the channel's maximum packet size or its current window size, whichever is smaller. Sending data decreases the remaining window size by the amount of data sent.</p> <p>{SendData} can send either of the following messages:</p> <p>Channel-Data (SSH_MSG_CHANNEL_DATA)</p> <p>Channel-Extended-Data (SSH_MSG_CHANNEL_EXTENDED_DATA)</p>
CloseCommand	<p>A CloseCommand terminates an SSH session. It sends the folmessages or commands:</p> <p>Channel-Close (if needed) (SSH_MSG_CHANNEL_CLOSE)</p> <p>Disconnect (SSH_MSG_DISCONNECT)</p>

Command	Description
ThinkCommand	The THINKCommand causes the client to become idle for a specified length of time, to simulate real-world usage scenarios in which a user may spend time absorbing or processing information received from the server before sending the next command. If you specify identical values for the minimum and maximum intervals, the client will be inactive for a fixed length of time. If you specify different values for the minimum and maximum intervals, IxLoad will select a value within the range and cause the client to be inactive for that length of time.
LoopBeginCommand	The Loop BeginCommand is an IxLoad command that you can add to the Command List to cause the commands between it and the LoopEndCommand to be executed a specified number of times.
LoopEndCommand	LoopEndCommand ends the list of commands that will be executed by the preceding LoopBeginCommand.

Arguments for id = AuthenticateUserCommand

`optionSet`

Name of option set. A value for this argument must be one of the `name` objects from the `optionSet` object. Minimum length = 1. (Default = "Default Option Set").

`userName`

Name of simulated user to be authenticated.

`password`

The password required for the authentication.

`authMethod`

Method used to authenticate the user. You can select from the following method:

`password`: Password-based authentication

`serverIPAddr`

Address of the SSH server. (Default = "198.18.0.101").

Arguments for id = OpenChannelCommand

`serverIPAddr`



Address of the SSH server. (Default = "198.18.0.101").

optionSet

Name of option set. A value for this argument must be one of the name objects from the optionSet object. Minimum length = 1. (Default = "Default Option Set").

userName

Name of simulated user to be authenticated.

password

The password required for the authentication.

authMethod

Method used to authenticate the user. You can select from the following method:

password: Password-based authentication

initialWindowSize

Initial size of the channel window, in bytes.

maximumpacketSize

Maximum size of the packets sent over the channel, in bytes.

Arguments for id = OpenShellCommand

serverIPAddr

Address of the SSH server. (Default = "198.18.0.101").

optionSet

Name of option set. A value for this argument must be one of the name objects from the optionSet object. Minimum length = 1. (Default = "Default Option Set").

userName

Name of simulated user to be authenticated.

password

The password required for the authentication.

`authMethod`

Method used to authenticate the user. You can select from the following method:

`password`: Password-based authentication

`initialWindowSize`

Initial size of the channel window, in bytes.

`maximumpacketSize`

Maximum size of the packets sent over the channel, in bytes.

`wantReply`

If enabled, the server returns a message indicating the success or failure of the Channel-Request. The IxLoad client does not display the actual text of the response.

Arguments for `id = ExecCommand`

`serverIPAddr`

Address of the SSH server. (Default = "198.18.0.101").

`optionSet`

Name of option set. A value for this argument must be one of the name objects from the `optionSet` object. Minimum length = 1. (Default = "Default Option Set").

`userName`

Name of simulated user to be authenticated.

`password`

The password required for the authentication.

`authMethod`

Method used to authenticate the user. You can select from the following methods:

`password`: Password-based authentication



`initialWindowSize`

Initial size of the channel window, in bytes.

`maximumpacketSize`

Maximum size of the packets sent over the channel, in bytes.

`wantReply`

If enabled, the server returns a message indicating the success or failure of the Channel-Request. The IxLoad client does not display the actual text of the response.

`commandName`

Name of the command to be executed on the server.

Arguments for id = SendDataCommand

`dataType`

Type of data to be sent to the SSH server.

- `normalData`: Sends normal channel data.
- `extendedData`: Sends extended channel data. The only type of extended data available is `stderr` (`SSH_EXTENDED_DATA-STDERR`).

`fileName`

If data is imported from a file, then the path is mentioned here.

Arguments for id = CloseCommand

`reasonCode`

Reason for ending the session. The values are:

Code	Description
1	<code>protocolError</code> Disconnecting because a protocol error occurred on the client or the server.
2	<code>keyExchangeFailed</code> Disconnecting because the key exchange failed on the client or server.

Code	Description
3	macError Disconnecting because the Message Authentication Code (MAC) failed on the client or server.
4	compressionError Disconnecting because a compression error occurred on the client or server.
5	versionNotSupported Disconnecting because the client or server does not support the protocol version indicated in the message.
6	hostKeyNotVerifiable Disconnecting because the host key could not be verified.
7	connectionLost Disconnecting because the connection was lost.
8	disconnectByApplication Disconnection caused by an application.
9	tooManyConnections Disconnected because the internal connection limit has been exceeded.
10	noMoreAuthenticationMethodsAvailable Disconnecting because there are no more authentication to try. This generally means that the client has failed in all the authenmethods available on the server.

description

Description of the reason for ending the session. The text must be in ISO-10646 UTF-8 encoding.

Arguments for id = ThinkCommand

minimumInterval

Minimum length of time that the client is idle. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

maximumInterval

Maximum length of time that the client is idle. Minimum = "1," maximum = "2,147,483,647." (Default = "1").

Arguments for id = LoopBeginCommand

loopCount



Number of times to repeat the enclosed commands. '0' treated as infinity. Mini= "0," maximum = "2,147,483,647." (Default = "5").

Arguments for id = LoopEndCommand

None.

EXAMPLE

```
$Activity_SSHClient1 agent.pm.commands.appendItem \  
-id "AuthenticateUserCommand" \  
-userName "ixia-user" \  
-password "password" \  
-optionSet "Default Option Set" \  
-authMethod "password" \  
-serverIPAddr "198.18.0.101"
```

SEE ALSO

SSH Client Agent

Option Set

Options Set—Configures the algorithm and language preferences that the IxLoad SSH client sends with some commands that require or allow those preferences to be specified.

SYNOPSIS

```
set Activity_SSHClient1 [$Traffic1_Network1 activityList.appendItem \  
$Activity_SSHClient1 agent.pm.optionSet.config \ options...
```

DESCRIPTION

An `Options Set` is a list of options, their arguments, and the commands for which those options are used. Configure the list using the same subcommands as for `ixConfig` (see the example below).

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOption` subcommands defined in the `ixConfig` command.

OPTIONS

`name`

Name of option set list. Minimum length = 1. (Default = "No Name")

`optionsList`

List of options and their arguments. See `Option Set Manager`. (Default = "{}").

`predefined`

If `true`, then the options in this option set are predefined for the SSH server to expose as available options. (Default = "0").

`inUse`

The option set that is configured through the `Option Set Manager`. (Default = 0).

EXAMPLE

```
$Activity_SSHClient1 agent.pm.optionSet.config \  
-predefined                false \  
-name                      "No Name" \  
-inUse                     0
```

SEE ALSO

SSH Client Agent, Option Set Manager

Option Set Manager

Options Set Manager—Configures the list of Option Sets.

SYNOPSIS

```
set Activity_SSHClient1 [$Traffic1_Network1 activityList.appendItem \
$Activity_SSHClient1 agent.pm.optionSetManager.optionSetList.appendItem \ options...
```

DESCRIPTION

To configure an Option Set Manager, use the `appendItem` command on the `pm.optionSetManager` component of the SSH Client Agent.

SUBCOMMANDS

None.

OPTIONS

`id`

Key exchange algorithm to be used. Select an algorithms from the table below. Each algorithm takes arguments, which are also listed in the table.

Option/Arguments	Description
0 kexAlgoName	kexAlgoElements Algorithms that the IxLoad SSH client proposes to protect the exchange of public keys between itself and the SSH server.
1 serverHostKeyAlgoName	serverHostKeyAlgoElements Algorithms that the client offers to accept for generthe server's host key.
2 encC2SAIgoName	encC2SAIgoElements Algorithms that the client proposes to encrypt traffic it sends to the server.
3 encS2CAIgoName	encS2CAIgoElements Algorithms that the client offers to accept for encrypttraffic it receives from the server.
4 macC2SAIgoName	macC2SAIgoElements Algorithms that the client proposes for ensuring the integrity of data it sends to the server.
5 macS2CAIgoName	macS2CAIgoElements Algorithms that the client offers to accept for ensurthe integrity of data it receives from the server.



Option/Arguments	Description
6 compC2SAlgoName	compC2SAlgoElements Algorithms that the client proposes for compressing the data it sends to the server.
7 compS2CAIgoName	compS2CAIgoElements Algorithms that the client offers to accept for comthe data it receives from the server.
8 languageC2SName	C2SLanguageElements Languages that the client proposes for messages it sends to the server.
9 languageS2CName	S2CLanguageElements Languages that the client offers to accept for mesit receives from the server.

EXAMPLE

```
$Activity_SSHClient1 agent.pm.optionSetManager.optionSetList.appendItem \  
-id          "OptionSet" \  
-predefined  true \  
-name       "Default Option Set" \  
-inUse      0  
  
$Activity_SSHClient1  
agent.pm.optionSetManager.optionSetList(0).optionsList.clear  
  
$Activity_SSHClient1  
agent.pm.optionSetManager.optionSetList(0).optionsList.appendItem \  
-id          "KexAlgos"  
  
$Activity_SSHClient1  
agent.pm.optionSetManager.optionSetList(0).optionsList(0).kexAlgoElements.cle  
ar  
  
$Activity_SSHClient1  
agent.pm.optionSetManager.optionSetList(0).optionsList(0).kexAlgoElements.app  
endItem \  
-id          "KexAlgoElement" \  
-kexAlgoName "diffie-hellman-group1-sha1"  
  
$Activity_SSHClient1 agent.pm.optionSetManager.optionSetList.appendItem \  
-id          "OptionSet" \  

```

```
-predefined 0 \  
-name      "Option Set - 0" \  
-inUse     0
```

SEE ALSO

Option Set



Global Config

Options Set Manager—Configures the parameters that define the way the IxLoad SSH client performs overall.

SYNOPSIS

```
set Activity_SSHClient1 [$Traffic1_Network1 activityList.appendItem \  
$Activity_SSHClient1 agent.pm.globalConfig.config \ options...
```

DESCRIPTION

To configure the parameters that define the way the IxLoad SSH client performs overall. Use the `appendItem` command on the `pm.optionSetManager` component of the SSH Client Agent.

SUBCOMMANDS

None.

OPTIONS

`defaultSshPort`

The default Listening Port of an SSH Server. `minimum = "1" maximum = "65535" default = "22"`.

`timeout`

Amount of time an SSH Client will wait for getting a response from the Server. `minimum = "1" maximum = "2000" default = "600"`.

`defaultUserName`

The default user name used to login if no other user name is specified in `authentimethod` configuration. `minimum = "1" maximum = "255" default = "ixia-user"`.

`password`

The password to be sent to the server for password authentication. `minimum = "1" maximum = "255" default = "password"`.

EXAMPLE

```
$Activity_SSHClient1 agent.pm.globalConfig.config \  
-defaultSshPort                22 \  
-implicitLoopCheck             true \  
-password                       "password" \  

```

```
-defaultUserName          "ixia-user" \
-timeout                  600
```

SEE ALSO

SSH Client Agent

SSH Client Statistics

The table below describes the SSH client statistics.

Statistic	Description
Objective Statistics	
Simulated Users objective	
User Count	If the objective is Simulated Users, this is the number of users created.
Concurrent Sessions objective	
SSH Concurrent Sessions	If the objective is Concurrent Sessions, this is the number of concurrent SSH sessions established.
Transaction Rate objective	
SSH Total Transactions	If the objective is Transaction Rate, this is the number of SSH transactions completed.
SSH Transaction Rate	If the objective is Transaction Rate, this is the rate at which the client completed SSH transactions.
Connection Rate objective	
SSH Connections Established	If the objective is Connection Rate, this is the number of SSH connections established.
SSH Connection Rate	If the objective is Connection Rate, this is the rate at which the client completed SSH transactions.
Authentication Statistics	
User Authentication statistics	
Total User Authentication Attempted	Total number of user authentication attempts of all types.
Total User Authentication Succeeded	Total number of user authentications of all types that succeeded.
Total User Authentication Failed	Total number of user authentications of all types that failed.
NEWKEYS statistics	
Total NEWKEYS Sent	Number of NEWKEYS messages sent by the client.
Total NEWKEYS Received	Number of NEWKEYS messages received by the client.



KEXINIT statistics	
Total KEXINIT Sent	Number of KEXINIT messages sent by the client.
Total KEXINIT Received	Number of KEXINIT messages received by the client.
Service Request statistics	
Total Service Request Sent	Total number of Service-Request messages sent by the client for all SSH protocols.
Total Service Accept Received	Number of Service-Accept messages received by the client.
Total Service Request Sent - SSH-Userauth	Number of Service-Request messages sent by the client for the SSH user authentication protocol (SSH-USERAUTH).
Total Service Request Sent - SSH-Connection	Number of Service-Request messages sent by the client for the SSH connection protocol (SSH-CONNECTION).
Total Service Accept Received - SSH-Userauth	Number of Service-Accept messages received for the SSH user authentication protocol (SSH-USERAUTH).
Total Service Accept Received - SSH-Connection	Number of Service-Accept messages received for the SSH connection protocol (SSH-CONNECTION).
Total Disconnect Received	Total number of Disconnect messages received for all SSH protocols.
Total Disconnect Received - SSH-Userauth	Number of Disconnect messages received for the SSH user authentication protocol (SSH-USERAUTH).
Total Disconnect Received - SSH-Connection	Number of Disconnect messages received for the SSH connection protocol (SSH-CONNECTION).
Channel Request statistics	
Total Channel Open Sent	Number of Channel Open messages sent by the client.
Total Channel Open Confirmation Received	Number of Channel Open confirmation messages received by the client.
Total Channel Open Failure Received	Number of Channel Open failure messages received by the client.
Total Channel Data Sent	Number of Channel Data messages sent by the client.
Total Channel Extended Data Sent	Number of Channel Data messages received by the client.
Total Channel Request Sent	Number of Channel-Request messages sent.
Total Channel Success Received	Number of Channel-Success messages sent.
Total Channel Failure Received	Number of Channel-Failure messages sent.
Total Channel EOF Sent	Number of Channel-EOF (End of File) messages sent.
Total Channel Close Sent	Number of Channel-Close messages sent.

Total Channel Request Sent - Shell	Number of shell-related Channel-Request messages sent.
Total Channel Request Sent - Exec	Number of exec-related Channel-Request messages sent.
Total Channel Success Received - Shell	Number of shell-related Channel-Success messages received.
Total Channel Success Received - Exec	Number of exec-related Channel-Success messages received.
Total Channel Failure Received - Shell	Number of shell-related Channel-Failure messages received.
Total Channel Failure Received - Exec	Number of exec-related Channel-Failure messages received.
Total Channel EOF Received	Total number of Channel-EOF messages received for all channels.
Total Channel Close Received	Total number of Channel-Close messages received for all channels.
Request / Response statistics	
Total Request Sent	Total number of requests of all kinds sent.
Total Response Received	Total number of responses of all kinds received.
Total Failure	Total number of failures of kinds that occurred.
Throughput statistics	
Total Bytes Sent	Total number of bytes sent in SSH messages.
Total Bytes Received	Total number of bytes received in SSH messages.
Server Response Time statistics	
Service Request Response Time	<p>Average time elapsed, in ms, between the time the client sent a Service-Request message and the time it received a Service-Accept or Disconnect message in response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
Userauth Request Response Time	<p>Average time elapsed, in ms, between the time the client sent a UserAuth-Request message and the time it received a UserAuth-Success or UserAuth-Failure message in response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>
Channel Open Request Response Time	<p>Average time elapsed, in ms, between the time the client sent a Channel-Open-Request message and the time it received a Channel-Open-Confirmation or Channel-Open-Failure message in response.</p> <p>Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.</p>

Channel Request Response Time	Average time elapsed, in ms, between the time the client sent a Channel-Request message and the time it received a Channel-Success or Channel-Failure message in response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
NewKeys Response Time	Average time elapsed, in ms, between the time the client sent a KEXINIT message and the time it received a NEWKEYS message in response. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.

28

Stateless Peer

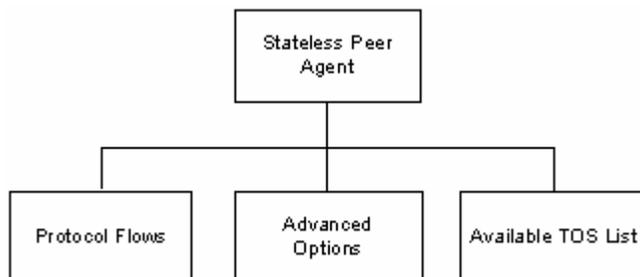
This section describes the Stateless Peer Tcl API objects.

In this chapter:

[Stateless Peer Overview](#) 1601

Stateless Peer Overview

The Stateless Peer API consists of the Stateless Peer Agent and its commands.



Objectives

The objectives (userObjective) you can set for the Stateless Peer are listed below. Test objectives are set in the ixTimeline object.

- simulatedUsers
- peerCount (displays as Initiator Peer Count in the GUI)
- connectionRate
- concurrentConnections
- throughputMBps
- throughputKBps
- transactionRate

Stateless Peer Commands

This section lists the Application Replay Peer's commands.



Stateless Peer Agent

Stateless Peer Agent - create a Stateless Peer agent

SYNOPSIS

DESCRIPTION

A Stateless Peer agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time .

`userIpMapping`

Mapping between IP address usage and simulated users. Default = "1:1"

`enableConstraint`

If true, the `constraintValue` is applied. Default = false

`userObjectiveValue`

User objective value. Default = 100

`constraintValue`

If `enableConstraint` is true, this option specifies the constraint that is applied. Default = 100

`userObjectiveType`

Objective of the activity. Default = "peerCount"

timeline

Name of the timeline used for this activity. Default = \$Timeline1

STATISTICS

EXAMPLE

```
set Activity_StatelessPeer1 [$myNetTraffic activityList.appendItem \
-protocolAndType                "stateless Peer" ]

set Timeline1 [::IxLoad new ixTimeline]
$Timeline1 config \
-rampUpValue                      1 \
-offlineTime                      0 \
-rampDownTime                    20 \
-name                            "Timeline1" \
-rampUpInterval                  1 \
-sustainTime                     20 \
-standbyTime                     0 \
-timelineType                    0 \
-rampUpType                      0

$Activity_StatelessPeer1 config \
-enable                          true \
-name                            "StatelessPeer1" \
-userIpMapping                   "1:1" \
-enableConstraint                false \
-userObjectiveValue              100 \
-constraintValue                 100 \
-userObjectiveType               "peerCount" \
-timeline                       $Timeline1
```

SEE ALSO

ixNetTraffic

Stateless Peer Advanced Options

Advanced Options - configure the Stateless Peer's global options

SYNOPSIS

DESCRIPTION

The Advanced Options object configures the Stateless Peer's global options.



SUBCOMMANDS

None.

OPTIONS

`enableTOS`

If `true`, TOS bits are included in packets from this activity. Default = `false`.

`typeOfService`

If `enableTOS` is `true`, this option configure the TOS bit used. Default = "Best Effort (0x0)"

`parallelCmdCnt`

Number of commands to execute simultaneously. Default = 1.

STATISTICS

EXAMPLE

```
$Activity_StatelessPeer1 agent.pm.advOptions.config \  
-enableTOS                false \  
-typeOfService    "Best Effort (0x0)" \  
-parallelCmdCnt    1
```

SEE ALSO

Stateless Peer Protocol Flows

Protocol Flows - configure the Stateless Peer's commands

SYNOPSIS

```
$Activity_StatelessPeer1 agent.pm.protocolFlows.appendItem \  
-id                "LoopBeginCommand" \  
-LoopCount        5
```

DESCRIPTION

An option is added to the list of protocol flows using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None.

OPTIONS

Options for LoopBeginCommand

`id`

Name of the command to be executed. Default = "LoopBeginCommand".

`LoopCount`

Number of times to execute the loop. Default = 5.

Options for GenerateStream

`id`

Name of the command to be executed . Default = "GenerateStream"

`remotePeer`

Name of the Stateless Peer that is the destination of the traffic. Default = "".

`destination`

Destination of traffic from this initiator. Default = "None".

- If the destination is a DUT, specify its IP address.
- If the destination is another Stateless Peer activity, specify its name.

`minPacketFreq`

Minimum rate at which packets will be sent. Default = 100.

`maxpacketFreq`

Maximum rate at which packets will be sent. Default = 100.

`streamDur`

Length of time, in seconds, to transmit the stream. Default = 20.

`destinationPort`

Port number on destination Stateless Peer to which traffic is sent. This can be a single port or a range (for example: 1024-2048). Default = 0.



sourcePort

Port number on the source Stateless Peer to which traffic is sent. Default = 0.

minContentSize

Minimum size of the IP payload. Default = 1024.

maxContentSize

Maximum size of the IP payload. Default = 1024.

Options for GenerateIPStream

id

Name of the command to be executed . Default = "GenerateIPStream".

remotePeer

Name of the Stateless Peer that is the destination of the traffic. Default = "None".

proto

Protocol ID contained in traffic from the peer. The list of protocol IDs is at <http://www.iana.org/assignments/protocol-numbers/>. Default = 0.

streamDur

Length of time, in seconds, to transmit the stream. Default = 20.

minPacketFreq

Minimum rate at which packets will be sent. Default = 100.

maxpacketFreq

Maximum rate at which packets will be sent. Default = 100.

minContentSize

Minimum size of the IP payload. Default = 1024.

maxContentSize

Maximum size of the IP payload. Default = 1024.

Options for LoopEndCommand

`id`

Name of the command to be executed . Default = "LoopEndCommand".

Options for Think

`id`

Name of the command to be executed . Default = "Think".

`minimumInterval`

Minimum length of time to think for. Default = 1000.

`minimumInterval`

Maximum length of time to think for. Default = 1000.

STATISTICS

EXAMPLE

```
$Activity_StatelessPeer1 agent.pm.protocolFlows.clear

$Activity_StatelessPeer1 agent.pm.protocolFlows.appendItem \
-id                "LoopBeginCommand" \
-LoopCount 5

$Activity_StatelessPeer1 agent.pm.protocolFlows.appendItem \
-id                "GenerateStream" \
-remotePeer       "None" \
-packetFreq       100 \
-streamDur        20 \
-destinationPort  23 \
-sourcePort       22 \
-contentSize      1024

$Activity_StatelessPeer1 agent.pm.protocolFlows(1).payloadHeaderList.clear

$Activity_StatelessPeer1 agent.pm.protocolFlows(1).payloadHeaderList.appendItem \
-id                "payloadHeaderRow" \
-streamIdentifier  false \
-length           1 \
-type             1 \
-value           "255"

$Activity_StatelessPeer1 agent.pm.protocolFlows.appendItem \
-id                "LoopEndCommand"

$Activity_StatelessPeer1 agent.pm.protocolFlows.appendItem \
-id                "Think" \
-minimumInterval  1000 \
-maximumInterval  1000
```

SEE ALSO

Stateless Peer Payload Header List

Protocol Header List - list of headers in the UDP packets.

SYNOPSIS

```
$Activity_StatelessPeer1 agent.pm.protocolFlows(1).payloadHeaderList.appendItem \
-id                "payloadHeaderRow" \
-streamIdentifier  false \
-length           1 \
-type             1 \
-value           "255"
```

DESCRIPTION

`payloadHeaderList` defines the list of headers in the UDP packet. This list is of type `ixConfigSequenceContainer`; items are added to the list via the `appendItem` sub-command. (Default = {}).

```
$Activity_StatelessPeer1 agent.pm.protocolFlows(1).payloadHeaderList.appendItem
```

Before you add items to the list, you should clear it. For example:

```
$Activity_StatelessPeer1 agent.pm.protocolFlows(1).payloadHeaderList.clear
```

SUBCOMMANDS

None.

OPTIONS

`id`

Name of the header. Deafult = "payloadHeaderRow".

`streamIdentifier`

A boolean that indicates whether or not this header is used to identify the stream. At least one header row must have this flag set. Default = false.

`length`

length of the data in the `value` field. Min = 1, Max = 65535, Default = 1.

`type`

Type of the data in the `value` field. Default = 1. The choices are:

Choice	Description
1	1 byte
2	2 bytes
3	3 bytes
4	4 bytes
5	Fixed binary
6	Fixed ascii

`value`

Value of the header field. Default = "".

STATISTICS

EXAMPLE

```
$Activity_StatelessPeer1 agent.pm.protocolFlows(1).payloadHeaderList.clear

$Activity_StatelessPeer1 agent.pm.protocolFlows(1).payloadHeaderList.appendItem \
-id                          "payloadHeaderRow" \
-streamIdentifier            false \
-length                      1 \
-type                        1 \
-value                       "255"
```

SEE ALSO

Stateless Peer Available TOS List

Available TOS List - list of TOS values in the UDP packets.

SYNOPSIS

```
$Activity_StatelessPeer1 agent.pm.availableTosList.appendItem \
-id                          "AvailableTypeOfService" \
-tos_value "Best Effort (0x0)"
```

DESCRIPTION

`availableTosList` defines the list of TOS values in the UDP packet. This list is of type `ixConfigSequenceContainer`; items are added to the list via the `appendItem` sub-command. (Default = {}).

```
$Activity_StatelessPeer1 agent.pm.availableTosList.appendItem
```

Before you add items to the list, you should clear it. For example:

```
$Activity_StatelessPeer1 agent.pm.availableTosList.clear
```

SUBCOMMANDS

None.

OPTIONS

`id`

Name of the TOS value. Default = "AvailableTypeOfService".

`tos_value`

TOS value. The possible TOS values are listed below. Default = "".

"Best Effort (0x0)"

"Class 1 (0x20)"

"Class 2 (0x40)"

"Class 3 (0x60)"

"Class 4 (0x80)"

"Express Forwarding (0xA0)"

"Control (0xC0)"

STATISTICS

EXAMPLE

```
$Activity_StatelessPeer1 agent.pm.availableTosList.clear

$Activity_StatelessPeer1 agent.pm.availableTosList.appendItem \
  -id "AvailableTypeOfService" \
  -tos_value "Best Effort (0x0)"
```

SEE ALSO

29

Telnet

Telnet is a protocol that provides a simple, general-purpose host-to-host communications facility. Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other.

A Telnet session consists of a single TCP connection that carries data and Telnet control information (interspersed with the data) bidirectionally between the two hosts (see the figure below).

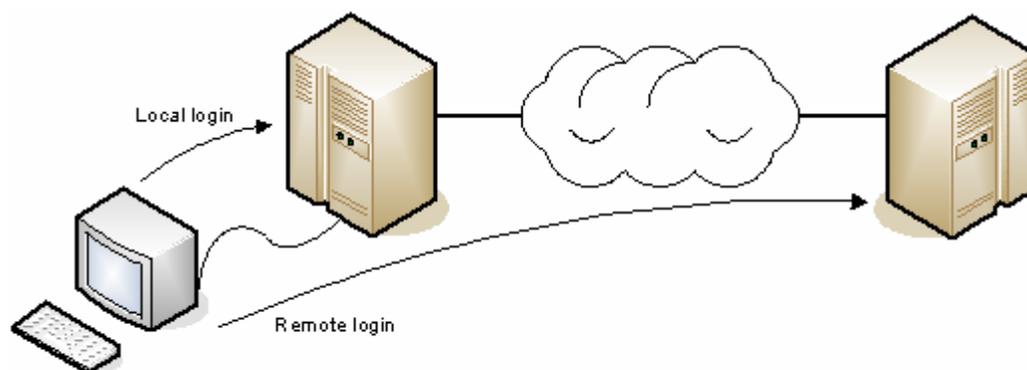


Figure 2: Telnet

When a Telnet connection is being established, each host is initially assumed to be an imaginary device called a "Network Virtual Terminal" (NVT). An NVT represents a generic host that supports a minimal, standardized set of input, display, and reception/transmission capabilities.

After the initial connection between the two NVTs is established, each side can begin negotiating options. Options are additional capabilities that one side requests that either or both hosts begin using, such as a richer character set or a different echo mode. To indicate that it accepts or rejects a particular option, a host sends DO, DON'T, WILL, or WON'T responses.

- If a host wants to begin using option XXX, it sends WILL XXX to the other host. The recipient responds with DO XXX to accept the option, or DON'T XXX to reject it.
- If one host wants the other host to begin using option XXX, it sends DO XXX to the other host. The recipient responds with WILL XXX to accept the option, or WON'T XXX to reject it.
- Similarly, if one host wants the other host to stop using option XXX, it sends DON'T XXX to the other host. The recipient responds with WON'T XXX to stop using the option (accepting the DON'T request), or WILL XXX to continue using it (rejecting the DON'T request).

If a host accepts an option, that option takes effect immediately. If a host receives a request for an option that it does not support, it responds by rejecting the option.

In this chapter:

Telnet RFCs	1614
IxLoad Telnet Client and Server	1614
Creating a Telnet Test	1617
Telnet Statistics	1628

Telnet RFCs

Telnet is defined in multiple RFCs; see the table below for a list of the most important ones.

RFC	Subject
854	Basic Telnet protocol specification
855	Telnet option specifications
856	Telnet binary transmission
857	Echo option
858	Suppress Go Ahead option
859	Status option
860	Timing Mark option
1091	Terminal Type option
1073	Window Size option
1079	Terminal Speed option
1372	Remote Flow Control option
1184	Linemode option
1408	Environment Variables option

IxLoad Telnet Client and Server

IxLoad includes two activities for testing Telnet connections: a Telnet client and a Telnet server.

Telnet Server (on page 1614)

Telnet Client (on page 1616)

Telnet Server

The IxLoad Telnet server is an "echo" server. After establishing a connection with a client, the server echoes back all the lines that it receives from the client. The basic functioning of the Telnet server is as follows:

1. After startup, the server listens for requests for new Telnet connections on the configured listening port.

2. After accepting a request for a connection, the server begins negotiating the Telnet options. It sends its configured options to the client, then it reads the options received from client and updates its list of negotiated options.
3. When it finishes negotiating the options, it sends the "Login:" prompt to the client. It treats any line that it receives in response as a valid username. The response must include a linefeed (LF) to indicate the end of the response.
4. It then sends the "Password:" prompt. It treats any line that it receives in response as a valid password. The response must include a linefeed (LF) to indicate the end of the response.
5. After receiving the response to the "Password:" prompt, the server is ready to accept commands from the client. In this mode, it echoes back to the client any line that it receives from the client. Each line from the client must include a linefeed (LF) to indicate the end of the line.
6. When the server receives an "Exit" command, it closes the connection.

Telnet Command Handling

The Telnet server responds to the following Telnet commands:

- Option Negotiation commands:
 - WILL
 - DO
 - WONT
 - DONT
- Sub-option Negotiation commands:
 - SB
 - SE



All other commands are silently ignored.

Line Mode Sub-option Negotiation

If you enable the Line Mode option (on the Telnet tab), the server behaves as follows:

LINEMODE suboption MODE: The server sends an IAC SB LINEMODE MODE mask IAC SE command with only the EDIT bit set in the mask field. If a client sends an IAC SB LINEMODE MODE mask IAC SE command without the MODE_ACK bit set, the server responds by sending another IAC SB LINEMODE MODE mask IAC SE command with only the EDIT and MODE_ACK bits set.

LINEMODE suboption FORWARDMASK: The server also sends an IAC SB LINEMODE DO FORWARDMASK mask0 mask1 ... mask31 IAC SE command with only the mask bit associated with LF set. If the client sends an IAC SB LINEMODE DO FORWARDMASK mask0 mask1 ... mask31 IAC SE command, the server responds with an IAC SB LINEMODE WONT FORWARDMASK IAC SE command. The server will silently ignore a IAC SB LINEMODE DONT FORWARDMASK IAC SE command from the client.

Set Local Character (SLC) requests: The server does not send SLC requests. If it receives an SLC 0 SLC_DEFAULT 0 or SLC 0 SLC_VALUE 0 request, the server send SLC xxx SLC_DEFAULT 0 for all SLC functions xxx defined in RFC 1184 (<http://www.faqs.org/rfcs/rfc1184.html>). If it receives any other SLC requests, server returns the same command with the SLC_ACK bit set in the second octet of the response.

See RFC 1184 (<http://www.faqs.org/rfcs/rfc1184.html>) for information on Telnet Line Mode options.

Telnet Client

You can use the IxLoad Telnet client to simulate a complete Telnet session. You can configure it to:

1. Open a new Telnet session with the Telnet server.
2. Send a login and password to the Telnet server.
3. Send simple commands to be executed on the server, and wait for a specific string in response to each command.
 - If the client receives a response within the timeout period, it immediately sends the next command in the command list, and again waits for a response.
 - If the timeout period expires and the client has not received a response from the server, the client sends the next command and again waits for a response.
4. Close the connection to the server.

The Telnet client supports the following commands:

- OPEN (see "[Telnet Client LOGIN Command](#)" on page 1623) opens a new Telnet connection to a server.
- LOGIN (see "[Telnet Client LOGIN Command](#)" on page 1623) sends a login (user name) to the server.
- PASSWORD (see "[Telnet Client PASSWORD Command](#)" on page 1624) sends a password to the server.
- SEND (see "[Telnet Client SEND Command](#)" on page 1624) sends a command to be executed on the server, such as "ls" (list).
- EXIT (see "[Telnet Client EXIT Command](#)" on page 1625) closes the Telnet connection.
- {Telnet Session} (see "[Telnet Client {Telnet Session} Command](#)" on page 1622) is a single command that performs an entire Telnet session. It is an alternative to using a standard sequence of OPEN (see "[Telnet Client LOGIN Command](#)" on page 1623), LOGIN (see "[Telnet Client LOGIN Command](#)" on page 1623), PASSWORD (see "[Telnet Client PASSWORD Command](#)" on page 1624), and EXIT (see "[Telnet Client EXIT Command](#)" on page 1625) commands.
- {Think} (see "[Telnet Client {Think} Command](#)" on page 1625) pauses execution of the command list.
- {Loop Begin} (see "[Telnet Client {Loop Begin} Command](#)" on page 1626) and {Loop End} (see "[Telnet Client {Loop End} Command](#)" on page 1626) allow you to repeat a subset of the commands within the command list. {Loop Begin} (see "[Telnet Client {Loop Begin} Command](#)" on page 1626) defines the beginning of the subset of commands, and {Loop End} (see "[Telnet Client {Loop End} Command](#)" on page 1626) defines the end of the subset. The commands listed between {Loop Begin} (see "[Telnet Client {Loop Begin} Command](#)" on page 1626) and {Loop End} (see "[Telnet Client {Loop End} Command](#)" on page 1626) will be repeated for the number of times you specify, then the main command list will resume execution, starting with the command following {Loop End} (see "[Telnet Client {Loop End} Command](#)" on page 1626).

Creating a Telnet Test

To create a Telnet test, follow the procedures in [Creating and Running an IxLoad Test](#) to create a basic IxLoad test configuration, then refer to this section for information on configuring the Telnet activities.



To create a Telnet Test:

1. If IxLoad is not already running, start it. See [Starting IxLoad](#).
2. Create the NetTraffics for the Telnet activities. See [Creating a NetTraffic](#).
 - To create a Telnet client Activity, set the client NetTraffic activity to TELNET CLIENT.
 - To create a Telnet server Activity, set the server NetTraffic activity to TELNET SERVER.
3. Add the DUT to the test configuration. See [Creating a DUT](#).
4. Connect the DUT to the NetTraffics. See [Connecting a NetTraffic to a DUT](#).
5. Configure the client or server Activities:
 - To configure a Telnet server agent, see [Configuring a Telnet Server](#) (on page 1618).
 - To configure a Telnet client agent, see [Configuring a Telnet Client](#) (on page 1620).



Note: If the destination of the client traffic will be a server Activity, you should configure the server Activity first.

6. Set the timeline and objectives for each NetTraffic (or each Activity). See [Timeline and Objective](#).
7. Select the statistics that you want to record. See [Selecting Statistics](#).



8. Select the ports that you will use for the test. See Port Assignments.
9. Run the test. See Running a Test.
View the graphs and test results. See Viewing the Test Results.

Configuring a Telnet Server



To configure a Telnet server, configure the controls on the following tabs:

Telnet tab (on page 1618)

Advanced Options tab (on page 1620)

Telnet Tab

The server Telnet tab contains controls that define the Telnet server's basic behavior.

The following table describes the fields.

Parameter	Description
Settings	
Default Command Prompt	Character sent by the server to the client to indicate that it is ready to accept the next command. The default command prompt is the pound sign (#).
Close Command	Command used to end the Telnet session.
Listening Port(s)	Port that the Telnet server listens on for new connections. The default is the well-known port for Telnet, 23. You can configure the Telnet server to listen on multiple ports. To specify multiple listening ports, enter the port numbers, separated by commas (.). You can specify up to 50 listening ports. When you configure the Telnet server to listen on multiple ports, IxLoad lists each port as a separate entry in the Destination list of the client's command list.
Select Option Negotiation	

<p>Suppress Go Ahead</p>	<p>If you check this box, the server attempts to negotiate suppression of Go Ahead (GA) signals with the client. To negotiate this option, it sends a DO SGA (requesting the client to suppress its GA signals) and a WILL SGA (offering to suppress its own GA signals). The client must agree to the SGA options in order for them to take effect.</p> <p>A Telnet server sends a GA signal to shift control of its keyboard to the client. The need for the GA signal dates from the time of 'lockable' terminal keyboards that operated in half-duplex mode. Either the server or the host could control the keyboard, but not both.</p> <p>GA signals are normally used by Telnet hosts than can only handle half-duplex communications. Telnet hosts that can handle full-duplex communications can set the Suppress Go Ahead command to improve the communications speed on the Telnet connection.</p> <p>Note: If you enable the Suppress Go Ahead and Line Mode options at the same time, the client will send an Interrupt Process signal to the server.</p>
<p>Line Mode</p>	<p>If you enable this option, the server echoes back to the client each line of input. This is also known as line-at-a-time mode. The server will also negotiate the Line Mode sub-options described in RFC 1184.</p> <p>If you do not enable this option, the server echoes back to the client each character as it is typed. This is known as character-at-a-time mode.</p> <p>The Line Mode option exists for several reasons. The primary motivation for line mode is that in networks with long delays, typing a character and waiting for the remote end to echo it back becomes tedious for the user at the local terminal.</p> <p>As an additional reason, sending each character separately requires more packets than sending the entire line of characters all at once; in older networks that charged for every packet sent, this increased the cost of the Telnet session.</p> <p>Finally, some computer systems (especially supercomputers) do not handle single-character inputs as efficiently as they do lines of input.</p> <p>See Telnet Server (on page 1614) for more information on how the IxLoad Telnet server functions in Line Mode.</p> <p>Note: If you enable the Suppress Go Ahead and Line Mode options at the same time, the client will send an Interrupt Process signal to the server.</p>
<p>Echo</p>	<p>If you enable this option, the server returns to the client every character it receives from the client.</p> <p>Note: If a client tries to echo the server's data, the server rejects it by sending DONT commands.</p>



Advanced Options tab

The Advanced Options tab contains the controls for configuring the Telnet server's global options. The table below describes the controls on the tab.

Parameter	Description
Enable ESM	<p>You can use this field to cause the server to transmit segments that are smaller than the maximum size advertised by the client.</p> <p>If Effective Send MSS (TX) is enabled, the server transmits segments of the size defined in this field or the size advertised by the client -- whichever is smaller.</p> <p>If Effective Send MSS (TX) is not enabled, the server transmits segments that respect the MSS advertised by the client.</p> <p>To configure the MSS advertised by the server, see IP and MAC Tab.</p>
Type of Service (TOS/DSCP)	<p>If enabled, allows you to select the Type of Service (TOS) bits that will be set in the packets sent by the client:</p> <p>Best Effort (0x0): Routine service.</p> <p>Class 1 (0x20): Priority service, Assured Forwarding class 1.</p> <p>Class 2 (0x40): Immediate service, Assured Forwarding class 2.</p> <p>Class 3 (0x60): Flash, Assured Forwarding class 3</p> <p>Class 4 (0x80): Flash-override, Assured Forwarding class 4</p> <p>Express Forwarding (0xA0): Critical-ecp</p> <p>Control (0xC0): Internet-control</p>

Configuring a Telnet Client



To configure a Telnet client, configure the controls on the following tabs:

Telnet tab

Command List tab

Advanced Options tab (see "[Telnet Client Advanced Options Tab](#)" on page 1627)

Telnet tab

The client Telnet tab contains the controls that control the Telnet client's basic behavior.

The table below describes the fields.

Parameter	Description
Default Command Prompt	Character expected from the server as the command prompt. The server sends the command prompt to indicate that it is ready to receive the next command from the client. The default command prompt is the pound sign (#).
Timeout	Amount of time that the client waits for a response after sending a command to a server. If the client does not receive a response within the timeout period, it closes the connection.
Terminating Character Set	Characters used to indicate the end of a line (newline).
Enable Options Negotiation	If this option is enabled, the client attempts to negotiate any Telnet options offered by the server. If this option is not enabled, the client rejects any options offered by the server by sending DONT or WONT responses to the offered options.

Telnet Client Commands

This section describes the commands available for the Telnet client. The command list contains the list of commands that the Telnet client executes during a test. To manage the command list, you use the Command Editor. See Command Editor for information on using the Command Editor.



Telnet Client {Telnet Session} Command

{Telnet Session} is an IxLoad command that performs a complete Telnet session. {Telnet Session} opens a connection to a server, performs a command, and then logs out. The table below describes the Telnet Client {Telnet Session} Command parameters.

Parameter	Description
Session Login Parameters	
Server IP	In the Server IP field, select the server activity (for an IxLoad Telnet server) or enter the IP address or hostname (for an external Telnet server) that the client will send this command to. If the external server listens for Telnet connections on a port other than the default port (23), add a colon (:) after the IP address or host name and then specify the port number.
User Name	User name used to login to the server. You can insert sequence generators into this field to create unique entries automatically. This field supports the following Sequence Generator types: <ul style="list-style-type: none">▪ Basic Independent▪ Enhanced Independent▪ Interdependent For information on how to use sequence generators, see Sequence Generators (on page 1626).
Password	Password string. You can insert sequence generators into this field to create unique entries automatically. This field supports the following Sequence Generator types: <ul style="list-style-type: none">▪ Basic Independent▪ Enhanced Independent▪ Interdependent For information on how to use sequence generators, see Sequence Generators (on page 1626).
Expected Login Prompt	String expected to be returned by the server after being contacted by the client. The default string is "login:".
Expected Password Prompt	String expected to be returned by the server after receiving the user name. The default string is "Password:".
Expected Command Prompt	String expected to be returned by the server after receiving the password. The default string is the Default Command Prompt.
Transaction Parameters	
Command	Command string to be executed on the server.

Substring to be matched in Response	String expected to be returned by the server after receiving the command string. The default string is the Default Command Prompt.
Close Session Parameters	
Exit Command	String sent to server to end the Telnet session

Telnet Client OPEN Command

OPEN opens a new Telnet connection to a server. The OPEN command takes the following arguments:

Parameter	Description
Server IP	<p>In the Server IP field, enter the IP address or hostname of the Telnet server that the client will send this command to. If the server listens for Telnet connections on a port other than the default port (23), add a colon (:) after the IP address or host name and then specify the port number.</p> <p>Note: The OPEN command's Server IP field does not support symbolic destinations (IxLoad server names), only IP addresses. This enables you to establish a Telnet connection to a specific IP address. If you want to use the OPEN command with an IxLoad server, enter its IP address.</p> <p>As an alternative to the OPEN command, you can use the {TelnetSession} (see "Telnet Client {Telnet Session} Command" on page 1622) command, which does support symbolic destinations. If you use a symbolic destination with a {TelnetSession} (see "Telnet Client {Telnet Session} Command" on page 1622) command, connections will be established to all the IP addresses that the destination IxLoad Telnet server is running on.</p>
Expect	<p>String expected to be returned by the server. The default string is <code>login:</code>.</p> <p>The client sends the OPEN command to the address configured in the Server IP field.</p>

Telnet Client LOGIN Command

LOGIN logs in to the server. The LOGIN command takes the following arguments:

Command	Description
Send	<p>User name used to login to server.</p> <p>You can insert sequence generators into this field to create unique entries automatically.</p> <p>This field supports the following Sequence Generator types:</p> <ul style="list-style-type: none"> ▪ Basic Independent ▪ Enhanced Independent ▪ Interdependent <p>For information on how to use sequence generators, see Sequence Generators (on page 1626).</p>



Expect	String expected to be returned by the server. The default string is "password:". The client sends the LOGIN command to the same server configured in the previous OPEN command.
--------	--

Telnet Client PASSWORD Command

PASSWORD sends the password for the user name used to login to the server. The PASSWORD command takes the following arguments:

Parameter	Description
Send	Password string. You can insert sequence generators into this field to create unique entries automatically. This field supports the following Sequence Generator types: <ul style="list-style-type: none">▪ Basic Independent▪ Enhanced Independent▪ Interdependent For information on how to use sequence generators, see Sequence Generators (on page 1626).
Expect	String expected to be returned by the server. The default string is the Default Command Prompt. The client sends the PASSWORD command to the same server configured in the previous OPEN command.

Telnet Client SEND Command

SEND sends a command to be executed on the server. The SEND command takes the following arguments:

Parameter	Description
Send:	Command sent to server. You can specify any command, including arguments, that is valid for the Telnet server. For example, you could send an 'ls' command to a Unix or Linux server, or you could send a 'dir' command to a Windows server.
Expect:	String expected to be returned by the server. The default string is the Default Command Prompt. The client sends the command string to the same server configured in the previous OPEN command.

Telnet Client {Think} Command

{Think} pauses the command list execution. The {Think} command takes the following arguments:

Parameter	Description
Min Duration	Minimum length of pause, in milliseconds. Min Duration must be less than or equal to the Max Duration.
Max Duration	Maximum length of pause, in milliseconds. Max Duration must be greater than or equal to the Min Duration. To make the pause a random length of time, enter different values for the Min and Max durations. To make the pause a fixed length of time, enter the same values for the Min and Max durations.

Telnet Client EXIT Command

EXIT ends the Telnet session. The EXIT command takes the following arguments:

Parameter	Description
Send	String sent to server to end the Telnet session. The client sends the EXIT command to the same server configured in the previous OPEN command.



Telnet Client {Loop Begin} Command

{Loop Begin} is an IxLoad command that you can add to the Command List to cause the commands between it and the {Loop End} to be executed a specified number of times.

For example, in a Command List that contains the following commands:

```
{Loop Begin}  
LOGIN  
PASSWORD  
EXIT  
{Loop End}
```

the LOGIN, PASSWORD, and EXIT commands would be executed for the number of times specified in the Loop Count field.

Command	Description
Loop Count	Number of times (iterations) to repeat the commands between {Loop Begin} and {Loop End}.

Telnet Client {Loop End} Command

{Loop End} ends the list of commands that will be executed by the preceding {Loop Begin} command.

Sequence Generators

Sequence generators are variables that you can include in certain fields, and that IxLoad will replace with generated values when you run a test. For example, you can use sequence generators to cause the Telnet client to generate multiple entries for the *User Name:* and *Password:* fields. The description for each field indicates whether or not you can use sequence generators in it.

The table below lists the sequence generators you can use.

Sequence Generator	Type	Description
[0 - 9]	Numbers	Creates a numeric range.
[a - z], [A - Z]	Letters	Creates an alphabetic range.
For a complete description of sequence generators, see Appendix A, Using Automatic Sequence Generators.		

Telnet Client Advanced Options Tab

The Advanced Options tab contains the controls for configuring the Telnet client's global options. The table below describes the parameters.

Parameter	Description
Enable ESM	<p>You can use this field to cause the client to transmit segments that are smaller than the maximum size advertised by the server.</p> <p>If Effective Send MSS (TX) is enabled, the client transmits segments of the size defined in this field or the size advertised by the server -- whichever is smaller.</p> <p>If Effective Send MSS (TX) is not enabled, the client transmits segments that respect the MSS advertised by the server.</p> <p>To configure the MSS advertised by the clients, see IP and MAC Tab.</p>
Type of Service (TOS/DSCP)	<p>If enabled, allows you to select the Type of Service (TOS) bits that will be set in the packets sent by the client:</p> <p>Best Effort (0x0): Routine service.</p> <p>Class 1 (0x20): Priority service, Assured Forwarding class 1.</p> <p>Class 2 (0x40): Immediate service, Assured Forwarding class 2.</p> <p>Class 3 (0x60): Flash, Assured Forwarding class 3.</p> <p>Class 4 (0x80): Flash-override, Assured Forwarding class 4.</p> <p>Express Forwarding (0xA0): Critical-ecp.</p> <p>Control (0xC0): Internet-control.</p>



Telnet Statistics

For the Telnet statistics, see the following sections.

Telnet Client Statistics (on page 1629)

Telnet Server Statistics (on page 1634)

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).

Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

If you installed IxLoad using the default installation path, the results files are stored in the following directory:

```
C:\Program  
Files\Ixia\IxLoad\Results\  
_<version>\<repository>_  
<testconfiguration>_  
<yyyy><mm><  
dd>_  
<hh><mm><ss>
```

Telnet Client Statistics

The table below describes the Telnet client statistics.

Statistic	Description
Connection Statistics	
Telnet Active Connections	Number of active Telnet connections.
Telnet Total Connections Requested	Number Telnet connections requested by the client.
Telnet Total Connections Succeeded	Number of Telnet connections established by the client.
Telnet Total Connections Failed	Number of failed attempts to establish Telnet connections.
Telnet Total Connections Latency	Average amount of time required to establish a Telnet connection. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
Telnet Total Sessions Completed	Total number of Telnet connections that ended normally.
Telnet Total Sessions Not Completed	Total number of Telnet sessions that were established but did not end normally.
Telnet Average Session Length	Average Telnet session duration. Note for Tcl API users: This is a weighted statistic. If you are using this statistic in a Tcl script, use the <i>kWeightedAverage</i> aggregation type.
Login Statistics	
Telnet Total Login Prompts Received	Total number of login prompts received.
Telnet Total Login Prompts Not Received	Total number instances in which the client connected to the server but did not receive a login prompt in return.
Telnet Total Password Prompts Received	Total number of password prompts received.
Telnet Total Password Prompts Not Received	Total number of instances in which the client sent a user name to the server but did not receive a password prompt in return.
Telnet Total Logins Sent	Total number of logins sent by the clients.



Telnet Total Logins Succeeded	Total number of successful logins by the clients.
Telnet Total Logins Failed	Total number of login attempts that failed.
Telnet Total Logins Failed (Timed Out)	Total number of login attempts that failed because a response was not received within the timeout period.
Telnet Total Logins Failed (Other)	Total number of login attempts that failed for reasons other than a timeout.
Telnet Total Passwords Sent	Total number of passwords sent by the client.
Telnet Total Passwords Succeeded	Total number of passwords that succeeded.
Telnet Total Passwords Failed	Total number of passwords that failed.
Telnet Total Passwords Failed (Timed Out)	Total number of passwords that failed because no response was received within the timeout period.
Telnet Total Passwords Failed (Other)	Total number of passwords that failed for reasons other than a timeout.
Transaction Statistics	
Telnet Total Transactions	Total number of Telnet transactions completed. One transaction consists of a LOGIN, PASSWORD, SEND, or EXIT command and the response to it.
Telnet Total Commands Sent	Total number of SEND commands sent by the client.
Telnet Total Commands Succeeded	Number of SEND commands during a Telnet session that succeeded. IxLoad considers a command to be successful if the client sent a SEND command to the server and the server returned the expected string.
Telnet Total Commands Failed	Number of SEND commands that failed. IxLoad considers a command to have failed if the client sent a SEND command to the server and the server did not return the expected string.
Telnet Total Commands Failed (Timed Out)	Total number of SEND commands that failed because no response was received within the timeout period.

Telnet Total Commands Failed (Other)	Total number of SEND commands that failed for reasons other than a timeout.
Bytes Statistics	
Telnet Total Bytes Sent	Total number of bytes sent over Telnet connections.
Telnet Total Bytes Received	Total number of bytes received by the client over Telnet.
Telnet Total Bytes Sent And Received	Total number of bytes sent and received by the client over Telnet.
Telnet Total Throughput	Total throughput over Telnet.
Echo Options Statistics	
Telnet Total Echo Options Do Received	Number of requests received from the server to begin using the Echo option.
Telnet Total Echo Options Dont Received	Number of requests received from the server to stop using the Echo option.
Telnet Total Echo Options Will Received	Number of responses received from the server agreeing to use the Echo option.
Telnet Total Echo Options Wont Received	Number of responses received from the server rejecting use of the Echo option.
Telnet Total Echo Options Do Sent	Number of requests sent to the server to begin using the Echo option.
Telnet Total Echo Options Dont Sent	Number of requests sent to the server to stop using the Echo option.
Telnet Total Echo Options Will Sent	Number of responses sent by the client agreeing to use the Echo option.
Telnet Total Echo Options Wont Sent	Number of responses sent by the client rejecting use of the Echo option.
Suppress Go Ahead Options	
Telnet Total Suppress Go Ahead Options Do Received	Number of requests received from the server to begin using the Go Ahead option.
Telnet Total Suppress Go Ahead Options Dont Received	Number of requests received from the server to stop using the Go Ahead option.
Telnet Total Suppress Go Ahead Options Will Received	Number of responses received from the server agreeing to use the Go Ahead option.



Telnet Total Suppress Go Ahead Options Wont Received	Number of responses received from the server rejecting use of the Go Ahead option.
Telnet Total Suppress Go Ahead Options Do Sent	Number of requests sent to the server to begin using the Go Ahead option.
Telnet Total Suppress Go Ahead Options Don't Sent	Number of requests sent to the server to stop using the Go Ahead option.
Telnet Total Suppress Go Ahead Options Will Sent	Number of responses sent by the client agreeing to use the Go Ahead option.
Telnet Total Suppress Go Ahead Options Wont Sent	Number of responses sent by the client rejecting use of the Go Ahead option.
Line Mode Options Statistics	
Telnet Total Line Mode Options Do Received	Number of requests received from the server to begin using the Line Mode option.
Telnet Total Line Mode Options Dont Received	Number of requests received from the server to stop using the Line Mode option.
Telnet Total Line Mode Options Will Received	Number of responses received from the server agreeing to use the Line Mode option.
Telnet Total Line Mode Options Wont Received	Number of responses received from the server rejecting use of the Line Mode option.
Telnet Total Line Mode Options Do Sent	Number of requests sent to the server to begin using the Line Mode option.
Telnet Total Line Mode Options Dont Sent	Number of requests sent to the server to stop using the Line Mode option.
Telnet Total Line Mode Options Will Sent	Number of responses sent by the client agreeing to use the Line Mode option.
Telnet Total Line Mode Options Wont Sent	Number of responses sent by the client rejecting use of the Line Mode option.
Sub-Options Statistics	
Telnet Total Suboptions Received	Total number of Telnet sub-options received by the client.
Telnet Total Suboptions Sent	Total number of Telnet sub-options sent by the client.
Line Mode Sub-Options Edit Mask Statistics	
Telnet Total Line Mode Suboptions Edit Mask Received	Total number of Telnet sub-option edit masks received by the client.

Telnet Total Line Mode Suboptions Edit Mask Sent	Total number of Telnet sub-option edit masks sent by the client.
--	--



Telnet Server Statistics

The table below describes the Telnet server statistics.

Statistic	Description
Connection Statistics	
Telnet Active Connections	Number of Telnet connections up.
Telnet Total Accepted Connections	Total number of requests for Telnet connections accepted by the server.
Username Statistics	
Telnet Login Prompts Sent	Number of login prompts sent by the server to the clients.
Telnet UserNames Succeeded	Number of user names accepted by the server.
Telnet UserNames Failed	Number of instances in which the server did not receive a user name for any reason.
Telnet UserNames Failed (Timed Out)	Number of instances in which the server did not receive a user name within the timeout period.
Telnet UserNames Failed (Other)	Number of instances in which the server did not receive user names for reasons other than a timeout.
Password Statistics	
Telnet Password Prompts Sent	Number of password prompts sent by the server to the clients.
Telnet Passwords Succeeded	Number of passwords accepted by the server.
Telnet Passwords Failed	Number of passwords rejected for any reason by the server.
Telnet Passwords Failed (Timed Out)	Number of passwords rejected by the server because they were not received within the specified time.
Telnet Passwords Failed (Other)	Number of passwords rejected by the server for reasons other than a timeout.
Login Statistics	

Telnet Logins Succeeded	Number of successful logins to the server. A successful login occurs when the server received a user name and password from the client and then returned a command prompt.
Telnet Logins Failed	Number of attempts to login to the server that failed.
Transaction Statistics	
Telnet Commands Processed	Total number of SEND commands received over Telnet connections and executed on the server.
Byte Statistics	
Telnet Total Bytes Sent	Total number of bytes sent over Telnet connections by the server.
Telnet Total Bytes Received	Total number of bytes received over Telnet connections by the server.
Telnet Total Bytes Sent And Received	Total number of bytes sent and received over Telnet by the server.
Telnet Total Throughput	Total Telnet throughput.
Generic Option Statistics	
Telnet Option Negotiation Failed	Total number of attempts to negotiate Telnet options that failed for any reason.
Telnet Option Negotiation Failed (Timed Out)	Total number of attempts to negotiate Telnet options that failed because no response was received within the timeout period.
Telnet Option Negotiation Failed (Other)	Total number of attempts to negotiate Telnet options that failed for reasons other than a timeout.
Generic Sub-Option Statistics	
Telnet Suboption Negotiation Failed	Total number of attempts to negotiate Telnet sub-options that failed for any reason.
Telnet Suboption Negotiation Failed (Timed Out)	Total number of attempts to negotiate Telnet sub-options that failed because no response was received within the timeout period.
Telnet Suboption Negotiation Failed (Other)	Total number of attempts to negotiate Telnet sub-options that failed for reasons other than a timeout.
Echo Option Statistics	

Telnet Echo Options DO Sent	Number of requests sent by the server to the client to begin using the Echo option.
Telnet Echo Options WILL Sent	Number of responses sent by the server agreeing to begin using the Echo option.
Telnet Echo Options DONT Sent	Number of requests sent by the server to the client to stop using the Echo option.
Telnet Echo Options WONT Sent	Number of responses sent by the server rejecting use of the Echo option.
Telnet Echo Options DO Received	Number of requests received by the server to begin using the Echo option.
Telnet Echo Options WILL Received	Number of responses received by the server agreeing to begin using the Echo option.
Telnet Echo Options DONT Received	Number of responses received by the server rejecting use of the Echo option.
Telnet Echo Options WONT Received	Number of responses received by the server agreeing to stop using the Echo option.
GA Suppress Option Statistics	
Telnet GA Suppress Options DO Sent	Number of requests sent by the server to suppress Go Ahead messages. Effectively, the server requests the clients not to send Go Ahead messages.
Telnet GA Suppress Options WILL Sent	Number of the responses sent by the server agreeing to suppress Go Ahead messages.
Telnet GA Suppress Options DONT Sent	Number of requests sent by the server asking the clients not to suppress Go Ahead messages. Effectively, the server requests that the clients send Go Ahead messages.
Telnet GA Suppress Options WONT Sent	Number of responses sent by the server rejecting suppression of Go Ahead messages. Effectively, the server signals that it will send Go Ahead messages.
Telnet GA Suppress Options DO Received	Number of requests received by the server to suppress Go Ahead messages. Effectively, the server is being requested to not send Go Ahead messages.

Telnet GA Suppress Options WILL Received	<p>Number of responses received by the server indicating that the client will suppress Go Ahead messages.</p> <p>Effectively, the clients agree to not send Go Ahead messages.</p>
Telnet GA Suppress Options DONT Received	<p>Number of requests received by the server to not suppress Go Ahead messages.</p> <p>Effectively, the server is being requested to send Go Ahead messages.</p>
Telnet GA Suppress Options WONT Received	<p>Number of responses received by the server indicating that the client will not suppress Go Ahead messages.</p> <p>Effectively, the clients agree to send Go Ahead messages.</p>
Line-Mode Option Statistics	
Telnet Line-mode Options DO Sent	Number of requests sent by the server to the clients to begin using the Line Mode option.
Telnet Line-mode Options WILL Sent	Number of responses sent by the server agreeing to begin using the Line mode option.
Telnet Line-mode Options DONT Sent	Number of requests sent by the server to the clients to stop using the Line Mode option.
Telnet Line-mode Options WONT Sent	Number of responses sent by the server agreeing to stop using the Line mode option.
Telnet Line-mode Options DO Received	Number of requests received by the server to begin using the Line Mode option.
Telnet Line-mode Options WILL Received	Number of responses received by the server agreeing to begin using the Line Mode option.
Telnet Line-mode Options DONT Received	Number of requests received by the server to stop using the Line Mode option.
Telnet Line-mode Options WONT Received	Number of responses received by the server rejecting use of the Line Mode option.
Line-Mode Sub-Option Statistics	
Telnet Line-mode Sub-options Sent	Number of messages setting the Line Mode sub-options sent by the server.
Telnet Line-mode Sub-options Received	Number of messages setting the Line Mode sub-options received by the server.
Special Statistics	



Telnet Line-mode Edit Mask Sent	Number of Line Mode Edit Mask messages sent by the server.
Telnet Line-mode Edit Mask Received	Number of Line Mode Edit Mask messages received by the server.

30

TFTP

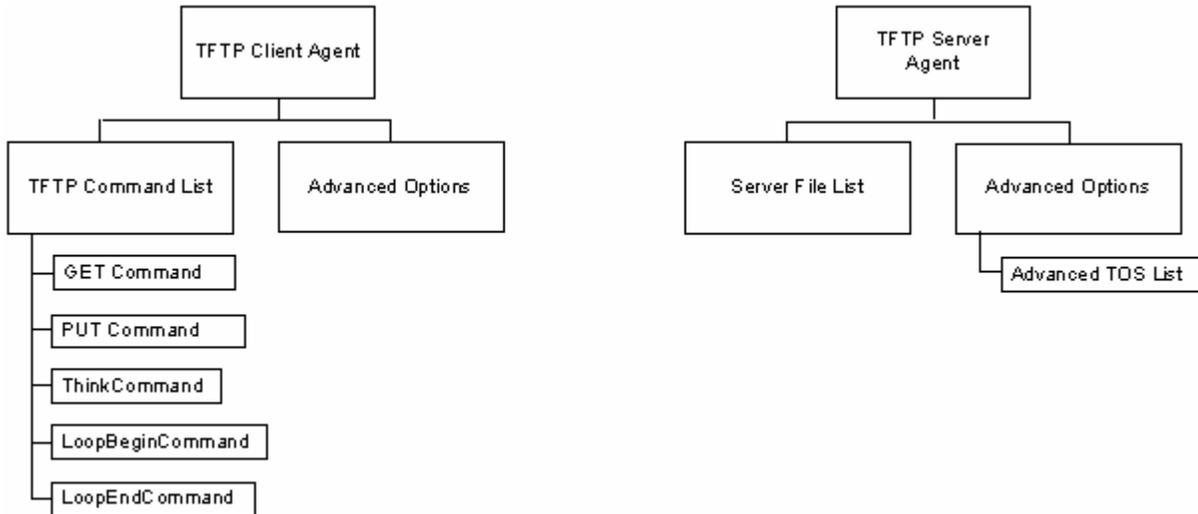
This section describes the TFTP Tcl API objects.

In this chapter:

Overview	1640
TFTP Client Agent	1642
TFTP Server Agent	1651
TFTP Client Statistics	1658
TFTP Server Statistics	1660

Overview

The IxLoad TFTP API consists of a client agent and its commands, structured as shown below.



Objectives

The objectives (userObjective) you can set for TFTP are listed below. Test objectives are set in the ixTimeline object.

- transactionRate
- simulatedUsers

TFTP Client Agent

Trivial File Transfer Protocol (TFTP) is a very simple file transfer protocol that functions essentially like a stripped-down version of FTP. Refer to `TFTP Client Agent` on page 26-3 for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of the TFTP client agent.
name	Name associated with the client agent.

TFTP Command List

The TFTP Command List creates the list of TFTP commands that the client will send to a TFTP server. Refer to `TFTP Command List` on page 26-8 for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	Command that client will send.

TFTP Client Advanced Options

The TFTP client advanced options define additional connection options. Refer to `TFTP Client Advanced` for a full description of this command. The important options of this command are listed below.

Option	Usage
responseTimeout	Time, in seconds, that the client waits for a response from the server.
ipPreference	If you have a mixture of IPv4 and IPv6 subnets configured on the client network, these fields select the order that the TFTP client will use the subnets.
numberOfRetries	Number of times that the TFTP client will re-send an un-acknowledged GET (RRQ packet) or PUT (WRQ packet) command.



TFTP Client Agent

TFTP Client Agent - create a TFTP client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_TFTPClient1 [$Traffic1_Network1 activityList.appendItem \
options...]
$Activity_TFTPClient1 agent.config \
```

DESCRIPTION

A TFTP client agent is added to the `activityList` option of the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. It is customary to set all the options of the client agent during the `appendItem` call.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#-----
# Activity TFTPClient1 of NetTraffic Traffic1@Network1
#-----
```

```
set Activity_TFTPClient1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType           "tftp Client" ]  
$Activity_TFTPClient1 agent.config \  
-enable                     true \  
-name                        "TFTPClient1"
```

SEE ALSO

`ixNetTraffic`



TFTP Command List

TFTP Command List—Creates the list of TFTP commands that the client will send to a TFTP server.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
$Activity_TFTPClient1 agent.pm.cmdList.appendItem \ options...
```

DESCRIPTION

A command is added to the TFTP Command List object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

TFTP command to be executed. One of the following:

Command	Description
GETCommand	The GET command retrieves a file from a TFTP Server by sending an RRQ (Read request) packet.
PUTCommand	The PUT command stores a file on a server by sending a WRQ request. The file can be an actual file, or a dummy file that consists of generated data.
ThinkCommand	The THINK Command causes the client to become idle for a specified length of time, to simulate real-world usage scenarios in which a user may spend time absorbing or processing information received from the server before sending the next command. If you specify identical values for the minimum and maxintervals, the client will be inactive for a fixed length of time. If you specify different values for the minand maximum intervals, IxLoad will select a value within the range and cause the client to be inactive for that length of time.
LoopBeginCommand	The Loop BeginCommand is an IxLoad command that you can add to the Command List to cause the combetween it and the LoopEndCommand to be executed a specified number of times.
LoopEndCommand	LoopEndCommand ends the list of commands that will be executed by the preceding LoopBeginCommand.

Arguments for id = GET Command

`getFileName`

Specifies the name and path of the file to be retrieved. The file path must be Unix-style. For example: `/abcd/foo.txt`

You can include sequence generators in this field to generate requests for multiple files automatically. (Default = "").

`enableFileSizeOption`

If enabled, the client includes the Transfer Size (tsize) option in the RRQ packet, with the value set to 0. (Default = "0").

`transportMode`

Type of data contained in file to be transferred:

Value	Description
<code>netascii</code> (Default)	0
<code>octet</code>	1

`enableBlkSizeOption`

If `true`, the client suggests the size of the Data field to be used in DATA packets from the server. (Default = "0").

`enableTimeoutOption`

If enabled, the client includes the Timeout (tout) option in the RRQ packet, with the value configured on the Advanced Options. (Default = "0").

`blksize`

Specifies the value of the block size, if it is enabled. (Default = "512").

`serverAddr`

IP address and port number of the external TFTP server. If you do not specify a port number, the IxLoad client uses port 69. (Default = "198.18.0.100").

Arguments for id = PUT Command

fileType

The file type can be of:

Value	Description
real file (Default)	0
dummy file	1

transportMode

Type of data contained in file to be transferred:

Value	Description
netascii (Default)	0
octet	1

enableBlkSizeOption

If true, the client suggests the size of the Data field to be used in DATA packets from the server. (Default = "0").

remoteFileName

Name and path that the file will be stored on the remote server. (Default = "").

dummyFileRange

If selected as fileType, the IxLoad TFTP client transfers a file composed of generrandom data. (Default = "8-8").

blksize

Specifies the value of the block size, if it is enabled. (Default = "512").

serverAddr

IP address and port number of the external TFTP server. If you do not specify a port number, the IxLoad client uses port 69. (Default = "198.18.0.100").

putFileName

Specifies the name of the file. (Default = "").

Arguments for id = ThinkCommand

minimumInterval

Minimum length of time that the client is idle. Minimum = "1000," maximum = "2,147,483,647."
(Default = "1000").

maximumInterval

Maximum length of time that the client is idle. Minimum = "1000," maximum = "2,147,483,647."
(Default = "1000").

Arguments for id = LoopBeginCommand

loopCount

Number of times to repeat the enclosed commands. '0' treated as infinity. Mini= "0," maximum =
"2,147,483,647." (Default = "5").

Arguments for id = LoopEndCommand

None.

EXAMPLE

```
$Activity_TFTPCliet1 agent.pm.cmdList.appendItem \  
-id "GET" \  
-getFileName "" \  
-enableFileSizeOption false \  
-transportMode 0 \  
-enableBlkSizeOption false \  
-enableTimeoutOption false \  
-blksize "512" \  
-serverAddr "198.18.0.100"
```

SEE ALSO

TFTP Client Agent



TFTP Client Advanced

TFTP Client Advanced Options - configure a TFTP client's advanced options

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_TFTPClient1 [$Traffic1_Network1 activityList.appendItem \.]
$Activity_TFTPClient1 agent.pm.advOptions.config options...
```

DESCRIPTION

A TFTP client's advanced options are set by modifying the options of the `pm.advanced` option of the `TFTP Client Agent` object using its `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`responseTimeout`

Time, in seconds, that the client waits for a response from the server. You can enter values from 1 to 255 seconds.

This value is included as the value for the `Tout` option included with an `RRQ` (GET command) or `WRQ` (PUT command). Minimum = "1", Maximum = "2147483", Default = "20".

`ipPreference`

If you have a mixture of IPv4 and IPv6 subnets configured on the client network, these fields select the order that the TFTP client will use the subnets. The values are:

Value	Description
0	IPv4: The client will use addresses from the IPv4 subnets only.
1	IPv6: The client will use addresses from the IPv6 subnets only.
2 (default)	Both, IPv4 first: The client will use addresses from the IPv4 subnets first, then if it needs more addresses, it will use addresses from the IPv6 subnets.
3	Both, IPv6 first: The client will use addresses from the IPv6 subnets first, then if it needs more addresses, it will use addresses from the IPv4 subnets.

`numberOfRetries`

Number of times that the TFTP client will re-send an un-acknowledged GET (RRQ packet) or PUT (WRQ packet) command. (Default = "3").

EXAMPLE

```
$Activity_TFTPClient1 agent.pm.advOptions.config \  
-responseTimeout           120 \  
-implicitLoopCheck        true \  
-ipPreference              2 \  
-numberOfRetries           3
```

SEE ALSO

TFTP Client Agent, TFTP Command List

TFTP Server Agent

TFTP Server Agent - create a TFTP server

SYNOPSIS

```
set Activity_TFTPServer1 [$myNetTraffic activityList.appendItem \  
    -protocolAndType          "tftp Server" ]
```

DESCRIPTION

A TFTP server agent is added to the `activityList` option of the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. It is customary to set all the options of the client agent during the `appendItem` call.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this server agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

`timeline`

The name of the timeline to be used with this object.

STATISTICS

EXAMPLE

```
set Activity_TFTPServer1 [$myNetTraffic activityList.appendItem \  
    -protocolAndType          "tftp Server" ]  
  
set _Match_Longest_ [::IxLoad new ixMatchLongestTimeline]  
  
$Activity_TFTPServer1 config \  
    -enable          true \  
    -name            "TFTPServer1" \  
    -timeline        $_Match_Longest_
```

SEE ALSO

`ixNetTraffic`

fileList

fileList - add files to a TFTP server

SYNOPSIS

```
$Activity_TFTPServer1 agent.pm.files.fileList.appendItem \  
-id "File" \  
-filePath "<Dummy File>" \  
-fileName "/#1"
```

DESCRIPTION

The `fileList` object adds files to the list of files hosted by a TFTP server. Files can be real files or simulated ("dummy") files.

To add a file to the list, you use the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `fileList`. It is customary to set all the options of the `fileList` during the `appendItem` call.

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands.

Before you add items to the `fileList`, you should initialize the list by using the `clear` subcommand of the `ixConfigSequenceContainer` command.

SUBCOMMANDS

OPTIONS

`id`

Server file list name. (Default = "File").

`filePath`

For an actual file, this is the name and full path of the file. For a simulated file, this is `<Dummy File>`. (Default = "`<Dummy File>`").

`fileName`

For an actual file, `fileName` is the label of the file, as advertised by the server. For a simulated file, `fileName` is the size of the file, in the format `/#n`, where `n` is the size in bytes. For example, for a 64-byte simulated file, specify `/#64`. (Default = `"/#1"`).

STATISTICS

EXAMPLE

```
$Activity_TFTPServer1 agent.pm.files.fileList.clear

$Activity_TFTPServer1 agent.pm.files.fileList.appendItem \
  -id                "File" \
  -filePath          "<Dummy File>" \
  -fileName          "/#1"
```

SEE ALSO

advanced

advanced - configure a TFTP server's global properties

SYNOPSIS

```
$Activity_TFTPServer1 agent.pm.advanced.config \  
    -enableTOS                                false \  
    
```

DESCRIPTION

The advanced object configures the TFTP server's global properties. .

SUBCOMMANDS

OPTIONS

enableTOS

Enables use of Type of Service (TOS) bits in TFTP packets. Configure the selected ToS type in `typeOfService`. Default = "false".

enableFileSizeOption

If enabled and the server receives a GET or PUT request from client with the File Size Option set, the server responds by sending an OACK with containing the size of file size, in octets. Default = "false".

enableBlkSizeOption

Causes the server to negotiate a Block Size with the client. Configure the server's Block Size value in the `blkSize` option. Default = "false".

enableTimeoutOption

Causes the server to negotiate a timeout interval with the client. Configure the server's timeout value in the `responseTimeOut` field. Default = "false".

typeOfService

ToS bit set in traffic from the TFTP server. To configure the list of allowed ToS settings, create an `availableTosList` object. Default = "Best Effort (0x0)".

blkSize

Block size used when `enableBlkSizeOption` is true. Default = "512".



retryCount

Number of DATA or ACK/OACK packets to be re-sent to the client if no response is received. Default = "3".

responseTimeOut

Length of time, in seconds, that the server waits for a response from the client. Default = "120".

tftpPort

Port number that the TFTP server listens on. Default = "69".

STATISTICS

EXAMPLE

```
$Activity_TFTPServer1 agent.pm.advanced.config \  
-enableTOS                false \  
-enableFileSizeOption     false \  
-enableBlkSizeOption      false \  
-enableTimeoutOption      false \  
-typeOfService            "Best Effort (0x0)" \  
-blkSize                   "512" \  
-retryCount                3 \  
-responseTimeOut          120 \  
-tftpPort                  69
```

SEE ALSO

availableTosList

availableTosList - configure the list of ToS levels for a TFTP server.

SYNOPSIS

```
$Activity_TFTPServer1 agent.pm.availableTosList.appendItem \  
    -id                "AvailableTypeOfService" \  
    -tos_value         "Best Effort (0x0)"
```

DESCRIPTION

The `availableTosList` object configures the list of available ToS levels.

To add a ToS level to the list, you use the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `availableTosList`. It is customary to set all the options of the `availableTosList` during the `appendItem` call.

Each member of the list can be separately addressed and modified using the `ixConfig` subcommands.

Before you add items to the `availableTosList`, you should initialize the list by using the `clear` subcommand of the `ixConfigSequenceContainer` command.

SUBCOMMANDS

OPTIONS

`id`

ToS list name. (Default = "AvailableTypeOfService").

`tos_value`

ToS level to be added to the list. Default = "" (null).

STATISTICS

EXAMPLE

```
$Activity_TFTPServer1 agent.pm.availableTosList.clear

$Activity_TFTPServer1 agent.pm.availableTosList.appendItem \
  -id "AvailableTypeOfService" \
  -tos_value "Best Effort (0x0)"
```

SEE ALSO

TFTP Client Statistics

The table below describes the TFTP client statistics.

Statistic	Description
Test Objective Statistics	
TFTP Simulated Users	Number of TFTP users simulated.
TFTP Transactions	Number of TFTP transactions completed.
TFTP Transaction Rate	Rate at which TFTP transactions were completed.
Upload / Download Statistics	
TFTP Total File Download Requests Sent	Number of GET commands (RRQ requests) sent by the client.
TFTP Total File Download Requests Successful	Number of files that the client successfully downloaded.
TFTP Total File Download Requests Failed	Number of files that the client failed to download.
TFTP Total File Upload Requests Sent	Number of PUT commands (WRQ requests) sent by the client.
TFTP Total File Upload Requests Successful	Number of files that the client successfully uploaded.
TFTP Total File Upload Requests Failed	Number of files that the client failed to upload.
ACK / OACK Statistics	
TFTP Total Acknowledgement (ACK) Sent	Number of ACK packets received by the client.
TFTP Total Acknowledgement (ACK) Received	Number of ACK packets received by the client.
TFTP Total Option Acknowledgement (OACK) Received	Number of OACK packets received by the client.
Bytes Sent / Received Statistics	

TFTP Total Bytes Sent	Total number of bytes sent in TFTP packets of all types.
TFTP Total Bytes Received	Total number of bytes received in TFTP packets of all types.
TFTP Total Bytes Sent And Received	Combined total of bytes sent and received in TFTP packets of all types.
Data Bytes Sent / Received Statistics	
TFTP Total Data Bytes Sent	Number of bytes sent in DATA packets.
TFTP Total Data Bytes Received	Number of bytes received in DATA packets.
TFTP Total Data Bytes Sent per sec	Rate, in bytes per second, at which the client sent DATA packets.
TFTP Total Data Bytes Received per sec	Rate, in bytes per second, at which the client received DATA packets.
Bytes Sent / Received Rate Statistics	
TFTP Total Bytes Sent per sec	Rate, in bytes per second, at which the client sent TFTP packets of all types.
TFTP Total Bytes Received per sec	Rate, in bytes per second, at which the client received TFTP packets of all types.
TFTP Total Bytes Sent And Received per sec	Combined rate, in bytes per second, at which the client sent and received TFTP packets of all types.
TFTP Total Out of Sequence Packets Received	Number of TFTP packets that were received out of order.
Error Statistics	
TFTP Total Timeouts Received	Number of timeouts received.
TFTP Total Errors	Number of ERROR packets received by the client.
TFTP Total Errors Received In Response to Read Request	Number of ERROR packets that the client received in response to a GET command (RRQ request).
TFTP Total Errors Received In Response to Write Request	Number of ERROR packets that the client received in response to a PUT command (WRQ request).
TFTP ERROR Received (code 0)	Number of ERROR packets received with error code 0 (Not defined, see error message (if any)).
TFTP ERROR Received (code 1)	Number of ERROR packets received with error code 1 (File not found).
TFTP ERROR Received (code 2)	Number of ERROR packets received with error code 2 (Access violation).



TFTP ERROR Sent (code 3)	Number of ERROR packets sent with error code 3 (Disk full or allocation exceeded). In addition to sending error code 3 packets in case of <i>disk full</i> or <i>allocation exceeded</i> errors, the client will also send an error code 3 packet if the server responds to the client's RRQ with an OACK that contains a tsize that the client cannot handle.
TFTP ERROR Received (code 3)	Number of ERROR packets received with error code 3 (Disk full or allocation exceeded). In addition to receiving error code 3 packets in case of <i>disk full</i> or <i>allocation exceeded</i> errors on the server, if the client sends a WRQ with a tsize that the server cannot handle, the server returns an error code 3 packet.
TFTP ERROR Received (code 4)	Number of ERROR packets received with error code 4 (Illegal TFTP operation).
TFTP ERROR Received (code 5)	Number of ERROR packets received with error code 5 (Unknown transfer ID).
TFTP ERROR Received (code 6)	Number of ERROR packets received with error code 6 (File already exists).
TFTP ERROR Received (code 7)	Number of ERROR packets received with error code 7 (No such user).
TFTP ERROR Sent (code 8)	Number of ERROR packets sent with error code 8 (Block size rejected).
TFTP ERROR Received (code 8)	Number of ERROR packets received with error code 8 (Block size rejected).
TFTP Other error	Number of ERROR packets received that were not error codes 1-8.

TFTP Server Statistics

The table below describes the TFTP server statistics.

Statistic	Description
TFTP Request / Response Statistics	
TFTP Total Download Request Received	Number of GET requests received.
TFTP Total Download Request Succeeded	Number of GET requests completed successfully.
TFTP Total Download Request Failed	Number of GET requests that failed.
TFTP Total Upload Request Received	Number of PUT requests received.
TFTP Total Upload Request Succeeded	Number of PUT requests completed successfully.
TFTP Total Upload Request Failed	Number of PUT requests that failed.

TFTP Total ACK Sent	Number of ACK responses sent.
TFTP Total ACK Received	Number of ACK responses received.
TFTP Total OACK Sent	Number of OACK responses sent.
Total Bytes Stats	
TFTP Total Bytes Sent	Number of bytes sent in TFTP packets of all types.
TFTP Total Bytes Received	Number of bytes received in TFTP packets of all types.
TFTP Total Bytes Sent And Received	Combined total of bytes sent and received in TFTP packets of all types.
TFTP Throughput	Combined rate, in bytes per second, at which the server sent and received TFTP packets of all types.
Data Bytes Statistics	
TFTP Total Data Bytes Sent	Number of bytes sent in DATA packets.
TFTP Total Data Bytes Received	Number of bytes received in DATA packets.
TFTP Total Data Bytes Sent And Received	Combined total of bytes sent and received in DATA packets.
TFTP Data Throughput	Combined rate, in bytes per second, at which the server sent and received DATA packets.
Error Statistics	
TFTP Total Errors Received	Total number of TFTP error messages received.
TFTP Total Timeout Errors	Number of times that the server did not receive a response within the timeout period.
TFTP ERROR Sent (code 0)	Number of error code 0 messages sent. Error code 0 is undefined; a description of the error may be in the string portion of the message.
TFTP ERROR Sent (code 1)	Number of error code 1 messages sent. Error code 1 is: File not found.
TFTP ERROR Sent (code 2)	Number of error code 2 messages sent. Error code 2 is: Access violation.
TFTP ERROR Received (code 3)	Number of error code 3 messages received. Error code 3 is: Disk full or allocation exceeded.
TFTP ERROR Sent (code 3)	Number of error code 3 messages sent. Error code 3 is: Disk full or allocation exceeded.
TFTP ERROR Sent (code 4)	Number of error code 4 messages received. Error code 4 is: Illegal TFTP operation.
TFTP ERROR Sent (code 5)	Number of error code 5 messages received. Error code 5 is: Unknown transfer ID.
TFTP ERROR Sent (code 6)	Number of error code 6 messages received. Error code 6 is: File already exists.



TFTP ERROR Sent (code 7)	Number of error code 7 messages received. Error code 7 is: No such user.
TFTP ERROR Received (code 8)	Number of error code 8 messages received. Error code 8 is sent to terminate a transfer due to a failure in option negotiation.
TFTP ERROR Sent (code 8)	Number of error code 8 messages received. Error code 8 is sent to terminate a transfer due to a failure in option negotiation.
TFTP Other error	Number of TFTP errors that were not classified as error code 0-8.

31

Trace File Replay

This section describes the Trace File Replay Tcl API objects.

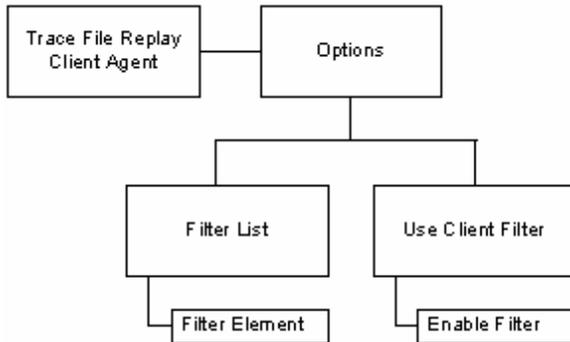
In this chapter:

Overview	1664
Trace File Replay Client Agent	1668
Trace File Replay Server Agent	1675

Overview

The IxLoad Trace File Replay API consists of client agent and server agents and their commands.

Figure 22-1. Trace File Replay Client API Structure



Objectives

The objectives (userObjective) you can set for Trace File Replay are listed below. Test objectives are set in the ixTimeline object.

- simulatedUsers

Trace File Replay Client Commands

This section lists the Trace File Replay client commands.

Trace File Replay Client Agent

The Trace File Replay Client Agent command defines a client that will transmit a packet stream to a Trace File Replay Server Agent. Refer to *Trace File Replay Client Agent* on page 22-5 for a full description of this command. The most sigoptions of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time.
protocol	Protocol used by the client agent.
type	Defines the agent as either a client or server.

Options

The Options command configures the Trace File Replay client's options. Refer to `Options` on page 22-7 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>destinationServerActivity</code>	Name of the IxLoad Trace File Replay server that the client will connect to.
<code>traceFileName</code>	Name and path of the pcap-format trace file that the client will use to generate the traffic stream.
<code>replayBidirectionalTraffic</code>	If <code>true</code> , the client uses the same trace (pcap) file as selected for the server agent.
<code>enableFilter</code>	If <code>true</code> , the filters in the client's <code>filterList</code> are applied to the incoming packet stream from the server.
<code>filterList</code>	List of filters applied to incoming packet stream.

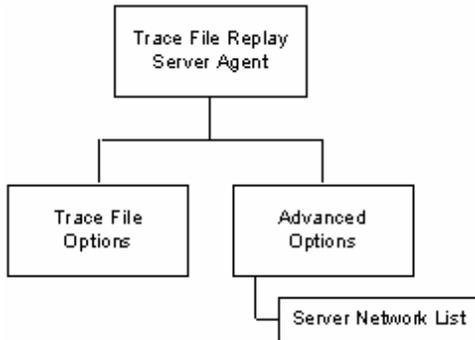
Filter List

The Filter List command configures a filter to be applied to the packet stream. Refer to `Filter List` on page 22-9 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>protocol</code>	Protocol to be filtered.
<code>srcDest</code>	Address type that <code>ipSubnet</code> applies to.
<code>ipSubnet</code>	IP address to be filtered.
<code>prefixLength</code>	Subnet mask applied to address to be filtered.
<code>srcDestPort</code>	Port type that <code>portNumber</code> applies to.
<code>portNumber</code>	Port number to be filtered.

Trace File Replay Server Commands

The Trace File Replay Server API structure is shown below.



Trace File Replay Server Agent

The Trace File Replay Server Agent command defines a server that transmits a packet stream to a Trace File Replay client. Refer to [Trace File Replay Client Agent](#) on page 22-5 for a full description of this command. The most significant options of this command listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time .
protocol	Protocol used by the client agent.
type	Defines the agent as either a client or server.

Trace File Options

The Trace File Options command configures the list of parameters for a Trace File Replay server. Refer to [Trace File Options](#) on page 22-13 for a full description of this command. The most significant options of this command are listed below.

Option	Description
sourceClientActivity	Name of the IxLoad Trace File Replay client that the server will connect to.
traceFileName	Name and path of the pcap-format trace file that the client will use to generate the traffic stream.
useDefaultTraceFile	If true, the client uses the same trace (pcap) file as selected for the server agent.
enableFilter	If true, the filters in the client's filterList are applied to the incoming packet stream from the server.
filterList	List of filters to be applied to the specified trace (pcap) file.

Advanced Options

The Trace File Server Advanced Options command configures the advanced options for a Trace File Replay server. Refer to *Advanced Options* on page 22-14 for a full description of this command. The most significant options of this comare listed below.

Option	Description
serverNetworkList	List of IP addresses in the trace (pcap) file identified as server addresses.
useSpecifiedServerAddr	If <code>true</code> , the server scans the trace file and automatically deterwhich addresses are server addresses.

Server Network List

The Server Network List command configures the list of server IP addresses in the trace (pcap) file. Refer to *Server Network List* on page 22-15 for a full description of this command. The most significant options of this command are listed below.

Option	Description
ipSubnet	IP address identified as a server IP address.
prefixLength	Width of subnet mask applied to subnetID.



Trace File Replay Client Agent

Trace File Replay Client Agent - create a Trace File Replay client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_TraceFileReplClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_TraceFileReplClient1 agent.config
```

DESCRIPTION

A Trace File Replay client agent is added to the `activityList` object. The `activityL` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity TraceFileReplClient1 of NetTraffic Traffic1@Network1
#####
set Activity_TraceFileReplClient1 [$Traffic1_Network1 activityList.appendItem
\
    -protocolAndType                                "capturereplay Client" ]
```

```
$Activity_TraceFileReplClient1 agent.config \  
-enable true \  
-name "TraceFileReplClient1"
```

SEE ALSO

`ixNetTraffic`



Options

Options—Configures the list of parameters for a Trace File Replay client.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_TraceFileReplClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_TraceFileReplClient1 agent.pm.options.config
```

DESCRIPTION

An option is added to the list of Options using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`destinationServerActivity`

Name of the IxLoad Trace File Replay server that the client will connect to. (Default = {}).

`traceFileName`

Name and path of the pcap-format trace file that the client will use to generate the traffic stream. (Default = {}).

`replayBidirectionalTraffic`

If `true`, the client uses the same trace (pcap) file as selected for the server agent. (Default = `true`).

`enableFilter`

If `true`, the filters in the client's `filterList` are applied to the incoming packet stream from the server.

`filterList`

List of filters to be applied to the specified trace (pcap) file. This is a list of `Filter List` objects. (Default = {}).

EXAMPLE

```
$Activity_TraceFileReplClient1 agent.pm.options.config \  
-traceFileName           "" \  
-destinationServerActivity "" \  
-serverAddrString        "" \  
-enableFilter             false \  
-replayBidirectionalTraffic true
```

SEE ALSO

Trace File Replay Client Agent



Filter List

Filter List—Configures a filter to be applied to the packet stream.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_TraceFileReplClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_newClientActivity1 agent.pm.filterList.appendItem
```

DESCRIPTION

The Filter List command configures a filter that can be applied to the incoming packet stream. This command is added to the list of Trace File Replay client agent object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`protocol`

Protocol to be filtered. The choices are:

- (default) TCP
- UDP
- ICMP
- Any

`srcDest`

Address type that `ipSubnet` applies to. The choices are:

- (default) Source
- Destination
- Both
- Any

`ipSubnet`

IP address to be filtered. This is one of the IP addresses contained within the trace file.

`prefixLength`

Subnet mask applied to address to be filtered. Packets matching the subnet mask will be accepted. The choices are "1" through "32." (Default = "24").

srcDestPort

Port type that portNumber applies to. The choices are:

- Source
- (default) Destination
- Any

portNumber

Port number to be filtered.

EXAMPLE

```
$Activity_newClientActivity1 agent.pm.filterList2.appendItem \  
-id "FilterElement" \  
-ipSubnet "198.18.1.1" \  
-portNumber "33729" \  
-srcDestPort "Source" \  
-protocol "TCP" \  
-prefixLength "32" \  
-srcDest "Any"
```



Enable Filter

Enable Filter—Enables the client's list of filters to be applied to the incoming packet stream.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_TraceFileReplClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_TraceFileReplClient1 agent.pm.options.config
```

DESCRIPTION

Enable Filter causes the Trace File Replay client to use the filters configured on the client to be applied to the trace (pcap) file configured for the Trace File Replay client. The Trace File Replay client can specify its filters only if the `useDefaultTraceFile` option is disabled.

This command is added to the list of Trace File Replay client agent object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enableFilter`

If `true`, the client applies the client-side filters to the incoming packet stream. (Default = 0).

EXAMPLE

```
$Activity_TraceFileReplClient1 agent.pm.options.config \  
-enableFilter true \  

```

SEE ALSO

Options

Trace File Replay Server Agent

Trace File Replay Server Agent

SYNOPSIS

```
set ServerTraffic1_ServerNetwork1 [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$ServerTraffic1_ServerNetwork1
activityLoption...]
$Activity_newServerActivity1 agent.config
```

DESCRIPTION

A Trace File Replay server agent is added to the `activityList` object. The `activi` object is added to the `ixNetTraffic` object using the `addItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this server agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set ServerTraffic1_ServerNetwork1 [::IxLoad new ixNetTraffic]
#####
# Activity newServerActivity1 of NetTraffic ServerTraffic1@ServerNetwork1
#####
set Activity_newServerActivity1 [$ServerTraffic1_ServerNetwork1
activityList.addItem \
    -protocolAndType "capturereplay Server" ]
```



```
$Activity_newServerActivity1 agent.config \  
-enable true \  
-name "newServerActivity1"
```

SEE ALSO

`ixNetTraffic`

Trace File Options

Trace File Options—Configures the list of parameters for a Trace File Replay server.

SYNOPSIS

```
set ServerTraffic1_ServerNetwork1 [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$ServerTraffic1_ServerNetwork1
activityOption...]
$Activity_newServerActivity1 agent.pm.traceFileOptions.config
```

DESCRIPTION

An option is added to the list of Options using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`sourceClientActivity`

Name of the IxLoad Trace File Replay client that the server will connect to.

`traceFileName`

Name and path of the pcap-format trace file that the server will use to generate the traffic stream. (Default = {}).

`enableFilter`

If `true`, the filters in the client's `filterList` are applied to the incoming packet stream from the server.

`filterList`

List of filters applied to incoming packet stream. This is a list of `Filter List` objects. (Default = {}).

EXAMPLE

```
$Activity_newServerActivity1 agent.pm.traceFileOptions.config \
```



```
-enableFilter          true \  
-traceFileName        "C:/Program Files/Ixia/IxLoad/ \  
\Repository/Samples/TraceFileReplay/Captures/oracle1.cap" \  
-clientAddrString     "sym:newServerActivity1!ClientTraffic1_newClientActivity1" \  
-sourceClientActivity "ClientTraffic1_newClientActivity1"
```

SEE ALSO

Filter List

Advanced Options

Advanced Options—Configures the list of advanced options for a Trace File Replay server.

SYNOPSIS

```
set ServerTraffic1_ServerNetwork1 [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$ServerTraffic1_ServerNetwork1
activityOption...]
$Activity_newServerActivity1 agent.pm.advancedOptions.config
```

DESCRIPTION

The Advanced Options command configures the global options of a Trace File Replay server. The command is configured using the `config` subcommand of the `ixConfig` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOption` subcommands defined in the `ixConfig` command.

OPTIONS

`serverNetworkList`

List of IP addresses in the trace (pcap) file identified as server addresses. See `Server Network List`.

`useSpecifiedServerAddr`

If `true`, the server scans the trace file and automatically determines which addresses are server addresses. `IxLoad` adds the IP addresses to the `Server Network List`.

EXAMPLE

```
$Activity_newServerActivity1 agent.pm.advancedOptions.config \
    -useSpecifiedServerAddr                true
```

SEE ALSO

`Trace File Replay Server Agent`, `Server Network List`



Server Network List

Server Network List—Lists of server IP addresses contained in trace (pcap) file.

SYNOPSIS

```
set ServerTraffic1_ServerNetwork1 [::IxLoad new ixNetTraffic]
set Activity_newServerActivity1 [$ServerTraffic1_ServerNetwork1
activityOption...]
$Activity_newServerActivity1 agent.pm.advancedOptions.serverNet
```

DESCRIPTION

Server Network List is a list of IP addresses and subnet masks contained within the trace (pcap) file that are determined (either manually by the user/application or automatically by IxLoad) to be server addresses.

This command is added to the list of Trace File Replay server agent advancedOptions using the appendItem subcommand from the ixConfigSequenceContainer command.

SUBCOMMANDS

The options for this command are configured and read using the standard config, cget, and getOptions subcommands defined in the ixConfig command.

OPTIONS

ipSubnet

IP address identified as a server IP address.

prefixLength

Width of subnet mask applied to ipSubnet.

EXAMPLE

```
$Activity_newServerActivity1
agent.pm.advancedOptions.serverNetworkList.appendItem \
    -id "Network" \
    -prefixLength "32" \
    -ipSubnet "198.18.1.11"
```

SEE ALSO

Advanced Options

Statistics

For Trace File Replay client statistics, see Trace File Replay Client Statistics (on page 1682).

For Trace File Replay server statistics, see Trace File Replay Server Statistics (on page 1684).

For TCP statistics, see TCP, Run State, and Curve Segment Statistics (see "[IP, TCP, Run State, and Curve Segment Statistics](#)" on page 2096).



Trace File Replay Client Statistics

The following table describes the statistics for the Trace File Replay client.

Statistic	Description
Trace File Replay Client Initializing State	Number of users currently being initialized.
Trace File Replay Client Synchronizing State	Number of users currently awaiting synchronization.
Trace File Replay Client Active State	Number of users currently active.
TraceFileReplay Client Packets Sent	Number of packets sent by the Trace File Replay client.
TraceFileReplay Client TCP Packets Sent	Number of TCP packets sent by the Trace File Replay client.
TraceFileReplay Client UDP Packets Sent	Number of UDP packets sent by the Trace File Replay client.
TraceFileReplay Client ARP Packets Sent	Number of ARP packets sent by the Trace File Replay client.
TraceFileReplay Client ICMP Packets Sent	Number of ICMP packets sent by the Trace File Replay client.
TraceFileReplay Client Other Packets Sent	Number of packets sent by the Trace File Replay client that were not TCP, UDP, ARP, or ICMP packets.
TraceFileReplay Client Discarded Packets	Number packets discarded by the Trace File Replay client.
TraceFileReplay Client Bytes Sent	Number of bytes sent by the Trace File Replay client.
TraceFileReplay Client TCP Bytes Sent	Number of TCP bytes sent by the Trace File Replay client.
TraceFileReplay Client UDP Bytes Sent	Number of UDP bytes sent by the Trace File Replay client.
TraceFileReplay Client ARP Bytes Sent	Number of ARP bytes sent by the Trace File Replay client.
TraceFileReplay Client ICMP Bytes Sent	Number of ICMP bytes sent by the Trace File Replay client.
TraceFileReplay Client Other Bytes Sent	Number of bytes sent by the Trace File Replay client that were not TCP, UDP, ARP, or ICMP bytes.
TraceFileReplay Client Discarded Bytes	Number of bytes discarded by the Trace File Replay Client.



Trace File Replay Server Statistics

The following table describes the statistics for the Trace File Replay server.

Statistic	Description
Trace File Replay Server Initializing State	Number of users currently being initialized.
Trace File Replay Server Synchronizing State	Number of users currently awaiting synchronization.
Trace File Replay Server Active State	Number of users currently active.
TraceFileReplay Server Packets Sent	Number of packets sent by the Trace File Replay server.
TraceFileReplay Server TCP Packets Sent	Number of TCP packets sent by the Trace File Replay server.
TraceFileReplay Server UDP Packets Sent	Number of UDP packets sent by the Trace File Replay server.
TraceFileReplay Server ARP Packets Sent	Number of ARP packets sent by the Trace File Replay server.
TraceFileReplay Server ICMP Packets Sent	Number of ICMP packets sent by the Trace File Replay server.
TraceFileReplay Server Other Packets Sent	Number of packets sent by the Trace File Replay server that were not TCP, UDP, ARP, or ICMP packets.
TraceFileReplay Server Discarded Packets	Number packets discarded by the Trace File Replay server.
TraceFileReplay Server Bytes Sent	Number of bytes sent by the Trace File Replay server.
TraceFileReplay Server TCP Bytes Sent	Number of TCP bytes sent by the Trace File Replay server.
TraceFileReplay Server UDP Bytes Sent	Number of UDP bytes sent by the Trace File Replay server.
TraceFileReplay Server ARP Bytes Sent	Number of ARP bytes sent by the Trace File Replay server.
TraceFileReplay Server ICMP Bytes Sent	Number of ICMP bytes sent by the Trace File Replay server.

TraceFileReplay Server Other Bytes Sent	Number of bytes sent by the Trace File Replay server that were not TCP, UDP, ARP, or ICMP bytes.
TraceFileReplay Server Discarded Bytes	Number of bytes discarded by the Trace File Replay server.



32

VoIP H.248 Peer

The IxLoad VoIP H.248 Peer Tcl API consists of a VoIP MGW and VOIP MGC Peer agent, with separate APIs for configuring each major aspect of the agent's functionality.

There is also a TermGroup Agent with separate configuration parameters.

- When defined on a MGW activity, a TermGroup refers to terminations present on that gateway.
- When defined on a MGC activity, a TermGroup refers to terminations managed by that controller.

In this chapter:

Limitations	1688
VoIP H248 Peer API Commands	1689
VoIP H248 Peer Agent	1693
Simulated MGC	1699
Simulated MGW	1701
H248 TermGroups	1703
MGW Automatic	1705
MGC Automatic	1708
Profiles	1714
Packages	1716
Events	1718
Properties	1720
Signals	1722
Statistics	1724
H248 Settings	1726
Codec Settings	1729
Data Codecs	1731
Codecs	1733
Other Settings	1740
SDP Settings	1743
RTP Settings	1745
Audio Settings	1747
Execution Settings	1751
Scenario Settings	1753

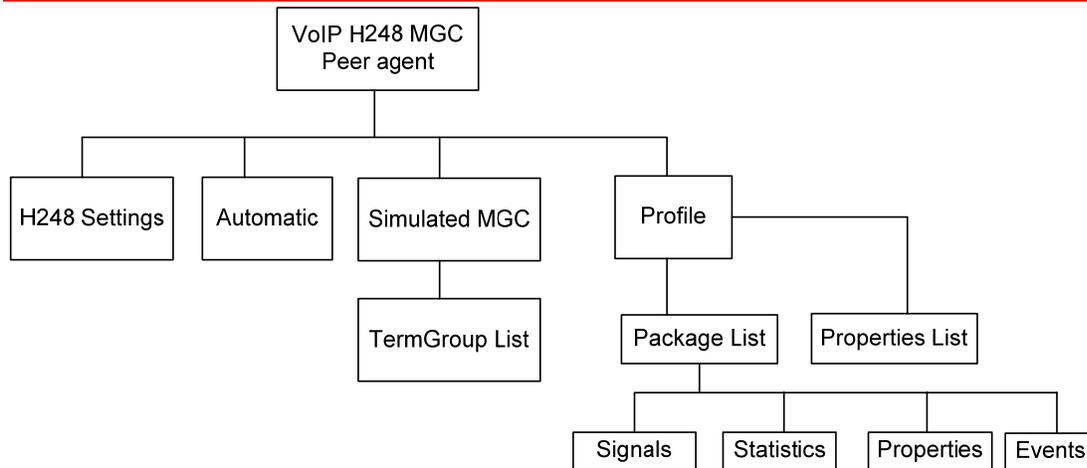
Limitations

The following restrictions and limitations of the VoIP H.248 Peer API exist:

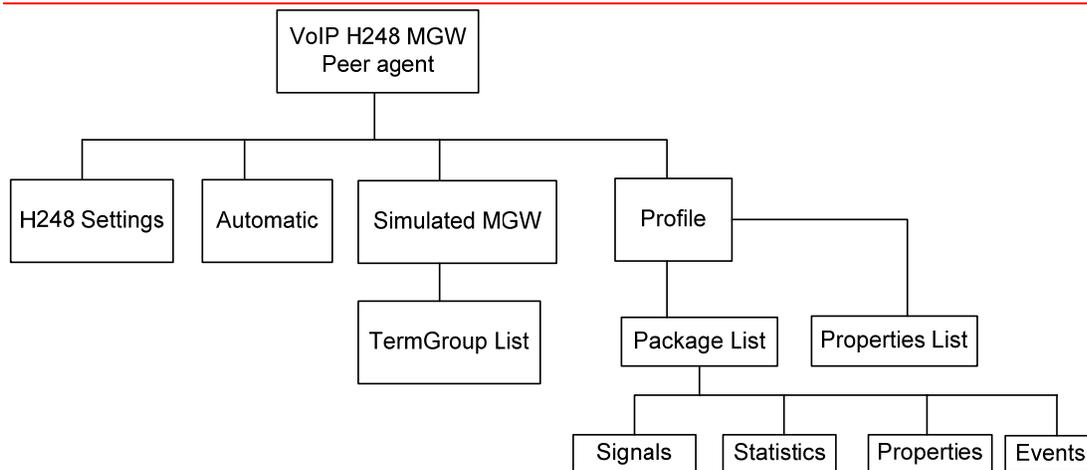
- Individual VoIP H248 script functions can not be added and edited from the Tcl API. Instead, you must add and configure the test scenario in the Scenario Editor, then save the test scenario file and pass it as an argument to the `ScenarioSettings` API class.

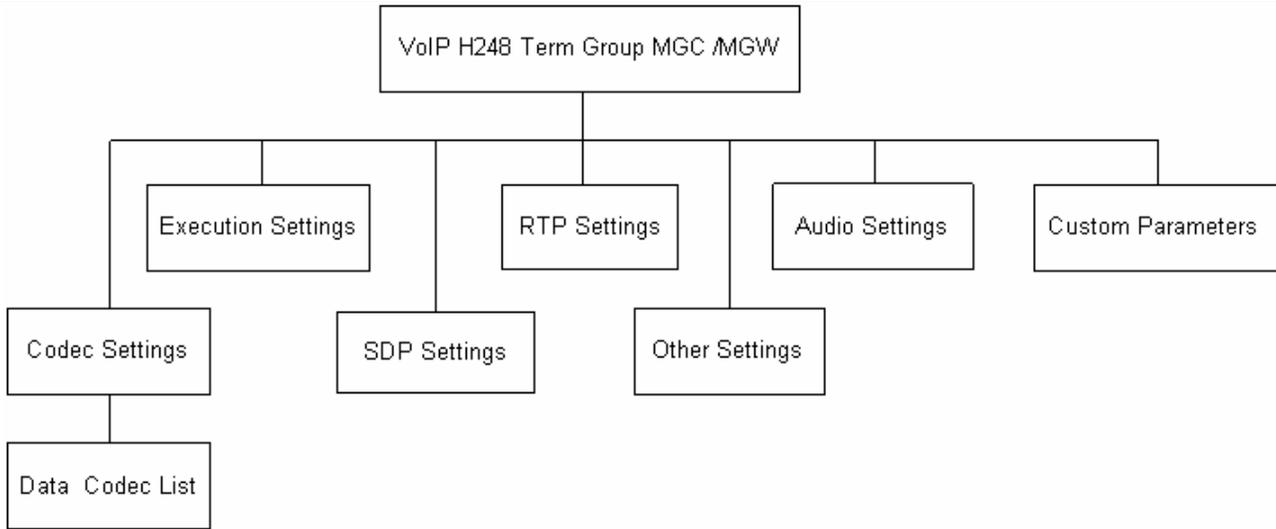
VoIP H248 Peer API Commands

The IxLoad VoIP H248 Peer API commands are organized as shown in the figures below.



VoIP H248 MGW Peer API Structure





VoIP H248 MGC/MGW Peer API Objects

The following table summarizes the VoIP H248 MGC/MGW Peer API Objects

Object	Description
VoIP H248 Peer Agent	Top-level object defining the VoIP H248 Peer activity.
H248 Settings	Configures the H.248 Settings separately for the Media Gateway or Media Controller.
Automatic	Sets the automatic functionality parameters for the MGC and MGW side.
SimulatedMGC / MGW	Simulates the source address in H.248 messages, designates the simulation type, and so on. Also contains the list of all termination groups associated with the gateway or controller.
TermGroups	Contains the list of all termination groups associated with the gateway or controller. Each termination groups has a name and two expressions to generate termination names. When a new TermGroup is added, a new activity is added in the same NetTraffic.
Profiles	A collection of packages where each package is a collection of events, signals, statistics, properties, and procedures. During registration, an MGW declares a supported profile and MGC sends audit commands to find the packages that are supported by a particular profile. The profile selected in Profiles depends on the SimulatedMGW type declared in <i>SimulatedMGW/MGC</i> .
Packages	A collection of events, signals, statistics, properties, and procedures.

Object	Description
Properties	H.248 has two basic components: Terminations and Contexts. Terminations have properties, which can be inspected and modified by the MGC.
Signals	Represents the signals of a transmission.
Statistics	Represents the statistics available for MGC and MGW.
Events	Represents the events of a transmission.

VoIP H248 TermGroup Peer API Objects

The following table summarizes the VoIP H248 TermGroup API Objects

Object	Description
VoIP H248 MGC/MGW TermGroup Agent	Top-level object defining the VoIP H248 MGC/MGW TermGroup agent activity.
Scenario Settings	Selects the Test Scenario file; corresponds to the Scenario Settings GUI tab.
Codec Settings	List of <code>Data Codecs</code> and <code>Codecs</code> objects.
Data Codecs	Data codec with parameters.
Codecs	Audio codec with parameters.
Other Settings	VoIP H323 Peer miscellaneous parameters; corresponds to the Other Settings GUI tab.
SDP Settings	H.248 uses SDP for specification and negotiation of media capabilities of GW terminations. SDP information is sent using a Stream descriptor that specifies as a single bi-directional media stream.
RTP Settings	RTP transport configuration; corresponds to the RTP Settings GUI tab.
Audio Settings	Audio settings; corresponds to the Audio GUI tab.
Custom Activity Link Settings, CustomParameters	BHCA objective configuration; corresponds to the Custom Parameters GUI tab.
Execution Settings	Run-time test configuration; corresponds to the Execution Settings GUI tab.



VoIP H248 Peer Agent

VoIP H248MGW or H248MGC Peer Agent

SYNOPSIS

```
set Activity_H248MGC1 [ $Traffic1_Network1 activityList.appendItem \  
-protocolAndType "H248MGC Peer" ]
```

DESCRIPTION

A VoIP H.248 Peer agent is added to the `agentList` option of the `ixConfig` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. See the following example:

```
set Activity_H248MGC1 [ $Traffic1_Network1 activityList.appendItem \  
-protocolAndType "H248MGC Peer" ]
```

```
$Activity_H248MGC1 config \  
-enable 1 \  
-name "H248MGC1"
```

```
$Activity_H248MGC1 agent.config \  
-enable 1 \  
-name "H248MGC1" \  
-uniqueID 1
```

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands. For example, the first agent uses an index of 0 and its name may be modified by:

```
$Activity_H248MGC1 agent(0).config -name "H248MGC Peer new"
```

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

`uniqueID`

The unique ID of this object. (Default = 1)

STATISTICS

The available H248/MEGACO statistics are listed in below:

Statistic	Description
H.248 MGC Transport	
Bytes Sent	The number of bytes sent by MGC.
Bytes Received	The number of bytes received by MGC.
Messages Sent	The number of messages sent by MGC.
Messages Received	The number of messages received by MGC.
Avg Sent/Received Message Size	The average sent/received messages size by MGC.
H.248 MGC Transactions	
Transaction Requests Sent	The number of transaction requests sent by MGC.
Transaction Requests Received	The number of transaction requests received by MGC.
Transaction Replies Sent	The number of transaction replies sent by MGC.
Transaction Replies Received	The number of transaction replies received by MGC.
Transaction Pending Sent	The number of 'Transaction pending' responses sent by MGC.
Transaction Pending Received	The number of 'Transaction pending' responses received by MGC.
Transaction Response Ack Sent	The number of Ack transaction responses sent by MGC.
Transaction Response Ack Received	The number of Ack transaction responses received by MGC.
H.248 MGC Retransmissions	
Retransmitted Transaction Requests Sent	The number of retransmitted transaction requests sent.
Retransmitted Transaction Requests Received	The number of retransmitted transaction requests received.
Retransmitted Transaction Replies Sent	The number of retransmitted transaction replies sent.
Retransmitted Transaction Replies Received	The number of retransmitted transaction replies received.
H.248 MGC Commands	
Add command requests sent	The number of Add command requests sent by MGC.
Add command replies received	The number of Add command replies received by MGC.

Statistic	Description
Modify command requests sent	The number of Modify command requests sent by MGC.
Modify command replies received	The number of Modify command replies received by MGC.
Move command requests sent	The number of Move command requests sent by MGC.
Move command replies received	The number of Move command replies received by MGC.
Subtract command requests sent	The number of Move command requests sent by MGC.
Subtract command replies received	The number of Move command replies received by MGC.
AuditValue command requests sent	The number of AuditValue command requests sent by MGC.
AuditValue command replies received	The number of AuditValue command replies received by MGC.
AuditCapability command requests sent	The number of AuditCapability command requests sent by MGC.
AuditCapability command replies received	The number of AuditCapability command replies received by MGC.
ServiceChange command requests sent	The number of ServiceChange command requests sent by MGC.
ServiceChange command requests received	The number of ServiceChange command requests received by MGC.
ServiceChange command replies sent	The number of ServiceChange command replies sent by MGC.
ServiceChange command replies received	The number of ServiceChange command replies received by MGC.
Notify command requests received	The number of Notify command requests received by MGC.
Notify command replies sent	The number of Notify command replies sent by MGC.
H.248 MGC Protocol Errors	
4xx Errors	The number of 4xx error messages sent and received by MGC.
5xx Errors	The number of 5xx error messages sent and received by MGC.
H.248 MGC Errors	
Transport Errors	The number of transport protocol errors.
SDP Errors	The number of SDP errors.
Parser Errors	The number of parser errors.



Statistic	Description
H.248 MGC Received Requests/Replies	
Transactions Not Matched	The number of transactions not matched.
Transactions Matched	The number of transactions matched.
Discarded Transactions	The number of discarded transactions.
Processed Transactions	The number of processed transactions.
Auto Processed Transactions	The number of automatically processed transactions.
H.248 MGW Transport	
Bytes Sent	The number of bytes sent by MGW.
Bytes Received	The number of bytes received by MGW.
Messages Sent	The number of messages sent by MGW.
Messages Received	The number of messages received by MGW.
Avg Sent/Received Message Size	The average sent/received messages size by MGW.
H.248 MGW Transactions	
Transaction Requests Sent	The number of transaction requests sent by MGW.
Transaction Requests Received	The number of transaction requests received by MGW.
Transaction Replies Sent	The number of transaction replies sent by MGW.
Transaction Replies Received	The number of transaction replies received by MGW.
Transaction Pending Sent	The number of 'Transaction pending' responses sent by MGW.
Transaction Pending Received	The number of 'Transaction pending' responses received by MGW.
Transaction Response Ack Sent	The number of Ack transaction responses sent by MGW.
Transaction Response Ack Received	The number of Ack transaction responses received by MGW.
H.248 MGW Retransmissions	
Retransmitted Transaction Requests Sent	The number of retransmitted transaction requests sent.
Retransmitted Transaction Requests Received	The number of retransmitted transaction requests received.
Retransmitted Transaction Replies Sent	The number of retransmitted transaction replies sent.
Retransmitted Transaction Replies Received	The number of retransmitted transaction replies received.

Statistic	Description
H.248 MGW Commands	
Add command requests received	The number of Add command requests received by MGW.
Add command replies sent	The number of Add command replies sent by MGW.
Modify command requests received	The number of Modify command requests received by MGW.
Modify command replies sent	The number of Modify command replies sent by MGW.
Move command requests received	The number of Move command requests received by MGW.
Move command replies sent	The number of Move command replies sent by MGW.
Subtract command requests received	The number of Move command requests received by MGW.
Subtract command replies sent	The number of Move command replies sent by MGW.
AuditValue command requests received	The number of AuditValue command requests received by MGW.
AuditValue command replies sent	The number of AuditValue command replies sent by MGW.
AuditCapability command requests received	The number of AuditCapability command requests received by MGW.
AuditCapability command replies sent	The number of AuditCapability command replies sent by MGW.
ServiceChange command requests sent	The number of ServiceChange command requests sent by MGW.
ServiceChange command requests received	The number of ServiceChange command requests received by MGW.
ServiceChange command replies sent	The number of ServiceChange command replies sent by MGW.
ServiceChange command replies received	The number of ServiceChange command replies received by MGW.
Notify command requests sent	The number of Notify command requests received by MGW.
Notify command replies received	The number of Notify command replies sent by MGW.
H.248 MGW Protocol Errors	
4xx Errors	The number of 4xx error messages sent and received by MGW.
5xx Errors	The number of 5xx error messages sent and received by MGW.
H.248 MGW Errors	
Transport Errors	The number of transport protocol errors.



Statistic	Description	
SDP Errors	The number of SDP errors.	
Parser Errors	The number of parser errors.	
H.248 MGW Received Requests/Replies		
Transactions Not Matched	The number of transactions not matched.	
Transactions Matched	The number of transactions matched.	
Discarded Transactions	The number of discarded transactions.	
Processed Transactions	The number of processed transactions.	
Auto Processed Transactions	The number of automatically processed transactions.	
H.248 Loop Rate		
Loops-per-second	The per polling interval loops-per-second value.	Global

EXAMPLE

```
#####
# Activity H248MGC1 of NetTraffic Traffic1@Network1
#####
set Activity_H248MGC1 [$Traffic1_Network1 activityList.appendItem \
-protocolAndType           "H248MGC Peer" ]

$Activity_H248MGC1 config \
-enable                     1 \
-name                       "H248MGC1"

$Activity_H248MGC1 agent.config \
-enable                     1 \
-name                       "H248MGC1" \
-uniqueID                   1
```

SEE ALSO

ixConfig

Simulated MGC

VoIP H248 Simulated MGC settings

SYNOPSIS

```
set Activity_H248MGC1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
  
$Activity_H248MGC1 agent.config  
  
$Activity_H248MGC1 agent.pm.simulatedMGC.config
```

DESCRIPTION

Simulates the source address in H.248 messages, designates the simulation type, and so on. Also contains the list of all termination groups associated with the gateway or controller.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`sourcePort`

Indicates the port number of the source address. Default = "2944"

`mgcName`

Indicates the device name or DNS name. It is not used when MID format is IP Address or IP Address:port.

NOTE: Sequence generator expressions are also supported, for example, MEGACOCA|00-|.

`controlledGWType`

Indicates the gateway types. The types are:

- Border Gateway (IP2IP)
- Trunking Gateway (PSTN2IP)
- Residential Gateway (PSTN2IP)
- Access Gateway (PSTN2IP)



mid

Indicates the format of the source address in H.248 messages. The options are:

- IP Address
- IP Address:port
- Device Name
- MGC DNS Name
- MGC DNS Name:port

EXAMPLE

```
set Activity_H248MGC1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType "H248MGC Peer" ]  
  
$Activity_H248MGC1 config \  
-enable 1 \  
-name "H248MGC1"  
  
$Activity_H248MGC1 agent.config \  
-enable 1 \  
-name "H248MGC1" \  
-uniqueID 1  
  
$Activity_H248MGC1 agent.pm.simulatedMGC.config \  
-sourcePort "2944" \  
-mgcName "MEGACOCA\[00-\]" \  
-controlledGWType 1 \  
-mid 1  
  
$Activity_H248MGC1 agent.pm.simulatedMGC.termGroups.clear
```

SEE ALSO

VoIP H248 Peer Agent

Simulated MGW

VoIP H248 Simulated MGW settings

SYNOPSIS

```
set Activity_H248MGW1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
  
$Activity_H248MGW1 agent.config  
  
$Activity_H248MGW1 agent.pm.simulatedMGW.config
```

DESCRIPTION

Simulates the source address in H.248 messages, designates the simulation type, and so on. Also contains the list of all termination groups associated with the gateway.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`sourcePort`

Indicates the port number of the source address. Default = "2944"

`gwName`

Indicates the device name or DNS name. It is not used when MID format is IP Address or IP Address:port.

NOTE: Sequence generator expressions are also supported, for example, MEGACOCA|00-|.

`simulatedType`

Indicates the gateway types. The types are:

- Border Gateway (IP2IP)
- Trunking Gateway (PSTN2IP)
- Residential Gateway (PSTN2IP)
- Access Gateway (PSTN2IP)



dest

The address of the controlling MGC, specified as an activity name or an IP address, followed by a port number.

mid

Indicates the format of the source address in H.248 messages. The format options are:

- IP Address
- IP Address:port
- Device Name
- GW DNS Name
- GW DNS Name:port

EXAMPLE

```
$Activity_H248MGW1 agent.pm.simulatedMGW.config \  
-sourcePort          "2944" \  
-gwName              "MEGACOGW|00-|" \  
-simulatedType       1 \  
-dest                "Traffic1_H248MGC1:2944" \  
-mid                 1  
  
$Activity_H248MGW1 agent.pm.simulatedMGW.termGroups.clear
```

SEE ALSO

VoIP H248 Peer Agent

H248 TermGroups

VoIP H248 MGW/MGC TermGroup settings

SYNOPSIS

H248 MGW TermGroup

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
$Activity_H248MGW1 agent.config \  
$Activity_H248MGW1 agent.pm.simulatedMGW.termGroups.appendItem\  

```

H248 MGC TermGroup

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
$Activity_H248MGC1 agent.config \  
$Activity_H248MGC1 agent.pm.simulatedMGC.termGroups.appendItem\  

```

DESCRIPTION

The list of all termination groups associated with the selected gateway or controller type.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`enabled`

If true, the term group is enabled. Default=false

`id`

Indicates the term group identification ID. Default=TermGroup



name

Indicates the name of the term group. For example: "H248TermGroupMGC1" \

physicalId

The physical identification of the term group. If a message with termination name \$ and without a physical name is received, the message can not be handled.

mgw

Indicates the media gateway.

rootTermination

Enables a request that is addressed to the ROOT termination. The request may be processed only by an user from a Termination Group marked as Root in the SimulatedMGC/MGW. Default = 0

rtpId1

The RTP termination ID.

rtpId2

The second RTP termination ID.

EXAMPLE

```
$Activity_H248MGC1 agent.pm.simulatedMGC.termGroups.appendItem \  
-id "TermGroup" \  
-name "H248TermGroupMGC1" \  
-physicalId "tdm/s_0/e1_{000-}/{00-29}" \  
-enabled true \  
-mgw "Traffic2_H248MGW1:2944" \  
-rootTermination 0 \  
-rtpId1 "Ephemeral/0/0/[00000-]" \  
-rtpId2 ""
```

SEE ALSO

Simulated MGW, Simulated MGC

MGW Automatic

VoIP H248 MGW Automatic settings

SYNOPSIS

H248 MGW Automatic

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ] \  
$Activity_H248MGW1 agent.config \  
$Activity_H248MGW1 agent.pm.automatic.config \  

```

DESCRIPTION

Automatic Settings specifies the automatic functionality parameters for the MGW side.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`autoReplyServiceChange`

If `true`, enables auto reply for Service Change. In this condition, Service Change requests are not dispatched to TermGroup users. `Default=false`.

`autoReplyAuditRequests`

If `true`, auto reply for Audit requests are not dispatched to TermGroup users.

Note: If a TermGroup associated with this MGW has `WaitAuditVal` or `WaitAuditCap` in scenario, the functions end with a Timeout status.

`sendTransAck`

If `true`, enables sending of Transaction Acknowledgement. In the enabled state, Ack is sent after each reply is received, and a Transaction Ack is expected after each reply sent. `Default=false`.

`sendTransPend`



If `true`, enables sending of `TransactionPending` reply. `TransactionPending` is sent when a `TransactionRequest` is received. The request is a retransmission. `Default=false`.

`autoReplyModifyOnRoot`

If `true`, `Modify` requests with `Termination ROOT` are not dispatched to `TermGroup` users.

Note: If a `TermGroup` associated with `ROOT` on this `MGW` has `WaitModify` in scenario, the function ends with a `Timeout` status.

`sendModifyOnRoot`

If `true`, `Modify` requests with `Termination ROOT` are not dispatched to `TermGroup` users. `Default=false`.

`enableRetransmissions`

If `true`, enables retransmissions of messages for which a response has not been received. `Default=false`

`maxRetransmissions`

When `enableRetransmissions` is configured `true`, this is the maximum number of retransmissions. `Default=1`

`retransmissionInterval`

When `enableRetransmissions` is configured `true`, this is the time in milliseconds for the first retransmission. `Default=10`

`commonDigitMap`

If `true`, the Gateway uses a specified default digit map. `Default=false`.

`digitMapName`

When `commonDigitMap` is `true`, this specifies the name of the default digit map. `Default="dgmap"`

`digitMapValue`

When `commonDigitMap` is `true`, this specifies the value of the default digit map. `Default="1234567890"`

`startWithRestart`

If true, the simulated MGW automatically registers with the MGC.

Note: Retransmissions for each transaction are not counted.

retryCount

Indicates the number of transactions with ServiceChange(Restart) generated.

Note: Retransmissions for each transactions are not counted.

timeoutBetweenRetries

Indicates the time between two transactions with ServiceChange(Restart) generated.

maxInactivityTime

The maximum inactivity time, after which an Inactivity event is generated.

EXAMPLE

```
$Activity_H248MGW1 agent.pm.automatic.config \  
-enableRetransmissions           false \  
-_gbCommonDigitMap              false \  
-maxInactivityTime               0 \  
-startWithRestart                true \  
-sendTransPend                   false \  
-commonDigitMap                  false \  
-retransmissionInterval          10 \  
-autoReplyToModifyOnRoot         true \  
-digitMapName                     " " \  
-digitMapValue                    " " \  
-autoReplyServiceChange          false \  
-sendTransAck                     false \  
-autoReplyAuditRequests          false \  
-retryCount                       5 \  
-timeoutBetweenRetries           5 \  
-maxRetransmissions              1
```

SEE ALSO

VoIP H248 Peer Agent



MGC Automatic

VoIP H248 MGC Automatic settings

SYNOPSIS

```
H248 MGC Automatic

set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType                "H248MGC Peer" ]

$Activity_H248MGC1 agent.config \

$Activity_H248MGC1 agent.pm.automatic.config \
```

DESCRIPTION

Automatic Settings specifies the automatic functionality parameters for the MGC side.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`autoReplyService`
Change

If `true`, enables auto reply for Service Change. In this condition, Service Change requests are not dispatched to TermGroup users. Default=false.

`autoReplyNotify`

If `true`, enables auto reply for Notify. Default=false.

`sendModifyOnRoot`

If `true`, Modify requests with Termination ROOT are not dispatched to TermGroup users. Default=false

`sendTransAck`

If `true`, enables sending of Transaction Acknowledgement. In the enabled state, Ack is sent after each reply is received, and a Transaction Ack is expected after each reply sent. Default=false.

`sendTransPend`

If `true`, enables sending of `TransactionPending` reply. `TransactionPending` is sent when a `TransactionRequest` is received. The request is a retransmission. `Default=false`.

`enableRetransmissions`

If `true`, enables retransmissions of messages for which a response has not been received. `Default=false`

`retransmissionInterval`

If `enableRetransmissions` is `true`, this specifies the time in milliseconds for the first retransmission. `Default=10`

`maxRetransmissions`

If `enableRetransmissions` is `true`, this specifies the maximum number of retransmissions. `Default=1`

`waitRestart`

If `true`, enables the restart procedure. `Default=true`.

`timeoutForRestart`

If `waitRestart` is `true`, this indicates the time before restart. `Default=0`

`auditValue`

If `true`, sends an `AuditValue` request with the specified context, termination, and descriptors specified. `Default=false`.

`auditContextVal`

When `auditValue` is `true`, this specifies the context ID to which the `AuditValue` request is sent. `Default="-"`.

`auditTerminationVal`

When `auditValue` is `true`, this specifies the termination ID to which the `AuditValue` request is sent. `Default="ROOT"`

`digitMapVal`

If `true`, includes the descriptor in the request. `Default=false`



eventsVal

If true, includes the descriptor in the request. Default=false

eventBufferVal

If true, includes the descriptor in the request. Default=false

mediaVal

If true, includes the descriptor in the request. Default=false

modemVal

If true, includes the descriptor in the request. Default=false

multiplexerVal

If true, includes the descriptor in the request. Default=false

observedEventsVal

If true, includes the descriptor in the request. Default=false

packagesVal

If true, includes the descriptor in the request. Default=true

statisticsVal

If true, includes the descriptor in the request. Default=false

signalsVal

If true, includes the descriptor in the request. Default=false

auditCapabilities

If true, sends an AuditCapabilities request with context, termination, and descriptors as specified.
Default=false

auditContextCap

When auditCapabilities is true, this specifies the context ID to which the AuditCapabilities request is sent.
Default="-"

auditTerminationCap

When `auditCapabilities` is `true`, this specifies the termination ID to which the `AuditCapabilities` request is sent. Default="ROOT"

digitMapVal

If `true`, includes the descriptor in the request. Default=`false`.

eventsCap

If `true`, includes the descriptor in the request. Default=`false`.

eventBufferCap

If `true`, includes the descriptor in the request. Default=`false`

mediaCap

If `true`, includes the descriptor in the request. Default=`false`

modemCap

If `true`, includes the descriptor in the request. Default=`false`

multiplexerCap

If `true`, includes the descriptor in the request. Default=`false`

observedEventsCap

If `true`, includes the descriptor in the request. Default=`false`

statisticsCap

If `true`, includes the descriptor in the request. Default=`false`

signalsCap

If `true`, includes the descriptor in the request. Default=`false`.

digitMapPerMGW

If `true`, sets a default `DigitMap` for the gateway. Default=`false`.



`digitMapName`

When `digitMapPerGW` is `true`, this specifies the name of the default Digit Map. Default="dgmap".

`digitMapValue`

When `digitMapPerGW` is `true`, this specifies the value of the default digit map. Default="1234567890".

`enableKeepAlive`

If `true`, activates the MGC keepalive mechanism by sending an inactivity timeout parameter to the GW. Default=`false`.

`maxInactivityTime`

If the Inactivity Timer Package is selected and the `enableKeepAlive` option is enabled, the configured timeout value value is sent to the GW. Default=10000.

EXAMPLE

```
$Activity_H248MGC1 agent.pm.automatic.config \  
-enableKeepAlive           false \  
-signalsCap                 false \  
-eventBufferVal            false \  
-sendTransPend             false \  
-modemVal                   false \  
-digitMapName               "dgmap" \  
-autoReplyNotify           false \  
-multiplexerVal            false \  
-eventsVal                  false \  
-auditTerminationCap       "ROOT" \  
-timeoutForRestart         0 \  
-packagesVal               true \  
-mediaCap                   false \  
-statisticsCap             false \  
-autoReplyServiceChange    false \  
-auditTerminationVal       "ROOT" \  
-auditCapabilities         false \  
-sendTransAck              false \  
-observedEventsCap        false \  
-maxRetransmissions        1 \  
-auditValue                 false \  
-digitMapPerMGW            false \  
-maxInactivityTime         10000 \  
-signalsVal                 false \  
-auditContextVal           "-" \  
-statisticsVal             false \  
-digitMapValue             "1234567890" \  
-eventsCap                  false \  
-sendModifyOnRoot          false \  
-enableRetransmissions     false \  
-mediaVal                   false \  
-retransmissionInterval    10 \  
-modemCap                   false \  
-eventBufferCap            false \  
-observedEventsVal        false \  
-digitMapVal               false \  
-waitRestart               true \  
-auditContextCap           "-" \  
-multiplexerCap            false
```

SEE ALSO

VoIP H248 Peer Agent

Profiles

VoIP H248 MGC/MGW Profiles Settings

SYNOPSIS

MGC Profile

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
  
$Activity_H248MGC1 agent.config \  
$Activity_H248MGC1 agent.pm.profiles.config \  

```

MGW Profile

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
  
$Activity_H248MGW1 agent.config \  
$Activity_H248MGW1 agent.pm.profiles.config \  

```

DESCRIPTION

A collection of packages where each package is a collection of events, signals, statistics, properties, and procedures.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`profile`

The name of the profile. Default="ETSI_TGW/1"

EXAMPLE

```
$Activity_H248MGC1 agent.pm.profiles.config \  
-profile                "ETSI_TGW/1"
```

SEE ALSO

VoIP H248 Peer Agent

Packages

VoIP H248 MGC/MGW Packages

SYNOPSIS

MGC Packages

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
  
$Activity_H248MGC1 agent.config \  
$Activity_H248MGC1 agent.pm.profiles.config \  
$Activity_H248MGC1 agent.pm.profiles.packages.appendItem
```

MGW Packages

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
  
$Activity_H248MGW1 agent.config \  
$Activity_H248MGW1 agent.pm.profiles.config \  
$Activity_H248MGW1 agent.pm.profiles.packages.appendItem
```

DESCRIPTION

A collection of events, signals, statistics, properties, and procedures.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`id`

Shows the package type and package description of the supported packages for a particular profile type.
Default="Package"

`supported`

Indicates the packages supported by the selected profile.

NOTE: The is editable when a custom profile is selected.

hexid

Indicates the hexadecimal id. Default=1

version

Indicates the protocol version. Allowed values are 1, 2, or 3. Default=2

EXAMPLE

```
$Activity_H248MGC1 agent.pm.profiles.packages.appendItem \  
-id "Package" \  
-supported 0 \  
-hexid 1 \  
-version 2
```

SEE ALSO

Profiles

Events

VoIP H248 MGC/MGW Events

SYNOPSIS

MGC Packages

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
  
$Activity_H248MGC1 agent.config \  
$Activity_H248MGC1 agent.pm.profiles.config \  
$Activity_H248MGC1 agent.pm.profiles.packages.appendItem  
$Activity_H248MGC1 agent.pm.profiles.packages(0).events.appendItem \  

```

MGW Packages

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
  
$Activity_H248MGW1 agent.config \  
$Activity_H248MGW1 agent.pm.profiles.config \  
$Activity_H248MGW1 agent.pm.profiles.packages.appendItem  
$Activity_H248MGW1 agent.pm.profiles.packages(0).events.appendItem \  

```

DESCRIPTION

Configures a collection of events.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`id`

Indicates the identification for events. Default="CID"

hexid

Indicates the hexadecimal id. Default=1

EXAMPLE

```
$Activity_H248MGC1 agent.pm.profiles.packages(1).events.appendItem \  
-id "CID" \  
-hexid 1
```

SEE ALSO

Packages

Properties

VoIP H248 MGC/MGW Properties

SYNOPSIS

MGC Packages

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
  
$Activity_H248MGC1 agent.config \  
$Activity_H248MGC1 agent.pm.profiles.config \  
$Activity_H248MGC1 agent.pm.profiles.packages.appendItem  
$Activity_H248MGC1 agent.pm.profiles.packages(2).properties.appendItem \  

```

MGW Packages

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
  
$Activity_H248MGW1 agent.config \  
$Activity_H248MGW1 agent.pm.profiles.config \  
$Activity_H248MGW1 agent.pm.profiles.packages.appendItem  
$Activity_H248MGW1 agent.pm.profiles.packages(2).properties.appendItem \  

```

DESCRIPTION

Configures a collection of properties.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`id`

Indicates the identification for properties. Default="CID"

hexid

Indicates the hexadecimal id. Default=1

EXAMPLE

```
$Activity_H248MGC1 agent.pm.profiles.packages(2).properties.appendItem \  
-id "CID" \  
-hexid 1
```

SEE ALSO

Packages

Signals

VoIP H248 MGC/MGW Signals

SYNOPSIS

MGC Packages

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
  
$Activity_H248MGC1 agent.config \  
$Activity_H248MGC1 agent.pm.profiles.config \  
$Activity_H248MGC1 agent.pm.profiles.packages.appendItem  
$Activity_H248MGC1 agent.pm.profiles.packages(5).signals.appendItem \  

```

MGW Packages

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
  
$Activity_H248MGW1 agent.config \  
$Activity_H248MGW1 agent.pm.profiles.config \  
$Activity_H248MGW1 agent.pm.profiles.packages.appendItem  
$Activity_H248MGW1 agent.pm.profiles.packages(5).signals.appendItem \  

```

DESCRIPTION

Configures a collection of signals.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`id`

Indicates the identification for signals. Default="CID"

hexid

Indicates the hexadecimal id. Default=1

EXAMPLE

```
$Activity_H248MGC1 agent.pm.profiles.packages(5).signals.appendItem \  
-id "CID" \  
-hexid 1
```

SEE ALSO

Packages

Statistics

VoIP H248 MGC/MGW Statistics

SYNOPSIS

MGC Packages

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
  
$Activity_H248MGC1 agent.config \  
$Activity_H248MGC1 agent.pm.profiles.config \  
$Activity_H248MGC1 agent.pm.profiles.packages.appendItem  
$Activity_H248MGC1 agent.pm.profiles.packages(16).statistics.appendItem \  

```

MGW Packages

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
  
$Activity_H248MGW1 agent.config \  
$Activity_H248MGW1 agent.pm.profiles.config \  
$Activity_H248MGW1 agent.pm.profiles.packages.appendItem  
$Activity_H248MGW1 agent.pm.profiles.packages(16).statistics.appendItem \  

```

DESCRIPTION

Configures a collection of statistics.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`id`

Indicates the identification for statistics. Default="CID"

hexid

Indicates the hexadecimal id. Default=1

EXAMPLE

```
$Activity_H248MGC1 agent.pm.profiles.packages(16).statistics.appendItem \  
-id "CID" \  
-hexid 5
```

SEE ALSO

Packages

H248 Settings

VoIP H248 Settings for MGW or MGC

SYNOPSIS

H248 Settings for MGW

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
$Activity_H248MGW1 agent.config  
$Activity_H248MGW1 agent.pm.h248Settings.config
```

H248 Settings for MGC

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
$Activity_H248MGC1 agent.config  
$Activity_H248MGC1 agent.pm.h248Settings.config \  

```

DESCRIPTION

H248 Settings specifies the H248 protocol settings for MGC and MGW.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`enableTos`

Enables the type of service for H248.

`transportType`

Indicates the transport type of type UDP.

ipPreference

Indicates the IP preference of IPv4 or IPv6.

tos

Indicates whether the TOS/DSCP byte setting is taken into consideration when sending SIP packets.

textEncoding

Indicates the type of text encoding:

- Compact
- Normal
- Pretty

nUdpMaxSize

Indicates the maximum size of UDP. Default=1024

protocolVersion

Indicates the versions of the protocol type. Allowed values of protocol versions are 1, 2, or 3.

encodingType

Indicates the encoding type of type text.

EXAMPLE

```
$Activity_H248MGW1 agent.pm.h248Settings.config \  
-enableTos                false \  
-transportType            0 \  
-ipPreference             0 \  
-tos                      0 \  
-textEncoding             3 \  
-nUdpMaxSize              1024 \  
-protocolVersion          3 \  
-encodingType             0  
  
$Activity_H248MGC1 agent.pm.h248Settings.config \  
-enableTos                false \  
-transportType            0 \  
-ipPreference             0 \  
-tos                      0 \  
-textEncoding             3 \  
-nUdpMaxSize              1024 \  
-protocolVersion          3 \  
-encodingType             0
```

SEE ALSO

VoIP H248 Peer Agent



Codec Settings

VoIP H248 MGC/MGW Term Group Codec settings

SYNOPSIS

VoIP H248 MGC TermGroup Codec Settings

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGC Peer" ]  
$Activity_H248MGC1 agent.config \  
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.config \  
VoIP H248 MGW TermGroup Codec Settings
```

VoIP H248 MGW TermGroup Codec Settings

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
-protocolAndType                "H248MGW Peer" ]  
$Activity_H248MGW1 agent.config \  
$Activity_H248TermGroupMGW1 agent.pm.codecSettings.config \  

```

DESCRIPTION

Codec Settings contains the list of codecs that is used by the VoIP H248 MGC/MGW Term Groups in the test. Codec Settings is a list of one or more `codec` (audio codec) objects. To add `codec` objects, use the `appendItem` command. To clear the codec settings, use the `clear` subcommand.

SUBCOMMANDS

None

OPTIONS

`codecs_number`

Indicates the codec numbers. Default=2

EXAMPLE

```
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.config \  
-codecs_number                2
```

SEE ALSO

VoIP H248 Peer Agent



Data Codecs

VoIP H248 MGC/MGW Term Group Data Codecs

SYNOPSIS

VoIP H248 MGC TermGroup Data Codec Settings

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType                "H248MGC Peer" ]
$Activity_H248MGC1 agent.config \
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.config \
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.dataCodecs.appendItem \
```

VoIP H248 MGW TermGroup Data Codec Settings

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \
    -protocolAndType                "H248MGW Peer" ]
$Activity_H248MGW1 agent.config \
$Activity_H248TermGroupMGW1 agent.pm.codecSettings.config \
$Activity_H248TermGroupMGW1 agent.pm.codecSettings.dataCodecs.appendItem \
```

DESCRIPTION

Data Codecs configures a data codec object, which is added to the `Codec Settings` list of codecs.

SUBCOMMANDS

None.

OPTIONS

`id`

Codec type. One of the following:

Codec	Description
Rtp2833Events	Named Events Payload format used for carrying DTMF digits and other line and trunk signals as events.
Rtp2833Tones	RTP Payload format that can represent tones consisting of one or more frequencies.



dPayloadType

Payload type used for RTP data packets. Default=(see table) min="96" max="127"

Codec	Default value for dPayloadType
Rtp2833Events	100
Rtp2833Tones	101

EXAMPLE

```
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.dataCodecs.clear
```

```
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.dataCodecs.appendItem \  
-id "Rtp2833Events" \  
-dPayloadType 100
```

```
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.dataCodecs.appendItem \  
-id "Rtp2833Tones" \  
-dPayloadType 101
```

SEE ALSO

Codec Settings

Codecs

VoIP H248 MGC/MGW Term Group Audio Codecs

SYNOPSIS

VoIP H248 MGC TermGroup Codecs

```
set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \  
    -protocolAndType                "H248MGC Peer" ]  
  
$Activity_H248MGC1 agent.config \  
  
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.config \  
  
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.codecs.appendItem \  

```

VoIP H248 MGW TermGroup Data Codecs

```
set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \  
    -protocolAndType                "H248MGW Peer" ]  
  
$Activity_H248MGW1 agent.config \  
  
$Activity_H248TermGroupMGW1 agent.pm.codecSettings.config \  
  
$Activity_H248TermGroupMGW1 agent.pm.codecSettings.codecs.appendItem \  

```

DESCRIPTION

Codecs configures an audio codec object, which is added to the `Codec Settings` list of codecs. To add a codec object, use the `appendItem` command.

SUBCOMMANDS

None.

OPTIONS

`id`

The codec type, which is one of the following:

Codec	Description
CodecAMR	Adaptive multi-rate codec
CodecG711u	G.711 mu-law codec
CodecG711a	G.711 A-law codec



Codec	Description
CodecG723x153	G.723.1 codec @ 5.3 kbps
CodecG723x163	G.723.1 codec @ 6.3 kbps
CodecG726x16	G.726 codec @ 16 Kbps
CodecG726x24	G.726 codec @ 24 Kbps
CodecG726x32	G.726 codec @ 32 Kbps
CodecG726x40	G.726 codec @ 40 Kbps
CodecG729A	G.729 Annex-A codec
CodeciLBC	iLBC codec

Options for CodecAMR

dPayloadIn

Incoming dynamic payload type. Default="98" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="98" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 14. Default=14.

payloadFormat

Payload format.

Value	Usage
0 (default)	Bandwidth-efficient format
1	Octet-aligned format

mode

Codec bit rate. One of the following:

Mode	Description
0 (default)	4.75 kbps
1	5.15 kbps

Mode	Description
2	5.90 kbps
3	6.70 kbps
4	7.40 kbps
5	7.95 kbps
6	10.20 kbps
7	12.20 kbps

Options for CodecG711u

dPayloadIn

Incoming dynamic payload type. Default="0" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="0" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG711a

dPayloadIn

Incoming dynamic payload type. Default="8" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="8" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG723x153

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut



Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 20. Default=20.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG726x16

dPayloadIn

Incoming dynamic payload type. Default="102" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="102" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 20, 40, 60. Default=20.

Options for CodecG726x24

dPayloadIn

Incoming dynamic payload type. Default="103" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="103" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 30, 60, 90. Default=30.

Options for CodecG726x32

dPayloadIn

Incoming dynamic payload type. Default="104" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="104" min="0" max="127".

byteOrder

Byte order.



Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 40, 80, 120. Default=40.

Options for CodecG729

dPayloadIn

Incoming dynamic payload type. Default="18" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="18" min="0" max="127".

cbxFrameSize

Bytes per frame. Must be one of the following: 10, 20, 30, 40, 50, Custom. Default=10.

customFrameSize

If `cbxFrameSize` is Custom, this option configures the custom frame size. Default="120" min="10" max="200".

EXAMPLE

```
$Activity_H248TermGroupMGC1 agent.pm.codecSettings.codecs.clear

$Activity_H248TermGroupMGC1 agent.pm.codecSettings.codecs.appendItem \
-id "CodecG711u" \
-dPayloadOut 0 \
-dPayloadIn 0 \
-frameSize 160

$Activity_H248TermGroupMGC1 agent.pm.codecSettings.codecs.appendItem \
-id "CodecG711a" \
-dPayloadOut 8 \
-dPayloadIn 8 \
-frameSize 160
```

SEE ALSO

Codec Settings

Other Settings

VoIPH248 MGC/MGW Term Group Peer Other Settings

SYNOPSIS

```
VoIP H248 MGC TermGroup Other Settings

set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType                "H248MGC Peer" ]

$Activity_H248MGC1 agent.config \

$Activity_H248TermGroupMGC1 agent.pm.otherSettings.config

VoIP H248 MGW TermGroup Other Settings

set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType                "H248MGW Peer" ]

$Activity_H248MGW1 agent.config \

$Activity_H248TermGroupMGW1 agent.pm.otherSettings.config
```

DESCRIPTION

This object configures the VoIP H248 MGC/MGW Term Group Peer activity's miscellaneous options.

SUBCOMMANDS

None.

OPTIONS

VOIP_Var0

The VOIP_Var1...VOIP_Var5 and VOIP_IPAddr1...VOIP_IPAddr5 string-type variables supporting generator expressions enable you to generate 10 series of global variables whose values are used at runtime by the simulated H.248 Term Group phones/channels. Default="".

Use the VOIP_Var1...VOIP_Var5 variables to represent phone numbers, and the VOIP_IPAddr1...VOIP_IPAddr5 to represent IP addresses.

VOIP_Var1

See VOIP_Var0.

VOIP_Var2

See VOIP_Var0.

VOIP_Var3

See VOIP_Var0.

VOIP_Var4

See VOIP_Var0.

VOIP_IPAddress0

See VOIP_Var0.

VOIP_IPAddress1

See VOIP_Var0.

VOIP_IPAddress2

See VOIP_Var0.

VOIP_IPAddress3

See VOIP_Var0.

VOIP_IPAddress4

See VOIP_Var0.

ipPreference

Type of addressing to be used on the subnet that the VOIP H248 Term Group runs on.

Value	Usage
0 (default)	IPv4
1	IPv6

EXAMPLE

```
$Activity_H248TermGroupMGC1 agent.pm.otherSettings.config \  
-VOIP_Var1                " " \  
-VOIP_Var0                " " \  
-VOIP_Var3                " " \  
-VOIP_Var2                " " \  
-VOIP_Var4                " " \  
-VOIP_IPAddress4         " " \  
-VOIP_IPAddress1         " " \  
-VOIP_IPAddress0         " " \  
-VOIP_IPAddress3         " " \  
-VOIP_IPAddress2         " "
```

SEE ALSO

VoIP H248 Peer Agent

SDP Settings

VoIPH248 MGC/MGW Term Group SDP Settings

SYNOPSIS

```
VoIP H248 MGC TermGroup SDP Settings

set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType                "H248MGC Peer" ]

$Activity_H248MGC1 agent.config \

$Activity_H248TermGroupMGC1 agent.pm.sdpSettings.config \

VoIP H248 MGW TermGroup SDP Settings

set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType                "H248MGW Peer" ]

$Activity_H248MGW1 agent.config \

$Activity_H248TermGroupMGW1 agent.pm.sdpSettings.config \
```

DESCRIPTION

H.248 uses SDP for specification and negotiation of media capabilities of GW terminations. SDP information is sent using a Stream descriptor that specifies as a single bi-directional media stream.

SUBCOMMANDS

None.

OPTIONS

replaceAutoSdpTemplate

If `true`, the auto SDP template constructed from the codec list can be overridden by editing the SDP template. Default=`false`

encodeRtpMap

If `true`, a static payload type is included in the auto SDP. Default=`false`

replaceAutoSDP

RTP Settings

VoIPH248 MGC/MGW TermGroup RTP settings

SYNOPSIS

VoIP H248 MGC TermGroup RTP Settings

```
$Activity_H248TermGroupMGC1 agent.pm.rtpSettings.config
```

VoIP H248 MGW TermGroup RTP Settings

```
$Activity_H248TermGroupMGW1 agent.pm.rtpSettings.config
```

DESCRIPTION

The RTP Settings configure the VoIPH248 MGC/MGW TermGroup RTP transport settings.

SUBCOMMANDS

None.

OPTIONS

enableRTP

If true, enables use of RTP to transport the media traffic. Default= False

rtpPort

The port used for RTP streaming. Default="10000".

enableRTCP

Enables the sending and receiving of RTCP packets.

chEnableHwAcc

If true, enables hardware acceleration for RTP traffic. Default=false.

enableAdvStatCalc

If true, enables the computation of advanced RTP statistics.

enablePerStream



Enables computation of per-stream statistics.

enableMDI

Enables computation of MDI DF and MDI MLR statistics.

enableNBExec

If `true`, all RTP functions from a scenario execute in a non-blocking mode, i.e the current function from a channel executes in the background, allowing the execution to continue on that channel with the next script function. Default= `False`.

EXAMPLE

```
$Activity_H248TermGroupMGC1 agent.pm.rtpSettings.config \  
-enableRTP true \  
-enableRTCP false \  
-enableMDI false \  
-chEnableHwAcc true \  
-chDisableHwAcc false \  
-enableAdvStatCalc false \  
-enablePerStream false \  
-rtpPort "[10000-65535,4]" \  
-enableNBExec false
```

SEE ALSO

VoIP H248 Peer Agent

Audio Settings

H.248 TermGroup audio settings

SYNOPSIS

```
$Activity_H248TermGroupMGCl agent.pm.rtpSettings.config\  
$Activity_H248TermGroupMGWl agent.pm.rtpSettings.config
```

DESCRIPTION

The Audio Settings configure the VoIPH248 TermGroup audio RTP settings.

SUBCOMMANDS

None.

OPTIONS

`enableAudio`

If selected, audio script functions are executed, otherwise they are skipped.

`audioClip`

The played audio clip file.

`playTypeAudio`

The mode in which the clip is played.

Value	Usage
0 (default)	The clip is played for clip duration or for the duration of the Talk Time parameter in the case of BHCA/CPS/LPS objectives.
1	The clip is played for a user-defined duration.

`audioDurationUnit`

The play duration unit, which can be milliseconds (0), seconds (1), minutes (2), or hours (3).

`outputLevel`

The output level of the played clip.



`enableTosRtp`

Enables use of TOS/DSCP. Use the `rtpTos` option to specify the TOS/DSCP value. `Default= False`

`rtpTosVal`

The Type of Service (TOS/DSCP) byte setting in the sent RTP packets has one of the following values:

- Best Effort (0x00): Routine service
- Class 1 (0x20): Priority service, Assured Forwarding class 1
- Class 2 (0x40): Immediate service, Assured Forwarding class 2
- Class 3 (0x60): Flash, Assured Forwarding class 3
- Class 4 (0x80): Flash-override, Assured Forwarding class 4
- Express Forwarding (0xA0): Critical-ecp
- Control (0xC0): Internet-control
- Custom: A user-specified value.

`useMOS`

Enables the computation of MOS scores. `Default= False`.

`enableAudioOWD`

If `true`, IxLoad computes the One-way Delay metric, a network measurement specifying the amount of time (in ms) that a packet has spent on the network before it was received on the destination side. `Default= False`

`useJitter`

If `true`, enables use of a jitter buffer. `Default= False`.

`jitMs`

If `useJitter` is 1, this option configures the size of the jitter buffer, in milliseconds. `Default="20"`
`min="1" max="3000"`.

`useJitComp`

If `true`, enables dynamic modification of the jitter buffer size. `Default= False`.

`jitCms`

If `useJitComp` is 1, this option configures the maximum size in of the jitter buffer, in milliseconds. `Default="1000" min="0" max="3000"`.

jitCMaxDrop

If `useJitComp` is 1, this option configures the condition - a maximum number of consecutive packets dropped - that determines the jitter buffer size to be increased.

enableQoV

If `true`, this enables QoV P.862 PESQ and P.56 QoV computation. `Default= False`.

channelTypeQoV

When `enableQoV` is `true`, this specifies the objective type as either of the following:

- Number of channels (0)
- Percentage (1)

valueQoV

When `enableQoV` is `true`, this specifies the number of channels for which PESQ and P.56 QoV metrics are computed (when `channelTypeQoV` is 0). Alternatively this represents the percentage of channels for which PESQ and P.56 QoV metrics are computed (when `channelTypeQoV` is 1).

unitsQoV

The channels selection mode, which can be any of the following:

- First channels (0)
- Last channels (1)
- Evenly-spaced channels (2)
- Random (3)

metricsQoV

When `enableQoV` is `true`, this specifies the metric that is calculated by the Zion card. Available options are:

- PESQ and P.56 (0)
- PESQ (1)
- P56 (2)

useSilence

If `true`, RTP packets containing artificial background noise are sent when no other media (DTMF, MF, real payload, and so on) is sent over the communication channel. `Default= False`.

silenceMode



If useSilence is 1, this option configures the silence mode.

Value	Usage
0	Null data encoded
1 (default)	Comfort noise.

EXAMPLE

```
$Activity_H248TermGroupMGC1 agent.pm.audioSettings.config \  
-enableAudio true \  
-audioClip "US_042.wav" \  
-playTypeAudio 0 \  
-audioDurationUnit 1 \  
-audioDuration 10 \  
-outputLevel -20 \  
-enableAudioOWD false \  
-enableTosRtp false \  
-rtpTosVal 32 \  
-useMos false \  
-useJitter false \  
-jitMs 20 \  
-useJitComp false \  
-jitCMs 1000 \  
-jitCMaxDrop 7 \  
-enableQoV false \  
-channelTypeQoV 0 \  
-valueQoV 100 \  
-unitsQoV 0 \  
-metricsQoV 0 \  
-useSilence false \  
-silenceMode 1 \  

```

SEE ALSO

Execution Settings

VoIP H248 MGC/MGW Term Group Execution Settings

SYNOPSIS

```
VoIP H248 MGC TermGroup Execution Settings

set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType                "H248MGC Peer" ]

$Activity_H248MGC1 agent.config \

$Activity_H248TermGroupMGC1 agent.pm.executionSettings.config \

VoIP H248 MGW TermGroup Execution Settings

set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType                "H248MGW Peer" ]

$Activity_H248MGW1 agent.config \

$Activity_H248TermGroupMGW1 agent.pm.executionSettings.config \
```

DESCRIPTION

This object defines the execution settings for the VoIP H248 MGC/MGW Term Group.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`gracefulRampDown`

If enabled, allows the user to exit gracefully after a session. Default=1

`loopCount`

If `loopMode` is 1, this option defines the number of loops that the test performs. Default="1".

`loopPreDelay`



Delay before first loop (ms). Default="0".

loopMode

Defines how many loops are executed for every voice channel corresponding to this activity.

Value	Description
0 (default)	Loop for the entire test duration.
1	Execute a number of loops. Specify the number of loops in loopCount.

loopMidDelay

Delay between loops (ms). Default="0".

EXAMPLE

```
$Activity_H248TermGroupMGCl agent.pm.executionSettings.config \  
-gracefulRampDown          1 \  
-loopMidDelay              0 \  
-loopPreDelay              0 \  
-loopCount                 2 \  
-loopMode                  1
```

SEE ALSO

VoIP H248 Peer Agent

Scenario Settings

VoIP H248 MGC/MGW TermGroup Scenario Settings

SYNOPSIS

```
VoIP H248 MGC TermGroup Scenario Settings

set Activity_H248MGC1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType           "H248MGC Peer" ]

$Activity_H248MGC1 agent.config \

$Activity_H248TermGroupMGC1 agent.pm.scenarioSettings.config \

VoIP H248 MGW TermGroup Scenario Settings

set Activity_H248MGW1 [$Traffic2_Network2 activityList.appendItem \
-protocolAndType           "H248MGW Peer" ]\

$Activity_H248MGW1 agent.config \

$Activity_H248TermGroupMGW1 agent.pm.scenarioSettings.config \
```

DESCRIPTION

Scenario Settings specifies the test scenario file that will be used by the Tcl script.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`scenarioFile`

The full path to the test scenario file for the activity.

`activeScenarioChannel`

Test scenario channel (0-based index) that is associated with the VoIP H248 Peer activity. Default=0



EXAMPLE

```
$Activity_H248TermGroupMGC1 agent.pm.scenarioSettings.config \  
-scenarioFile "C:\\Documents and Settings\\bmoraru\\My \  
Documents\\Load3.70\\test1.tst" \  
-activeScenarioChannel 0
```

SEE ALSO

VoIP H248 Peer Agent



33

VoIP H.323 Peer

The IxLoad VoIP H.323 Peer Tcl API consists of a VoIP H.323 Peer agent, with separate APIs for configuring each major aspect of the agent's functionality.

In this chapter:

API Overview	1757
Limitations	1758
VoIP H323 Peer API Commands	1759
VoIP H323 Peer Agent	1761
Codec Settings	1771
Codecs	1772
Data Codecs	1778
Other Settings	1780
RTP Settings	1783
Audio Settings	1785
Video Settings	1789
Alternative Capability Value Set List	1795
Capability List	1797
Custom Activity Link Settings	1799
Execution Settings	1801
Simultaneous Capability	1803
H323 Settings	1804
Simultaneous Capability Value Set List	1809
Alternative Capability List	1811
Alternative Capability	1813
Dial Plan	1814
Terminal Capability Set	1817
Simultaneous Capability List	1818
Scenario Settings	1820
Custom Parameters	1822



API Overview

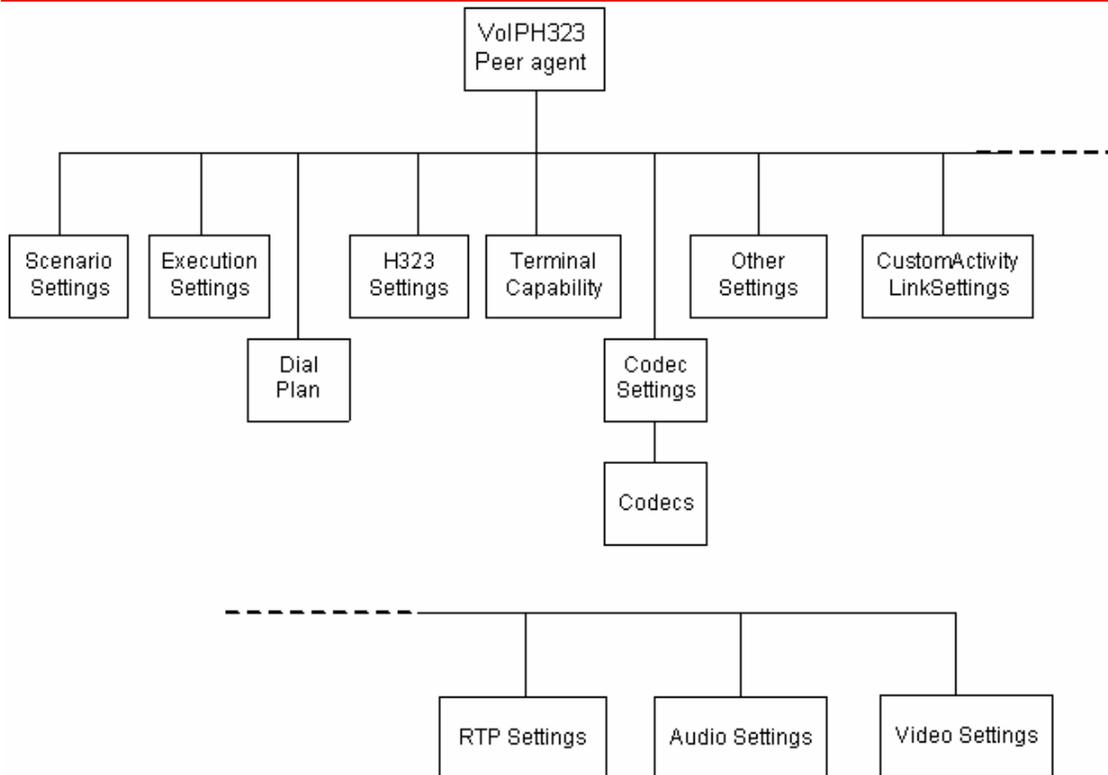
Limitations

The following restrictions and limitations of the VoIP.H323 Peer API exist:

- Individual VoIP H323 script functions can not be added and edited from the Tcl API. Instead, you must add and configure the commands in the Scenario Editor, save the test scenario file, then pass it as an argument to the `ScenarioSettings` API class.
- Implementation of the BHCA objective features relies on two classes, `CustomParameters` and `CustomActivityLinkSettings` that have to be configured using the same parameters.

VoIP H323 Peer API Commands

The IxLoad VoIP H323 Peer API commands are organized as shown in the following figure.



VoIP H323 Peer API Objects

The following table summarizes the VoIP H323 Peer API Objects

Object	Description
VoIP H323 Peer Agent	Top-level object defining the VoIP H323 Peer activity.
Scenario Settings	Selects the Test Scenario file; corresponds to the Scenario Settings GUI tab.
Codec Settings	List of <code>Codecs</code> objects.
Codecs	Audio codec with parameters.
H323 Settings	VoIP H323 Peer parameters; corresponds to the H323 Settings GUI tab.
Execution Settings	Run-time test configuration; corresponds to the Execution Settings GUI tab.
Terminal Capability	Configures the terminal capability settings.



Object	Description
Dial Plan	Configures the registration names, phone numbers, and source, destination, and transfer addresses for the channels/phones; corresponds to the Dial Plan GUI tab.
RTP Settings	RTP transport configuration; corresponds to the RTP Settings GUI tab.
Audio Settings	Audio settings; corresponds to the Audio GUI tab.
Other Settings	VoIP H323 Peer miscellaneous parameters; corresponds to the Other Settings GUI tab.
Custom Activity Link Settings, CustomParameters	BHCA objective configuration; corresponds to the Custom Parameters GUI tab.

VoIP H323 Peer Agent

VoIP H323 Peer Agent

SYNOPSIS

```
set Activity_VoIPH323Peer1 \[$Traffic1_Network1 activityList.appendItem \  
-protocolAndType "VoIPH323 Peer" ]
```

DESCRIPTION

A VoIP H.323 Peer agent is added to the `agentList` option of the `ixConfig` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. See the following example:

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType "VoIPH323 Peer" ]  
$Activity_VoIPH323Peer2 config \  
-enable true \  
-name "VoIPH323Peer1" \  
-enableConstraint false \  
-userObjectiveValue 1 \  
-constraintValue 100 \  
-userObjectiveType "channels" \  
-timeline $Timeline1 \  
  
$Activity_VoIPH323Peer1 agent.config \  
-enable true \  
-name "VoIPH323Peer1"
```

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands. For example, the first agent uses an index of 0 and its name may be modified by:

```
$Activity_VoIPH323Peer1 agent(0).config -name "VoIPH323Peer new"
```

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

`uniqueID`



The unique ID of this object. (Default = 1)

STATISTICS

The available H.323 statistics are listed below.

Statistic	Description	Per Channel/Global
VoIPH323 Channels		
Total Channels	The per polling interval total number of channels, a sum of active and non-active channels.	Global
Completed Channels	The per polling interval number of COMPLETED channels. A channel is COMPLETED if all the channel loops were COMPLETED.	Global
Warning Channels	The per polling interval number of WARNING channels. A channel is WARNING if all the channel loops were COMPLETED or WARNING and at least one loop had a WARNING result.	Global
Failed Channels	The per polling interval number of FAILED channels. A channel is FAILED if all the channel loops were COMPLETED or WARNING, and at least one loop was FAILED.	Global
Aborted Channels	The per polling interval number of ABORTED channels. A channel is ABORTED if all the channel loops were COMPLETED, WARNING, FAILED, or ABORTED and at least one loop was ABORTED.	Global
Active Channels	The per polling interval number of active channels. Active channels are the channels executing a scenario channel functions flow.	Global
VoIPH323 Loops		
Completed Channel Loops	The cumulative count of COMPLETED channel loops. A channel loop is COMPLETED if all executed script functions in the corresponding scenario channel produced SKIPPED or COMPLETED results.	Global
Warning Channel Loops	The cumulative count of WARNING channel loops. A channel loop has a WARNING result if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, or WARNING results and at least one script function had a WARNING result.	Global
Failed Channel Loops	The cumulative count of FAILED channel loops. A channel loop is FAILED if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, WARNING, or FAILED results and at least one script function had a FAILED result.	Global

Statistic	Description	Per Channel/Global
Aborted Channel Loops	The cumulative count of ABORTED channel loops. A channel loop is FAILED if all the executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, WARNING, FAILED, or ABORTED results and at least one script function had an ABORTED result.	Global
Total Channel Loops	The cumulative count of executed loops.	Global
Interloop Duration (Avg) [ms]	The time gap between loops.	Global
VoIPH323 Calls		
Attempted Calls	The number of initiated calls.	Global
Connected Calls	The number of established calls.	Global
Received Calls	The number of received incoming calls.	Global
Answered Calls	The number of answered incoming calls.	Global
End Calls Initiated	The count of initiated end call procedures, incremented whenever an Initiate EndCall (EndCall with "Wait other party to disconnect" option is disabled) command execution is started.	Global
End Calls Received	The count of received end call procedures, incremented whenever an Await EndCall(EndCall with "Wait other party to disconnect" option is enabled) command execution is started.	Global
End Calls Completed	The count of completed end call procedures.	Global
Active Calls	The number of active calls at one time. For the initiator side a call is active after having sent a SETUP message and until receiving or sending a RELEASE COMPLETE message. For the terminating side, a call is active after having received SETUP message and until sending or receiving RELEASE COMPLETE message.	Global
Busy Calls	Updated when an incoming call is received for an alias/number on which an existing call is in progress.	Global
VoIPH323 Call Rates		
Attempted calls/s, Connected Calls/s, Received Calls/s, Answered Calls/s, Rejected Calls/s, Busy Calls/s	The per polling interval rates for the above <i>VoIPH323 Call</i> statistics.	Global



Statistic	Description	Per Channel/Global
VoIPH323 Busy Hour Call Measurements		
BHCA	The per polling interval Busy Hour Call Attempts rate that represents the number of calls initiated in one hour.	Global
BHCC	The per polling interval Busy Hour Call Completions rate that represents the number of calls initiated and connected in one hour.	Global
VoIPH323 Call Times		
Call Setup Time (Avg) [ms]	The average duration between the moment a call is initiated and call is connected.	Global
Talk Time (Avg) [ms]	The average talk time (the duration between the moment the call is connected and the moment the call is disconnected by one of the parties).	Global
End Call Time (Avg) [ms]	From the time the EndCall is executed to the time it takes to tear down the call and complete the EndCall execution.	Global
Total Call Duration (Avg) [ms]	The average call duration. When referring to a single call: (Entire) Call Length = Call Setup-Time + Talk Time + Call Teardown Time.	Global
VoIPH323 Registrations		
Attempted Registrations	This statistic is updated when a RRQ is sent. Note: It is not updated when a light-weight RRQ is sent.	Global
Successful Registrations	This statistic is updated when a RFC is received for an RRQ which is not a light weight RRQ.	Global
Failed Registrations	This statistic is updated when a RRJ is received for an RRQ which is not a light weight RRQ.	Global
Attempted DeRegistrations	This statistic is updated when an URQ is sent.	Global
Successful De-Registrations	The cumulative count of successful de-registrations, incremented when the phone receives the reply <i>UnregisterConfirm</i> message from the gatekeeper.	Global
Failed De-Registrations	This statistic is updated when an URJ is received.	Global
VoIPH323 Registration Rates		
Attempted Registrations /sec	The per polling interval rate of attempted registrations.	Global
Successful Registrations /sec	The per polling interval rate of successful registrations.	Global
Attempted DeRegistrations /sec	The per-polling interval rate of attempted de-registrations.	Global
Successful DeRegistrations /sec	The per polling interval rate of successful registrations.	Global

Statistic	Description	Per Channel/Global
VoIPH323 Registration Times		
Registration Time (Avg) [ms]	The registration time from the time a RRQ is sent to the time a RCF is received. Note: This statistic is not updated for light-weight RRQ transactions.	Global
DeRegistration Time (Avg) [ms]	The de-registration time from the time an URQ is sent to the time an UCF is received.	Global
VoIPH323 Gatekeeper Discovery Requests		
VoIPH323 GK Request GRQ Transmitted	The number of Gatekeeper requests transmitted.	Global
VoIPH323 GK Confirm GCF Received	The number of Gatekeeper confirmations received.	Global
VoIPH323 GK Reject GRJ Received	The number of Gatekeeper Rejects received.	Global
VOIPH323 GRQ Timed Out	The number of sent Gatekeeper requests that timed out.	Global
VoIPH323 Gatekeeper Registration Requests		
VoIPH323 Registration Request RRQ Transmitted	The number of sent Registration Requests messages transmitted.	Global
VoIPH323 Registration Confirm RCF Received	The number of received confirmation messages.	Global
VoIPH323 Registration Reject RRJ Received	The number of received reject messages.	Global
VOIPH323 RRQ Timed Out	The number of sent request messages that timed out.	Global
VoIPH323 Gatekeeper Admission Requests		
VoIPH323 Admission Request ARQ Transmitted	The number of admission request messages transmitted.	Global
VoIPH323 Admission Confirm ACF Received	The number of admission confirmations received.	Global
VoIPH323 Admission Reject ARJ Received	The number of admission rejects received.	Global
VOIPH323 ARQ Timed Out	The number of sent admission request messages that timed out.	Global
VoIPH323 Gatekeeper Disengage Requests		
VoIPH323 Disengage Request DRQ Transmitted	The number of disengage requests transmitted.	Global



Statistic	Description	Per Channel/Global
VoIPH323 Disengage Confirm DCF Received	The number of disengage confirmation messages received.	Global
VoIPH323 Disengage Reject DRJ Received	The number of disengage reject messages received.	Global
VOIPH323 DRQ Timed Out	The number of sent request messages that timed out.	Global
VoIPH323 Disengage Request DRQ Received	The number of disengage requests received.	Global
VoIPH323 Disengage Confirm DCF Transmitted	The number of disengage confirmations received.	Global
VoIPH323 Gatekeeper Unregistration Requests		
VoIPH323 Unregister Request URQ Transmitted	The number of unregister requests transmitted.	Global
VoIPH323 Unregister Confirm UCF Received	The number of unregister confirmations received.	Global
VoIPH323 Unregister Reject URJ Received	The number of unregister reject messages received.	Global
VOIPH323 URQ Timed Out	The number of unregister messages that timed out.	Global
VoIPH323 URQ Received	The number of unregister request messages received.	
VoIPH323 UCF Transmitted	The number of unregister confirmation messages transmitted.	
VoIPH323 H225 Requests and Responses		
VoIPH323 Setup Transmitted	The number of Setup messages transmitted.	Global
VoIPH323 Setup Received	The number of Setup messages received.	Global
VoIPH323 CallProceeding Transmitted	The number of CallProceeding messages transmitted.	Global
VoIPH323 CallProceeding Received	The number of CallProceeding messages received.	Global
VoIPH323 Alerting Transmitted	The number of Alerting messages transmitted.	Global
VoIPH323 Alerting Received	The number of Alerting messages received.	Global
VoIPH323 Connect Transmitted	The number of Connect messages transmitted.	Global

Statistic	Description	Per Channel/Global
VoIPH323 Connect Received	The number of Connect messages received.	Global
VoIPH323 releaseComplete Transmitted	The number of releasecomplete messages transmitted.	Global
VoIPH323 H245 Requests and Responses		
VoIPH323 TCS Transmitted	The number of TerminalCapabilitySet messages transmitted.	Global
VoIPH323 TCS Received	The number of TerminalCapabilitySet messages received.	Global
VoIPH323 TCSAck Transmitted	The number of TerminalCapabilitySetAcknowledgement messages transmitted.	Global
VoIPH323 TCSAck Received	The number of TerminalCapabilitySetAcknowledgement messages received.	Global
VoIPH323 masterSlaveDetermination Transmitted	The number of MasterSlaveDetermination messages transmitted.	Global
VoIPH323 masterSlaveDetermination Received	The number of MasterSlaveDetermination messages received.	Global
VoIPH323 masterSlaveDeterminationAck Transmitted	The number of MasterSlaveDeterminationAcknowledgement messages transmitted.	Global
VoIPH323 masterSlaveDeterminationAck Received	The number of MasterSlaveDeterminationAcknowledgement messages received.	Global
VoIPH323 openLogicalChannel Transmitted	The number of OpenLogicalChannel messages transmitted.	Global
VoIPH323 openLogicalChannel Received	The number of OpenLogicalChannel messages received.	Global
VoIPH323 openLogicalChannelAck Transmitted	The number of OpenLogicalChannelAcknowledgement messages transmitted.	Global
VoIPH323 openLogicalChannelAck Received	The number of OpenLogicalChannelAcknowledgement messages received.	Global
VoIPH323 closeLogicalChannel Transmitted	The number of CloseLogicalChannel messages transmitted.	Global

Statistic	Description	Per Channel/Global
VoIPH323 closeLogicalChannel Received	The number of CloseLogicalChannel messages received.	Global
VoIPH323 closeLogicalChannelAck Transmitted	The number of CloseLogicalChannelAcknowledgement messages transmitted.	Global
VoIPH323 closeLogicalChannelAck Received	The number of CloseLogicalChannelAcknowledgement messages received.	Global
VoIPH323 H245 Reject Messages		
VoIPH323 TCSReject Transmitted	The number of TerminalCapabilitySet reject messages transmitted.	Global
VoIPH323 TCSReject Received	The number of TerminalCapabilitySet reject messages received.	Global
VoIPH323 masterSlaveDeterminati on- Reject Transmitted	The number of MasterSlaveDetermination reject messages transmitted.	Global
VoIPH323 masterSlaveDeterminati on- Reject Received	The number of MasterSlaveDetermination reject messages received.	Global
VoIPH323 openLogicalChannelRej ect Transmitted	The number of OpenLogicalChannel reject messages transmitted.	Global
VoIPH323 openLogicalChannelRej ect Received	The number of OpenLogicalChannel reject messages received.	Global
VoIPH323 Errors		
Trigger Errors	The total number of trigger errors.	Global
RTP Errors	<p>The total number of RTP related errors, incremented when any RTP script function is failing or exiting on the Warning or Timeout outputs. Possible causes include media sessions that have been closed by the signaling engine, or Generate DTMF/MF/Tone or Detect DTMF/MF/Tone functions that failed.</p> <p>This statistic is also incremented when the signaling engine cannot start a media session, such as when the negotiated codec or the negotiated <i>ptime</i> is unsupported.</p>	Global
Internal Errors	The total number of internal errors.	Global
Timeout Errors	The total number of script functions that have timed out.	Global

Statistic	Description	Per Channel/Global
Transport Errors	The total number of transport errors reported during I/O operation.	Global
VoIPH323 Specific Errors		
Parser Error	The total number of parser errors encountered during parsing.	Global
Call Flow Errors	The total number of H323 call flow errors.	Global
VoIPH323 Throughput		
VoIPH323 Bytes Transmitted/sec	The total number of bytes transmitted in H.323 call signaling and call control packets (excluding RTP packets).	Global
VoIPH323 Bytes Received/sec	The total number of bytes received in H.323 call signaling and call control packets (excluding RTP packets).	Global
VoIPH323 Bytes Transmitted And Received/sec	The total number of bytes transmitted and received in H.323 call signaling and call control packets (excluding RTP packets).	Global

EXAMPLE

```

set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \
-protocolAndType                "VoIPH323 Peer" ]

set Timeline1 [::IxLoad new ixTimeline]
$Timeline1 config \
-rampUpValue                      1 \
-rampUpType                        0 \
-offlineTime                       0 \
-rampDownTime                      20 \
-standbyTime                       0 \
-iterations                        1 \
-rampUpInterval                    1 \
-sustainTime                       20 \
-timelineType                      0 \
-name                              "Timeline1"

$Activity_VoIPH323Peer1 config \
-enable                            true \
-name                              "VoIPH323Peer1" \
-enableConstraint                   false \
-userObjectiveValue                 1 \
-constraintValue                    100 \
-userObjectiveType                  "channels" \
-timeline                          $Timeline1

$Activity_VoIPH323Peer1 agent.config \
-enable                             true \
-name                               "VoIPH323Peer1" \

```

SEE ALSO

ixConfig



Codec Settings

VoIP H323 Peer Codec Settings

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.codecSettings.config \  

```

DESCRIPTION

Codec Settings contains the list of codecs that will be used by the VoIP H323 Peers in the test. Codec Settings is a list of one or more `codec` (audio codec) objects. To add `codec` objects, use the `appendItem` command. To clear the codec settings, use the `clear` subcommand.

SUBCOMMANDS

```
clear  
  
Clears the list of codec settings. For example:  
$Activity_VoIPH323Peer1 agent.pm.codecSettings.codecs.clear
```

OPTIONS

```
codecs_number  
  
Indicates the codec numbers. Default= 0
```

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.codecSettings.config \  
-codecs_number                0
```

SEE ALSO

VoIP H323 Peer Agent

Codecs

VoIP H323 Peer Audio Codec

SYNOPSIS

```
set Activity_VoIPH323Peer1 [${Traffic1_Network1} activityList.appendItem \  
    -protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.codecSettings.config \  
  
$Activity_VoIPH323Peer1 agent.pm.codecSettings.codecs.appendItem \  

```

DESCRIPTION

Codecs configures an audio codec object, which is added to the `Codec Settings` list of codecs. To add a `codec` object, use the `appendItem` command.

SUBCOMMANDS

None.

OPTIONS

`id`

Codec type. One of the following:

Codec	Description
CodecAMR	Adaptive multi-rate codec
CodecG711u	G.711 mu-law codec
CodecG711a	G.711 A-law codec
CodecG723x153	G.723.1 codec @ 5.3 kbps
CodecG723x163	G.723.1 codec @ 6.3 kbps
CodecG726x16	G.726 codec @ 16 Kbps
CodecG726x24	G.726 codec @ 24 Kbps
CodecG726x32	G.726 codec @ 32 Kbps
CodecG726x40	G.726 codec @ 40 Kbps
CodecG729A	G.729 Annex-A codec
CodecLBC	Internet low-bitrate codec

Options for CodecAMR

dPayloadIn

Incoming dynamic payload type. Default="98" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="98" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 14. Default=14.

payloadFormat

Payload format.

Value	Usage
0 (default)	Bandwidth-efficient format
1	Octet-aligned format

mode

Codec bit rate. One of the following:

Mode	Description
0 (default)	4.75 kbps
1	5.15 kbps
2	5.90 kbps
3	6.70 kbps
4	7.40 kbps
5	7.95 kbps
6	10.20 kbps
7	12.20 kbps

Options for CodecG711u

dPayloadIn

Incoming dynamic payload type. Default="0" min="0" max="127".



dPayloadOut

Outgoing dynamic payload type. Default="0" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG711a

dPayloadIn

Incoming dynamic payload type. Default="8" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="8" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG723x153

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 20. Default=20.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG726x16

dPayloadIn

Incoming dynamic payload type. Default="102" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="102" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 20, 40, 60. Default=20.

Options for CodecG726x24

dPayloadIn

Incoming dynamic payload type. Default="103" min="0" max="127".



dPayloadOut

Outgoing dynamic payload type. Default="103" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 30, 60, 90. Default=30.

Options for CodecG726x32

dPayloadIn

Incoming dynamic payload type. Default="104" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="104" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 40, 80, 120. Default=40.

Options for CodecG729

dPayloadIn

Incoming dynamic payload type. Default="18" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="18" min="0" max="127".

cbxFrameSize

Bytes per frame. Must be one of the following: 10, 20, 30, 40, 50, Custom. Default=10.

customFrameSize

If `cbxFrameSize` is Custom, this option configures the custom frame size. Default="120" min="10" max="200".

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.codecSettings.codecs.clear

$Activity_VoIPH323Peer1 agent.pm.codecSettings.codecs.appendItem \
-id "CodecG711u" \
-dPayloadOut 0 \
-dPayloadIn 0 \
-frameSize 160

$Activity_VoIPH323Peer1 agent.pm.codecSettings.codecs.appendItem \
-id "CodecG711a" \
-dPayloadOut 8 \
-dPayloadIn 8 \
-frameSize 160
```

SEE ALSO

Codec Settings

Data Codecs

VoIP H248 MGC/MGW Term Group Data Codecs

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.codecSettings.config \  
  
$Activity_VoIPH323Peer1 agent.pm.codecSettings.dataCodecs.appendItem \  

```

DESCRIPTION

Data Codecs configures a data codec object, which is added to the Codec Settings list of codecs.

SUBCOMMANDS

None.

OPTIONS

id

Codec type. One of the following:

Codec	Description
Rtp2833Events	Named Events Payload format used for carrying DTMF digits and other line and trunk signals as events.
Rtp2833Tones	RTP Payload format that can represent tones consisting of one or more frequencies.

dPayloadType

Payload type used for RTP data packets. Default=(see table) min="96" max="127"

Codec	Default value for dPayloadType
Rtp2833Events	100
Rtp2833Tones	101

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.codecSettings.dataCodecs.clear

$Activity_VoIPH323Peer1 agent.pm.codecSettings.dataCodecs.appendItem \
-id "Rtp2833Events" \
-dPayloadType 100

$Activity_VoIPH323Peer1 agent.pm.codecSettings.dataCodecs.appendItem \
-id "Rtp2833Tones" \
-dPayloadType 101

$Activity_VoIPH323Peer1 agent.pm.codecSettings.codecs.clear
```

SEE ALSO

Codec Settings



Other Settings

VoIPH323 Peer Other Settings

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.otherSettings.config \  

```

DESCRIPTION

This object configures the VoIP H323 Peer activity's miscellaneous options.

SUBCOMMANDS

None.

OPTIONS

VOIP_Var0

The VOIP_Var1...VOIP_Var5 and VOIP_IPAddr1...VOIP_IPAddr5 string-type variables supporting generator expressions enable you to generate 10 series of global variables whose values are used at runtime by the simulated H.323 phones/channels. Default=" ".

Use the VOIP_Var1...VOIP_Var5 variables to represent phone numbers, and the VOIP_IPAddr1...VOIP_IPAddr5 to represent IP addresses.

VOIP_Var1

See VOIP_Var0.

VOIP_Var2

See VOIP_Var0.

VOIP_Var3

See VOIP_Var0.

VOIP_Var4

See VOIP_Var0.

VOIP_IPAddress0

See VOIP_Var0.

VOIP_IPAddress1

See VOIP_Var0.

VOIP_IPAddress2

See VOIP_Var0.

VOIP_IPAddress3

See VOIP_Var0.

VOIP_IPAddress4

See VOIP_Var0.

ipPreference

Type of addressing to be used on the subnet that the VOIP H323 Peer runs on.

Value	Usage
0 (default)	IPv4
1	IPv6

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.otherSettings.config \  
-ipPreference                0 \  
-stackConfigFilename         "" \  
-VOIP_Var1                   "" \  
-VOIP_Var0                    "" \  
-VOIP_Var3                    "" \  
-VOIP_Var2                    "" \  
-VOIP_Var4                    "" \  
-VOIP_IPAddress4              "" \  
-totalUserCount              0 \  
-VOIP_IPAddress1              "" \  
-VOIP_IPAddress0              "" \  
-VOIP_IPAddress3              "" \  
-VOIP_IPAddress2              ""
```

SEE ALSO

VoIP H323 Peer Agent

RTP Settings

VoIPH323 Peer RTP Settings

SYNOPSIS

```
$Activity_VoIPH323Peer1 agent.pm.rtpSettings.config \  
-option      value
```

DESCRIPTION

RTP Settings configures the VoIPH323Peer RTP transport settings.

SUBCOMMANDS

None.

OPTIONS

enableRTP

Enables use of RTP to transport the media traffic.

0 = disabled (default)

1 = enabled

rtpPort

RTP port number. Default="10000".

Note: Valid port numbers are between 1000 and 65534.

enableRTCP

Enables the sending and receiving of RTCP packets.

chEnableHwAcc

If true, enables hardware acceleration for RTP traffic. Default=false.

enableAdvStatCalc

Enables the computation of advanced RTP statistics.

enablePerStream

Audio Settings

VoIPH323 Peer audio settings

SYNOPSIS

```
$Activity_VoIPH323Peer1 agent.pm.audioSettings.config \
```

DESCRIPTION

The Audio Settings configure the VoIPH323 Peer audio RTP settings.

SUBCOMMANDS

None.

OPTIONS

`enableAudio`

If selected, audio script functions are executed, otherwise they are skipped.

`audioClip`

The played audio clip file.

`playTypeAudio`

The mode in which the clip is played.

Value	Usage
0 (default)	The clip is played for clip duration or for the duration of the Talk Time parameter in the case of BHCA/CPS/LPS objectives.
1	The clip is played for a user-defined duration.

`audioDurationUnit`

The play duration unit, which can be milliseconds (0), seconds (1), minutes (2), or hours (3).

`outputLevel`

The output level of the played clip.

`enableTosRtp`



Enables use of TOS/DSCP. Use the `rtpTos` option to specify the TOS/DSCP value. Default= `False`

`rtpTosVal`

- The Type of Service (TOS/DSCP) byte setting in the sent RTP packets has one of the following values:
- Best Effort (0x00): Routine service
- Class 1 (0x20): Priority service, Assured Forwarding class 1
- Class 2 (0x40): Immediate service, Assured Forwarding class 2
- Class 3 (0x60): Flash, Assured Forwarding class 3
- Class 4 (0x80): Flash-override, Assured Forwarding class 4
- Express Forwarding (0xA0): Critical-ecp
- Control (0xC0): Internet-control
- Custom: A user-specified value.

`useMOS`

Enables the computation of MOS scores. Default= `False`.

`enableAudioOWD`

If `true`, `IxLoad` computes the One-way Delay metric, a network measurement specifying the amount of time (in ms) that a packet has spent on the network before it was received on the destination side. Default= `False`

`useJitter`

If `true`, enables use of a jitter buffer. Default= `False`.

`jitMs`

If `useJitter` is 1, this option configures the size of the jitter buffer, in milliseconds. Default= "20" min="1" max="3000".

`useJitComp`

If `true`, enables dynamic modification of the jitter buffer size. Default= `False`.

`jitCMs`

If `useJitComp` is 1, this option configures the maximum size in of the jitter buffer, in milliseconds. Default="1000" min="0" max="3000".

`jitCMaxDrop`

If `useJitComp` is 1, this option configures the condition - a maximum number of consecutive packets dropped - that determines the jitter buffer size to be increased.

`enableQoV`

If `true`, this enables QoV P.862 PESQ and P.56 QoV computation. `Default= False`.

`channelTypeQoV`

When `enableQoV` is `true`, this specifies the objective type as either of the following:

- Number of channels (0)
- Percentage (1)

`valueQoV`

When `enableQoV` is `true`, this specifies the number of channels for which PESQ and P.56 QoV metrics are computed (when `channelTypeQoV` is 0). Alternatively this represents the percentage of channels for which PESQ and P.56 QoV metrics are computed (when `channelTypeQoV` is 1).

`unitsQoV`

The channels selection mode, which can be any of the following:

- First channels (0)
- Last channels (1)
- Evenly-spaced channels (2)
- Random (3)

`metricsQoV`

When `enableQoV` is `true`, this specifies the metric that is calculated by the Zion card. Available options are:

- PESQ and P.56 (0)
- PESQ (1)
- P56 (2)

`useSilence`

If `true`, RTP packets containing artificial background noise are sent when no other media (DTMF, MF, real payload, and so on) is sent over the communication channel. `Default= False`.

`silenceMode`

If `useSilence` is 1, this option configures the silence mode.



Value	Usage
0	Null data encoded
1 (default)	Comfort noise.

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.audioSettings.config \  
-enableAudio true \  
-audioClip "US_042.wav" \  
-playTypeAudio 0 \  
-audioDurationUnit 1 \  
-audioDuration 10 \  
-outputLevel -20 \  
-enableAudioOWD false \  
-enableTosRtp false \  
-rtpTosVal 32 \  
-useMos false \  
-useJitter false \  
-jitMs 20 \  
-useJitComp false \  
-jitCMs 1000 \  
-jitCMaxDrop 7 \  
-enableQoV false \  
-channelTypeQoV 0 \  
-valueQoV 100 \  
-unitsQoV 0 \  
-metricsQoV 0 \  
-useSilence false \  
-silenceMode 1 \  

```

SEE ALSO

Video Settings

VoIPH323 Peer Video Settings

SYNOPSIS

```
$Activity_VoIPH323Peer1 agent.pm.VideoSettings.config \  
-option          value
```

DESCRIPTION

Video Settings configures the VoIPH323 Peer's video settings.

SUBCOMMANDS

None.

OPTIONS

enableVideo

Enables use of video as media traffic.

0 = disabled (default)

1 = enabled

videoClip

Name of the video file. Default = "Fire_avc.mp4"

playTypeVideo

Determines parameters for running video. Following values are available:

Value	Usage
0 (default)	Play for clip duration
1	Play for specified duration.
2	Conference mode

videoDuration

If playTypeVideo = 1, determines duration of video. Maximum value = 25920000.

videoDurationUnit



Unit of duration. The following values are available:

Value	Usage
0	milliseconds
1	seconds
2	minutes
3	hours

`useConference`

If `playTypeVideo = 2`, enables use of conference mode. The following values are available:

Value	Usage
0	All speak
1	Sequential
2	Random

`confVideoDuration`

If `playTypeVideo = 2`, enables selection of conference video duration.

`confVideoDurationUnit`

If `playTypeVideo = 2`, enables selection unit of conference video duration. The following values are available:

Value	Usage
0	milliseconds
1	seconds
2	minutes
3	hours

`confDuration`

If `playTypeVideo = 2`, enables selection of conference audio duration.

`confDurationUnit`

If `playTypeVideo = 2`, enables selection unit of conference audio duration. The following values are available:

Value	Usage
0	milliseconds

Value	Usage
1	seconds
2	minutes
3	hours

`enableTosVideo`

Enables use of TOS/DSCP. Use the `tosVideo` option to specify the TOS/DSCP value.

`tosVideo`

The following values are available:

Value	Usage
0	Best Effort (0x00)"
1	Class 1 (0x20)
2	Class 2 (0x40)
3	Class 3 (0x60)
4	Class 4 (0x80)
5	Express Forwarding (0xA0)
6	Control (0xC0)
7	Custom

`useMosVideo`

Enables computation of MOS.

0 = disabled (default)

1 = enabled

Note: If MOS computation is enabled, the `enableVideoOWD` option also has to be enabled.

`enableVideoOWD`

If enabled, the One-way Delay metric is computed, a network measurement specifying the amount of time (in ms) that a packet has spent on the network before it was received on the destination side. Default = disabled.

`ignoreHintTrack`

If enabled, the hint track (if any) in the video clip is ignored. The video streaming uses a new hint track which is recreated using one of the packetization modes defined by `hintTrackType`. Default = disabled.



hintTrackType

Allows to select the packetization mode. The following values are available:

Value	Usage
0 (default)	Single NAL Unit
1	STAP-A, with FU-A fragmentation

AdvancedVideoSettings

If enabled, allows selecting the advanced settings. Valid only for H323 activities.

enableCustomMaxMBPS

If enabled, allows selecting the maximum number of macroblocks per second supported. Default = disabled.

customMaxMBPS

The maximum number of macroblocks per second supported.

enableCustomMaxFS

If enabled, allows selecting the maximum frame size supported. Default = disabled.

customMaxFS

The maximum frame size supported.

enableCustomMaxDPB

If enabled, allows selecting the maximum decoded picture buffer size supported. By default it is disabled.

customMaxDPB

The maximum decoded picture buffer size supported.

enableCustomMaxBRandCPB

If enabled, allows selecting the maximum supported video bitrate and coded picture buffer.

customMaxBRandCPB

The maximum number of static macroblocks per second.

enableMaxStaticMBPS

If enabled, allows selecting the maximum number of static macroblocks per second. Default = disabled.

maxStaticMBPS

The maximum number of static macroblocks per second.

enableMaxRcmdNalUnitSize

If enabled, allows selecting the maximum recommended NALU size. Default = disabled.

enableMaxNalUnitSize

If enabled, allows selecting the maximum NALU size supported. Default = disabled.

maxNalUnitSize

The maximum NALU size supported.

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.videoSettings.config \  
-rotationScheme          0 \  
-confDuration            1 \  
-useMosVideo             false \  
-enableVideoOWD         false \  
-ignoreHintTrack        false \  
-enableTosVideo         true \  
-enableVideo            true \  
-videoClip              "Fire_avc.mp4" \  
-useH323AdvancedSettings false \  
-videoDuration          5 \  
-confVideoDurationUnit  1 \  
-useConference          false \  
-confDurationUnit       1 \  
-confVideoDuration      1 \  
-videoDurationUnit      1 \  
-hintTrackType          1 \  
-fmtp                   "" \  
-rtptime                 "" \  
-playTypeVideo          0 \  
-tosValVideo            32
```

SEE ALSO



Alternative Capability Value Set List

VoIP H323 Alternative Capability Value Set List

SYNOPSIS

```
set Activity_VoIPH323Peer1 [${Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1  
agent.pm.alternativeCapabilitySetList.alternativeCapabilityValueSetList.append  
dItem \  

```

DESCRIPTION

Helps to configure the alternative capability value list.

SUBCOMMANDS

None.

OPTIONS

id

Indicates the id of the alternative capability name.

Default= "AlternativeCapability"

alternativeCapabilityName

The name of the alternative capability list.

Default= "Default_Alternative_Capability"

refCount

The reference count that is used to deallocate objects which are no longer referenced. Default= 0



EXAMPLE

```
$Activity_VoIPH323Peer1
agent.pm.alternativeCapabilitySetList.alternativeCapabilityValueSetList.appendItem \
-id                                     "AlternativeCapability" \
-alternativeCapabilityName           "Default_Alternative_Capability" \
-refCount                             0
```

SEE ALSO

VoIP H323 Peer Agent

Capability List

VoIP H323 Alternative Capability Value Set List

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1  
agent.pm.alternativeCapabilitySetList.alternativeCapabilityValueSetList.append  
dItem \  
  
$Activity_VoIPH323Peer1  
agent.pm.alternativeCapabilitySetList.alternativeCapabilityValueSetList.capab  
ilityList.appendItem \  

```

DESCRIPTION

Helps to configure the capability list.

SUBCOMMANDS

None.

OPTIONS

id

Indicates the id of the capability list. Default= "Capability".

transportType

The transport type used for the VoIP data for the various versions. Default= 3.

capabilityTableEntryNumber

The number that is entered in the capability table. This table is referred to take policy actions based on whether the system has a particular capability. Default= 1

EXAMPLE

```
$Activity_VoIPH323Peer1
agent.pm.alternativeCapabilitySetList.alternativeCapabilityValueSetList.capabilityList
.appendItem \
  -id "Capability" \
  -transportType 3 \
  -capabilityTableEntryNumber 1
```

SEE ALSO

VoIP H323 Peer Agent

Custom Activity Link Settings

VoIP H323 Peer CustomActivityLinkSettings

SYNOPSIS

```
set Activity_VoIPH323Peer1 [${Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.customActivityLinkSettings.config \  

```

DESCRIPTION

CustomActivityLinkSettings configures the settings for the BHCA objective for VoIPH323 Peer activities. This options in this object correspond to the controls on the Custom Parameters tab for a NetTraffic/ActivityLink in the Timeline and Objective branch of the Test Configuration tree in the IxLoad GUI.

Note: The CustomActivityLinkSettings class must be configured alongside the CustomParameters class that implements the same functionality.

Note: CPS objective related settings are not available for VoIPH323 Peer activities.

SUBCOMMANDS

None.

OPTIONS

bhcaObjectiveValue

The BHCA test objective value. Default="80000".

bhcaType

Determines how the BHCA objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	BHCA will be met by specifying the talk time. Specify the talk time in talkTime.
1	BHCA will be met by specifying the number of channels. Specify the number of channels in channelsNo.

talkTime



If `bhcaType` is 0, this option specifies the Talk Time that will be used to attain the BHCA test objective. Default="40000".

`channelsNo`

If `bhcaType` is 1, this option specifies the number of channels that will be used to attain the BHCA test objective. Default="100".

`callSetupTime`

Estimated call setup time. Default="500".

`callTeardownTime`

Estimated call teardown time. Default="500".

`interCallDuration`

Inter-call duration. Default="4000".

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.customActivityLinkSettings.config \  
-talkTime 40000 \  
-cpsObjectiveValue 100 \  
-cpsType 0 \  
-cpsInterCallDuration 150 \  
-channelsNo 1 \  
-cpsTalkTime 750 \  
-cpsOverheadTime 100 \  
-cpsChannelsNo 100 \  
-bhcaType 0 \  
-callTeardownTime 500 \  
-interCallDuration 4000 \  
-bhcaObjectiveValue 80000 \  
-callSetupTime 500
```

SEE ALSO

VoIP H323 Peer Agent

Execution Settings

VoIP H323 Peer Execution Settings

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.executionSettings.config \  

```

DESCRIPTION

This object defines the execution settings for the VoIP H323 Peer.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`gracefulRampDown`

If enabled, the execution is stopped gracefully and the call is closed before the rampdown period ends.

`loopCount`

If `loopMode` is 1, this option defines the number of loops that the test performs.

Default="1".

`loopPreDelay`

Delay before first loop (ms). Default="0".

`loopMode`

Defines how many loops are executed for every voice channel corresponding to this activity.

Value	Description
0 (default)	Loop for the entire test duration.



Value	Description
1	Execute a number of loops. Specify the number of loops in loopCount.

loopMidDelay

Delay between loops (ms). Default="0".

phoneRule

Defines how phone numbers are incremented for H323 activity.

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.executionSettings.config \  
-gracefulRampDown          true \  
-loopCount                  1 \  
-loopPreDelay               0 \  
-loopMode                   0 \  
-loopMidDelay               0 \  
-phoneRule                   1
```

SEE ALSO

VoIP H323 Peer Agent

Simultaneous Capability

VoIP H323 Simultaneous Capability

SYNOPSIS

```
set Activity_VoIPH323Peer1 [${Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.simultaneousCapability.config \  

```

DESCRIPTION

Configures the simultaneous capability name.

SUBCOMMANDS

None.

OPTIONS

refCount

The reference count that is used to deallocate objects which are no longer referenced. Default= 0

simultaneousCapabilityName

The name of the simultaneous capability. Default= ""

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.simultaneousCapability.config \  
-refCount                0 \  
-simultaneousCapabilityName    ""
```

SEE ALSO

VoIP H323 Peer Agent



H323 Settings

VoIP H323 Peer Signaling Settings

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.h323Settings.config \  

```

DESCRIPTION

This object defines the VoIP H323 Peer settings.

SUBCOMMANDS

None.

OPTIONS

`enableParallelH245`

If true, H.323 initiates parallel H.245 channel establishment simultaneously with FastStart. Default= False

Note: This option is enabled only if `enableFastStart` is true.

`useGKforAdmission`

If true, `MakeCall` sends ARQ request to gatekeeper and waits for ACF request before establishing the call. Default= False

`H225Version`

The version specified in the protocol-identifier field of the Q.931 and RAS packet. Default= 5

`rasRetryCount`

Sets the number of retries to be done for RAS requests. Default= 1

`enableDisengage`

If true, `EndCall` request sends a Disengage message to the gatekeeper. Default= False

textUserUser

Specifies the user information to be sent in the Q.931 User-User IE. This can be either a character string or a byte stream encoded in hexadecimal digits. Default= False

bandwidth

The value of bandwidth requested by the endpoint and also advertised in RAS messages. Default= 64 Kbps

textDisplay

Specifies the display information that is sent in the Q.931 Display IE.

enableH245tunneling

If true, H323 uses tunnel H.245 payloads within Q.931/H.225 packets. Default= True

enableCallAlerting

If true, sends out a call alerting message. Default= False

GKAdresstext

If `enableAutoGKDiscovery` is false, you can specify IP address or hostname for up to three gatekeepers. The plug-in, in this case, accepts redirection requests for the gatekeepers. Default= False

enableTos

If true, allows to configure Type of Service.TOS value is set for all UDP and TCP packets originating from the IxLoad H.323 stack.

useRegistration
Parameters

If true, the values sent by the gatekeeper during registration overrides the corresponding values set by the user for various parameters. Default= False

enableFastStart

If true, H.323 call establishment tries to use the FastStart mechanism. Default= True

ckHexUserUserData

If true, the user information that is sent in the Q.391 UserUser IE can be sent as a byte stream encoded in hexadecimal digits. Default= False



rasTimeout

Sets the number of seconds after which a RAS request reaches time-out state if no response to that request is received in the specified period. After a timeout, retry happens if so configured. Default= 4 seconds

enableCallProceeding

If true, sends out a call proceeding message. Default= False

enableRas

If true, RAS message is sent and received. Default= False

enableKeepAliveReg

If true, sends keep alive ARQ to the gatekeeper. Default= False

GKAddress

If enableAutoGKdiscovery is true, the IP address or hostname can be specified up to three gatekeepers.

tosVal

If enableTos is true, this option sets the value of the TOS bits.

Value	Usage
0 (default)	Best Effort (0x00)
1	Class 1 (0x20)
2	Class 2 (0x40)
3	Class 3 (0x60)
4	Class 4 (0x80)
5	Express Forwarding (0xA0)
6	Control (0xC0)

displayData

Specifies the display information to be sent in the Q.931 Display IE. This is IA5 character string. This field supports sequence generators. Default= "Ixia\[00-\]" \

terminalType

Indicates the type of the endpoint. The terminal types are:

- Terminal Entity Without MC
- Gatekeeper Entity Without MC

H245Version

Specified version in the protocol-identifier field of the H.245 packet. Default= 9.

autoRegisterToGk

If true, H.323 MakeCall option sends RAS signaling (including registration) to establish the call and EndCall unregisters with the gatekeeper. Default= False.

userUserData

Specifies the user information to be sent in the Q.931 User-User IE. This can be either IA5 character string or a byte stream encoded in hexadecimal digits. This field supports sequence generators for IA5 characters. Default= "1234\[00-\]" \.

enableAutoGKdiscovery

If true, Automatic Gatekeeper Discovery is attempted by sending GRQ to the well-known Discovery Multicast Address. Default= False.

callSignalingViaUDP

If true, call signaling is done over UDP and not over TCP as per Annex E specification. Default= False.

enableH323

If true, the H323 script functions are executed; otherwise they are skipped. Default= True.

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.h323Settings.config \  
-enableParallelH245           false \  
-useGKforAdmission            false \  
-H225Version                   5 \  
-rasRetryCount                 1 \  
-enableDisengage              false \  
-textUserUser                 false \  
-bandwidth                     64 \  
-textDisplay                   false \  
-enableH245tunneling          true \  
-enableCallAlerting           false \  
-GKAddressstext               false \  
-enableTos                     false \  
-useRegistrationParameters    false \  
-enableFastStart              true \  
-bandwidthtext                false \  
-ckHexUserUserData            false \  
-rasTimeout                    4 \  
-enableCallProceeding         false \  
-enableRas                     false \  
-enableKeepAliveReg           false \  
-GKAddress                    "198.18.80.80" \  
-tosVal                        0 \  
-displayData                  "Ixia\[00-\]" \  
-terminalType                 50 \  
-H245Version                   9 \  
-autoRegisterToGk             false \  
-userUserData                 "1234\[00-\]" \  
-enableH323                   true
```

SEE ALSO

VoIP H323 Peer Agent

Simultaneous Capability Value Set List

VoIP H323 Simultaneous Capability Value Set List

SYNOPSIS

```
set Activity_VoIPH323Peer1 [${Traffic1_Network1} activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.terminalCapabilitySet.config \  
  
$Activity_VoIPH323Peer1  
agent.pm.terminalCapabilitySet.simultaneousCapabilityList.appendItem
```

DESCRIPTION

Helps to configure the simultaneous capability list.

SUBCOMMANDS

None.

OPTIONS

id

Indicates the id of the simultaneous capability name. Default= "SimultaneousCapabilityName"

refCount

The reference count that is used to deallocate objects which are no longer referenced. Default= 0

simultaneousCapabilityName

The name of the simultaneous capability list. Default= "Default_Simultaneous_Capability"

EXAMPLE

```
$Activity_VoIPH323Peer1  
agent.pm.simultaneousCapabilitySetList.simultaneousCapabilityValueSetList.appendItem \  
-id                                "SimultaneousCapability" \  
-refCount                            0 \  
-simultaneousCapabilityName        "Default_Simultaneous_Capability"
```

SEE ALSO

Terminal Capability Set



Alternative Capability List

VoIP H323 Alternative Capability List

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1  
agent.pm.simultaneousCapabilitySetList.simultaneousCapabilityValueSetList.alt  
ernativeCapabilityList.appendItem \  

```

DESCRIPTION

Configures the alternative capability list.

SUBCOMMANDS

None.

OPTIONS

id

Indicates the id of the alternative capability name. Default= "AlternativeCapabilityName"

alternativeCapabilityName

The name of the alternative capability list. Default= "Default_Alternative_Capability"

EXAMPLE

```
$Activity_VoIPH323Peer1  
agent.pm.simultaneousCapabilitySetList.simultaneousCapabilityValueSetList.alternativeC  
apabilityList.appendItem \  
-id                "AlternativeCapabilityName" \  
-alternativeCapabilityName "Default_Alternative_Capability"
```

SEE ALSO

Simultaneous Capability Value Set List



Alternative Capability

VoIP H323 Alternative Capability

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.alternativeCapability.config \  

```

DESCRIPTION

Configures the alternative capability descriptors.

SUBCOMMANDS

None.

OPTIONS

alternativeCapabilityName

The name of the alternative capability. Default= "".

refCount

The reference count that is used to deallocate objects which are no longer referenced. Default= 0.

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.alternativeCapability.config \  
-alternativeCapabilityName      "" \  
-refCount                      0
```

SEE ALSO

VoIP H323 Peer Agent

Dial Plan

VoIP H323 Peer Dial Plan

SYNOPSIS

```
set Activity_VoIPH323Peer1 [${Traffic1_Network1} activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 agent.pm.dialPlan.config \  

```

DESCRIPTION

The Dial Plan object configures the registration names, phone numbers, and source, destination, and transfer addresses for the channels/phones emulated by the VoIP H323 Peer activity.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`_useSPB`

Method used to select phone number.

Value	Usage
0	Use the phone number specified by pattern.
1	Use the phone number specified by Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (`_useSPb=1`). The generated Tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

`_symDestStr`

String identifying the VoIP H323 Peer that is the destination for traffic from this VoIP H323 Peer activity.
Default="None".

`_sPhone`

If `_useSPb` is 0, this option specifies the phone number. You can use sequence generators in this field to generate multiple phone numbers. See the sequence generator appendix.
 Default="160[00000000-]".

`_sBp`

If `_useSPb` is 1, this option specifies the phone book entry name.
 Default="<None>".

`srcPhoneType`

Indicates the type of source phone number.

Value	Usage
0	Specified by <code>sourcePhoneSpecified</code> as digits (default).
1	Specified by <code>sourcePhoneBook</code> as a file name.

`_dBp`

If `_useDPb` is 1, this option specifies the phone book file name.
 Default="<None>".

`ovrDestPhone`

Enables overriding of phone number from the destination VoIP H323 Peer.

Default= False.

`_dPhone`

If `_useDPb` is 0, this option specifies the phone number. Default="170[00000000-]".

`_useDPb`

Method used to select the phone number used to override destination phone number.

Value	Usage
0 (default)	Specify pattern.
1	Specify Phonebook entry.

Note: This option appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (`useDestPhoneBook=1`). The generated Tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

`destPhoneType`



Method used to select phone number.

Value	Usage
0 (default)	Use the phone number specified by pattern.
1	Use the phone number specified by Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (`useSourcePhoneBook=1`). The generated Tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.dialPlan.config \  
-_useSPb 0 \  
-symDestStr "Traffic2_VoIPH323Peer2" \  
-_sPhone "160\[00000000-\]" \  
-_sBp "&lt;None&gt;" \  
-srcPhoneType 0 \  
-_dBp "&lt;None&gt;" \  
-ovrDestPhone false \  
-_dPhone "170\[00000000-\]" \  
-_useDPb 0 \  
-destPhoneType 0
```

SEE ALSO

VoIP H323 Peer Agent

Terminal Capability Set

VoIP H323 Terminal Capability Set

SYNOPSIS

```
set Activity_VoIPH323Peer1 [${Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ] \  
$Activity_VoIPH323Peer1 agent.config \  
$Activity_VoIPH323Peer1 agent.pm.terminalCapabilitySet.config \  

```

DESCRIPTION

Configures the terminal capability descriptors.

SUBCOMMANDS

None.

OPTIONS

defaultCodecIndex

Helps to edit the default codex index. Default = 0.

defaultCodecName

Helps to edit the default codec name. Default = "".

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.terminalCapabilitySet.config \  
-defaultCodecIndex      0 \  
-defaultCodecName      ""
```

SEE ALSO

VoIP H323 Peer Agent



Simultaneous Capability List

VoIP H323 Simultaneous Capability List

SYNOPSIS

```
set Activity_VoIPH323Peer1 [${Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ] \  
$Activity_VoIPH323Peer1 agent.config \  
$Activity_VoIPH323Peer1 agent.pm.terminalCapabilitySet.config \  
$Activity_VoIPH323Peer1 \  
agent.pm.terminalCapabilitySet.simultaneousCapabilityList.appendItem
```

DESCRIPTION

Helps to configure the simultaneous capability list.

SUBCOMMANDS

None.

OPTIONS

id

Indicates the id of the simultaneous capability name. Default= "SimultaneousCapabilityName".

simultaneousCapabilityName

The name of the simultaneous capability list. Default= "Default_Simultaneous_Capability".

EXAMPLE

```
$Activity_VoIPH323Peer1 \  
agent.pm.terminalCapabilitySet.simultaneousCapabilityList.appendItem \  
-id \  
  "SimultaneousCapabilityName" \  
-simultaneousCapabilityName      "Default_Simultaneous_Capability"
```

SEE ALSO

Terminal Capability Set



Scenario Settings

VoIP H323 Peer Scenario Settings

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "VoIPH323 Peer" ]  
$Activity_VoIPH323Peer1 agent.config \  
$Activity_VoIPH323Peer1 agent.pm.scenarioSettings.config \  

```

DESCRIPTION

Scenario Settings specifies the test scenario file that will be used by the Tcl script.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`scenarioFile`

The full path to the test scenario file for the activity.

`activeScenarioChannel`

Test scenario channel (0-based index) that is associated with the VoIP H323 Peer activity. Default=0.

EXAMPLE

```
$Activity_VoIPH323Peer1 agent.pm.scenarioSettings.config \  
-scenarioFile                "C:\\Documents and Settings\\supanda\\Desktop \  
\\H323-rxf\\Simple H323 calls with FirstConnect..tst" \  
-activeScenarioChannel      0
```

SEE ALSO

VoIP H323 Peer Agent

Custom Parameters

VoIPH323 Peer CustomParameters

SYNOPSIS

```
set Activity_VoIPH323Peer1 [$Traffic1_Network1 activityList.appendItem \  
    -protocolAndType                "VoIPH323 Peer" ]  
  
$Activity_VoIPH323Peer1 agent.config \  
  
$Activity_VoIPH323Peer1 customParameters.config \  

```

DESCRIPTION

CustomParameters configures the settings for the BHCA objective for VoIPH323Peer activities. This options in this object correspond to the controls on the Custom Parameters tab for a NetTraffic/ActivityLink in the Timeline and Objective branch of the Test Configuration tree in the GUI.

Note: The CustomParameters class has to be configured alongside the CustomActivityLinkSettings class that implements the same functionality.

Note: CPS objective related settings are not available for VoIPH323 Peer activities.

SUBCOMMANDS

None.

OPTIONS

bhcaObjectiveValue

The BHCA test objective value. Default="80000".

bhcaType

Determines how the BHCA objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	BHCA will be met by specifying the talk time. Specify the talk time in talkTime.
1	BHCA will be met by specifying the number of channels. Specify the number of channels in channelsNo.

talkTime

If `bhcaType` is 0, this option specifies the Talk Time that will be used to attain the BHCA test objective. Default="40000".

`channelsNo`

If `bhcaType` is 1, this option specifies the number of channels that will be used to attain the BHCA test objective. Default="100".

`interCallDuration`

Inter-call duration. Default="4000".

`callSetupTime`

Estimated call setup time. Default="500".

`callTeardownTime`

Estimated call teardown time. Default="500".

EXAMPLE

```
$Activity_VoIPH323Peer1 customParameters.config \  
-talkTime 40000 \  
-cpsObjectiveValue 100 \  
-cpsType 0 \  
-cpsInterCallDuration 150 \  
-channelsNo 1 \  
-cpsTalkTime 750 \  
-cpsOverheadTime 100 \  
-cpsChannelsNo 100 \  
-bhcaType 0 \  
-callTeardownTime 500 \  
-interCallDuration 4000 \  
-bhcaObjectiveValue 80000 \  
-callSetupTime 500
```

SEE ALSO

VoIP H323 Peer Agent

34

VoIP MGCP

The IxLoad VoIP MGCP Peer Tcl API consists of VoIP GW and VOIP MGC agents with separate APIs for configuring each major aspect of the agent’s functionality.

There is also an Endpoint Agent with separate configuration parameters.

- When defined on a GW activity, an Endpoint agent refers to endpoints present on that gateway.
- When defined on a CA activity, an Endpoint agent refers to endpoints managed by that controller.

In this chapter:

Limitations	1826
VoIP MGCP Peer API Commands	1827
MGCP GW Agent	1830
MGCP Settings (GW)	1838
Automatic Settings (GW)	1841
Endpoints	1843
MGCP CA Agent	1845
MGCP Settings (CA)	1847
Automatic Settings (CA)	1850
Endpoints	1852
Gateways	1854
Scenario Settings	1856
Execution Settings	1857
Custom Activity Link Settings	1859
Simulated Endpoints	1863
Data Codecs	1865
Codecs	1867
SDP Settings	1872
RTP Settings	1874
Audio Settings	1876
Other Settings	1880

Limitations

The following restrictions and limitations of the VoIP MGCP Peer API exist:

- Individual VoIP MGCP script functions can not be added and edited from the Tcl API. Instead, you must add and configure the test scenario in the Scenario Editor, then save the test scenario file and pass it as an argument to the `ScenarioSettings` API class.

VoIP MGCP Peer API Commands

The IxLoad VoIP MGCP Peer API commands are organized as shown in the figure below.

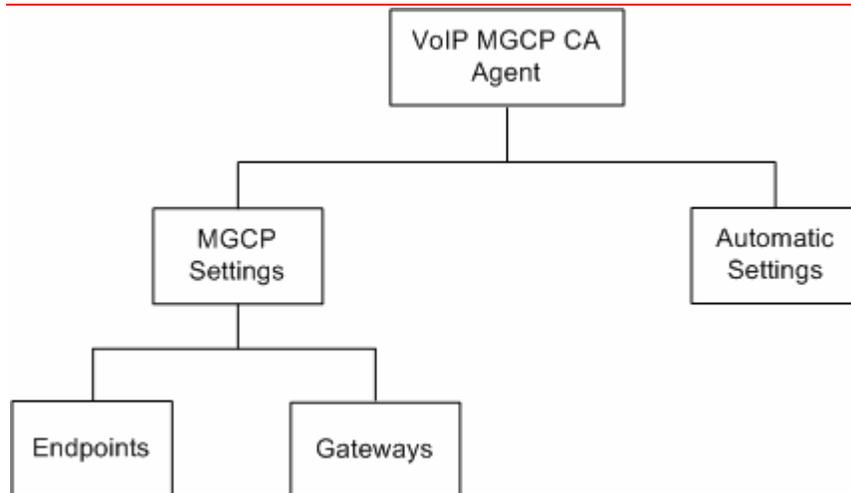
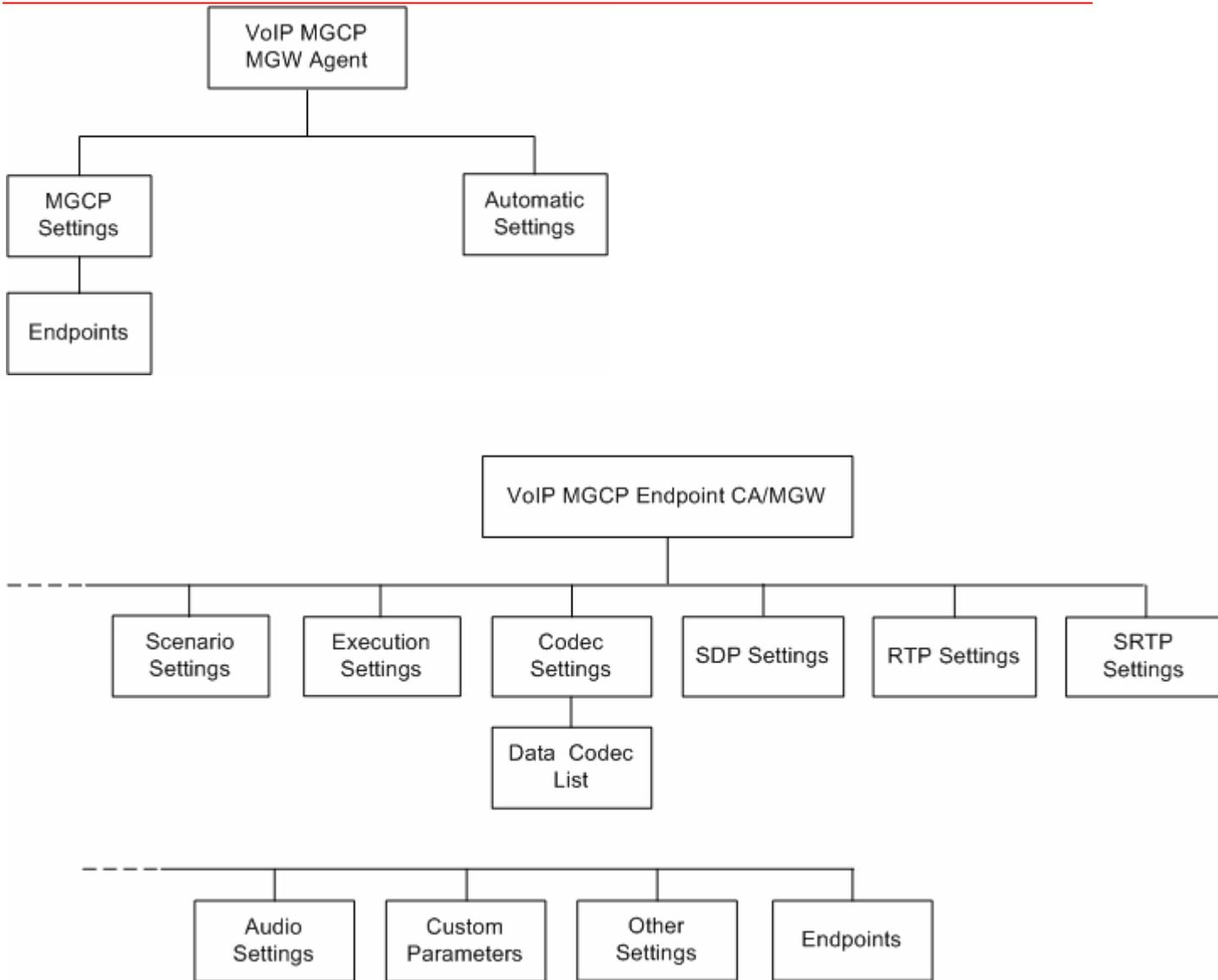


Figure 30-2.VoIP MGCP GW Peer API Structure



VoIP MGCP CAMGW Peer API Objects

The table below summarizes the VoIP MGCP CA/GW API objects.

Object	Description
VoIP MGCP CA Agent	Top-level object defining the VoIP CA activity.
VoIP MGCP GW Agent	Top-level object defining the VoIP GW activity.
MGCP Settings	Configures the MGCP Settings separately for the Media Gateway or Media Controller. Also contains the list of all endpoint groups associated with the gateway or controller.
Automatic Settings	Sets the automatic functionality parameters for the MGC and GW side.

Object	Description
Endpoints	Contains the list of all endpoint groups associated with the gateway or controller. When a new Endpoint is added, a new activity is added in the same NetTraffic.
Gateways	The list of CA-controlled Gateways.

VoIP MGCP Endpoint Peer API Objects

The table below summarizes the MGCP Endpoint API objects.

Object	Description
VoIP MGCP Endpoint Agent	Top-level object defining the VoIP MGCP CA/GW Endpoint agent activity.
Scenario Settings	This object corresponds to the Scenario Settings GUI tab and enables the selection of the scenario channel.
Execution Settings	Run-time test configuration; corresponds to the Execution Settings GUI tab.
Endpoints	The list of simulated endpoint groups.
Codec Settings	List of <code>Data Codecs</code> and <code>Codecs</code> objects.
Data Codecs	Data codec with parameters.
Codecs	Audio codecs with parameters.
SDP Settings	The SDP settings for the simulated endpoints.
RTP Settings	RTP transport configuration corresponding to the RTP Settings GUI tab.
SRTP Settings	SRTP settings corresponding to the SRTP GUI tab.
Other Settings	VoIP MGCP Peer miscellaneous parameters which corresponds to the Other Settings GUI tab.
Custom Activity Link Settings, CustomParameters	BHCA objective configuration which corresponds to the Custom Parameters GUI tab.



MGCP GW Agent

VoIP MGCPGW Agent

SYNOPSIS

```
set Activity_MGCPGW1 [${Traffic1_Network1} activityList.appendItem \  
-protocolAndType "MgcpGw Peer" ]
```

DESCRIPTION

A VoIP MGCPGW agent is added to the `agentList` option of the `ixConfig` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. See the following example:

```
set Activity_MGCPGW1 [${Traffic1_Network1} activityList.appendItem \  
-protocolAndType "MgcpGw Peer" ]  
  
$Activity_MGCPGW1 config \  
-enable 1 \  
-name "MGCPGW1"  
  
$$Activity_MGCPGW1 agent.config \  
-cmdListLoops 0
```

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands. For example, the first agent uses an index of 0 and its name may be modified by:

```
$Activity_MGCPGW1 agent(0).config -name "MgcpGw Peer new"
```

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = 1).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

The available MGCP statistics are listed in the table below.

Statistic	Description	Applies To
MGCP Transactions		
Transactions Sent	Number of transactions sent from test start.	MGW, CA

Statistic	Description	Applies To
Transactions Received	Number of transactions received from test start.	MGW, CA
Transactions Failed	Number of transactions initiated and failed from test start.	MGW, CA
MGCP Transaction Rates		
Transactions Sent/sec	Number of transactions sent per second from test start.	MGW, CA
Transactions Received/sec	Number of transactions received per second from test start.	MGW, CA
Transactions Failed/sec	Number of transactions failed per second from test start.	MGW, CA
MGCP Transaction Times		
Transactions Sent Duration (ms) (min, avg, max)	The min/max/average duration of sent transactions.	MGW, CA
Transactions Received Duration (ms) (min, avg, max)	The min/max/average duration of sent transactions.	MGW, CA
MGCP Calls		
Calls Attempted	The number of attempted calls since test start. This statistic is incremented when the phone number digits are dialed.	MGW, CA
Calls Received	The number of received calls since test start. This statistic is incremented when a receiver connection becomes <i>sendreceive</i> .	MGW, CA
Calls Connected	The number of connected calls since test start. This statistic is incremented when a connection state becomes <i>sendreceive</i> following an originated call.	MGW, CA
Calls Answered	The number of calls since test start that were not answered. This statistic is incremented when a receiver endpoint sends an RQNT (S:L/rg).	MGW, CA
Calls Busy	The number of calls since test start that resulted in an endpoint busy condition. This statistic is incremented when an originating call is in <i>hd</i> state.	MGW, CA
Calls Rejected	The number of calls since test start that were rejected. This statistic is incremented when a delete call message is received before the call is connected.	MGW, CA
MGCP Call Times		
MGCP Call Setup Time for MGW (ms) (min, avg, max)	The min/max/average call setup time from the MGW perspective. This is defined as the time duration between the first digit is sent and the moment when the MGCP connection becomes <i>sendrecv</i> .	MGW

Statistic	Description	Applies To
MGCP Call Setup Time for CA (ms) (min, avg, max)	The min/max/average call setup time from the CA perspective. This is the time duration between the moment the RQNT with rg (or wt) is sent and the moment when the MGCP connection becomes <i>sendrecv</i> .	CA
MGCP End Call Time MGW (ms) (min, avg, max)	The min/max/average end call call time from the MGW perspective. This is the duration between the DLCX is received and the NTFY (<i>hu</i>) is sent, or inversed flow.	MGW
MGCP End Call Time CA (ms) (min, avg, max)	The min/max/average end call call time from the CA perspective. This is the time duration between the DLCX is sent and the NTFY (<i>hu</i>) is received.	CA
MGCP Post Dial Delay (ms) (min, avg, max)	The min/max/average post dial delay. This is defined as the time between the last digit is sent and moment when the MGCP connection becomes <i>sendrecv</i> .	MGW
MGCP Post Pickup Delay (ms) (min, avg, max)	The min/max/average post pickup delay. This is defined as the time duration between the NTFY (<i>hd</i>) sent and the moment when the MGCP connection becomes <i>sendrecv</i> .	MGW
MGCP Media Delay Tx (ms)	The transmitting side media delay. This is the time duration between the moment the call setup is finished and the moment the first RTP packet is received.	MGW
MGCP Media Delay Rx (ms)	The receiving side media delay. The time duration between the moment the call setup is finished and the moment the first RTP packet is received.	CA
MGCP Total Call Time (min, avg, max)	The min/max/average total call time, which includes the call setup time, talk time, and end call time. This is defined as the time duration between the NTFY (<i>hd</i>) sent/received and the moment when the End Call is finished (NTFY(<i>hu</i>) after DLCX, or DLCX after NTFY(<i>hu</i>)).	MGW, CA
MGCP Call Rates		
Calls Attempted/sec	The number of attempted calls per second since test start.	MGW, CA
Calls Received/sec	The number of received calls per second since test start.	MGW, CA
Calls Connected/sec	The number of connected calls per second since test start.	MGW, CA
MGCP Commands		

Statistic	Description	Applies To
MGCP RSIP Sent	Number of MGCP RSIP commands sent by the MGW since test start, including retransmitted commands.	MGW
MGCP NTFY Sent	Number of MGCP NTFY commands sent by the MGW since test start, including retransmitted commands.	MGW
MGCP DLCX Sent	Number of MGCP DLCX commands sent by the CA since test start, including retransmitted commands.	CA
MGCP RQNT Received	Number of MGCP RQNT commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP CRCX Received	Number of MGCP CRCX commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP MDCX Received	Number of MGCP MDCX commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP DLCX Received	Number of MGCP DLCX commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP AUCX Received	Number of MGCP AUCX commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP AUPE Received	Number of MGCP AUPE commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP EPCF Received	Number of MGCP EPCF commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP RSIP Received	Number of MGCP RSIP commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP NTFY Received	Number of MGCP NTFY commands received by the CA since test start, including retransmitted commands.	CA
MGCP DLCX Received	Number of MGCP DLCX commands received by the MGW since test start, including retransmitted commands.	MGW
MGCP RQNT Sent	Number of MGCP RQNT commands sent by the CA since test start, including retransmitted commands.	CA
MGCP CRCX Sent	Number of MGCP CRCX commands sent by the CA since test start, including retransmitted commands.	CA
MGCP MDCX Sent	Number of MGCP MDCX commands sent by the CA since test start, including retransmitted commands.	CA



Statistic	Description	Applies To
MGCP DLCX Sent	Number of MGCP DLCX commands sent by the CA since test start, including retransmitted commands.	CA
MGCP AUCX Sent	Number of MGCP AUCX commands sent by the CA since test start, including retransmitted commands.	CA
MGCP AUPEP Sent	Number of MGCP AUPEP commands sent by the CA since test start, including retransmitted commands.	CA
MGCP EPCF Sent	Number of MGCP EPCF commands sent by the CA since test start, including retransmitted commands.	CA
MGCP Responses		
MGCP 1xx Received	The number of 1xx responses received since test start, including retransmissions.	MGW, CA
MGCP 1xx Sent	The number of 1xx responses sent since test start, including retransmissions.	MGW, CA
MGCP 2xx Received	The number of 2xx responses received since test start, including retransmissions.	MGW, CA
MGCP 2xx Sent	The number of 2xx responses sent since test start, including retransmissions.	MGW, CA
MGCP 4xx Received	The number of 4xx responses received since test start, including retransmissions.	MGW, CA
MGCP 4xx Sent	The number of 4xx responses sent since test start, including retransmissions.	MGW, CA
MGCP 5xx Received	The number of 5xx responses received since test start, including retransmissions.	MGW, CA
MGCP 5xx Sent	The number of 5xx responses sent since test start, including retransmissions.	MGW, CA
MGCP 5xx Received	The number of 5xx responses received since test start, including retransmissions.	MGW, CA
MGCP Messages		
MGCP msgs tx	The number of sent MGCP messages since test start.	MGW, CA
MGCP msgs rx	The number of received MGCP messages since test start.	MGW, CA
MGCP matched msgs	The number of matched MGCP messages since test start.	MGW, CA
MGCP commands tx	The number of sent MGCP commands since test start.	MGW, CA
MGCP commands rx	The number of received MGCP commands since test start.	MGW, CA
MGCP responses tx	The number of sent MGCP responses since test start.	MGW, CA
MGCP responses rx	The number of received MGCP responses since test start.	MGW, CA

Statistic	Description	Applies To
MGCP Retransmissions TX		
MGCP retransmitted messages sent	The percentage of retransmitted messages sent.	MGW, CA
MGCP messages sent	The percentage of distinct messages sent, not including retransmissions sent.	MGW, CA
MGCP Retransmissions RX		
MGCP retransmitted messages received	The percentage of retransmitted messages received.	MGW, CA
MGCP messages received	The percentage of distinct messages received, not including received retransmissions.	MGW, CA
MGCP Restart Duration		
MGCP RSIP-Restart Duration	The duration of MGW restarts with a <i>restart</i> reason. This is the time elapsed between the sending of an RSIP (<i>reason=restart</i>) command and the receiving of a 200 OK response.	MGW
MGCP RSIP-Forced Duration	The duration of MGW restarts with a <i>forced</i> reason. This is the time elapsed between the sending of an RSIP (<i>reason=forced</i>) command and the receiving of a 200 OK response.	MGW
MGCP Active Calls/Transactions		
Number of active calls	The number of active calls.	MGW, CA
Number of active transactions	The number of active transactions.	MGW, CA
MGCP Errors		
MGCP Call Flow Errors	The number of call flow errors. Such an error occurs when: <ul style="list-style-type: none"> ▪ A gateway has no IP assigned; ▪ A script function exits on the Error output 	MGW, CA
MGCP Parser Errors	The number of MGCP parser errors. This statistic is incremented when a MGCP message is received with the first line malformed.	MGW, CA
MGCP SDP Errors	The number of SDP errors. This statistic is incremented in any of the following situations: <ul style="list-style-type: none"> ▪ The SDP body cannot be constructed (invalid or empty custom SDP); ▪ The SDP parsing error occurred; ▪ The SDP negotiation failed; ▪ The SDP Glare; 	MGW, CA
MGCP RTP Errors	The number of RTP errors.	MGW, CA



Statistic	Description	Applies To
MGCP Transport Errors (failure to send a message)	The number of errors resulting in the incapability to send a message.	MGW, CA
MGCP Protocol Errors	The number of MGCP protocol errors. This statistic is incremented in any of the following situations: <ul style="list-style-type: none"> An RSIP with restart method is received for a gateway that is already restarted; An RSIP with force method is received for a gateway that is already force-restarted; An invalid message is intended to be sent; A message with an incompatible version is received; An RSIP message is received on a gateway; An inappropriate parameters has been received for a certain endpoint state; An invalid message is received (with not allowed parameters for that message); 	MGW, CA
MGCP Timeout Errors	The number of MGCP timeout errors. This statistic is incremented when a Wait-type script function timeout period has expired or when the transaction timeout period has expired.	MGW, CA
Unknown Phone No	The number of errors having an unknown destination phone number as a cause. This statistic is incremented only on the Call Agent when the dialed phone number received through a NTFY command is not defined in the Call Agent.	CA

EXAMPLE

```
#####
# Activity MGCPGW1 of NetTraffic Traffic1@Network1
#####
set Activity_MGCPGW1 [${Traffic1_Network1} activityList.appendItem \
-protocolAndType           "MgcpGw Peer" ]

$Activity_MGCPGW1 config \
-enable                     1 \
-name                       "MGCPGW1"

$Activity_MGCPGW1 agent.config \
-cmdListLoops               0
```

SEE ALSO

ixConfig

MGCP Settings (GW)

VoIP MGCP simulated GW settings

SYNOPSIS

```
set Activity_MGCPGW1 [ $Traffic1_Network1 activityList.appendItem \  
-protocolAndType                "MgcpGw Peer" ] \  
$Activity_MGCPGW1 config \  
$Activity_MGCPGW1 agent.pm.mgcpSettings.config
```

DESCRIPTION

Simulates the source address in MGCP messages and contains the list of all endpoint groups provisioned on the GW.

SUBCOMMANDS

Endpoints.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`port`

The MGCP listening port. Default = "2427".

`callAgent`

The address of the controlling CA as a symbolic link or as an IP address.

`enableTos`

Enables use of TOS/DSCP settings. When `enableTos` is configured to 1, the `tos` option specifies the TOS/DSCP value.

0 = TOS disabled (default)

1 = TOS enabled

`tosVal`

If `enableTos` is configured 1, this option sets the value of the TOS bits.

Value	Usage
0 (default)	Best Effort (0x00)
1	Class 1 (0x20)
2	Class 2 (0x40)
3	Class 3 (0x60)
4	Class 4 (0x80)
5	Express Forwarding (0xA0)
6	Control (0xC0)

domainName

The MGCP GW domain name. Sequence generator expressions are supported, for example, gw[001-100].ixialab.com which defines a number of 100 hosts, gw001 through gw100.

mgcpVersion

The currently supported MGCP version.

ncsTgcpVersion

The currently supported NCS version.

udpMaxSize

The maximum UDP size for MGCP traffic.

ipPreference

The IP preference, IPv4 or IPv6.

EXAMPLE

```
$Activity_MGCPGW1 agent.pm.mgcpSettings.config \  
-enableTos                false \  
-tosVal                    0 \  
-domainName                "gw[001-100].ixialab.com" \  
-ncsTgcpVersion            "NCS 1.0" \  
-callAgent                 "Traffic2_MGCPCA1" \  
-udpMaxSize                1470 \  
-mgcpActivitiesCount       0 \  
-ipPreference              0 \  
-mgcpVersion               "1.0" \  
-port                      "2427"
```

```
$Activity_MGCPGW1 agent.pm.mgcpSettings.endpoints.clear
```

SEE ALSO

VoIP MGCPGW Agent



Automatic Settings (GW)

VoIP MGCP GW automatic settings.

SYNOPSIS

```
s$Activity_MGCPGW1 agent.pm.automaticSettings.config \
```

DESCRIPTION

Defines automatic settings for the MGCP GW.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`rsipAtBegin`

If configured to `true`, an RSIP command is sent at the test start.

`restartDelay`

If `rsipAtBegin` is configured `true`, this specifies a restart delay value (Default=0).

`rsipAtEnd`

If configured to `true`, an RSIP command is sent at the test end.

`retransmit`

If configured to `true`, this option enables retransmissions of message for whom a reply has not been received.

`ignoreRecvRetransmit`

If configured `true`, retransmissions are ignored.

`retransmTimerType`

Specifies a timer type as either of the following:



- 0 = A timer value is specified
- 1 = The timer value is calculated internally according to RFC 3435

retransmTimerDuration

If retransmTimerType is configured 0, this specifies a retransmission timer value (default = 3000 ms).

transactionTimeout

Specifies a transaction timeout (default = 50000 ms).

waitTimeout

Specifies a wait timeout (default = 50000 ms).

EXAMPLE

```
$Activity_MGCPGW1 agent.pm.automaticSettings.config \  
-restartDelay           "0" \  
-ignoreRecvRetransmit  true \  
-rsipAtBegin           true \  
-transactionTimeout    50000 \  
-retransmTimerType     0 \  
-rsipAtEnd             true \  
-waitTimeout           50000 \  
-retransmTimerDuration 3000 \  
-retransmit            true
```

SEE ALSO

MGCPSettings (GW)

Endpoints

VoIP MGCP GW endpoint group settings.

SYNOPSIS

```
$Activity_MGCPGW1 agent.pm.mgcpSettings.endpoints.appendItem
```

DESCRIPTION

Defines the properties of an endpoint group provisioned on the GW.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`id`

Indicates the endpoint group id.

`endpoint`

Indicates the endpoint group name.

NOTE: Sequence generator expressions are supported, such as for example, `aa1n[1-3]`.

`enabled`

Indicates if the endpoint group is active (`true`) or not (`disabled = false`).

`activity`

The name of the corresponding Endpoint activity.

EXAMPLE

```
$Activity_MGCPGW1 agent.pm.mgcpSettings.endpoints.appendItem \  
-id "Endpoint" \  
-endpoint "aaln1" \  
-enabled true \  
-activity "Endpoint1"
```

SEE ALSO

MGCP CA Agent

VoIP MGCP CA Agent

SYNOPSIS

```
set Activity_MGCPA1 [${Traffic2_Network2} activityList.appendItem \  
-protocolAndType "MgcpCa Peer" ]  
  
$Activity_MGCPA1 config \  
-enable 1 \  
-name "MGCPA1"
```

DESCRIPTION

A VoIP MGCP CA agent is added to the `agentList` option of the `ixConfig` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. See the following example:

```
set Activity_MGCPA1 [${Traffic1_Network1} activityList.appendItem \  
-protocolAndType "MgcpCa Peer" ]  
  
$Activity_MGCPA1 config \  
-enable 1 \  
-name "MGCPA1"  
  
$$Activity_MGCPA1 agent.config \  
-cmdListLoops 0
```

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands. For example, the first agent uses an index of 0 and its name may be modified by:

```
$Activity_MGCPA1 agent(0).config -name "MgcpCa Peer new"
```

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = 1).

`name`

The name associated with this object, which must be set at object creation time.

EXAMPLE

```
#####  
# Activity MGCPA1 of NetTraffic Traffic1@Network1  
#####  
set Activity_MGCPA1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType           "MgcpCa Peer" ]  
  
$Activity_MGCPA1 config \  
-enable                     1 \  
-name                       "MGCPA1"
```

SEE ALSO

ixConfig

MGCP Settings (CA)

VoIP MGCP Simulated CA settings

SYNOPSIS

```
$Activity_MGCPCA1 agent.pm.mgcpSettings.config \
```

DESCRIPTION

Simulates the source address in MGCP messages, designates the MGCP CA settings. Also contains the list of all gateways and endpoint groups controlled by the CA.

SUBCOMMANDS

Endpoints, Gateways.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`port`

The MGCP CA listening port (default 2727).

`mgcpVersion`

The currently supported MGCP version.

`ncsTgcpVersion`

The currently supported NCS version.

`enable Tos`

Enables use of TOS/DSCP settings. When `enableTos` is configured to 1, the `tos` option specifies the TOS/DSCP value.

0 = TOS disabled (default)

1 = TOS enabled

`tosval`

If `enableTos` is configured 1, this option sets the value of the TOS bits.



Value	Usage
0 (default)	Best Effort (0x00)
1	Class 1 (0x20)
2	Class 2 (0x40)
3	Class 3 (0x60)
4	Class 4 (0x80)
5	Express Forwarding (0xA0)
6	Control (0xC0)

useDigitMap

If configured true, a digit map is specified.

digitMap

The digit map to be send to the GW.

ipPreference

The IP address preference, IPv4 or IPv6.

udpMaxSize

The maximum UDP size for MGCP traffic.

EXAMPLE

```
$Activity_MGCPA1 agent.pm.mgcpSettings.config \  
-enableTos                false \  
-tosVal                    0 \  
-digitMap                  "160xxxxxx" \  
-ipPreference              0 \  
-ncsTgcpVersion            "NCS 1.0" \  
-useDigitMap               true \  
-mgcpActivitiesCount       0 \  
-udpMaxSize                1470 \  
-mgcpActivityId            0 \  
-mgcpVersion                "1.0" \  
-port                      2727
```

SEE ALSO

Endpoints, Gateways

Automatic Settings (CA)

VoIP MGCP CA automatic settings

SYNOPSIS

```
s$Activity_MGCPCA1 agent.pm.automaticSettings.config \
```

DESCRIPTION

The automated settings for the simulated CA.

SUBCOMMANDS

None

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`rsipAtBegin`

If configured to `true`, an RSIP command is awaited at test start.

`restartDelay`

If `rsipAtBegin` is configured `true`, this specifies a restart delay value (Default=0).

`rsipAtEnd`

If configured to `true`, an RSIP command is awaited at test end.

`retransmit`

If configured to `true`, retransmissions are enabled.

`ignoreRecvRetransmit`

If configured `true`, retransmissions are ignored.

`retransmTimerType`

Specifies a timer type as either of the following:

- 0 = A timer value is specified

- 1 = The timer value is calculated internally according to RFC 3435

retransmTimerDuration

If `retransmTimerType` is configured 0, this specifies a retransmission timer value (default = 3000 ms).

transactionTimeout

Specifies a transaction timeout (default = 50000 ms).

waitTimeout

Specifies a wait timeout (default = 50000 ms).

EXAMPLE

```
$Activity_MGCPA1 agent.pm.automaticSettings.config \  
-restartDelay           "0" \  
-ignoreRecvRetransmit  true \  
-rsipAtBegin           true \  
-transactionTimeout    50000 \  
-retransmTimerType     0 \  
-rsipAtEnd             true \  
-waitTimeout           50000 \  
-retransmTimerDuration 3000 \  
-retransmit            true
```

SEE ALSO

Endpoints

VoIP MGCP controlled Endpoint settings.

SYNOPSIS

```
$Activity_MGCPA1 agent.pm.mgcpSettings.endpoints.clear
$Activity_MGCPA1 agent.pm.mgcpSettings.endpoints.appendItem \
```

DESCRIPTION

The properties of endpoint groups controlled by the CA.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`id`

Indicates the endpoint group id.

`set`

Indicates the endpoint set id.

`endpoint`

Indicates the endpoint group name.

NOTE: Sequence generator expressions are supported, for example, `aa1n[1-3]`.

`enabled`

Indicates if the endpoint group is active (`true`) or not (`disabled = false`).

`activity`

The name of the corresponding MGCP Endpoint activity.

EXAMPLE

```
$Activity_MGCPA1 agent.pm.mgcpSettings.endpoints.clear

$Activity_MGCPA1 agent.pm.mgcpSettings.endpoints.appendItem \
-id                               "Endpoint" \
-set                              "Set1" \
-endpoint                         "aaln1" \
-enabled                          true \
-activity                         "Endpoint2"
```

SEE ALSO

Gateways

Gateways

VoIP MGCP controlled GW settings

SYNOPSIS

```
$Activity_MGCPCA1 agent.pm.mgcpSettings.gateways.clear
```

```
$Activity_MGCPCA1 agent.pm.mgcpSettings.gateways.appendItem \
```

DESCRIPTION

The properties of gateways controlled by the CA.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`id`

Indicates the gateway id.

`set`

Indicates the name of the endpoint set.

`endpoint`

Indicates the endpoint group name.

NOTE: Sequence generator expressions are supported, for example, `aa1n[1-3]`.

`gateway`

A controlled gateway.

NOTE: Sequence generator expressions are supported, for example `gw[001-100].ixialab.com`.

EXAMPLE

```
$Activity_MGCPA1 agent.pm.mgcpSettings.gateways.clear

$Activity_MGCPA1 agent.pm.mgcpSettings.gateways.appendItem \
-id "Gateway" \
-set "Set1" \
-endpoint "aaln1" \
-gateway "gw\[001-100\].ixialab.com"
```

SEE ALSO

Endpoints



Scenario Settings

VoIP MGCP Endpoint scenario settings.

SYNOPSIS

```
$Activity_Endpoint2 agent.pm.scenarioSettings.config \
```

DESCRIPTION

Specifies the test scenario file and channel executed by the Tcl script.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`scenarioFile`

The full path to the test scenario file for the activity.

`activeScenarioChannel`

Test scenario channel (0-based index) that is associated with the VoIP MGCP Endpoint activity.
Default=0.

EXAMPLE

```
$Activity_Endpoint2 agent.pm.scenarioSettings.config \  
-scenarioFile           "C:\\Documents and  
Settings\\user11\\Desktop\\MGCP rxf\\MBCP_BC_GWvsCA.tst" \  
-activeScenarioChannel           1
```

SEE ALSO

Activity Endpoint

Execution Settings

VoIP MGCP Endpoint execution settings.

SYNOPSIS

```
$Activity_Endpoint2 agent.pm.executionSettings.config \
```

DESCRIPTION

This object defines the execution settings for the VoIP MGCP Endpoint.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`gracefulRampDown`

If enabled, allows the user to exit gracefully after a session. `Default=1`

`loopCount`

If `loopMode` is 1, this option defines the number of loops that the test performs.

`Default="1"`.

`loopPreDelay`

Delay before first loop (ms). `Default="0"`.

`loopMode`

Defines how many loops are executed for every voice channel corresponding to this activity.

Value	Description
0 (default)	Loop for the entire test duration.
1	Execute a number of loops. Specify the number of loops in <code>loopCount</code> .

`loopMidDelay`



Delay between loops (ms). Default="0".

EXAMPLE

```
$Activity_Endpoint2 agent.pm.executionSettings.config \  
-gracefulRampDown      true \  
-loopMidDelay          0 \  
-loopPreDelay          0 \  
-loopCount             1 \  
-loopMode              1
```

SEE ALSO

Activity Endpoint

Custom Activity Link Settings

VoIP MGCP Endpoint link settings.

SYNOPSIS

```
$Activity_Endpoint1 agent.pm.customActivityLinkSettings.config \
```

DESCRIPTION

This object defines the link settings for the VoIP MGCP Endpoint.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`bhcaObjectiveValue`

The BHCA test objective value. Default="80000".

`bhcaType`

Determines how the BHCA objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	BHCA will be met by specifying the talk time. Specify the talk time in <code>talkTime</code> .
1	BHCA will be met by specifying the number of channels. Specify the number of channels in <code>channelsNo</code> .

`talkTime`

If `bhcaType` is 0, this option specifies the Talk Time that will be used to attain the BHCA test objective. Default="40000".

`channelsNo`

If `bhcaType` is 1, this option specifies the number of channels that will be used to attain the BHCA test objective. Default="100".



`callSetupTime`

Estimated call setup time. Default="500".

`callTeardownTime`

Estimated call teardown time. Default="500".

`interCallDuration`

Inter-call duration. Default="4000".

`cpsObjectiveValue`

The Calls per Second test objective value. Default="100"

`cpsType`

Determines how the CPS objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	CPS objective will be met by specifying the talk time. Based on the the talk time value specified in <code>cpstalkTime</code> , the <code>cpsChannelsNo</code> value is computed.
1	CPS objective will be met by specifying the number of channels. Based on the the channels number value specified in <code>cpsChannelsNo</code> , the <code>cpstalkTime</code> value is computed.

`cpsTalkTime`

If `cpsType` is 0, this option specifies the Talk Time that will be used to attain the CPS test objective. Default="40000".

`cpsChannelsNo`

If `bhcaType` is 1, this option specifies the number of channels that will be used to attain the CPS test objective. Default="100".

`cpsOverheadTime`

Indicates the duration of all other actions on the channel except the talk time and minimum inter-call duration.

`cpsInterCallDuration`

The minimum time interval between the end of a call on a Voice channel and the start of a new call on the same voice channel.

`lpsObjectiveValue`

The `Loops-per-Second` test objective value. Default="100".

`lpsType`

The mode in which the `Loops-per-Second` objective is met, either by specifying the talk time or the number of channels, as follows:

Value	Usage
0 (default)	LPS will be met by specifying the talk time. Specify the talk time in <code>lpstalkTime</code> .
1	LPS will be met by specifying the number of channels. Specify the number of channels in <code>lpschannelsNo</code> .

`lpsTalkTime`

If `lpsType` is configured 0, this is the estimated talk time value. Default="750".

`lpsChannelsNo`

If `lpsType` is configured 1, this is the estimated talk time value. Default="100".

`lpsOverheadTime`

The estimated overhead time. Default="1500".

`lpsInterloopDuration`

The estimated interloop duration. Default="2000".

`lpsActiveChannel`

The referenced test scenario channel.

`activeUsersNo`

The total number of simulated VoIP users (for `ActiveCallers` test objective). Default="100".

`activeUsersObjectValue`



The `ActiveCallers` test objective value. Active callers at any time represent a subset of the total number of users. Default="100".

`activeUserChannel`

The referenced test scenario channel.

`activeUsersTalkTime`

The estimated talk time for the `ActiveCallers` test objective. Default="750".

EXAMPLE

```
$Activity_Endpoint1 agent.pm.customActivityLinkSettings.config \  
-lpsActiveChannel          0 \  
-talkTime                  40000 \  
-lpsTalkTime               750 \  
-interCallDuration         4000 \  
-callSetupTime             500 \  
-lpsChannelsNo             100 \  
-activeUsersNo             100 \  
-cpsType                   0 \  
-lpsObjectiveValue         100 \  
-lpsInterLoopDuration      2000 \  
-cpsOverheadTime           1500 \  
-activeUsersChannel        0 \  
-activeUsersObjectiveValue 100 \  
-cpsInterCallDuration      2000 \  
-lpsOverheadTime           1500 \  
-lpsType                   0 \  
-callTeardownTime          500 \  
-bhcaObjectiveValue        80000 \  
-cpsObjectiveValue         100 \  
-activeUsersTalkTime       750 \  
-cpsTalkTime               750 \  
-bhcaType                  0 \  
-channelsNo                100 \  
-cpsChannelsNo             100
```

SEE ALSO

Activity Endpoint

Simulated Endpoints

Simulated endpoint settings.

SYNOPSIS

```
$Activity_Endpoint2 agent.pm.endpoints.config \
```

DESCRIPTION

This object configures the simulated endpoint settings of an Endpoint activity. This object can reside both under an MGCP GW and MGCP CA object.

SUBCOMMANDS

None.

OPTIONS

`endpointName`

The endpoint name.

Note: Sequence generator expressions are supported, for example, `aaln[1-3]`.

`gwIpAsName`

If configured `true`, the `gwName` parameter is specified as an IP address, otherwise it is specified as a fully qualified host name.

`gwName`

The GW(s) the endpoint is provisioned on.

Note: Sequence generator expressions are supported, for example, `gw[001-100].ixialab.com`.

`destPhoneSource`

Defines the mode in which the destination phone number (for GW) or source phone number (for CA) is specified:

0 = The phone number is specified by the `destPhoneUser` parameter

1 = The phone number is specified by a phone book entry (`destPhonePB` parameter)

2 = The phone number is taken from the CA activity (available for an MGCP GW activity)



destPhonePB

If destPhoneSource is configured with a value of 2, this specifies a phone book entry.

destPhoneUser

If destPhoneSource is configured with a value of 0, this specifies the call destination phone (for GW) or source phone (for CA).

Note: Sequence generator expressions are supported, for example, 170[000000-].

EXAMPLE

```
$Activity_Endpoint2 agent.pm.endpoints.config \  
-gwIpAsName           false \  
-destPhone            "170\[000000-\]" \  
-_labelDestPhone     false \  
-destPhonePreview    "" \  
-gwName               "gw\[01-10\].ixialab.com" \  
-endpointName        "aaln1" \  
-destPhoneType       0 \  
-destPhoneSource     0 \  
-destPhonePB         "<None>" \  
-_phoneNoFromCA      false \  
-destPhoneUser       "170\[000000-\]"
```

SEE ALSO

VoIP MGCPGW, MGCPCA Agent

Data Codecs

VoIP MGCP Endpoint Group Data Codecs

SYNOPSIS

```
$Activity_Endpoint1 agent.pm.codecSettings.dataCodecs.clear  
$Activity_Endpoint1 agent.pm.codecSettings.dataCodecs.appendItem \  

```

DESCRIPTION

Data Codecs configures a data codec object, which is added to the `Codec Settings` list of codecs.

SUBCOMMANDS

None.

OPTIONS

`id`

Codec type. One of the following:

Codec	Description
Rtp2833Events	Named Events Payload format used for carrying DTMF digits and other line and trunk signals as events.
Rtp2833Tones	RTP Payload format that can represent tones consisting of one or more frequencies.

`dPayloadType`

Payload type used for RTP data packets. Default=(see table) min="96" max="127"

Codec	Default value for dPayloadType
Rtp2833Events	100
Rtp2833Tones	101

EXAMPLE

```
$Activity_Endpoint1 agent.pm.codecSettings.dataCodecs.clear

$Activity_Endpoint1 agent.pm.codecSettings.dataCodecs.appendItem \
  -id "Rtp2833Events" \
  -dPayloadType 100
```

SEE ALSO

Codec Settings

Codecs

VoIP MGCP CA/GW Endpoint audio codecs

SYNOPSIS

```
$Activity_Endpoint1 agent.pm.codecSettings.codecs.clear  
  
$Activity_Endpoint1 agent.pm.codecSettings.codecs.appendItem \  

```

DESCRIPTION

Codecs configures an audio codec object, which is added to the `Codec Settings` list of codecs. To add a codec object, use the `appendItem` command.

SUBCOMMANDS

None.

OPTIONS

`id`

The codec type, which is one of the following:

Codec	Description
CodecG711u	G.711 mu-law codec
CodecG711a	G.711 A-law codec
CodecG723x153	G.723.1 codec @ 5.3 kbps
CodecG723x163	G.723.1 codec @ 6.3 kbps
CodecG726x16	G.726 codec @ 16 Kbps
CodecG726x24	G.726 codec @ 24 Kbps
CodecG726x32	G.726 codec @ 32 Kbps
CodecG726x40	G.726 codec @ 40 Kbps
CodecG729A	G.729 Annex-A codec
CodecILBC	iLBC codec

Options for CodecG711u

`dPayloadIn`

Incoming dynamic payload type. Default="0" min="0" max="127".



dPayloadOut

Outgoing dynamic payload type. Default="0" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG711a

dPayloadIn

Incoming dynamic payload type. Default="8" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="8" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG723x153

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 20. Default=20.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG726x16

dPayloadIn

Incoming dynamic payload type. Default="102" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="102" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 20, 40, 60. Default=20.

Options for CodecG726x24

dPayloadIn

Incoming dynamic payload type. Default="103" min="0" max="127".



dPayloadOut

Outgoing dynamic payload type. Default="103" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 30, 60, 90. Default=30.

Options for CodecG726x32

dPayloadIn

Incoming dynamic payload type. Default="104" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="104" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 40, 80, 120. Default=40.

Options for CodecG729

dPayloadIn

Incoming dynamic payload type. Default="18" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="18" min="0" max="127".

cbxFrameSize

Bytes per frame. Must be one of the following: 10, 20, 30, 40, 50, Custom. Default=10.

customFrameSize

If `cbxFrameSize` is Custom, this option configures the custom frame size. Default="120" min="10" max="200".

EXAMPLE

```
$Activity_Endpoint1 agent.pm.codecSettings.codecs.clear
$Activity_Endpoint1 agent.pm.codecSettings.codecs.appendItem \
-id "CodecG711u" \
-dPayloadOut 0 \
-dPayloadIn 0 \
-frameSize 160
```

SEE ALSO

Codec Settings



SDP Settings

VoIP MGCP Endpoint SDP Settings

SYNOPSIS

```
$Activity_Endpoint1 agent.pm.sdpSettings.config \
```

DESCRIPTION

MGCP uses SDP for specification and negotiation of media capabilities of GW endpoints. SDP information is sent using a stream descriptor that specifies a single bi-directional media stream.

SUBCOMMANDS

None.

OPTIONS

useCustomSdp

If `true`, the auto SDP template constructed from the codec list can be overridden by editing the SDP template. Default=`false`.

customSPDP

The SDP string that is used when the Auto option is selected for an SDP descriptor.

EXAMPLE

```
$Activity_Endpoint1 agent.pm.sdpSettings.config \  
-useCustomSDP                false \  
-customSDP                    "v=0 \  
o=- 0 0 IN IP4 <\$VOIP_SignalingIP> \  
s=session \  
c=IN IP4 <\$VOIP_MediaIP> \  
t=0 0 \  
m=audio <\$VOIP_MediaBasePort> RTP/AVP 0 8 100 101 \  
a=rtpmap:0 PCMU/8000 \  
a=rtpmap:8 PCMA/8000 \  
a=rtpmap:100 telephone-event/8000 \  
a=rtpmap:101 tone/8000 \  
a=ptime:20"
```

SEE ALSO

VoIP MGCPGW Agent



RTP Settings

VoIP MGCP Endpoint RTP settings

SYNOPSIS

```
$Activity_Endpoint1 agent.pm.rtpSettings.config \
```

DESCRIPTION

The RTP Settings configure the VoIP MGCP MGC/GW Endpoint group RTP settings.

SUBCOMMANDS

None.

OPTIONS

enableRTP

If true, enables use of RTP to transport the media traffic. Default= false.

enableRTCP

If true, enables use of RTCP for RTP traffic. Default= false.

rtpPort

The port used for audio/video RTP streaming. Default="10000".

chEnableHwAcc

If true, enables hardware acceleration for RTP traffic. Default=false.

enableAdvStatCalc

If true, enables the computation of advanced audio RTP statistics.

enableMDI

Enables computation of MDI DF and MDI MLR statistics.

enableNBExec

If `true`, all RTP functions from a scenario execute in a non-blocking mode, i.e the current function from a channel executes in the background, allowing the execution to continue on that channel with the next script function. Default= `False`.

EXAMPLE

```
$Activity_Endpoint1 agent.pm.rtpSettings.config \  
-enableRTP                false \  
-enableRTCP               false \  
-chEnableHwAcc            true \  
-chDisableHwAcc           false \  
-enableAdvStatCalc        false \  
-enablePerStream          false \  
-enableMDI                false \  
-rtpPort                  "[10000-65535,4]" \  
-enableNBExec             false
```

SEE ALSO

VoIP MGCPGW Agent



Audio Settings

VoIP MGCP MGC/GW Endpoint audio settings

SYNOPSIS

```
$Activity_Endpoint1 agent.pm.audioSettings.config \
```

DESCRIPTION

The Audio Settings configure the VoIP MGCP MGC/GW Endpoint audio RTP settings.

SUBCOMMANDS

None.

OPTIONS

`enableAudio`

If selected, audio script functions are executed, otherwise they are skipped.

`audioClip`

The played audio clip file.

`playTypeAudio`

The mode in which the clip is played.

Value	Usage
0 (default)	The clip is played for clip duration or for the duration of the Talk Time parameter in the case of BHCA/CPS/LPS objectives.
1	The clip is played for a user-defined duration.

`audioDurationUnit`

The play duration unit, which can be milliseconds (0), seconds (1), minutes (2), or hours (3).

`outputLevel`

The output level of the played clip.

`enableTosRtp`

Enables use of TOS/DSCP. Use the `rtpTos` option to specify the TOS/DSCP value. `Default= False`

`rtpTosVal`

The Type of Service (TOS/DSCP) byte setting in the sent RTP packets has one of the following values:

- Best Effort (0x00): Routine service
- Class 1 (0x20): Priority service, Assured Forwarding class 1
- Class 2 (0x40): Immediate service, Assured Forwarding class 2
- Class 3 (0x60): Flash, Assured Forwarding class 3
- Class 4 (0x80): Flash-override, Assured Forwarding class 4
- Express Forwarding (0xA0): Critical-ecp
- Control (0xC0): Internet-control
- Custom: A user-specified value.

`useMOS`

Enables the computation of MOS scores. `Default= False`.

`enableAudioOWD`

If `true`, `IxLoad` computes the One-way Delay metric, a network measurement specifying the amount of time (in ms) that a packet has spent on the network before it was received on the destination side. `Default= False`

`useJitter`

If `true`, enables use of a jitter buffer. `Default= False`.

`jjitMs`

If `useJitter` is 1, this option configures the size of the jitter buffer, in milliseconds. `Default="20"`
`min="1" max="3000"`.

`useJitComp`

If `true`, enables dynamic modification of the jitter buffer size. `Default= False`.

`jitCMs`

If `useJitComp` is 1, this option configures the maximum size in of the jitter buffer, in milliseconds. `Default="1000" min="0" max="3000"`.

`jitCMaxDrop`



If `useJitComp` is 1, this option configures the condition - a maximum number of consecutive packets dropped - that determines the jitter buffer size to be increased.

`enableQoV`

If `true`, this enables QoV P.862 PESQ and P.56 QoV computation. `Default= False`.

`channelTypeQoV`

When `enableQoV` is `true`, this specifies the objective type as either of the following:

- Number of channels (0)
- Percentage (1)

`valueQoV`

When `enableQoV` is `true`, this specifies the number of channels for which PESQ and P.56 QoV metrics are computed (when `channelTypeQoV` is 0). Alternatively this represents the percentage of channels for which PESQ and P.56 QoV metrics are computed (when `channelTypeQoV` is 1).

`unitsQoV`

The channels selection mode, which can be any of the following:

- First channels (0)
- Last channels (1)
- Evenly-spaced channels (2)
- Random (3)

`metricsQoV`

When `enableQoV` is `true`, this specifies the metric that is calculated by the Zion card. Available options are:

- PESQ and P.56 (0)
- PESQ (1)
- P56 (2)

`useSilence`

If `true`, RTP packets containing artificial background noise are sent when no other media (DTMF, MF, real payload, and so on) is sent over the communication channel. `Default= False`.

`silenceMode`

If `useSilence` is 1, this option configures the silence mode.

Value	Usage
0	Null data encoded
1 (default)	Comfort noise.

EXAMPLE

```
$Activity_Endpoint1 agent.pm.audioSettings.config \  
-enableAudio true \  
-audioClip "US_042.wav" \  
-playTypeAudio 0 \  
-audioDurationUnit 1 \  
-audioDuration 10 \  
-outputLevel -20 \  
-enableAudioOWD false \  
-enableTosRtp false \  
-rtpTosVal 32 \  
-useMos false \  
-useJitter false \  
-jitMs 20 \  
-useJitComp false \  
-jitCms 1000 \  
-jitCMaxDrop 7 \  
-enableQoV false \  
-channelTypeQoV 0 \  
-valueQoV 100 \  
-unitsQoV 0 \  
-metricsQoV 0 \  
-useSilence false \  
-silenceMode 1 \  

```

SEE ALSO

VoIP MGCPGW Agent



Other Settings

VoIPMGCP MGC/GW EndpointGroup Other Settings

SYNOPSIS

```
$Activity_Endpoint1 agent.pm.otherSettings.config \
```

DESCRIPTION

This object configures the VoIP MGCP MGC/GW Endpoint Group activity's miscellaneous options.

SUBCOMMANDS

None.

OPTIONS

VOIP_Var0

The VOIP_Var1...VOIP_Var5 and VOIP_IPAddr1...VOIP_IPAddr5 string-type variables supporting generator expressions enable you to generate 10 series of global variables whose values are used at runtime by the simulated MGCP Endpoint Group phones/channels. `Default=""`.

Use the VOIP_Var1...VOIP_Var5 variables to represent phone numbers, and the VOIP_IPAddr1...VOIP_IPAddr5 to represent IP addresses.

VOIP_Var1

See VOIP_Var0.

VOIP_Var2

See VOIP_Var0.

VOIP_Var3

See VOIP_Var0.

VOIP_Var4

See VOIP_Var0.

VOIP_IPAddress0

See VOIP_Var0.

VOIP_IPAddress1

See VOIP_Var0.

VOIP_IPAddress2

See VOIP_Var0.

VOIP_IPAddress3

See VOIP_Var0.

VOIP_IPAddress4

See VOIP_Var0.

EXAMPLE

```
$Activity_Endpoint1 agent.pm.otherSettings.config \  
-VOIP_Var1                " " \  
-VOIP_Var0                " " \  
-VOIP_Var3                " " \  
-VOIP_Var2                " " \  
-VOIP_Var4                " " \  
-VOIP_IPAddress4         " " \  
-VOIP_IPAddress1         " " \  
-VOIP_IPAddress0         " " \  
-VOIP_IPAddress3         " " \  
-VOIP_IPAddress2         " "
```

SEE ALSO

VoIP MGCPGW Agent



35

VoIP SIP Cloud

The IxLoad VoIP SIP Cloud Peer Tcl API consists of a VoIP SIP Cloud Peer agent, with separate APIs for configuring each major aspect of the agent's functionality.

In this chapter:

Limitations	1883
VoIP SIP Cloud API Commands	1884
VoIPSIP Cloud Agent	1885
Settings	1887
SIP Server List	1888

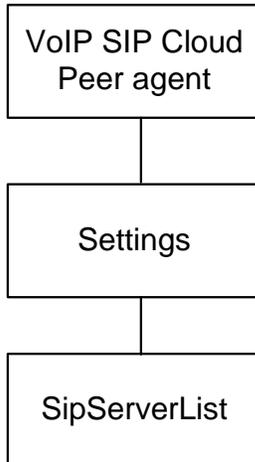
Limitations

The following restrictions and limitations of the VoIP SIP Cloud Peer API exist:

- A VoIPSIP Cloud Peer activity and the VoIPSIP Peer activity it is associated to must reside on the same NetTraffic.
- A network range assigned to a VoIPSIP Cloud necessarily has to be of the Round-Robin IP distribution type.

VoIP SIP Cloud API Commands

The IxLoad VoIP SIP Cloud API commands are organized as shown in the figure below.



API Objects

The following table lists the VoIP SIP Cloud Peer API objects

Object	Description
VoIP SIPCloud Peer Agent	Top-level object defining the VoIP SIP Cloud Peer activity.
Settings	VoIPSIP Cloud IP addressing type settings.
SipServerList	List of SIP Proxy servers emulated by the activity.

VoIPSIP Cloud Agent

VoIPSIPCloud Peer Agent

SYNOPSIS

```
set Activity_VoIPSIPPeer1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType           "VoIPSIPCloud Peer" ]
```

DESCRIPTION

A VoIPSIPCloud Peer agent is added to the `agentList` option of the `ixConfig` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command, as shown in the following example:

```
set Activity_VoIPSipCloud1 [$Traffic1_Network1 activityList.appendItem \  
-protocolAndType           "VoIPSipCloud Peer" ]  
  
$Activity_VoIPSipCloud1 config \  
-enable                    true \  
-name                      "VoIPSipCloud1"  
  
$Activity_VoIPSipCloud1 agent.config \  
-enable                    true \  
-name                      "VoIPSipCloud1"
```

Each member of the list may be separately addressed and modified using the `ixConfig` subcommands. For example, the first agent uses an index of 0 and its name may be modified by using the following command:

```
$Activity_VoIPSipCloud1 agent(0).config -name "VoIPSIP Cloud Peer2"
```

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

The following table lists the statistics published by this object.

Statistic	Description	Per Channel/Global
Dispatched messages	This statistic is incremented when a received message has been parsed and a matching dispatching rule was found for it.	Global
Undispatched messages	This statistic is incremented when a received message has been parsed and a matching dispatching rule was not found for it.	Global
Sent Messages	This statistics is incremented when an message was sent from a server in the cloud.	Global
Parsed Messages	This statistic is incremented when a message was successfully parsed.	Global
Parser errors	This statistic is incremented on the cloud in the data process function, after parsing the received stream, if we receive parser errors when we try to process the stream.	Global
Bytes Transmitted	This statistic is incremented only on the cloud for sent messages. Messages sent by a SIP peer through the cloud will not be accounted to the SIP peer's bytes sent.	Global
Bytes Received	This statistic is incremented only on the cloud for received messages. Messages received by a SIP peer through the cloud will not be accounted to the SIP peer's bytes received.	Global

EXAMPLE

```
set Activity_VoIPSipCloud1 [${Traffic1_Network1 activityList.appendItem \  
-protocolAndType          "VoIPSipCloud Peer" ]  
  
$Activity_VoIPSipCloud1 config \  
-enable                   true \  
-name                     "VoIPSipCloud1"  
  
$Activity_VoIPSipCloud1 agent.config \  
-enable                   true \  
-name                     "VoIPSipCloud1"
```

SEE ALSO

ixConfig

Settings

VoIPSIPCloud Peer Settings

SYNOPSIS

```
$Activity_VoIPSipCloud1 agent.pm.settings.config \  
-ipPreference           0  
$Activity_VoIPSipCloud1 agent.pm.settings.sipServerList.clear
```

DESCRIPTION

Contains the preferred IP addressing type used, IPv4- or IPv6-only.

SUBCOMMANDS

None.

OPTIONS

ipPreference

The preferred IP address type:

0 = Only IPv4 (default)

1 = Only IPv6

EXAMPLE

```
$Activity_VoIPSipCloud1 agent.pm.settings.config \  
-ipPreference           0
```

SEE ALSO

SIP Server List

SIP Proxy Servers List

SYNOPSIS

```
$Activity_VoIPSipCloud1 agent.pm.settings.sipServerList.appendItem \  
-option value
```

DESCRIPTION

A `SipServerList` contains the list of SIP Proxy Servers emulated by the VoIPSIP Cloud Peer. To add `SipServer` objects, use the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

Note: The `SipServerList` class has to be configured alongside the `CloudServers` class of a VoIP SIP Peer that implements the same functionality.

SUBCOMMANDS

None.

OPTIONS

`id`

The SIP server list ID.

`firstIp`

The first IP address in the network range associated with the SIP Proxy server. This is the SIP Proxy server that is located at the cloud boundary.

`name`

The server name (default `sip_server#1` and subsequent strings).

`rangeType`

The range type, which can be only IP for VoIP SIP Cloud peers.

`ipAddr`

The start IP address of the associated network range.

`netMask`

The network mask.

ipStep

The incrementation step of the start IP address (default "0.0.0.1").

attachedInfo

An extra string associated with the proxy, such as for example a domain name (default = sip-test.my-domain.com).

ipCount

The number of hosts (default = 1).

port

The SIP port (default = 5060).

ipType

The IP addressing type, IPv4 or or IPv6.

EXAMPLE

```
$Activity_VoIPSipCloud1 agent.pm.settings.sipServerList.appendItem \  
-id "SipServer" \  
-firstIp "172.20.13.1" \  
-name "sip_server#1" \  
-rangeType "IP" \  
-ipAddr "Network Range 2 in Network1 (172.20.13.1+1)" \  
-ipStep "0.0.0.1" \  
-attachedInfo "sip-test.my-domain.com" \  
-netMask "255.254.0.0" \  
-ipCount "1" \  
-port 5060 \  
-ipType "IPv4"
```

SEE ALSO



36

VoIP SIP Peer

The IxLoad VoIP SIP Peer Tcl API consists of a VoIP SIP Peer agent, with separate APIs for configuring each major aspect of the agent's functionality.

Note: IxLoad supports two different approaches for SIP protocol testing:

- Basic SIP protocol testing support, the simpler of the two supported approaches for SIP testing, is based on SIP Client and SIP Server activities having only limited call flow configuration capabilities.
The Tcl API configuration commands corresponding to this approach are covered in `SIP Protocol Support`.
- Advanced SIP testing support is based on VoIPSIPPeer activities capable of executing more complex, custom protocol message flows.
The Tcl API configuration commands for this approach are covered in this chapter.

In this chapter:

Limitations	1893
VoIP SIP Peer API Commands	1894
VoIP SIP Peer Agent	1897
Codec Settings	1916
Data Codecs	1917
Codecs	1919
Other Settings	1926
Signaling Settings	1929
Edit Contact	1934
RTP Settings	1936
Audio Settings	1938
Video Settings	1942
T.38 Settings	1946
T.30 Settings	1950
Timer Settings	1955
SRTP Settings	1957
Custom Activity Link Settings	1960
Execution Settings	1963
Transfer Address	1966
Scenario Settings	1968
Dial Plan	1969
TLS Settings	1972
TLS Cyphers	1975
Custom Parameters	1976
Advanced Settings	1979
Cloud Servers	1981
Server Rules	1983
Cloud Rules	1985
RuleData	1987



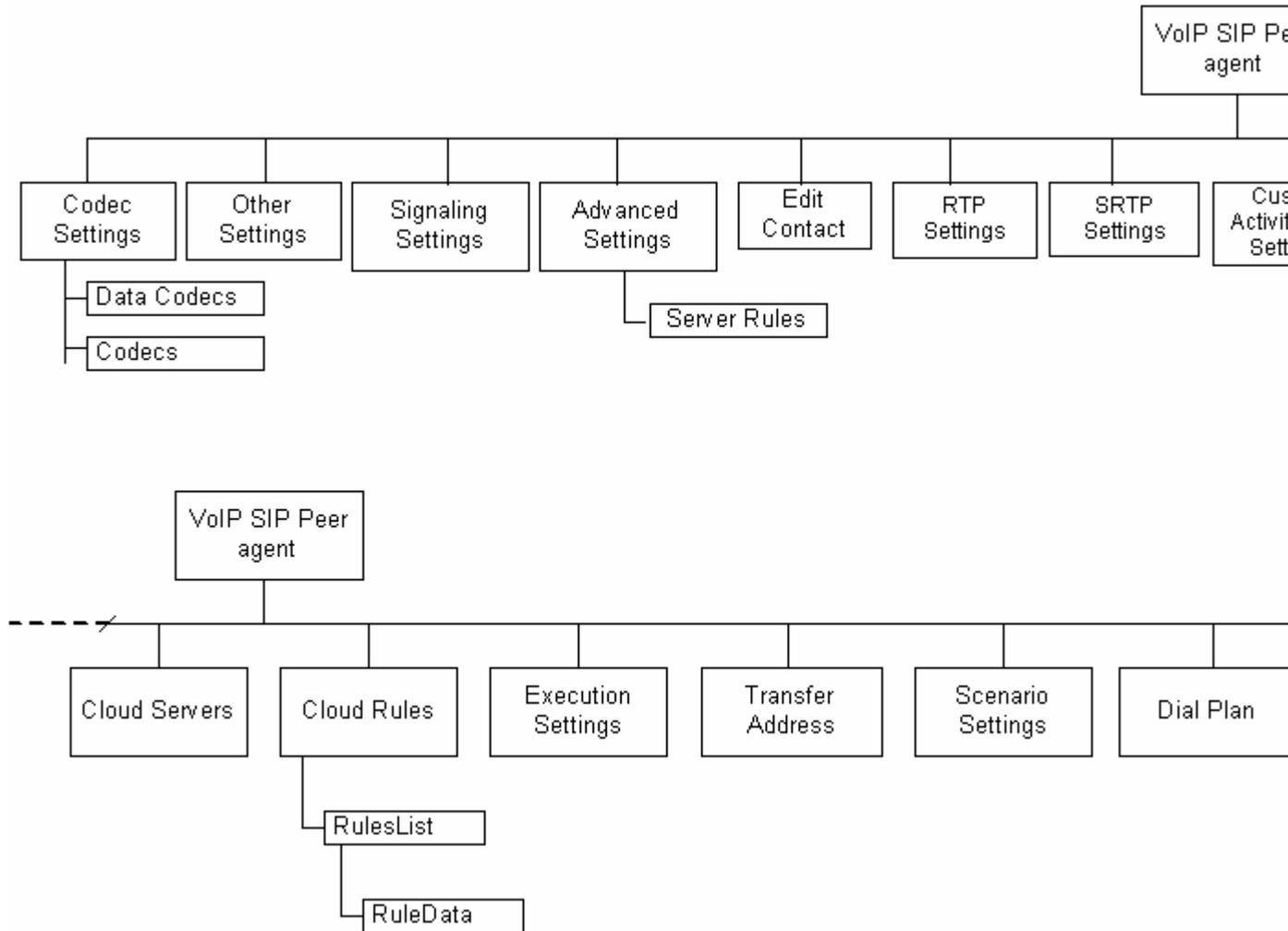
Limitations

The following restrictions and limitations of the VoIP SIP Peer API exist:

- The PhoneBook and other related classes, such as PhoneBookEntry, cannot be edited from the Tcl API.
- Individual VoIP SIP script functions cannot be added and edited from the Tcl API. Instead, you must add and configure the commands in the Scenario Editor, save the test scenario file, then pass it as an argument to the Scenario Settings API class.
- Implementation of the BHCA and CPS objective features relies on two classes, CustomParameters and CustomActivityLinkSettings that have to be configured using the same parameters.

VoIP SIP Peer API Commands

The IxLoad VoIP SIP Peer API commands are organized as shown in the figure below.



VoIP SIP Peer API Objects

The table below summarizes the objects in the VoIP SIP Peer API.

Object	Description
VoIP SIP Peer Agent	Top-level object defining the VoIP SIP Peer activity.
Codec Settings	List of <code>Data Codecs</code> and <code>Codecs</code> objects.
Data Codecs	Data codec with parameters.
Codecs	Audio codec with parameters.
Other Settings	VoIP SIP Peer miscellaneous parameters; corresponds to the Other Settings tab in GUI.

Object	Description
Signaling Settings	VoIP SIP Peer parameters corresponding to the SIP Settings GUI tab.
Edit Contact	Replacement contact information; corresponds to Override Contact control on the Contact Settings tab in GUI.
RTP Settings	RTP transport configuration; corresponds to the RTP Settings GUI tab.
Audio Settings	Audio settings; corresponds to the Audio GUI tab.
T.38 Settings	T.38 IP fax settings; corresponds to the T.38 GUI tab.
T.30 Settings	T.30 settings; corresponds to the T.30 GUI tab.
SRTP Settings	SRTP configuration corresponding to the SRTP Settings GUI tab.
Custom Activity Link Settings	BHCA and CPS objective configuration; corresponds to the Custom Parameters GUI tab.
Execution Settings	Run-time test configuration; corresponds to the Execution Settings GUI tab.
Transfer Address	Configures a SIP transfer address. corresponds to the Transfer Address window opened from the SIP Settings GUI tab.
Scenario Settings	Selects the Test Scenario file; corresponds to the Scenario Settings GUI tab.
Dial Plan	Configures the source, destination, and transfer addresses and phone numbers for the channels/endpoints; corresponds to the Dial Plan GUI tab.
Timer Settings	Configures session refresh and SIP message retransmission settings; corresponds to the Automatic GUI tab.
TLS Settings	TLS transport configuration corresponding to the TLS GUI tab.
TlsCyphers	Configures a list of cyphers supported by the VoIPsIPPeer activity.
CustomParameters	BHCA and CPS objective configuration; corresponds to the Custom Parameters GUI tab.
Advanced Settings	Configures a VoIPSIP Cloud Peer activity associated with the VoIPSIP Peer. Corresponds to the SIP Cloud Settings GUI tab.
CloudServers	Configures the list of SIP Proxy servers emulated by a VoIPSIP Cloud Peer.
ServerRules	Configures a list of rules associated with each emulated SIP Proxy server in the cloud.
CloudRules	Configures a list of dispatching rules that override the default VoIP SIP Cloud rules.
RuleData	Configures the processing operations applied to incoming SIP messages for extracting an overriding dispatching rule. Corresponds to the Edit Cloud Rule GUI.



VoIP SIP Peer Agent

VoIP SIP Peer Agent

SYNOPSIS

```
set Activity_VoIPSIPPeer1 [$SIP_Network1 activityList.appendItem \  
-protocolAndType                "VoIPSIP Peer" ]
```

DESCRIPTION

A VoIP SIP Peer agent is added to the `agentList` option of the `ixConfig` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. See the following example:

```
set Activity_VoIPSIPPeer1 [$SIP_Network1 activityList.appendItem \  
-protocolAndType                "VoIPSIP Peer" ]  
  
$Activity_VoIPSIPPeer1 config \  
-enable                        true \  
-name                          "VoIPSIPPeer1" \  
-enableConstraint              false \  
-userObjectiveValue            1 \  
-constraintValue                100 \  
-userObjectiveType              "channels" \  
-timeline                      $Timeline1  
  
$Activity_VoIPSIPPeer1 agent.config \  
-enable                        true \  
-name                          "VoIPSIPPeer1"
```

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands. For example, the first agent uses an index of 0 and its name may be modified by:

```
$Activity_VoIPSIPPeer1 agent(0).config -name "VoIPSIP Peer new"
```

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

The statistics published by this agent are listed below.

Statistic	Description	Per Channel/Global
VoIPSIP Channels		
Successful Channels	The per polling interval number of COMPLETED channels. A channel is COMPLETED if all the channel loops were COMPLETED.	Global
Warning Channels	The per polling interval number of WARNING channels. A channel is WARNING if all the channel loops were COMPLETED or WARNING and at least one loop had a WARNING result.	Global
Failed Channels	The per polling interval number of FAILED channels. A channel is FAILED if all the channel loops were COMPLETED or WARNING, and at least one loop was FAILED.	Global
Aborted Channels	The per polling interval number of ABORTED channels. A channel is ABORTED if all the channel loops were COMPLETED, WARNING, FAILED, or ABORTED and at least one loop was ABORTED.	Global
Active Channels	The per polling interval number of active channels. Active channels are the channels executing a scenario channel functions flow.	Global
Total Channels	The per polling interval total number of channels, a sum of active and non-active channels.	Global
VoIPSIP Loops		
Successful Channel Loops	The cumulative count of COMPLETED channel loops. A channel loop is COMPLETED if all executed script functions in the corresponding scenario channel produced SKIPPED or COMPLETED results.	Global
Warning Channel Loops	The cumulative count of WARNING channel loops. A channel loop has a WARNING result if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, or WARNING results and at least one script function had a WARNING result.	Global
Failed Channel Loops	The cumulative count of FAILED channel loops. A channel loop is FAILED if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, WARNING, or FAILED results and at least one script function had a FAILED result.	Global

Statistic	Description	Per Channel/Global
Aborted Channel Loops	The cumulative count of ABORTED channel loops. A channel loop is FAILED if all the executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, WARNING, FAILED, or ABORTED results and at least one script function had an ABORTED result.	Global
Total Channel Loops	The cumulative count of executed loops.	Global
Interloop Duration (Avg) [ms]	The time gap between loops.	Global
VoIPSIP Loop Rate		
Loops-per-second	The per polling interval loops-per-second value.	Global
VoIPSIP Calls		
Attempted Calls	The number of initiated calls.	Global
Connected Calls	The number of established calls.	Global
Received Calls	The number of received incoming calls.	Global
Answered Calls	The number of answered incoming calls.	Global
Rejected Calls	The number of rejected incoming calls.	Global
Failed Calls	This statistic is calculated at the end of the script loop as (Attempted Calls - Connected Calls) on originating endpoints and as (Received Calls - Answered Calls) on receiving endpoints. NOTE: This statistic deals only with failed originated/received calls - problems with the initiation part of the call and not with the successful conclusion of the call.	Global
Transferred Calls	The number of originated or incoming calls that were transferred.	Global
Busy Calls	The number of originated or incoming calls that were rejected with busy cause.	Global
Redirected Calls	The number of originated or incoming calls that were redirected.	Global
Calls with Authentication Required	The number of originated or incoming calls that required use of authentication.	Global
Calls Over UDP	The number of completed originated or incoming calls using UDP transport.	Global
Calls Over TCP	The number of completed originated or incoming calls using TCP transport.	Global
Calls Over TLS	The number of completed originated or incoming calls using TLS transport.	Global



Statistic	Description	Per Channel/Global
Calls Over Mixed Transport	The number of completed originated or incoming calls using mixed UDP/TCP transport.	Global
Active Calls	<p>The number of active calls at one time.</p> <p>For the initiator side a call is active after having sent an ACK message and until receiving or sending a 200 OK for BYE message, depending on who is disconnecting the session.</p> <p>For the terminating side, a call is active after having sent an 200 OK for INVITE message (with SDP negotiation having completed successfully) and until sending or receiving a 200 OK for BYE message, depending on the party disconnecting the session.</p>	Global
End Calls Initiated	The count of initiated end call procedures, incremented whenever a SIP BYE message is sent.	Global
End Calls Received	The count of received end call procedures, incremented whenever a SIP BYE message is received.	Global
End Calls Completed	The count of completed end call procedures.	Global
VoIPSIP Call Rates		
Attempted calls/s, Connected Calls/s, Received Calls/s, Answered Calls/s, Rejected Calls/s, Transferred Calls/s, Busy Calls/s, Redirected Calls/s, Calls with Authentication Required/s, Calls Over UDP /s, Calls Over TCP /s, Calls Over TLS /s,	The per polling interval rates for the above <i>VoIPSIP Call</i> statistics.	Global

Statistic	Description	Per Channel/Global
VoIPSIP Call Times		
Call Setup Time (Avg) [ms]	The average duration between the moment a call is initiated (e.g. a SIP INVITE request message is sent/received) and the moment the call is connected (e.g. SIP ACK for INVITE is sent/received).	Global
Talk Time (Avg) [ms]	The average talk time (the duration between the moment the call is connected and the moment the call is disconnected by one of the parties).	Global
Call End Time (Avg) [ms]	The average duration between the moment a call disconnect is initiated (e.g. a SIP BYE request message is sent) and the moment the call is cleared (e.g. a 200 OK response is received)	Global
Total Call Duration (Avg) [ms]	The average call duration. When referring to a single call: (Entire) Call Length = Call Setup-Time + Talk Time + Call Teardown Time.	Global
VoIPSIP Refreshes		
Attempted Session Refreshes	The total number of attempted session refreshes (re-registrations, re-invites, updates treated as refreshes).	
Received Session Refreshes	The total number of received session refreshes (re-invites, updates treated as refreshes).	
Successful Session Refreshes	The total number of successful session refreshes (ACK received for INVITE refresh, 200 Ok to UPDATE sent, or 200 Ok received for REGISTER)	
Failed Session Refreshes	The total number of failed session refreshes (non 2xx responses and transaction failures).	
VoIPSIP SSL Handshake		
Client Hello Sent	The cumulative count of Client Hello messages sent.	Global
Client Hello Received	The cumulative count of Client Hello messages received.	Global
Server Hello Sent	The cumulative count of Server Hello messages sent.	Global
Server Hello Received	The cumulative count of Server Hello messages received.	Global
SSL Negotiations Finished Successfully	The cumulative count of SSL sessions negotiations finished successfully.	Global
SSL Errors Sent	The cumulative count of errors sent.	Global
SSL Errors Received	The cumulative count of errors received.	Global
SSL Alerts Sent	The cumulative count of SSL alerts of all types sent.	Global
SSL Alerts Received	The cumulative count of SSL alerts of all types received.	Global



Statistic	Description	Per Channel/Global
VoIPSIP SSL Throughput		
SSL Throughput (Mbps)	The combined rate at which the peer sends and receives SSL data.	Global
SSL Tx Rate (Mbps), SSL Rx Rate (Mbps)	The total bytes sent/received over the SSL connection, including control and data bytes.	Global
VoIPSIP SSL Warning Alerts		
SSL Alerts Sent (user_canceled)	The cumulative count of <i>User Canceled</i> alerts sent.	Global
SSL Alerts Sent (unsupported_certificate)	The cumulative count of <i>Unsupported Certificate</i> alerts sent.	Global
SSL Alerts Sent (record_overflow)	The cumulative count of <i>Record Overflow</i> alerts sent.	Global
SSL Alerts Sent (no_renegotiation)	The cumulative count of <i>No Renegotiation</i> alerts sent.	Global
SSL Alerts Sent (no_certificate)	The cumulative count of <i>No Certificate</i> alerts sent.	Global
VoIPSIP SSL Fatal Alerts		
SSL Alerts Sent (unknown_ca)	The cumulative count of <i>Unknown CA</i> alerts sent.	Global
SSL Alerts Sent (unexpected_message)	The cumulative count of <i>Unexpected Message</i> alerts sent.	Global
SSL Alerts Sent (protocol_version)	The cumulative count of <i>Protocol Version</i> messages sent.	Global
SSL Alerts Sent (internal_error)	The cumulative count of <i>Internal Error</i> messages sent.	Global
SSL Alerts Sent (insufficient_security)	The cumulative count of <i>Insufficient Security</i> alerts sent.	Global
SSL Alerts Sent (illegal_parameter)	The cumulative count of <i>Illegal Parameter</i> alerts sent.	Global
SSL Alerts Sent (handshake_failure)	The cumulative count of <i>Handshake Failure</i> alerts sent.	Global
SSL Alerts Sent (export_restriction)	The cumulative count of <i>Export Restriction</i> alerts sent.	Global
SSL Alerts Sent (decompression_failure)	The cumulative count of <i>Decompression Failure</i> alerts sent.	Global
SSL Alerts Sent (decode_error)	The cumulative count of <i>Decode Error</i> alerts sent.	Global

Statistic	Description	Per Channel/Global
SSL Alerts Sent (bad_record_mac)	The cumulative count of <i>Bad MAC Record</i> alerts sent.	Global
SSL Alerts Sent (access_denied)	The cumulative count of <i>Access Denied</i> alerts sent.	Global
SSL Alerts Received (unknown_ca)	The cumulative count of <i>Unknown CA</i> alerts received. An <i>Unknown CA</i> alert is sent if a valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or could not be matched with a known, trusted CA.	Global
SSL Alerts Received (unexpected_message)	The cumulative count of <i>Unexpected Message</i> alerts received. An <i>Unexpected Message</i> alert is sent when an SSL peer receives a message that it was not expecting, for example if it received handshake data when it was expecting application data.	Global
SSL Alerts Received (protocol_version)	The cumulative count of <i>Protocol Version</i> alerts received. A <i>Protocol Version</i> alert is sent if the protocol version the client has attempted to negotiate is recognized, but not supported. (For example, old protocol versions might be avoided for security reasons).	Global
SSL Alerts Received (internal_error)	The cumulative count of <i>Internal Error</i> alerts received. An <i>Internal Error</i> alert is sent if an internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue, such as a memory allocation failure.	Global
SSL Alerts Received (insufficient_security)	The cumulative count of <i>Insufficient Security</i> messages received. An <i>Insufficient Security</i> alert is returned instead of a <i>Handshake Failure</i> when a negotiation has failed specifically because the peer requires ciphers more secure than those supported by the client.	Global
SSL Alerts Received (illegal_parameter)	The cumulative count of <i>Illegal Parameter</i> alerts received. An <i>Illegal Parameter</i> alert is sent if a field in the handshake was out of range or inconsistent with other fields.	Global
SSL Alerts Received (handshake_failure)	The cumulative count of <i>Handshake Failure</i> alerts received. The reception of a <i>Handshake Failure</i> alert message indicates that the sender was unable to negotiate an acceptable set of security parameters from the options available.	Global
SSL Alerts Received (decompression_failure)	The cumulative count of <i>Decompression Failure</i> alerts received. A <i>Decompression Failure</i> alert is sent when the decompression function received improper input (data that expanded to an excessive length).	Global

Statistic	Description	Per Channel/Global
SSL Alerts Received (decode_error)	The cumulative count of <i>Decode Error</i> alerts received. A <i>Decode Error</i> alert is sent when a message could not be decoded because a field was out of the specified range or the length of the message was incorrect.	Global
SSL Alerts Received (bad_record_mac)	The cumulative count of <i>Bad MAC Record</i> alerts received. A <i>Bad MAC Record</i> alert is sent when a message is received with an incorrect MAC (Message Authentication Code).	Global
SSL Alerts Received (access_denied)	The cumulative count of <i>Access Denied</i> alerts received. An <i>Access Denied</i> alert is sent if a valid certificate was received, but when access control was applied, the sender decided not to proceed with negotiation.	Global
SSL Alerts Received (export_restriction)	The cumulative count of <i>Export Restriction</i> alerts received. An <i>Export Restriction</i> alert is sent when a negotiation not in compliance with export restrictions was detected; for example, attempting to transfer a 1024 bit ephemeral RSA key for the RSA_EXPORT handshake method.	Global
VoIPSIP SSLv2 Errors		
SSL Errors Sent (unsupported certificate)	The cumulative count of <i>Unsupported Certificate</i> error messages sent.	Global
SSL Errors Sent (undefined error)	The cumulative count of <i>Undefined Error</i> messages sent.	Global
SSL Errors Sent (no cipher)	The cumulative count of <i>No Cipher</i> messages sent.	Global
SSL Errors Sent (bad certificate)	The cumulative count of <i>Bad Certificate</i> error messages sent.	Global
SSL Errors Received (unsupported certificate)	The cumulative count of <i>Unsupported Certificate</i> error messages received. This error is returned when a peer receives a certificate type that it does not support. An <i>Unsupported Certificate</i> error is recoverable for client authentication only.	Global
SSL Errors Received (undefined error)	The cumulative count of <i>Undefined Error</i> messages received. An <i>Undefined Error</i> is returned by the client to the peer when it cannot find a supported cipher or key size that is also supported by the peer. An <i>Undefined Error</i> is not recoverable.	Global

Statistic	Description	Per Channel/Global
SSL Errors Received (no cipher)	The cumulative count of <i>No Cipher</i> messages received. A <i>No Cipher</i> error is returned by the client to the peer when it cannot find a cipher or key size that it supports that is also supported by the peer. <i>A No Cipher</i> error is not recoverable.	Global
SSL Errors Received (no certificate)	The cumulative count of <i>No Certificate</i> error messages received. When a REQUEST-CERTIFICATE message is sent, this error may be returned if the client has no certificate to reply with. <i>A No Certificate</i> error is recoverable for client authentication only.	Global
SSL Errors Received (bad certificate)	The cumulative count of <i>Bad Certificate</i> error messages received. A <i>Bad Certificate</i> error is returned when a certificate is deemed `bad' by the receiving party, either because the signature of the certificate was bad, or the values in the certificate were inappropriate (for example, a name in the certificate did not match the expected name). <i>A Bad Certificate</i> error is recoverable for client authentication only.	Global
VoIPSIP Delays		
Post Dial Delay (Avg) [ms]	The per polling interval time elapsed between sending an INVITE message and receiving an answer from the peer endpoint. This statistic is relevant for the call originating endpoint.	Global
Media Delay TX (Avg) [ms], Media Delay TX (Max) [ms], Media Delay TX (Min) [ms]	The per polling interval average/min/max media delay computed as the time elapsed between the sending of the SIP INVITE and the receiving of the first RTP packet at the call initiating endpoint. The media delay value includes the full call setup time and the time it takes to receive the first media packet at the call initiating endpoint. This statistic is relevant for the call originating endpoint.	Global
Media Delay RX (Avg) [ms], Media Delay RX (Max) [ms], Media Delay RX (Min) [ms]	The per polling interval average/min/max time elapsed between receiving the initial SIP INVITE and receiving the first media packet. The media delay includes the call setup delay and post-pickup delay. This statistic is relevant for the call terminating endpoint.	Global

Statistic	Description	Per Channel/Global
Post-Pickup Delay (Avg) [ms], Post-Pickup Delay (Max) [ms], Post-Pickup Delay (Min) [ms]	<p>The per polling interval average/min/max time elapsed between answering the call and receiving the first media packet.</p> <p>The post pickup delay is computed as time between the sending of the SIP 200 OK response (after receiving the SIP INVITE) and the receiving of the first RTP packet.</p> <p>This statistic is relevant for the call terminating endpoint.</p>	Global
VoIPSIP Registrations		
Attempted Registrations	The cumulative count of generated registrations, retransmissions of REGISTER and REGISTER w/ Auth messages not taken into account. This count includes automatic registration or registration refreshes.	Global
Successful Registrations	The cumulative count of registration messages that completed successfully. This count includes automatic registration or registration refreshes.	Global
Failed Registrations	<p>The cumulative count of failed registrations, defined as an initial REGISTER request followed by a final response from the destination registrar or interim proxies indicating a failure. This count includes automatic registration or registration refreshes.</p> <p>A failure response is described as a 4XX (excepting the 401 and 407 responses), 5XX, or 6XX message.</p> <p>Notes:</p> <ul style="list-style-type: none"> ▪ An unsuccessful registration at loop end is considered a failed registration. ▪ Since registration attempts are often repeated, a failed scenario must identify a failure response associated with the final attempt. 	Global
Attempted De-Registrations	The cumulative count of de-registration attempts computed as the number of sent registrations having a zero Expires header value.	Global
Successful De-Registrations	The cumulative count of de-registration attempts that completed successfully.	Global
Failed De-Registrations	The cumulative count of failed de-registration attempts.	Global
Registration Time (Avg) [ms]	The time elapsed between sending a registration request and receiving a final successful response, in milliseconds.	Global
De-Registration Time (Avg) [ms]	The time elapsed between sending a de-registration request and receiving a final successful response, in milliseconds.	Global

Statistic	Description	Per Channel/Global
VoIPSIP Registration Rates		
Attempted Registrations /sec	The per polling interval rate of attempted registrations, including automatic registration or registration refreshes.	Global
Successful Registrations /sec	The per polling interval rate of successful registrations, including automatic registration or registration refreshes.	Global
Attempted De-Registrations /sec	The per-polling interval rate of attempted de-registrations, including automatic registration or registration refreshes.	Global
Successful De-Registrations /sec	The per polling interval rate of successful registrations.	Global
VoIPSIP SIP Messages		
Requests Sent	The total number of sent SIP requests.	Global
Requests Parsed	The total number of received and parsed SIP requests.	Global
Requests Matched	The total number of matched SIP requests.	Global
Responses Sent	The total number of sent SIP responses.	Global
Responses Parsed	The total number of received and parsed SIP responses.	Global
Responses Matched	The total number of matched SIP responses.	Global
INVITE Requests Sent	The total number of SIP INVITE messages sent by the client.	Global
INVITE Requests Parsed	The total number of received and parsed SIP INVITE requests.	Global
INVITE Requests Matched	The total number of received, parsed, and matched SIP INVITE requests.	Global
ACK Requests Sent	The total number of SIP ACK messages sent by the client.	Global
ACK Requests Parsed	The total number of received and parsed SIP ACK requests.	Global
ACK Requests Matched	The total number of received, parsed, and matched SIP ACK requests.	Global
BYE Requests Sent	The total number of SIP BYE messages sent by the client.	Global
BYE Requests Parsed	The total number of received and parsed SIP BYE requests.	Global
BYE Requests Matched	The total number of received, parsed, and matched SIP BYE requests.	Global



Statistic	Description	Per Channel/Global
BYE Requests Internally Matched	The total number of SIP BYE requests that caused a "disconnect" during an RTP function execution, but were not explicitly expected in the test scenario.	Global
CANCEL Requests Sent	The total number of SIP CANCEL messages sent by the client.	Global
CANCEL Requests Parsed	The total number of received and parsed SIP CANCEL requests.	Global
CANCEL Requests Matched	The total number of received, parsed, and matched SIP CANCEL requests.	Global
OPTIONS Requests Sent	The total number of SIP OPTIONS messages sent by the client.	Global
OPTIONS Requests Parsed	The total number of received and parsed SIP OPTIONS requests.	Global
OPTIONS Requests Matched	The total number of received, parsed, and matched SIP OPTIONS requests.	Global
REGISTER Requests Sent	The total number of sent SIP REGISTER messages.	Global
REGISTER Requests Parsed	The total number of received and parsed SIP REGISTER requests.	Global
REGISTER Requests Matched	The total number of received, parsed, and matched SIP REGISTER requests.	Global
NOTIFY Requests Sent	The total number of sent SIP NOTIFY requests.	Global
NOTIFY Requests Parsed	The total number of received and parsed SIP NOTIFY requests.	Global
NOTIFY Requests Matched	The total number of received, parsed, and matched SIP NOTIFY requests.	Global
SUBSCRIBE Requests Sent	The total number of sent SIP SUBSCRIBE requests.	Global
SUBSCRIBE Requests Parsed	The total number of received and parsed SIP SUBSCRIBE requests.	Global
SUBSCRIBE Requests Matched	The total number of received, parsed, and matched SIP SUBSCRIBE requests.	Global
REFER Requests Sent	The total number of sent SIP REFER requests.	Global
REFER Requests Parsed	The total number of received and parsed SIP REFER requests.	Global
REFER Requests Matched	The total number of received, parsed, and matched SIP REFER requests.	Global
MESSAGE Requests Sent	The total number of sent SIP MESSAGE requests.	Global
MESSAGE Requests Parsed	The total number of received and parsed SIP MESSAGE requests.	Global

Statistic	Description	Per Channel/Global
MESSAGE Requests Matched	The total number of received, parsed, and matched SIP MESSAGE requests.	Global
INFO Requests Sent	The total number of sent SIP INFO requests.	Global
INFO Requests Parsed	The total number of received and parsed SIP INFO requests.	Global
INFO Requests Matched	The total number of received, parsed, and matched SIP INFO requests.	Global
UPDATE Requests Sent	The total number of sent SIP UPDATE requests.	Global
UPDATE Requests Parsed	The total number of received and parsed SIP UPDATE requests.	Global
UPDATE Requests Matched	The total number of received, parsed, and matched SIP UPDATE requests.	Global
PRACK Requests Sent	The total number of sent SIP PRACK requests.	Global
PRACK Requests Parsed	The total number of received and parsed SIP PRACK requests.	Global
PRACK Requests Matched	The total number of received, parsed, and matched SIP PRACK requests.	Global
UNKNOWN Requests Parsed	The total number of received and parsed SIP UNKNOWN requests.	Global
UNKNOWN Requests Matched	The total number of unknown received, parsed and matched SIP request messages. A SIP request message is considered unknown if the method is none of the INVITE, ACK, OPTIONS, BYE, CANCEL, REGISTER, REFER, NOTIFY, SUBSCRIBE, MESSAGE, PRACK, INFO, UPDATE supported methods.	Global
UNKNOWN Responses Parsed	The total number of received and parsed SIP UNKNOWN responses.	Global
UNKNOWN Responses Matched	The total number of unknown received, parsed, and matched SIP response messages. A SIP response message is considered unknown if the response code is other than 100-699.	Global
1xx responses sent	The total number of sent SIP 1xx response messages.	Global
1xx responses parsed	The total number of received and parsed SIP 1xx response messages.	Global
1xx responses matched	The total number of received, parsed, and matched SIP 1xx response messages.	Global
2xx responses sent	The total number of sent SIP 2xx response messages.	Global
2xx responses parsed	The total number of received and parsed SIP 2xx response messages.	Global



Statistic	Description	Per Channel/Global
2xx responses matched	The total number of received, parsed, and matched SIP 2xx response messages.	Global
3xx responses sent	The total number of sent SIP 3xx response messages.	Global
3xx responses parsed	The total number of received and parsed SIP 3xx response messages.	Global
3xx responses matched	The total number of received, parsed, and matched SIP 3xx response messages.	Global
4xx responses sent	The total number of sent SIP 4xx response messages.	Global
4xx responses parsed	The total number of received and parsed SIP 4xx response messages.	Global
4xx responses matched	The total number of received, parsed, and matched SIP 4xx response messages.	Global
5xx responses sent	The total number of sent SIP 5xx response messages.	Global
5xx responses parsed	The total number of received and parsed SIP 5xx response messages.	Global
5xx responses matched	The total number of received, parsed, and matched SIP 5xx response messages.	Global
6xx responses sent	The total number of sent SIP 6xx response messages.	Global
6xx responses parsed	The total number of received and parsed SIP 6xx response messages.	Global
6xx responses matched	The total number of received, parsed, and matched SIP 6xx response messages.	Global
Retransmitted Msgs	The total number of retransmitted SIP messages.	Global
Ignored Re-transmissions	The number of parsed and ignored retransmitted messages following the enabling of the Ignore Retransmissions option.	Global
Requests Orphans	The number of failures to identify a call recipient for SIP request messages when running in a multiple channels per IP:port configuration.	Global
Responses Orphans	The number of failures to identify a call recipient for SIP response messages when running in a multiple channels per IP:port configuration.	Global
VoIPSIP Errors		
Transport Errors	The total number of transport errors, occurring when a SIP message could not be sent due to a socket error or a failed DNS server query.	Global
SIP Call Flows Errors	The total number of SIP call flow errors.	Global

Statistic	Description	Per Channel/Global
SIP Parser Errors	<p>The total number of SIP parser errors. A SIP parser error indicates a message with an invalid request/status line or invalid (malformed) message headers.</p> <p>Note: In case the message has parser errors in the mandatory headers (To, From, CSeq, Call-ID, Via), the message is dropped without the statistic being incremented.</p> <p>In case the message has parser errors in the non-mandatory headers, the parser error statistic is incremented and the execution continues with the malformed message, without the message being dropped.</p> <p>At the same time, a new <i>parser_errors.log</i> log file comprising the most recent 100 entries is created on the port CPU in the /tmp/ folder.</p>	Global
SIP SDP Errors	The total number of SIP SDP errors. A SDP error occurs when an invalid SDP is parsed, when two offers or two answers are received in a row in the same session or when the SDP negotiation fails as described in RFC 3264.	Global
SIP Internal Errors	The total number of SIP internal errors.	Global
Trigger Errors	The total number of trigger errors.	Global
RTP Errors	<p>The total number of RTP related errors, incremented when any RTP script function is failing or exiting on the Warning or Timeout outputs. Possible causes include media sessions that have been closed by the signaling engine, or Generate DTMF/MF/Tone or Detect DTMF/MF/Tone functions that failed.</p> <p>This statistic is also incremented when the signaling engine cannot start a media session, such as when the negotiated codec or the negotiatedptime is unsupported.</p>	Global
Internal Errors	The total number of internal errors.	Global
Timeout Errors	The total number of script functions that have timed out.	Global
VoIPSIP Busy Hour Call Measurements		
BHCA	The per polling interval Busy Hour Call Attempts rate that represents the number of calls initiated in one hour.	Global
BHCC	The per polling interval Busy Hour Call Completions rate that represents the number of calls initiated and connected in one hour.	Global



Statistic	Description	Per Channel/Global
VoIPSIP Other		
Extract Variables Errors	The number of encountered ExtractVariable function errors, occurring when at least one variable could not be extracted.	Global
Requests Sent /s	The number of sent SIP requests per second.	Global
Requests Parsed /s	The number of parsed SIP requests per second.	Global
Requests Matched /s	The number of matched SIP requests per second.	Global
Responses Sent/s	The number of sent SIP responses per second.	Global
Responses Parsed/s	The number of parsed SIP responses per second.	Global
Responses Matched /s	The number of parsed SIP responses per second.	Global
INVITE ACK CANCEL OPTIONS REGISTER NOTIFY SUBSCRIBE REFER MESSAGE INFO UPDATE PRACK UNKNOWN BYE Requests Sent /s	The rate of SIP requests of the given message type sent.	Global
INVITE ACK CANCEL OPTIONS REGISTER NOTIFY SUBSCRIBE REFER MESSAGE INFO UPDATE PRACK UNKNOWN BYE Requests Parsed /s	The rate of SIP requests of the given message type parsed.	Global
BYE Requests Internally Matched /s	The rate of BYE messages received when executing RTP functions.	Global
1xx 2xx 3xx 4xx 5xx 6xx Responses Sent /s	The rate of SIP responses of the given type sent.	Global
1xx 2xx 3xx 4xx 5xx 6xx Responses Parsed /s	The rate of SIP responses of the given type parsed.	Global
Retransmitted Msgs /s	The rate of retransmitted SIP messages.	Global
Requests Orphans /s	The rate of failures to identify a call recipient for SIP request messages when running in a multiple channels per IP:port configuration.	Global
Responses Orphans /s	The rate of failures to identify a call recipient for SIP response messages when running in a multiple channels per IP:port configuration.	Global
Bytes Received /s	The incoming SIP byte rate.	Global

Statistic	Description	Per Channel/Global
TX Messages, TX Messages /s	The number of outbound SIP messages, rate of SIP outbound messages.	Global
TX SIP Msg Length (Avg), TX SIP Msg Length (Min), TX SIP Msg Length (Max)	The Avg/Min/Max outbound SIP message length.	Global
Bytes Transmitted, Bytes Received, Bytes Transmitted /s	The number of SIP bytes sent, received, rate of SIP bytes sent	Global
RX SIP Msg Length (Min), RX SIP Msg Length (Avg), RX SIP Msg Length (Max)	The Avg/Min/Max inbound SIP message length	Global
RX Messages, RX Messages /sec	The number of inbound SIP messages, rate of inbound SIP messages	Global
Triggers Sent, Triggers Sent /s	The number of triggers sent, rate of triggers sent	Global
Triggers Received, Triggers Received /s	The number of triggers received, rate of triggers received	Global
Triggers Bytes Sent, Triggers Bytes Sent /s	The number of trigger bytes sent, rate of trigger bytes sent.	Global
Triggers Bytes Received, Triggers Bytes Received /s	The number of trigger bytes received, rate of trigger bytes received.	Global



Note: Statistics from the *Other* category are only stored in application-generated CSV files and are not displayed in any of the predefined views, but can be assigned to custom statistics views.



EXAMPLE

```
set my_network1 [::IxLoad new ixNetTraffic]

#####
# Activity VoIPSIPPeer2 of NetTraffic my_network1
#####
set Activity_VoIPSIPPeer2 [$my_network1 activityList.appendItem \
    -protocolAndType          "VoIPSIP Peer" ]
#####
# Timelinel for activitiy VoIPSIPPeer2, VoIPSIPPeer3
#####
set Timeline1 [::IxLoad new ixTimeline]

$Timeline1 config \
    -rampUpValue          1 \
    -rampUpType           0 \
    -offlineTime          0 \
    -rampDownTime        60 \
    -standbyTime          0 \
    -iterations           1 \
    -rampUpInterval       1 \
    -sustainTime          80 \
    -timelineType         0 \
    -name                 "Timeline1"

$Activity_VoIPSIPPeer2 config \
    -enable               true \
    -name                 "VoIPSIPPeer2" \
    -enableConstraint     false \
    -userObjectiveValue   1 \
    -constraintValue      100 \
    -userObjectiveType    "channels" \
    -timeline             $Timeline1

$Activity_VoIPSIPPeer2 agent.config \
    -enable               true \
    -name                 "VoIPSIPPeer2"
```

SEE ALSO

ixConfig

Codec Settings

VoIP SIP Peer Codec Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.codecSettings.codecs.appendItem \  
-option value  
  
$Activity_VoIPSIPPeer1 agent.pm.codecSettings.dataCodecs.appendItem \  
-option value
```

DESCRIPTION

Codec Settings contains the list of codecs that will be used by the VoIP SIP peers in the test. Codec Settings is a list of one or more `codec` (audio codec) or `dataCodec` objects. To add `codec` or `dataCodec` objects, use the `appendItem` command.

SUBCOMMANDS

None.

OPTIONS

None.

EXAMPLE

See the examples for `Data Codecs` and `Codecs`.

SEE ALSO

`Data Codecs`, `Codecs`

Data Codecs

VoIP SIP Peer Data Codecs

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.codecSettings.dataCodecs.appendItem \
    -option      value
```

DESCRIPTION

Data Codecs configures a data codec object, which is added to the `Codec Settings` list of codecs.

SUBCOMMANDS

None.

OPTIONS

`id`

Codec type. One of the following:

Codec	Description
Rtp2833Events	Named Events Payload format used for carrying DTMF digits and other line and trunk signals as events.
Rtp2833Tones	RTP Payload format that can represent tones consisting of one or more frequencies.

`dPayloadType`

Payload type used for RTP data packets. Default=(see table) min="96" max="127"

Codec	Default value for dPayloadType
Rtp2833Events	100
Rtp2833Tones	101

EXAMPLE

```
$Activity_VoIPSIPPeer1 agent.pm.codecSettings.dataCodecs.clear

$Activity_VoIPSIPPeer1 agent.pm.codecSettings.dataCodecs.appendItem \
  -id          "Rtp2833Events" \
  -dPayloadType 100

$Activity_VoIPSIPPeer1 agent.pm.codecSettings.dataCodecs.appendItem \
  -id          "Rtp2833Tones" \
  -dPayloadType 101
```

SEE ALSO

Codec Settings

Codecs

VoIP SIP Peer Audio Codec

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.codecSettings.codecs.appendItem \  
-option value
```

DESCRIPTION

Codecs configures an audio codec object, which is added to the `Codec Settings` list of codecs.

SUBCOMMANDS

None.

OPTIONS

`id`

Codec type. One of the following:

Codec	Description
CodecAMR	Adaptive multi-rate codec
CodecG711u	G.711 mu-law codec
CodecG711a	G.711 A-law codec
CodecG723x153	G.723.1 codec @ 5.3 kbps
CodecG723x163	G.723.1 codec @ 6.3 kbps
CodecG726x16	G.726 codec @ 16 Kbps
CodecG726x24	G.726 codec @ 24 Kbps
CodecG726x32	G.726 codec @ 32 Kbps
CodecG726x40	G.726 codec @ 40 Kbps
CodecG729A	G.729 Annex-A codec
CodecILBC	Internet Low Bit Rate Codec

Options for CodecAMR

`dPayloadIn`

Incoming dynamic payload type. Default="98" min="0" max="127".



dPayloadOut

Outgoing dynamic payload type. Default="98" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 14. Default=14.

payloadFormat

Payload format.

Value	Usage
0 (default)	Bandwidth-efficient format
1	Octet-aligned format

mode

Codec bit rate. One of the following:

Mode	Description
0 (default)	4.75 kbps
1	5.15 kbps
2	5.90 kbps
3	6.70 kbps
4	7.40 kbps
5	7.95 kbps
6	10.20 kbps
7	12.20 kbps

Options for CodecG711u

dPayloadIn

Incoming dynamic payload type. Default="0" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="0" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG711a

dPayloadIn

Incoming dynamic payload type. Default="8" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="8" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG723x153

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 20. Default=20.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG726x16

dPayloadIn

Incoming dynamic payload type. Default="102" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="102" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 20, 40, 60. Default=20.

Options for CodecG726x24

dPayloadIn

Incoming dynamic payload type. Default="103" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="103" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 30, 60, 90. Default=30.

Options for CodecG726x32

dPayloadIn

Incoming dynamic payload type. Default="104" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="104" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 40, 80, 120. Default=40.

Options for CodecG726x40

dPayloadIn

Incoming dynamic payload type. Default="105" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="105" min="0" max="127".



byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 50, 100, 150. Default=50.

Options for CodecG729

dPayloadIn

Incoming dynamic payload type. Default="18" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="18" min="0" max="127".

cbxFrameSize

Bytes per frame. Must be one of the following: 10, 20, 30, 40, 50, Custom. Default=10.

customFrameSize

If `cbxFrameSize` is `Custom`, this option configures the custom frame size. Default="120" min="10" max="200".

Options for CodecILBC

dPayloadIn

Incoming dynamic payload type. Default="97" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="97" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 38, 50, Custom. Default=38.

EXAMPLE

```
$Activity_VoIPPeer1 agent.pm.codecSettings.codecs.clear

$Activity_VoIPPeer1 agent.pm.codecSettings.codecs.appendItem \
-id          "CodecG711u" \
-dPayloadOut 0 \
-dPayloadIn  0 \
-frameSize   160

$Activity_VoIPPeer1 agent.pm.codecSettings.codecs.appendItem \
-id          "CodecG711a" \
-dPayloadOut 8 \
-dPayloadIn  8 \
-frameSize   160
```

SEE ALSO

Codec Settings

Other Settings

VoIP SIP Peer Other Settings

SYNOPSIS

```
$Activity_VoIPSIP Peer1 agent.pm.otherSettings.config \  
-option          value
```

DESCRIPTION

This object configures the VoIP SIP Peer activity's miscellaneous options.

SUBCOMMANDS

None.

OPTIONS

VOIP_Var0

The VOIP_Var1...VOIP_Var5 and VOIP_IPAddr1...VOIP_IPAddr5 string-type variables supporting generator expressions enable you to generate 10 series of global variables whose values are used at runtime by the simulated endpoints/channels. `Default=""`.

Use the VOIP_Var1...VOIP_Var5 variables to represent phone numbers, and the VOIP_IPAddr1...VOIP_IPAddr5 to represent IP addresses.

VOIP_Var1

See VOIP_Var0.

VOIP_Var2

See VOIP_Var0.

VOIP_Var3

See VOIP_Var0.

VOIP_Var4

See VOIP_Var0.

VOIP_IPAddress0

See VOIP_Var0.

VOIP_IPAddress1

See VOIP_Var0.

VOIP_IPAddress2

See VOIP_Var0.

VOIP_IPAddress3

See VOIP_Var0.

VOIP_IPAddress4

See VOIP_Var0.

ipPreference

Type of addressing you want to use on the subnet that the VOIP SIP Peer runs on.

Value	Usage
0 (default)	IPv4
1	IPv6

bUseStun

Enables use of a STUN server.

Value	Usage
0 (default)	Disabled
1	Enabled. Configure the STUN server's address and port number in <code>stunAddr</code> and <code>stunPort</code> .

stunAddr

If `bUseStun` is 1, this option configures the STUN server's address. You can include sequence generators in this field to generate multiple addresses. The STUN server address must be an IPv4 address. Default="127.0.0.1".

stunPort

If `bUseStun` is 1, this option configures the STUN port number. You can include sequence generators in this field to generate multiple port numbers. Default="3478".

EXAMPLE

```
$Activity_VoIPSIPPeer1 agent.pm.otherSettings.config \  
-ipPreference          0 \  
-stunAddr              "127.0.0.1" \  
-stunPort              "3478" \  
-bUseStun              false \  
-VOIP_Var1            "" \  
-VOIP_Var0            "" \  
-VOIP_Var3            "" \  
-VOIP_Var2            "" \  
-VOIP_Var4            "" \  
-VOIP_IPAddress4      "" \  
-VOIP_IPAddress1      "" \  
-VOIP_IPAddress0      "" \  
-VOIP_IPAddress3      "" \  
-VOIP_IPAddress2      ""
```

SEE ALSO

Signaling Settings

VoIP SIP Peer Signaling Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.signalingSettings.config \  
-option value
```

DESCRIPTION

This object defines the VoIP Peer SIP settings.

SUBCOMMANDS

None.

OPTIONS

enableSIP

Enables use of SIP signaling for the VoIPSIP Peer activity.

0 = SIP disabled

1 = SIP enabled (default)

port

Port used for SIP. You can use Sequence Generators in this field to generate multiple port numbers. See the Sequence Generator appendix for more information. Default="[5060-]".

Note: Valid port numbers are between 1000 and 65534.

realm

SIP registration realm (for User Agent Client (UAC) authentication with a registrar). Default="" (null).

user

User name of the emulated device (for User Agent Client (UAC) authentication with a registrar). Default="Anonymous".

passwd

SIP registration password (for User Agent Client (UAC) authentication with a registrar). Default="" (null).



`enableTos`

Enables use of TOS/DSCP. Use the `tos` option to specify the TOS/DSCP value.

0 = TOS disabled (default)

1= TOS enabled

`tosVal`

If `enableTos` is 1, this option sets the value of the TOS bits.

Value	Usage
0 (default)	Best Effort (0x00)
1	Class 1 (0x20)
2	Class 2 (0x40)
3	Class 3 (0x60)
4	Class 4 (0x80)
5	Express Forwarding (0xA0)
6	Control (0xC0)

`useServer`

Enables use of a proxy server.

0 = disabled (default)

1 = enabled

`srvAddr`

If `useServer` is 1, this option configures the proxy server address. You can use Sequence Generators in this field to generate multiple addresses. See the Sequence Generator appendix for more information. (Default = "").

`srvPort`

If `useServer` is 1, this option configures the proxy server port number. You can use Sequence Generators in this field to generate multiple port numbers. See the Sequence Generator appendix for more information. (Default="5060")

`srvDomain`

If `useServer` is 1, this option configures the proxy server domain or local IP address. Default="" (null).

outboundProxy

Enables the use of an outbound Proxy that receives requests from a client, even though it may not be the server resolved by the Request-URI.

0 = disabled (default)

1 = enabled

registrarSrv

Enables the use of a Registrar Proxy, a server that accepts register requests and places the information it receives in requests into the location service for the domain it handles.

0 = disabled (default)

1 = enabled

ovrContact

If set to 1, the default Contact message header (AUTO_CONTACT) is ignored and the settings specified by Edit Contact are used

0 = disabled (default)

1 = enabled

ovrDest

If set to 1, this parameter enables you to specify a new destination host setting overriding the default setting.

0 = disabled (default)

1 = enabled

ovrDestHostPort

If set to 1, this parameter enables you to specify a new destination port setting overriding the default setting.

0 = disabled (default)

1 = enabled

nUdpMaxSize

Specifies the maximum SIP message size, beyond which messages are truncated, when the used transport protocol is UDP. Min="1024" Max="4000" Default="1024".



telURISource, telURIDest

If configured `true`, a tel URI is used for source and destination.

enableRetransmissions

If configured `true`, enables the retransmission of certain SIP messages, both requests and responses, for script functions pertaining to the activity. Retransmission is a mechanism whereby messages are re-sent with a pattern, until either a response message is received or a maximum timeout value is reached.

T1, T2

Specifies the retransmission timers.

ignoreRetransmissions

When selected, this option determines ignoring all the received retransmissions.

ovrTrans

If configured `true`, the preferred transport type for SIP messages can be selected as either of the following, overriding the scenario-level settings:

ovrTransOption

If `ovrTrans` is configured `true`, this parameter specifies the preferred transport as listed in the table below.

0 = UDP only

1 = TCP only

2 = UDP

3 = TCP

tcpWriteImmediate

If configured `true`, SIP messages are sent immediately instead of being queued.

ovrTimeout

If configured `true`, the timeout of the Wait Response (...) and the Wait ACK script functions is specified by the global `64*T1` value, instead of the function-level value.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.signalingSettings.config \  
-tcpWriteImmediate          false \  
-telURISource               false \  
-enableSIP                  true \  
-srvPort                    "5060" \  
-port                       "\[5060-\]" \  
-realm                      "" \  
-ovrDest                    false \  
-nUdpMaxSize                1024 \  
-srvDomain                  "" \  
-telURIDest                 false \  
-ovrTimeout                 false \  
-enableRetransmissions     false \  
-enableTos                  false \  
-srvAddr                    "" \  
-ovrDestHostPort           "" \  
-passwd                     "" \  
-T2                         4000 \  
-T1                         500 \  
-outboundProxy              false \  
-user                       "Anonymous" \  
-useServer                  false \  
-registrarSrv               false \  
-tosVal                     0 \  
-ovrContact                 false \  
-ovrTrans                   false \  
-useDnsSrv                  false \  
-ovrTransOption             0 \  
-ignoreRetransmissions     true
```

SEE ALSO

Edit Contact

VoIP SIP Peer Edit Contact

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.editContact.config \  
-option value
```

DESCRIPTION

If the `ovrContact` option in `Contact Settings` is enabled, `Edit Contact` defines the replacement contact information.

SUBCOMMANDS

None.

OPTIONS

`useDomainName`

Domain name to be used.

Value	Usage
0 (default)	Use the domain associated with the source IP address
1	Use the domain specified in <code>domainName</code> .

`domainName`

If `useDomainName` is 1, this option specifies the domain name.
Default="mysipdomain.ixiacom.com"

`_useEPb`

Source of phone number.

Value	Usage
0 (default)	Use the phone number specified in <code>_ePhone</code> .
1	Use the phone number specified by a Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (`_useEPb=1`). The generated tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the tcl code.

`_ePhone`

If `_useEPb` is 0, this identifies the phone number to be used. Default="160[00000000-]"

`_ckETelURI`

Enable usage of Tel URI parameters.

0 = disabled (default)

1 = enabled. Specify the Tel URI parameters in `_eTelURIParams`.

`_eTelURIParams`

If `_ckETelURI` =1, this option specifies the Tel URI parameters.

Default="phone-context=example.com".

EXAMPLE

```
$Activity_VoIPSIPPeer1 agent.pm.editContact.config \  
-_useEPb                0 \  
-domainName             "mysipdomain.ixiacom.com" \  
-_eBp                   "&lt;None&gt;" \  
-_ePhone                "160\[00000000-\  
-ePhoneType             0 \  
-_eTelURIParams        "phone-context=example.com" \  
-useDomainName          0 \  
-editTelPar             "" \  
-ePhone                 "160\[00000000-\  
-_ckETelURI             false
```

SEE ALSO



RTP Settings

VoIP SIP Peer RTP Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.rtpSettings.config \  
-option value
```

DESCRIPTION

RTP Settings configures the VoIPSIPPeer RTP transport settings.

SUBCOMMANDS

None.

OPTIONS

enableRTP

Enables use of RTP to transport the media traffic.

0 = disabled (default)

1 = enabled

rtpPort

RTP port number. Default="10000".

Note: Valid port numbers are between 1000 and 65534.

enableRTCP

Enables the sending and receiving of RTCP packets.

chEnableHwAcc

If true, enables hardware acceleration for RTP traffic. Default=false.

enableAdvStatCalc

Enables the computation of advanced RTP statistics.

enablePerStream

Enables computation of per-stream statistics.

enableMDI

Enables computation of MDI DF and MDI MLR statistics.

enableNBExec

If `true`, all RTP functions from a scenario execute in a non-blocking mode, i.e the current function from a channel executes in the background, allowing the execution to continue on that channel with the next script function. Default= `False`.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.rtpSettings.config \  
-enableRTP true \  
-enableRTCP false \  
-enableMDI false \  
-chEnableHwAcc true \  
-chDisableHwAcc false \  
-enableAdvStatCalc false \  
-enablePerStream false \  
-rtpPort "[10000-65535,4]" \  
-enableNBExec false
```

SEE ALSO

Audio Settings

VoIPSIP Peer audio settings

SYNOPSIS

```
$Activity_VoIPSipPeer1 agent.pm.audioSettings.config \
```

DESCRIPTION

The Audio Settings configure the VoIPSIP Peer audio RTP settings.

SUBCOMMANDS

None.

OPTIONS

`enableAudio`

If selected, audio script functions are executed, otherwise they are skipped.

`audioClip`

The played audio clip file.

`playTypeAudio`

The mode in which the clip is played.

Value	Usage
0 (default)	The clip is played for clip duration or for the duration of the Talk Time parameter in the case of BHCA/CPS/LPS objectives.
1	The clip is played for a user-defined duration.

`audioDurationUnit`

The play duration unit, which can be milliseconds (0), seconds (1), minutes (2), or hours (3).

`outputLevel`

The output level of the played clip.

`enableTosRtp`

Enables use of TOS/DSCP. Use the `rtpTos` option to specify the TOS/DSCP value. `Default= False`

`rtpTosVal`

The Type of Service (TOS/DSCP) byte setting in the sent RTP packets has one of the following values:

- Best Effort (0x00): Routine service
- Class 1 (0x20): Priority service, Assured Forwarding class 1
- Class 2 (0x40): Immediate service, Assured Forwarding class 2
- Class 3 (0x60): Flash, Assured Forwarding class 3
- Class 4 (0x80): Flash-override, Assured Forwarding class 4
- Express Forwarding (0xA0): Critical-ecp
- Control (0xC0): Internet-control
- Custom: A user-specified value.

`useMOS`

Enables the computation of MOS scores. `Default= False`.

`enableAudioOWD`

If `true`, IxLoad computes the One-way Delay metric, a network measurement specifying the amount of time (in ms) that a packet has spent on the network before it was received on the destination side. `Default= False`

`useJitter`

If `true`, enables use of a jitter buffer. `Default= False`.

`jitMs`

If `useJitter` is 1, this option configures the size of the jitter buffer, in milliseconds. `Default="20"`
`min="1" max="3000"`.

`useJitComp`

If `true`, enables dynamic modification of the jitter buffer size. `Default= False`.

`jitCMs`

If `useJitComp` is 1, this option configures the maximum size in of the jitter buffer, in milliseconds. `Default="1000" min="0" max="3000"`.

`jitCMaxDrop`



If `useJitComp` is 1, this option configures the condition - a maximum number of consecutive packets dropped - that determines the jitter buffer size to be increased.

`enableQoV`

If `true`, this enables QoV P.862 PESQ and P.56 QoV computation. `Default= False`.

`channelTypeQoV`

When `enableQoV` is `true`, this specifies the objective type as either of the following:

- Number of channels (0)
- Percentage (1)

`valueQoV`

When `enableQoV` is `true`, this specifies the number of channels for which PESQ and P.56 QoV metrics are computed (when `channelTypeQoV` is 0). Alternatively this represents the percentage of channels for which PESQ and P.56 QoV metrics are computed (when `channelTypeQoV` is 1).

`unitsQoV`

The channels selection mode, which can be any of the following:

- First channels (0)
- Last channels (1)
- Evenly-spaced channels (2)
- Random (3)

`metricsQoV`

When `enableQoV` is `true`, this specifies the metric that is calculated by the Zion card. Available options are:

- PESQ and P.56 (0)
- PESQ (1)
- P56 (2)

`useSilence`

If `true`, RTP packets containing artificial background noise are sent when no other media (DTMF, MF, real payload, and so on) is sent over the communication channel. `Default= False`.

`silenceMode`

If `useSilence` is 1, this option configures the silence mode.

Value	Usage
0	Null data encoded
1 (default)	Comfort noise.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.audioSettings.config \  
-enableAudio true \  
-audioClip "US_042.wav" \  
-playTypeAudio 0 \  
-audioDurationUnit 1 \  
-audioDuration 10 \  
-outputLevel -20 \  
-enableAudioOWD false \  
-enableTosRtp false \  
-rtpTosVal 32 \  
-useMos false \  
-useJitter false \  
-jitMs 20 \  
-useJitComp false \  
-jitCMs 1000 \  
-jitCMaxDrop 7 \  
-enableQoV false \  
-channelTypeQoV 0 \  
-valueQoV 100 \  
-unitsQoV 0 \  
-activityIdQoV 0 \  
-metricsQoV 0 \  
-useSilence false \  
-silenceMode 1 \  

```

SEE ALSO

Video Settings

VoIP SIP Peer Video Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.VideoSettings.config \  
-option value
```

DESCRIPTION

Video Settings configures the VoIPSIP Peer's video settings.

SUBCOMMANDS

None.

OPTIONS

`enableVideo`

Enables use of video as media traffic.

0 = disabled (default)

1 = enabled

`videoClip`

Name of the video file. Default = "Fire_avc.mp4"

`playTypeVideo`

Determines parameters for running video. Following values are available:

Value	Usage
0 (default)	Play for clip duration
1	Play for specified duration.
2	Conference mode

`videoDuration`

If `playTypeVideo = 1`, determines duration of video. Maximum value = 25920000.

`videoDurationUnit`

Unit of duration. Following values are available:

Value	Usage
0	milliseconds
1	seconds
2	minutes
3	hours

`useConference`

If `playTypeVideo = 2`, enables use of conference mode. Following values are available:

Value	Usage
0	All speak
1	Sequential
2	Random

`confVideoDuration`

If `playTypeVideo = 2`, enables selection of conference video duration.

`confVideoDurationUnit`

If `playTypeVideo = 2`, enables selection unit of conference video duration. The following values are available:

Value	Usage
0	milliseconds
1	seconds
2	minutes
3	hours

`confDuration`

If `playTypeVideo = 2`, enables selection of conference audio duration.

`confDurationUnit`

If `playTypeVideo = 2`, enables selection unit of conference audio duration. The following values are available:

Value	Usage
0	milliseconds



Value	Usage
1	seconds
2	minutes
3	hours

`enableTosVideo`

Enables use of TOS/DSCP. Use the `tosVideo` option to specify the TOS/DSCP value.

`tosVideo`

The following values are available:

Value	Usage
0	Best Effort (0x00)"
1	Class 1 (0x20)
2	Class 2 (0x40)
3	Class 3 (0x60)
4	Class 4 (0x80)
5	Express Forwarding (0xA0)
6	Control (0xC0)
7	Custom

`useMosVideo`

Enables computation of MOS.

0 = disabled (default)

1 = enabled

Note: If MOS computation is enabled, the `enableVideoOWD` option also has to be enabled.

`enableVideoOWD`

If enabled, the One-way Delay metric is computed, a network measurement specifying the amount of time (in ms) that a packet has spent on the network before it was received on the destination side.
Default = disabled.

`ignoreHintTrack`

If enabled, the hint track present in the video clip is ignored. The video streaming uses a new hint track which is recreated using one of the packetization modes defined by `hintTrackType`. By default it is disabled.

hintTrackType

Allows to select the packetization mode. The following values are available:

Value	Usage
0 (default)	Single NAL Unit
1	STAP-A, with FU-A fragmentation

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.videoSettings.config \  
-rotationScheme 0 \  
-confDuration 1 \  
-useMosVideo false \  
-enableVideoOWD false \  
-ignoreHintTrack false \  
-enableTosVideo true \  
-enableVideo true \  
-videoClip "Fire_avc.mp4" \  
-useH323AdvancedSettings false \  
-videoDuration 5 \  
-confVideoDurationUnit 1 \  
-useConference false \  
-confDurationUnit 1 \  
-confVideoDuration 1 \  
-videoDurationUnit 1 \  
-hintTrackType 1 \  
-fmt " " \  
-rtptime " " \  
-playTypeVideo 0 \  
-tosValVideo 32
```

SEE ALSO

T.38 Settings

VoIP SIP Peer T.38 Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.Fax(T.38)Settings.config \  
-option value
```

DESCRIPTION

T.38 Settings configures the VoIP SIPPeer's fax T.38 settings.

SUBCOMMANDS

None.

OPTIONS

enableT38

Enables use of 'T.38 Fax Session' script function.

0 = disabled (default)

1 = enabled

t38Port

The T.38 listening port. Default = "40000". This parameter specifies a valid port (1000-65535) or simple sequence generator expression (e.g. [1000-2000,2])

faxImage

Fax image to be sent. Default = "Ixia2Pages.tif"

t38TransportType

The transport protocol used for carrying the T.38 traffic. Default = "1"

The following values are available

Value	Usage
0	TCP
1	UDP

t38UdpEncapsulation

If `t38TransportType = 1`, `t38UdpEncapsulation` defines the protocol used to encapsulate T.38 messages. The following values are available:

Value	Usage
0	UDPTL
1	RTP

`t38PayloadType`

The payload type identifier. Minimum = 0, Maximum = 127, and Default = 102

`useFaxVersion`

If enabled, allows selecting the T.38 protocol version.

`faxVersion`

If `useFaxVersion` is enabled, used to identify the T.38 protocol version, 0, 1, 2, or 3 (default = 0).

`useT38MaxBitrate`

If enabled, allows selecting the maximum fax transmission rate.

`t38MaxBitrate`

The maximum fax transmission rate supported by the endpoint (default = 5). The following values are allowed:

Value	Usage
0	2.4 kbps
1	4.8 kbps
2	7.2 kbps
3	9.6 kbps
4	12 kbps
5 (default)	14.4 kbps
6	16.8 kbps
7	19.2 kbps
8	21.6 kbps
9	24 kbps
10	26.4 kbps
11	28.8 kbps
12	31.2 kbps



Value	Usage
13	33.6 kbps

`useT38RateMgmt`

If enabled, allows selecting the fax rate management model.

`t38RateMgmt`

The fax rate management model as defined in T.38. Following values are allowed:

Value	Usage
0	Transferred TCF
1	Local TCF

`useErrorRecoverySchema`

If enabled, allows selecting the desired error correction scheme.

`errorRecoverySchema`

The desired error correction scheme. The following values are allowed:

Value	Usage
0 (default)	Redundancy
1	FEC

`useT38MaxDatagramSize`

If enabled, allows selecting the maximum datagram size.

`t38MaxDatagramSize`

The maximum datagram size (`default = 256`), which represents the maximum number of bytes that can be stored on the remote device before an overflow condition occurs. Minimum = 0, Maximum = 256.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.t38Settings.config \  
-enableT38 true \  
-t38TranscodingMMR false \  
-t38UdpEncapsulation 0 \  
-useT38MaxBitrate true \  
-t38RateMgmt 0 \  
-t38TranscodingJBIG false \  
-t38TransportType 1 \  
-t38Port "40000" \  
-t38FillBitRemoval 0 \  
-faxVersion 0 \  
-useT38FillBitRemoval false \  
-useT38RateMgmt true \  
-faxImage "Ixia2Pages.tif" \  
-useT38MaxBufferSize false \  
-errorRecoverySchema 0 \  
-t38MaxDatagramSize 256 \  
-t38MaxBufferSize 200 \  
-useFaxVersion true \  
-useT38MaxDatagramSize true \  
-t38MaxBitrate 5 \  
-t38PayloadType 102 \  
-useErrorRecoverySchema true
```

SEE ALSO

T.30 Settings

VoIP SIP Peer T.30 Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.Fax(T.30)Settings.config \  
-option value
```

DESCRIPTION

T.30 Settings configures the VoIP SIPPeer's fax T.30 settings.

SUBCOMMANDS

None.

OPTIONS

t30StationId

The fax station's identifier sent in CSI, TSI and CIG. Required valid station ID or sequence generator expression (e.g. '5551[000-]'). Default = "5551[000-]"

t30SendCoding

The highest coding scheme available to compress the page data when sending. The following values are available:

Value	Usage
0	MH
1	MR
2 (Default)	MMR

t30SendDataRate

The data rate for sending. The following values are available:

Value	Usage
0	V.27 ter 2.4
1	V.27 ter 4.8
2	V.17 7.2
3	V.17 9.6
4	V.17 12
5(default)	V.17 14.4

Value	Usage
6	V.29 7.2
7	V.29 9.6
8	V.34 16.8
9	V.34 19.2
10	V.34 21.6
11	V.34 24
12	V.34 26.4
13	V34 28.8
14	V.34 31.2
15	V34 33.6

`t30SendPageSize`

The page size for sending. The following values are available:

Value	Usage
0	A4 (210x297 mm)
1	B4 (255x364 mm)
2	A3 (297x420 mm)

`t30SendMSLT`

The minimum transmission time of one coded scan line. Default = 0

The following values are available:

Value	Usage
0 (default)	Auto (based on DIS)
1	5 ms T7.7 = T3.85
2	10 ms T7.7 = 1/2 T3.85
3	10 ms T7.7 = T3.85
4	20 ms T7.7 = 1/2 T3.85
5	20 ms T7.7 = T3.85
6	40 ms T7.7 = 1/2 T3.85
7	40 ms T7.7 = T3.85

`t30SendProtocol`

The protocol used for fax sending. The following values are available



Value	Usage
0	non-ECM
1 (default)	ECM.

t30SendResolution

The horizontal and vertical resolution of the page image. The following values are available

Value	Usage
0 (default)	R8x3.85 lines/mm
1	R8x7.7 lines/mm
2	R8x15.4 lines/mm
3	200x200 dots/inch

sendCNG

If enabled, CNG message is sent.

t30ReceiveCoding

The highest coding scheme available to compress the page data when receiving. The following values are available:

Value	Usage
0	MH
1	MR
2 (Default)	MMR

t30ReceivePageSize

The page size for receiving. The following values are available:

Value	Usage
0	A4 (210x297 mm)
1	B4 (255x364 mm)
2 (default)	A3 (297x420 mm)

t30ReceiveMSLT

The minimum transmission time of one coded scan line. Default = 0

The following values are available:

Value	Usage
0 (default)	0 ms T7.7 = T3.85
1	5 ms T7.7 = T3.85
2	10 ms T7.7 = 1/2 T3.85
3	10 ms T7.7 = T3.85
4	20 ms T7.7 = 1/2 T3.85
5	20 ms T7.7 = T3.85
6	40 ms T7.7 = 1/2 T3.85
7	40 ms T7.7 = T3.85

t30ReceiveProtocol

The protocol used for fax receiving. The following values are available:

Value	Usage
0	non-ECM
1 (default)	ECM.

sendCedBeforeDIS

If enabled, allows the answering fax to send a CED (Called station Id) signal.

t30ReceiveModulations

Allows to select the receiving protocol. The following values are available:

Value	Usage
0	V.27
1 (default)	V.27/V.29
2	V.27/V.29/V.17
3	V.27/V.29/V.17/V.34

t30ReceiveR8x3

If enabled, receive resolution is R8x3.85 lines/mm.

t30ReceiveR8x7

If enabled, receive resolution is R8x7.7 lines/mm.

t30ReceiveR8x15



If enabled, receive resolution is R8x15.4 lines/mm.

t30Receive200x200

If enabled, receive resolution is 200x200 dots/inch.

EXAMPLE

```
$Activity_VoIPsIPPeer1 agent.pm.t30Parameters.config \  
-t30SendResolution          0 \  
-sendCedBeforeDIS          1 \  
-t30ReceiveR8x7            true \  
-t30SendPageSize           0 \  
-t30ReceiveR8x3            true \  
-t30SendProtocol           1 \  
-t30ReceiveProtocol        1 \  
-sendCNG                    1 \  
-t30SendCoding              0 \  
-t30ReceiveMSLT            0 \  
-t30SendMSLT                0 \  
-t30ReceiveCoding          2 \  
-t30ReceivePageSize        2 \  
-t30ReceiveModulations     3 \  
-t30ReceiveR8x15           true \  
-t30StationId               "5551\[000-\]" \  
-t30SendDataRate           5 \  
-t30Receive200x200        true
```

SEE ALSO

Timer Settings

VoIP SIP Peer Timer Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.TimerSettings.config \  
-option value
```

DESCRIPTION

Timer Settings configures the VoIP SIPPeer's timer settings.

SUBCOMMANDS

None.

OPTIONS

`enableTimers`

If enabled, the session refresh mechanism according to RFC4028 (Session Timers in SIP) and the registration refresh mechanism according to RFC3261 (Session Initiation Protocol) is allowed.

`expirationValList`

Defines the expiration value for each message.

`sessionRefreshType`

Allows you to select the session refresh time. The following values are available:

Value	Usage
0	After specified seconds.
1 (default)	After specified % of negotiated value.
2	With specified seconds before expiration

`refreshAfterSecs`

If `sessionRefreshType` = 0, minimum = 1, maximum = 9999, and default = 3000

`refreshAfterPercent`

If `sessionRefreshType` = 1, minimum = 1, maximum = 100, and default = 50

`refreshInSecs`



If sessionRefreshType = 2, minimum = 1, maximum = 9999, and default = 32

enableRetransmissions

If enabled, allows retransmission of certain SIP messages, both requests and responses, for script functions pertaining to the activity.

ignoreRetransmissions

If enabled, determines ignoring all the received retransmissions.

retransmitACK

If enabled, allows retransmission of the 200 Ok final response to an INVITE transaction causes the ACK message, in accordance with the provisions of RFC3261.

autoEndCall

If enabled, allows automatic deletion of active calls at the end of test loops

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.timerSettings.config \  
-enableRetransmissions      false \  
-retransmitACK              true \  
-refreshInSecs              32 \  
-refreshAfterSecs          3000 \  
-T2                          4000 \  
-T1                          500 \  
-refreshAfterPercent        50 \  
-ignoreRetransmissions      true \  
-ovrTimeout                 false \  
-sessionRefreshType         1 \  
-autoEndCall2               true \  
-enableTimers                false
```

SEE ALSO

SRTP Settings

VoIP SIP Peer SRTP Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.srtpSettings.config \  
-option value
```

DESCRIPTION

SRTP Settings configures the VoIP SIPPeer's SRTP settings.

SUBCOMMANDS

None.

OPTIONS

`benableRTP`

Enables use of SRTP to transport the media traffic.

- `false` = disabled (default)
- `true` = enabled

`bDisableSRTPAuthentication`

If `true`, this option disables SRTP authentication.

`bDisableSRTPEncryption`

If `true`, this option disables SRTP stream encryption.

`bIncludeMKI`

If `true`, the generated SRTP packets include the MKI field.

`bDisableValidations`

If `true`, none of the validations below are performed on the received SRTP packets:

- SRTP packet authentication tag is not verified
- Master Key expiration is not verified
- SRTP packet MKI field is ignored



bDisableSRTPEncryption

If true, this option disables SRTP stream encryption.

bAllowOnlySecureStreams

If true, the SDP offer comprises only secure streams and SDP negotiates only secure streams.

bDisableMasterSalt

If true, the Master Salt value is null instead of it being randomly generated.

bStaticMasterKeySalt

If true, this option determines the use of a static master key and salt.

_masterKeySelection

Specifies if a single key or multiple keys are used:

- 0 = A single key is used. The key is specified by the `staticSingleKeySalt` parameter.
- 1 = Multiple static keys are used. Keys are obtained from a file specified by the `staticKeyFile` parameter.

staticSingleKeySalt

If `bStaticMasterKeySalt` is true, this parameter defines a key value.

staticKeyFile

If `bStaticMasterKeySalt` is true, this parameter defines a file containing multiple key values.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.srtpSettings.config \  
-bDisableSRTPAuthentication      false \  
-bIncludeMKI                     true \  
-bEnableSRTP                     true \  
-bDisableValidations            false \  
-bDisableSRTPEncryption          false \  
-bStaticMasterKeySalt            true \  
-bAllowOnlySecureStreams         false \  
-bDisableMasterSalt              false \  
-staticSingleKeySalt             "BjVfSzwVXnYB2Rtr6BbFfbvDkuFtUjjWUCClq4gP" \  
-staticKeyFile                   "" \  
-bDisableSRTPEncryption          false \  
-_masterKeySelection              0
```

SEE ALSO

Custom Activity Link Settings

VoIP SIP Peer CustomActivityLinkSettings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.customActivityLinkSettings.config \  
-option value
```

DESCRIPTION

CustomActivityLinkSettings configures the BHCA and CPS objective settings for VoIP SIP Peer activities. This options in this object correspond to the controls on the Custom Parameters tab for a NetTraffic/ActivityLink in the Timeline and Objective branch of the Test Configuration tree in the GUI.

Note: The CustomActivityLinkSettings class has to be configured alongside the CustomParameters class that implements the same functionality.

SUBCOMMANDS

None.

OPTIONS

bhcaObjectiveValue

The BHCA test objective value. Default="80000".

bhcaType

Determines how the BHCA objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	BHCA will be met by specifying the talk time. Specify the talk time in talkTime.
1	BHCA will be met by specifying the number of channels. Specify the number of channels in channelsNo.

talkTime

If bhcaType is 0, this option specifies the Talk Time that will be used to attain the BHCA test objective. Default="40000".

channelsNo

If bhcaType is 1, this option specifies the number of channels that will be used to attain the BHCA test objective. Default="100".

`callSetupTime`

Estimated call setup time. Default="500".

`callTeardownTime`

Estimated call teardown time. Default="500".

`interCallDuration`

Inter-call duration. Default="4000".

`cpsObjectiveValue`

The CPS test objective value.

`cpsType`

Determines how the CPS objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	CPS objective will be met by specifying the talk time. Based on the the talk time value specified in <code>talkTime</code> , the <code>cpsChannelsNo</code> value is computed.
1	CPS objective will be met by specifying the number of channels. Based on the the channels number value specified in <code>cpsChannelsNo</code> , the <code>talkTime</code> value is computed.

`cpsTalkTime`

If `cpsType` is 0, this option specifies the Talk Time that will be used to attain the CPS test objective. Default="40000".

`cpsChannelsNo`

If `bhcaType` is 1, this option specifies the number of channels that will be used to attain the CPS test objective. Default="100".

`cpsOverheadTime`

Indicates the duration of all other actions on the channel except the talk time and minimum inter-call duration.

`cpsInterCallDuration`

The minimum time interval between the end of a call on a Voice channel and the start of a new call on the same voice channel

EXAMPLE

```
$Activity_Make_Call agent.pm.customActivityLinkSettings.config \  
-talkTime                40000 \  
-cpsObjectiveValue       100 \  
-cpsType                 0 \  
-cpsInterCallDuration    2000 \  
-channelsNo              100 \  
-cpsTalkTime             750 \  
-cpsOverheadTime         1500 \  
-cpsChannelsNo           425 \  
-bhcaType                0 \  
-callTeardownTime        500 \  
-interCallDuration       4000 \  
-bhcaObjectiveValue      100 \  
-callSetupTime           500
```

SEE ALSO

Execution Settings

VoIP SIP Peer Execution Settings

SYNOPSIS

```
$Activity_<VoIPSIPPeer activity name>agent.pm.executionSettings.config \  
-option value
```

DESCRIPTION

This object defines the execution settings for the VoIP SIP Peer activity.

SUBCOMMANDS

None.

OPTIONS

loopMode

Defines how many loops are executed for every voice channel corresponding to this activity.

Value	Description
0 (default)	Loop for the entire test duration.
1	Execute a number of loops. Specify the number of loops in loopCount.

loopCount

If loopMode is 1, this option defines the number of loops that the test performs. Default="1".

loopPreDelay

Delay before first loop (ms). Default="0", min="0" max="3600000".

loopMidDelay

Delay between loops (ms). Default="0" min="0" max="3600000".

aliases

Number of aliases (phone numbers) per channel. Default="1", min="1" max="16000".

multipleUsersPerIO



Specifies if multiple VoIPSIP channels can share the same IP:port.

ipRule

A simulated VoIPSIP channel is uniquely identified by IP address, TCP/UDP/TLS port, and Phone number. This option selects the rule used for the IP address portion of the channel mapping rule.

- 0 = Use same value (per port)
- 1 = Use consecutive values (per port) (default)
- 2 = Use same value for every x channels. Specify the value for x in -ipRuleCh.

ipRuleCh

If ipRule is Use same value every, this specifies the number of channels. (Default="1" min="1" max="100000")

portRule

A simulated VoIPSIP channel is uniquely identified by IP address, TCP/UDP/TLS port, and Phone number. This option selects the rule used for the TCP/UDP portion of the channel mapping rule.

- 0 = Use same value (default)
- 1 = Use consecutive values (per port)
- 2 = Use consecutive values (per activity)
- 3 = Use same value for every x channels. Specify the value for x in -portRuleCh.

portRuleCh

If portRule is Use same value every, this specifies the number of channels. (Default="1" min="1" max="100000").

phoneRule

A simulated VoIPSIP channel is uniquely identified by IP address, TCP/UDP/TLS port, and Phone number. This option selects the rule used for the Phone number portion of the channel mapping rule.

- 0 = Use consecutive values (per port) (default)
- 1 = Use consecutive values (per activity)

rtpIpRule

A simulated RTP channel is uniquely identified by the IP address and UDP port. This option selects the rule used for the IP address portion of the RTP channel allocation.

- 0 = Use same value (per port) (default)
- 1 = Use consecutive values (per port)
- 2 = Use same value for every x channels. Specify the value for x in the rtpIpRuleCh parameter.

rtpIpRuleCh

If rtpIpRule is Use same value every, this parameter specifies the number of channels.

rtpPortRule

This option selects the rule used for the port portion of the RTP channel allocation.

- 0 = Use same value (default)
- 1 = Use consecutive values (per port)
- 2 = Use consecutive values (per activity)
- 3 = Use same value for every x channels. Specify the value for X in rtpPortRuleCh.

rtpPortRuleCh

If rtpPortRule is Use same value every, this parameter specifies the number of channels.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.executionSettings.config \  
-portRuleCh                1 \  
-rtpPortRule               0 \  
-multipleUsersPerIO       false \  
-loopMidDelay              0 \  
-loopCount                 1 \  
-rtpIpRule                 1 \  
-rtpIpRuleCh               1 \  
-rtpPortRuleCh             1 \  
-loopPreDelay              0 \  
-loopMode                  0 \  
-phoneRule                 0 \  
-portRule                  0 \  
-ipRule                    1 \  
-ipRuleCh                  1 \  
-aliases                   1
```

SEE ALSO

Transfer Address

VoIP SIP Peer Transfer Address

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.transferAddress.config \  
-option value
```

DESCRIPTION

Transfer Address configures a SIP transfer address (see RFC 3261).

SUBCOMMANDS

None.

OPTIONS

`symTransferStr`

Name of the VoIP SIP Peer configured as transfer destination (Default="None").

`overridePhoneNo`

Enables override of phone numbers from destination VoIP SIP Peer.

Value	Usage
0 (default)	Disabled
1	Enabled

`_useTPb`

If `overridePhoneNo` is 1, this option selects the source of the replacement phone numbers.

Value	Usage
0 (default)	Use phone number specified by <code>_tPhone</code> .
1	Use phone number specified by Phonebook entry.

Note: This options appears in the generated Tcl code only if the test configuration contains a reference to a Phonebook entry (`_useTPb=1`). The generated tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

`_tPhone`

If `_useTPb` is 0, this option specifies the replacement phone numbers. You can use sequence generators in this field. Default="150[00000000-]".

`_ckTTelURIParams`

Enables insertion of Tel URI parameters.

Value	Usage
0 (default)	Disabled
1	Enabled

`_tTelURIParams`

If `_ckTTelURIParams` is 1, this option specifies the Tel URI parameters. Default="phone-context=example.com".

EXAMPLE

```
$Activity_VoIPSIPPeer1 agent.pm.transferAddress.config \  
-overridePhoneNo      false \  
-useTPb                0 \  
-tPhone                "150\[00000000-\  
-transTelPar           "" \  
-tPhone                "150\[00000000-\  
-ckTTelURIParams      false \  
-symTransferStr        "None" \  
-tPhoneType            0 \  
-tTelURIParams         "phone-context=example.com" \  
-tBp                   "&lt;None&gt;"
```

SEE ALSO

Scenario Settings

VoIP SIP Peer Scenario Settings

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.scenarioSettings.config \  
-option      value
```

DESCRIPTION

Scenario Settings specifies the test scenario file used by the Tcl script.

SUBCOMMANDS

None.

OPTIONS

scenarioFile

The full path to the test scenario file for the activity.

activeScenarioChannel

Test scenario channel (0-based index) that is associated with the VoIP SIP Peer activity (Default=0).

EXAMPLE

```
$Activity_VoIPSIPPeer1 agent.pm.scenarioSettings.config \  
-scenarioFile  
  "E:\\ScenarioTestFiles\\Basic_Call_TCP.tst" \  
-activeScenarioChanne      10
```

SEE ALSO

Dial Plan

VoIP SIP Peer Dial Plan

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.dialPlan.config \  
-option      value
```

DESCRIPTION

The Dial Plan object configures the source, destination, and transfer addresses and phone numbers for the channels/endpoints simulated by the VoIPSIPPeer activity.

SUBCOMMANDS

None.

OPTIONS

sourceIPs

List of IPs taken from the associated network (read-only).

_useSPb

Method used to select phone number.

Value	Usage
0	Use the phone number specified by pattern.
1	Use the phone number specified by Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (_useSPb=1). The generated Tcl script will run only on the machine it has been generated on and only if the correspon-ding Phonebook entry has not yet been deleted since the generation of the Tcl code.

_sPhone

If _useSPb is 0, this option specifies the phone number. You can use sequence generators in this field to generate multiple phone numbers. See the sequence generator appendix.
Default="160[00000000-]" .

_ckSTelURIParams

Enables insertion of Tel URI parameters.



Value	Usage
0 (default)	Disabled
1	Enabled

`_sTelURIParams`

If `_ckSTelURIParams` is 1, this option specifies the Tel URI parameters. Default="phone-context=example.com".

`symDestStr`

String identifying the VoIP SIP Peer or VoIP Skinny Peer that is the destination for traffic from this VoIP SIP Peer activity. Default="None".

`ovrDestPhone`

Enables overriding of phone number from the destination VoIP Peer.

Value	Usage
0 (default)	Disabled
1	Enabled

`_useDPb`

Method used to select the phone number used to override destination phone number.

Value	Usage
0 (default)	Specify pattern.
1	Specify Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (`_useDPb=1`). The generated Tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

`_dPhone`

If `_useDPb` is 0, this option specifies the phone number. Default="170[00000000-]".

`_ckDTelURIParams`

Enables insertion of Tel URI parameter.

Value	Usage
0 (default)	Disabled
1	Enabled

`_dTelURIParams`

If `_ckDTelURIParams` is 1, this option configures the Tel URI parameters. Default="phone-context=example.com".

EXAMPLE

```
$Activity_VoIPSIPPeer1 agent.pm.dialPlan.config \
-_useSPb                                0 \
-symDestStr                              "sip server_VoIPSIPPeer2:5060" \
-_sTelURIParams                          "phone-context=example.com" \
-destPhoneType                            0 \
-_sPhone                                  "160\[00000000-\]" \
-_dTelURIParams                          "phone-context=example.com" \
-_sBp                                     "&lt;None&gt;" \
-srcPhoneType                             0 \
-_dBp                                     "&lt;None&gt;" \
-ovrDestPhone                             false \
-destTelPar                               "" \
-_ckSTelURIParams                        false \
-_dPhone                                  "170\[00000000-\]" \
-srcPhone                                  "160\[00000000-\]" \
-destPhone                                 "160\[00000000-\]" \
-_useDPb                                  0 \
-_ckDTelURIParams                        false \
-srcTelPar                                 ""
```

SEE ALSO

TLS Settings

Configures VoIP SIP Peer TLS settings.

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.tlsSettings.config \  
-option value
```

DESCRIPTION

Specifies TLS settings for SIP traffic.

SUBCOMMANDS

None.

OPTIONS

`enableTLS`

Enables use of TLS to transport the SIP traffic.

false = disabled (default)

true = enabled

`tlsProtocol`

Specifies the TLS protocol version used:

- 0 = TLS 1.0 Only (Default)
- 1 = SSL 3.0 Only
- 2 = TLS + SSL

`tlsPort`

Specifies the TLS listening port (default=5061).

`tlsEnableTcpKeepAlive`

If configured `true`, enables the TCP keep alive mechanism on the VoIPSIPPeer-emulated endpoints.

`tlsReuseConnection`

If configured `true`, an ``alias'` parameter is added in the Via header of SIP requests sent by the VoIPSipPeer activity, such as to enable the TLS connection reuse mechanism.

When this option is set to the `true` value, the Mutual Authentication option is automatically selected.

`tlsMutual`

If configured `true`, mutual authentication is performed. When this parameter is configured `true`, the `tlsAuthClient` option also has to be configured `true`.

`tlsAuthClient`

If configured `true`, client authentication at the TLS connection establishment stage is also performed. By default, only the server authenticates itself by presenting a certificate.

`tlsSessionRefresh`

If configured `true`, TLS renegotiation is enabled at the interval of time specified by the `tlsRefreshInterval` parameter.

`tlsRefreshInterval`

When the `tlsSessionRefresh` option is configured `true`, this parameter specifies the refresh interval.

`ignoreSubjectAltName`

If configured `true`, the verification of the Subject Alternative Name certificate parameter is not performed and the connection is re-used for which the ``alias'` parameter of the `Via` header was received.

`sipScheme`

Specifies the scheme, `sip` or `sips`, used for the construction of the Request-URI for the following SIP message headers: Contact, From, To, Reply-To, Via, Record-Route.

0 = sip

1 = sips

`tlsTransportType`

Specifies the transport protocol – TCP or TLS – used in the construction of SIP Request-URIs, the Contact message header and the ``sent-protocol'` parameter of Via message headers

0 = TCP

1 = TLS

`tlsDisableUdpAndTcp`

If `true`, the VoIPSIP peer only accepts TLS connections, rejecting any UDP or TCP connections.



tlsCertificatesPath

Specifies the certificates location, a folder containing the certificates files. Default = "".

tlsPublicKeyCertificate

Specifies the name of the certificate file containing the public key, or a sequence specifying a set of certificate file names.

tlsPrivateKeyCertificate

Specifies the name of the certificate file containing the private key, or a sequence specifying a set of certificate file names.

tlsPassword

Specifies an optional parameter, defined as a string or a sequence, representing the password used to encrypt the private key. Default = "".

EXAMPLE

```
$Activity_VoIPsipPeer1 agent.pm.tlsSettings.config \  
-tlsProtocol                2 \  
-tlsPublicKeyCertificate    "" \  
-tlsEnableTcpKeepAlive     false \  
-tlsReuseConnection        false \  
-tlsPort                    "5061" \  
-tlsSessionRefresh         false \  
-enableTLS                 false \  
-ignoreSubjectAltName      false \  
-tlsAuthClient             0 \  
-tlsPrivateKeyCertificate   "" \  
-tlsPassword               "" \  
-tlsMutual                  false \  
-tlsRefreshInterval        3600 \  
-sipScheme                 0 \  
-tlsTransportType          0 \  
-tlsDisableUdpAndTcp       true \  
-tlsCertificatesPath       ""
```

SEE ALSO

TLS Cyphers

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.tlsSettings.tlsCyphers.appendItem \  
-option value
```

DESCRIPTION

The `tlsCyphers` object configures a list of cyphers supported by the `VoIPSipPeer` activity. Cyphers are added to the list using the `appendItem` command.

SUBCOMMANDS

None.

OPTIONS

`id`

The TLS cypher list Id.

`enabled`

If configured `true`, the use of the given cipher is advertised (default = false).

`name`

The cypher name.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.tlsSettings.tlsCyphers.clear  
  
$Activity_VoIPSipPeer1 agent.pm.tlsSettings.tlsCyphers.appendItem \  
-id "TlsCyphers" \  
-enabled true \  
-name "AES128-SHA"
```

SEE ALSO

Custom Parameters

VoIP SIP Peer CustomParameters.

SYNOPSIS

```
$Activity_VoIPSIPPeer1 customParameters.config \  
-option value
```

DESCRIPTION

CustomParameters configures the settings for the BHCA objective for VoIP SIP Peer activities. This options in this object correspond to the controls on the Custom Parameters tab for a NetTraffic/ActivityLink in the Timeline and Objective branch of the Test Configuration tree in the GUI.

Note: The CustomParameters class has to be configured alongside the CustomActivityLinkSettings class that implements the same functionality.

SUBCOMMANDS

None.

OPTIONS

bhcaObjectiveValue

The BHCA test objective value. Default="80000".

bhcaType

Determines how the BHCA objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	BHCA will be met by specifying the talk time. Specify the talk time in talkTime.
1	BHCA will be met by specifying the number of channels. Specify the number of channels in channelsNo.

talkTime

If bhcaType is 0, this option specifies the Talk Time that will be used to attain the BHCA test objective. Default="40000".

channelsNo

If bhcaType is 1, this option specifies the number of channels that will be used to attain the BHCA test objective. Default="100".

`callSetupTime`

Estimated call setup time. Default="500".

`callTeardownTime`

Estimated call teardown time. Default="500".

`interCallDuration`

Inter-call duration. Default="4000".

`cpsObjectiveValue`

The CPS test objective value.

`cpsType`

Determines how the CPS objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	CPS objective will be met by specifying the talk time. Based on the the talk time value specified in <code>talkTime</code> , the <code>cpsChannelsNo</code> value is computed.
1	CPS objective will be met by specifying the number of channels. Based on the the channels number value specified in <code>cpsChannelsNo</code> , the <code>talkTime</code> value is computed.

`cpsTalkTime`

If `cpsType` is 0, this option specifies the Talk Time that will be used to attain the CPS test objective. Default="40000".

`cpsChannelsNo`

If `bhcaType` is 1, this option specifies the number of channels that will be used to attain the CPS test objective. Default="100".

`cpsOverheadTime`

Indicates the duration of all other actions on the channel except the talk time and minimum inter-call duration.

`cpsInterCallDuration`

The minimum time interval between the end of a call on a Voice channel and the start of a new call on the same voice channel.

EXAMPLE

```
$Activity_Make_Call customParameters.config \  
-talkTime                40000 \  
-cpsObjectiveValue       100 \  
-cpsType                 0 \  
-cpsInterCallDuration    2000 \  
-channelsNo              100 \  
-cpsTalkTime            750 \  
-cpsOverheadTime        1500 \  
-cpsChannelsNo          425 \  
-bhcaType                0 \  
-callTeardownTime       500 \  
-interCallDuration      4000 \  
-bhcaObjectiveValue     100 \  
-callSetupTime          500
```

SEE ALSO

Advanced Settings

Configures a VoIPSIP Cloud Peer activity that is associated with the VoIPSIP Peer.

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.advancedSettings.config \  
-option          value
```

DESCRIPTION

Advanced Settings configure the use of a specified VoIPSIP Cloud Peer conjointly with the SIP Peer.

The SIP Proxy servers emulated by the VoIPSIP Cloud Peer can be configured to add `via` and `Record-Route` message headers to SIP messages traversing them.

SUBCOMMANDS

None.

OPTIONS

`useCloud`

If `true`, this option enables use of a VoIPSIP cloud with the VoIPSIP Peer.

`false` = disabled (default)

`true` = enabled

`ovrCloudRules`

If configured `true`, default dispatching rules are being overridden.

`cloud`

Specifies the SIP cloud to use.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.advancedSettings.config \  
-useCloud          true \  
-ovrCloudRules     false \  
-cloud             "VoIPSipCloud1"
```

SEE ALSO

`CloudServers`



Cloud Servers

Configures a list of SIP Proxy Servers emulated by a VoIPSIP Loud Peer.

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.cloudServers.appendItem \  
-option          value
```

DESCRIPTION

This object configures a list of SIP Proxy Servers emulated by a VoIPSIP Cloud Peer activity. SIP proxies are added to the list using the `appendItem` command.

Note: The `CloudServers` class has to be configured alongside the `SipServerList` class of a VoIP SIP Cloud Peer that implements the same functionality.

SUBCOMMANDS

None.

OPTIONS

`id`

The cloud server's list ID.

`firstIp`

The first IP address in the network range associated with the SIP Proxy server. This is the SIP Proxy server that is located at the cloud boundary.

`name`

The server name (default `sip_server#1` and subsequent strings).

`rangeType`

The range type, which can be `Virtual IP` and `IP`.

`ipAddr`

The starting IP address of the associated network range.

`netMask`



The network mask.

ipStep

The increment step of the starting IP address (default "0.0.0.1").

attachedInfo

An extra string associated with the proxy, such as a domain name (default = sip-test.my-domain.com).

ipCount

The number of hosts (default = 1).

port

The SIP port (default = 5060).

ipType

The IP addressing type, IPv4 or IPv6.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.cloudServers.clear

$Activity_VoIPSipPeer1 agent.pm.cloudServers.appendItem \
-id "CloudServer" \
-firstIp "172.20.13.1" \
-name "sip_server#1" \
-rangeType "IP" \
-ipAddr "Network Range 2 in Network1 (172.20.13.1+1)" \
-ipStep "0.0.0.1" \
-attachedInfo "sip-test.my-domain.com" \
-netMask "255.254.0.0" \
-ipCount "1" \
-port 5060 \
-ipType "IPv4"
```

SEE ALSO

Server Rules

Configures a list of rules associated with each emulated SIP Proxy server in the VoIPSIP Cloud Peer.

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.advancedSettings.serverRules.appendItem \  
-option          value
```

DESCRIPTION

This object configures a rules list. For each SIP server in the cloud, its associated rule specifies if a `Via` or a `Record-Route` header are added to SIP messages traversing the server. Rules are added to the list using the `appendItem` command.

SUBCOMMANDS

None.

OPTIONS

`id`

The server rules list ID.

`recordRoute`

If `true`, a SIP Record-Route message header is added to SIP messages (default = `true`).

`via`

If `true`, a SIP Record-Route message header is added to SIP messages (default = `true`).

`name`

The name of the SIP Proxy server (default = `sip_server#<n>`).

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.advancedSettings.serverRules.clear  
  
$Activity_VoIPSipPeer1 agent.pm.advancedSettings.serverRules. \  
appendItem \  
-id          "ServerRule" \  
-recordRoute true \  
-via         true \  
-name       "sip_server#1"
```

SEE ALSO



Cloud Rules

Configures a list of dispatching rules that override the default VoIP SIP Cloud rules.

SYNOPSIS

```
$Activity_VoIPSIPPeer1 agent.pm.cloudRules.rulesList.appendItem \  
-option          value
```

DESCRIPTION

A new dispatching rule is added to the `rulesList` of the `cloudRules` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None.

OPTIONS

`id`

The cloud rules list Id.

`when`

Specifies the SIP message that is processed for extracting a rule.

`where`

- Extracts the variable from the request line, or from parts of it, as follows:
 - Entire First Line
 - Request Line - Method
 - Request Line - Request-URI
 - Request Line - Request-URI - Phone
 - Request Line - SIP Version

`refine`

Specifies if further processing is applied or not:

- N/A: No further processing is applied
- Refined: Further processing is applied, as defined by a `RuleData` object.

`formula`



A formula that is defined using the same syntax as a sequence generator expression. Form the extracted string matched against the dispatching formula, the message is dispatched to a specific SIP channel.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.cloudRules.rulesList.clear

$Activity_VoIPSipPeer1 agent.pm.cloudRules.rulesList.appendItem \
-id "CloudRule" \
-where "Request Line - Request-URI - Phone" \
-when "INVITE" \
-refine "N/A" \
-formula "160\[00000000-\]"
```

SEE ALSO

RuleData

RuleData

Configures the processing operations applied to incoming message for extracting a dispatching rule.

SYNOPSIS

```
$Activity_VoIPSipPeer agent.pm.cloudRules.rulesList(0).ruleData.config \  
-option value
```

DESCRIPTION

A `RuleData` object defining further processing that is applied to a string after it is extracted from a SIP message. This object corresponds to the Edit Cloud Rule GUI in the application.

SUBCOMMANDS

None.

OPTIONS

`what`

Defines the extraction scope as one of the following:

- 0 = Entire SIP request
- 1 = Request line
- 2 = Header
- 3 = SIP message body

`reqLine`

If `what` is configured to the value '1', this parameter specifies which part of the request line the string is extracted from:

- 0 = Entire First Line
- 1 = Request Line - Request-URI - Phone
- 2 = Request Line - Method
- 3 = Request Line - Request-URI
- 4 = Request Line - SIP Version

`headerType`

If `what` is configured to the value '2', this parameter specifies a header type that is being extracted (default = To).

`compactForm`



If `what` is configured to the value '2', this parameter defines the compact form of the SIP message header specified by the `headerType` parameter.

`occurFrom, endOccur`

If `what` is configured to the value '2', this parameter specifies between which occurrences extraction is done.

`whatExtract`

If `what` is configured to the value '2', this parameter specifies which part of the header is extracted:

- 0 = Whole header value
- 1 = Header value without parameters
- 2 = The parameter specified by `paramName`
- 3 = Phone value from URI

`extractHeaderName`

When the `whatExtract` parameter is configured to the value '0', if this option is configured `true`, the header name is also extracted.

`paramName`

When the `whatExtract` parameter is configured to the value '2', this option extracts the value of the named parameter.

`revHeaderOrder`

When configured `true`, this option to `true` processes the occurrences in reverse order, starting from the last up to the first.

`keepHeaderCrlf`

When configured `true`, the last Carriage Return/Line Feed character extracted into the variable is kept.

Note: The parameters above correspond to Step 2 in the dispatching rules definition window of the IxLoad GUI.

`usePosition`

Specifies the mode in which an extracted substring is delimited:

- 0 = The substring is marked by delimiters.
- 1 = The substring is marked by position.

`beginAfter`

If this parameter is configured `true`, a substring is delimited by the `afterStr` and `afterOccur` parameters.

This parameter is relevant when `usePosition` is configured to the value '0'.

`afterStr`, `afterOccur`

The substring start is indicated by these parameters.

`endBefore`

If this parameter is configured `true`, a substring is delimited by the `endStr` and `endOccur` parameters.

`endStr`, `endOccur`

The substring end is indicated by these parameters.

`positionFrom`, `positionTo`

If `usePosition` is configured to the value '1', these parameters specify the delimiting positions for position-based substring extraction.

`formula`

Specifies a formula that is defined using the same syntax as a sequence generator expression. The extracted string matched against the dispatching formula and the message is dispatched to a specific SIP channel.

Note: The parameters above correspond to Step 3 in the dispatching rules definition window of the IxLoad GUI.

EXAMPLE

```
$Activity_VoIPSipPeer1 agent.pm.cloudRules.rulesList.appendItem \  
-id "CloudRule" \  
-where "Request Line - Request-URI - Phone" \  
-when "INVITE" \  
-refine "Refined" \  
-formula "160\[00000000-\]" \  
  
$Activity_VoIPSipPeer1 agent.pm.cloudRules.rulesList(0).ruleData. \ config \  
-positionFrom "1" \  
-what 1 \  
-endBefore true \  
-extractHeaderName false \  
-headerType "To" \  
-whatExtract 3 \  
-occurFrom "1" \  
-formula "160\[00000000-\]" \  
-endStr ">" \  
-usePosition 0 \  
-endOccur "last" \  
-positionTo "last" \  
-reqLine 1 \  
-keepHeaderCrlf false \  
-compactForm "t" \  
-paramName "" \  
-afterOccur "1" \  
-beginAfter true \  
-afterStr "<" \  
-occurTo "1" \  
-revHeaderOrder false
```

SEE ALSO



37

VoIP Skinny Peer

The IxLoad VoIP Skinny Peer Tcl API consists of a VoIP Skinny Peer agent, with separate APIs for configuring each major aspect of the agent’s functionality.

In this chapter:

Limitations	1993
VoIP Skinny Peer API Commands	1994
VoIP Skinny Peer Agent	1996
Scenario Settings	2012
Execution Settings	2013
Dial Plan	2015
Skinny Settings	2020
Call Managers	2023
Codec Settings	2025
Data Codecs	2026
Codecs	2028
RTP Settings	2034
Audio Settings	2036
Other Settings	2040
Custom Activity Link Settings	2043
Custom Parameters	2045



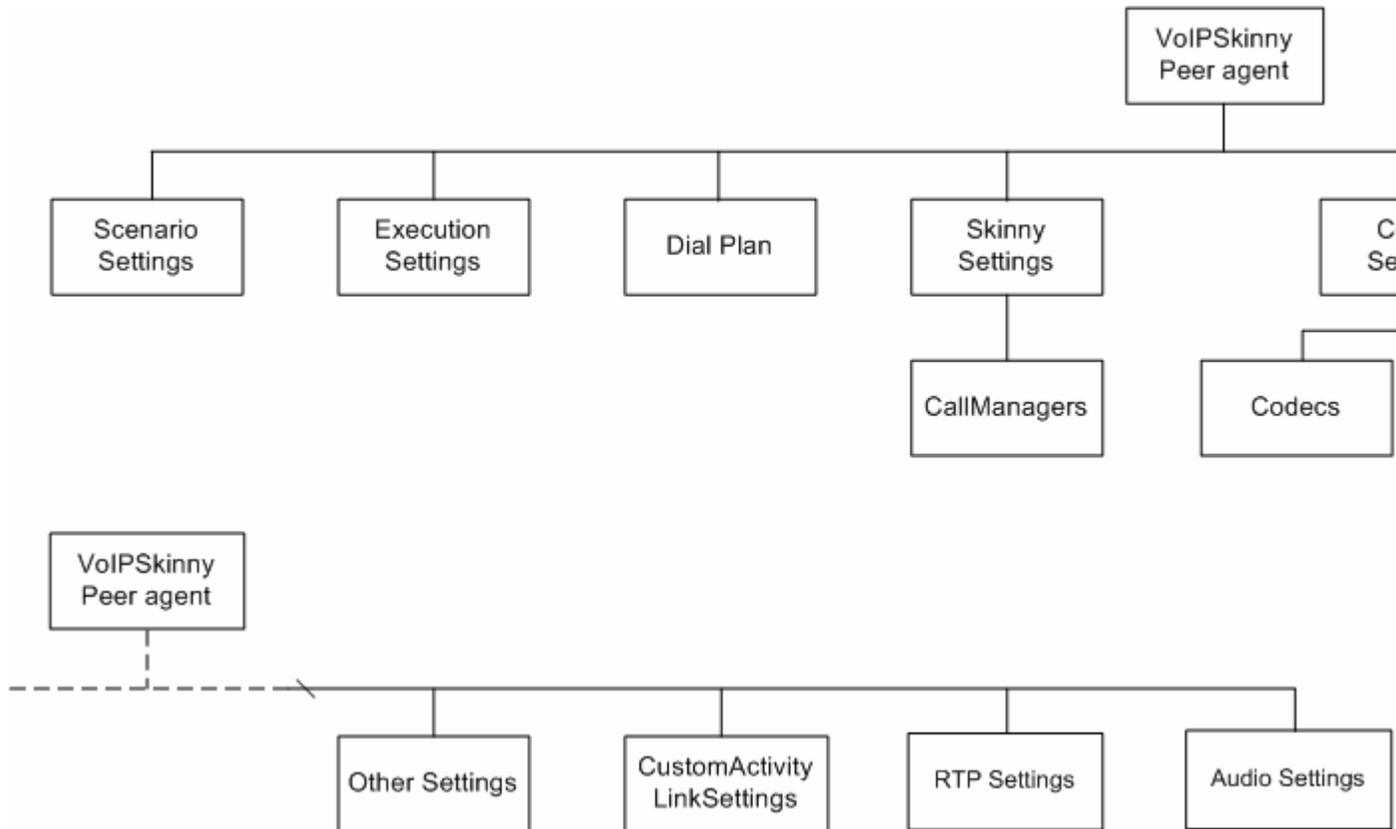
Limitations

The following restrictions and limitations of the VoIPSkinny Peer API exist:

- The PhoneBook and other related classes, such as PhoneBookEntry, can not be edited from the Tcl API.
- Individual VoIP Skinny script functions can not be added and edited from the Tcl API. Instead, you must add and configure the commands in the Scenario Editor, save the test scenario file, then pass it as an argument to the ScenarioSettings API class.
- Implementation of the BHCA objective features relies on two classes, CustomParameters and CustomActivityLinkSettings that have to be configured using the same parameters.

VoIP Skinny Peer API Commands

The IxLoad VoIP Skinny Peer API commands are organized as shown in the figure below.



VoIP Skinny API Objects

The table below lists the VoIP Skinny Peer API objects.

Object	Description
VoIP Skinny Peer Agent	Top-level object defining the VoIP Skinny Peer activity.
Scenario Settings	Selects the Test Scenario file; corresponds to the Scenario Settings GUI tab.
Codec Settings	List of Data Codecs and Codecs objects.
Data Codecs	Data codec with parameters.
Codecs	Audio codec with parameters.
Skinny Settings	VoIP Skinny Peer Skinny parameters; corresponds to the Skinny Settings GUI tab.
CallManager	CallManager object with parameters.

Object	Description
Execution Settings	Run-time test configuration; corresponds to the Execution Settings GUI tab.
Dial Plan	Configures the registration names, phone numbers, and source, destination, and transfer addresses for the channels/phones; corresponds to the Dial Plan GUI tab.
RTP Settings	RTP transport configuration; corresponds to the RTP Settings GUI tab.
Audio Settings	Audio settings; corresponds to the Audio GUI tab.
Other Settings	VoIP Skinny Peer miscellaneous parameters; corresponds to the Other Settings GUI tab.
Custom Activity Link Settings, CustomParameters	BHCA objective configuration; corresponds to the Custom Parameters GUI tab.



VoIP Skinny Peer Agent

VoIP Skinny Peer Agent

SYNOPSIS

```
set Activity_VoIPSkinnyPeer1 \  
[$ClientNetwork1 activityList.appendItem \  
-protocolAndType      "VoIP Skinny Peer" ]
```

DESCRIPTION

A VoIP Skinny Peer agent is added to the `agentList` option of the `config` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command. Other `ixConfigSequenceContainer` subcommands may be used to modify the `agentList`. See the following example:

```
set Activity_VoIPSkinnyPeer2 [$skinny_client_ClientNetwork1 \  
 \ activityList.appendItem \  
 \  
   -protocolAndType      "VoIP Skinny Peer" ]  
$Activity_VoIPSkinnyPeer2 config \  
  -enable                true \  
  -name                  "VoIPSkinnyPeer2" \  
  -enableConstraint     false \  
  -userObjectiveValue    1 \  
  -constraintValue      100 \  
  -userObjectiveType     "channels" \  
  -timeline              $Timeline3 \  
  
$Activity_VoIPSkinnyPeer2 agent.config \  
  -enable                true \  
  -name                  "VoIPSkinnyPeer2"
```

Each member of the list, however may be separately addressed and modified using the `ixConfig` subcommands. For example, the first agent uses an index of 0 and its name may be modified by:

```
$Activity_VoIPSkinnyPeer1 agent(0).config -name "VoIP Skinny Peer new"
```

SUBCOMMANDS

None.

OPTIONS

`enable`

Enables the use of this agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

The table below lists the statistics published by this object.



Note: Statistics from this category are not displayed in any of the pre-defined views, but can be assigned to custom statistics views.

Statistic	Description	Per Channel/Global
VoIPSkinny Channels		
Successful Channels	The per polling interval number of COMPLETED channels. A channel is COMPLETED if all the channel loops were COMPLETED.	Global
Warning Channels	The per polling interval number of WARNING channels. A channel is WARNING if all the channel loops were COMPLETED or WARNING and at least one loop had a WARNING result.	Global
Failed Channels	The per polling interval number of FAILED channels. A channel is FAILED if all the loops of the channel were COMPLETED, WARNING, or FAILED and at least one loop was FAILED.	Global
Aborted Channels	The per polling interval number of ABORTED channels. A channel is ABORTED if all the channel loops of the channel were COMPLETED, WARNING, FAILED, or ABORTED and at least one loop was ABORTED.	Global
Active Channels	The per polling interval number of active channels. Active channels are the channels executing a scenario channel functions flow.	Global
Total Channels	The per polling interval total number of channels, a sum of active and non-active channels.	Global
VoIPSkinny Loops		
Successful Channel Loops	The cumulative count of COMPLETED channel loops. A channel loop is COMPLETED if all executed script functions in the corresponding scenario channel produced SKIPPED or COMPLETED function results.	Global
Warning Channel Loops	The cumulative count of WARNING channel loops. A channel loop has a WARNING result if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, or WARNING function results and at least one script function had a WARNING result.	Global

Statistic	Description	Per Channel/Global
Failed Channel Loops	The cumulative count of FAILED channel loops. A channel loop is FAILED if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, WARNING, or FAILED function results and at least one script function had a FAILED result.	Global
Aborted Channel Loops	The cumulative count of ABORTED channel loops. A channel loop is FAILED if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, WARNING, FAILED, or ABORTED function results and at least one script function had an ABORTED result.	Global
Total Channel Loops	The cumulative count of total executed loops.	Global
Inter Loop Duration (Avg) (ms)	The average time gap between loops.	Global
VoIPSkinny Calls		
Attempted Calls	The number of originated calls - not necessarily answered or connected. This statistic is updated whenever the <i>skNewCall</i> , <i>SkRedial</i> , or <i>SkMeetMeConfrn</i> softkey is sent. It is also incremented when a transfer or a conference is initiated, i.e. a <i>SkTrnsfer</i> or a <i>SkConfrn</i> softkey is sent for the first time.	Global
Connected Calls	The number of calls successfully connected from the originator point of view. This statistic is incremented whenever the originating side receives a <i>StartMediaTransmission</i> message.	Global
Received Calls	The number of received calls, not necessarily answered. This statistic is incremented whenever the <i>CallState TsRingIn</i> or the <i>CallState TsWaitCalling</i> message is received by the call terminating side.	Global
Answered Calls	The number of calls received, successfully answered and connected. This statistic is incremented whenever the receiving side receives a <i>StartMediaTransmission</i> message.	Global
Transferred Calls	The number of transferred calls. This statistic is updated whenever the <i>CallState TsOnHook</i> message is received, after the second <i>SkTransfer</i> softkey was sent, to complete the transfer.	Global
Active Calls	Number of active calls at one time. This statistic is incremented when the <i>StartMediaTransmission</i> message is received, and is decremented when the <i>CallState TsOnHook</i> message is received or at the end of the loop.	Global

Statistic	Description	Per Channel/Global
Busy Calls	The number of calls that were rejected with the party being busy as a cause. This statistic is incremented when the <i>CallState TsBusy</i> message is received.	Global
End Call Initiated	The number of initiated end call operations. This statistic is updated whenever the <i>skEndCall</i> softkey is sent.	Global
End Call Received	This statistic is updated whenever the <i>CallState TsOnHook</i> message is received, without the <i>skEndCall</i> softkey being previously sent.	Global
End Calls Completed	This statistic is updated whenever the <i>CallState TsOnHook</i> message is received after having sent a <i>SkEndCall</i> softkey.	Global
Answered Calls TX	The number of acknowledged calls (answered), but not necessarily connected. This statistic is incremented when the <i>CallState TsConnected</i> message is received by originating side.	Global
Attempted Answered Calls RX	The number of calls received and answered, but not necessarily completed. This statistic is incremented whenever the <i>SkAnswer</i> softkey event is sent by the receiving side.	Global
VoIPSkinny Call Rates		
Attempted Calls /s, Connected Calls /s, Received Calls /sec, Answered Calls /s, Rejected Calls /s, Calls with Authentication Required /s, Transferred Calls /s, Busy Calls /s, Redirected Calls /s	The per polling interval rates corresponding to some of the previous VoIPSkinny Calls statistics.	Global
VoIPSkinny Call Times		
Call Setup Time TX Avg (ms)	The time it takes to setup a call and receive a call acceptance acknowledgement from the remote endpoint, including the post-dial delay and computed as the time between the <i>SkNewCall</i> and the <i>StartMediaTransmission</i> events.	Global
Call Setup Time RX Avg (ms)	The time from receiving the request for the call until receiving the final caller acknowledgment that the call setup has been successfully completed, computed as time between the <i>CallState TsRingIn</i> and the <i>StartMediaTransmission</i> events.	Global



Statistic	Description	Per Channel/Global
Talk Time (Avg)	The active conversational between the <i>StartMediaTransmission</i> message until the <i>CloseReceiveChannel</i> message.	Global
End Call Time (Avg)	The time between the sending the <i>SkEndCall</i> softkey and receiving the <i>CallState TsOnHook</i> message.	Global
Total Call Duration (Avg)	The total call duration comprising the call setup, talk, and end call times.	Global
VolpSkinny Delays		
Post Dial Delay (Avg) [ms]	The per polling interval time elapsed between sending of the last dialed number digit and receiving of a <i>CallStateMessage TsRingOut</i> , <i>TsCongestion</i> , <i>TsBusy</i> , or <i>TsInvalidNumber</i> message.	Global
Media Delay TX (Avg) [ms], Media Delay TX (Max) [ms], Media Delay TX (Min) [ms]	The per polling interval average/min/max media delay, including both the call setup delay and the post dial delay, is delimited in time by the sending of the last dialed digit and the receiving of the first RTP packet at the call initiating endpoint.	Global
Media Delay RX (Avg) [ms], Media Delay RX (Max) [ms], Media Delay RX (Min) [ms]	The per polling interval average/min/max time elapsed between receiving the <i>CallState TsRingIn</i> message and receiving the first media packet. The media delay includes both the call setup delay and post-pickup delay.	Global
Post-Pickup Delay (Avg) [ms], Post-Pickup Delay (Max) [ms], Post-Pickup Delay (Min) [ms]	The per polling interval average/min/max time elapsed between sending the <i>SoftKeyEventMessage</i> (SoftKeyEvent = Answer) and receiving the first media packet.	Global
Dial Tone Delay (Avg) [ms]	The per polling interval time elapsed between sending the <i>OffHook</i> message or the <i>NewCall</i> softkey and receiving the <i>StationStartToneMessage (DtDialTone)</i> .	Global
Busy Tone Delay (Avg) [ms]	The per polling interval time elapsed between sending the <i>NewCall</i> softkey included in the <i>OffHook</i> message and receiving the <i>StationStartToneMessage (DtLineBusyTone)</i> .	Global

Statistic	Description	Per Channel/Global
VoIPSkinny Registrations		
Attempted Registrations	The cumulative count of attempted registrations, incremented when a phone is starting registration with the primary CCM. If the phone is already registered, the statistic is not incremented. Note: If a problem occurs with the primary CCM, the phone tries to re-register with the second CCM and the statistic is incremented.	Global
Successful Registrations	The cumulative count of successful registrations, incremented when a registration completes, that is all the registration sequence messages have been sent and replies for them have been received from the primary CCM.	Global
Failed Registrations	The cumulative count of failed registrations, incremented whenever the CCM replies with the <i>RegisterReject</i> message, the timeout for the registration function expires, or in case of connection failure.	Global
Attempted De-Registrations	The cumulative count of attempted de-registrations, incremented when a phone is starting de-registration with the primary CCM by sending the <i>Unregister</i> message.	Global
Successful De-Registrations	The cumulative count of successful de-registrations, incremented when the phone receives the reply <i>UnregisterAck</i> message from the primary CCM.	Global
Failed De-Registrations	The cumulative count of failed de-registrations, incremented when the timeout for the Skinny Unregister Client script function expires.	Global
Registration Time (Avg) [ms]	The time it takes for the registration function to complete, including the time for the establishment of the primary and secondary CCM connections, the time spent to send all the registration sequence messages and to receive the replies from the primary CCM.	Global
DeRegistration Time (Avg) [ms]	The time it takes the phone to send the Unregister message and to wait for the reply <i>UnregisterAck</i> message from the primary CCM.	Global
VoIPSkinny Registration Rates		
Attempted Registrations /s	The per polling interval attempted registration rate.	Global
Successful Registrations /s	The per polling interval successful registration rate.	Global
Attempted De-Registrations /s	The per polling interval attempted de-registration rate.	Global



Statistic	Description	Per Channel/Global
Successful De-Registrations /s	The per polling interval successful de-registration rate.	Global
VoIPSkinny Errors		
Transport Errors	The number of Skinny transport errors, occurring when a Skinny message cannot be sent due to a socket error.	Global
Trigger Errors	The number of trigger errors.	Global
RTP Errors	The total number of RTP related errors, incremented when any RTP script function is failing or exiting on the Warning or Timeout outputs.	Global
Timeout Errors	The number of script functions timeout errors.	Global
Internal Errors	The total number of internal errors.	Global
VoIPSkinny Busy Hour Call Measurements		
BHCA	The Busy Hour Call Attempts rate that represents the number of calls initiated in one hour.	Global
BHCC	The Busy Hour Call Completions rate that represents the number of calls initiated and connected in one hour.	Global
VoIPSkinny Other		
Payload Bytes Received, Payload Bytes Received/s	The inbound RTP payload bytes number, inbound RTP payload bytes rate.	Both
Triggers Sent, Triggers Sent /s	The number of triggers sent, the rate of triggers sent.	Global
Triggers Received, Triggers Received /s	The number of triggers received, the rate of triggers received	Global
Triggers Bytes Sent, Triggers Bytes Sent /s	The number of trigger bytes sent, the rate of trigger bytes sent.	Global
Triggers Bytes Received, Triggers Bytes Received /s	The number of trigger bytes received, the rate of trigger bytes received.	Global

Table 29-1. VoIPSkinnyPeer Statistics

Statistic	Description	Per Channel/Global
-----------	-------------	--------------------

VoIPSkinny Channels

Statistic	Description	Per Channel/Global
Successful Channels	The per polling interval number of COMPLETED channels. A channel is COMPLETED if all the channel loops were COMPLETED.	Global
Warning Channels	The per polling interval number of WARNING channels. A channel is WARNING if all the channel loops were COMPLETED or WARNING and at least one loop had a WARNING result.	Global
Failed Channels	The per polling interval number of FAILED channels. A channel is FAILED if all the loops of the channel were COMPLETED, WARNING, or FAILED and at least one loop was FAILED.	Global
Aborted Channels	The per polling interval number of ABORTED channels. A channel is ABORTED if all the channel loops of the channel were COMPLETED, WARNING, FAILED, or ABORTED and at least one loop was ABORTED.	Global
Active Channels	The per polling interval number of active channels. Active channels are the channels executing a scenario channel functions flow.	Global
Total Channels	The per polling interval total number of channels, a sum of active and non-active channels.	Global
VoIPSkinny Loops		
Successful Channel Loops	The cumulative count of COMPLETED channel loops. A channel loop is COMPLETED if all executed script functions in the corresponding scenario channel produced SKIPPED or COMPLETED function results.	Global
Warning Channel Loops	The cumulative count of WARNING channel loops. A channel loop has a WARNING result if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, or WARNING function results and at least one script function had a WARNING result.	Global
Failed Channel Loops	The cumulative count of FAILED channel loops. A channel loop is FAILED if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, WARNING, or FAILED function results and at least one script function had a FAILED result.	Global



Statistic	Description	Per Channel/Global
Aborted Channel Loops	The cumulative count of ABORTED channel loops. A channel loop is FAILED if all executed script functions in the corresponding scenario channel produced SKIPPED, COMPLETED, WARNING, FAILED, or ABORTED function results and at least one script function had an ABORTED result.	Global
Total Channel Loops	The cumulative count of total executed loops.	Global
Inter Loop Duration (Avg) (ms)	The average time gap between loops.	Global
VoIPSkinny Calls		
Attempted Calls	The number of originated calls - not necessarily answered or connected. This statistic is updated whenever the <code>skNewCall</code> , <code>SkRedial</code> , or <code>SkMeetMeConfrn</code> softkey is sent. It is also incremented when a transfer or a conference is initiated, i.e. a <code>SkTrnsfer</code> or a <code>SkConfrn</code> softkey is sent for the first time.	Global
Connected Calls	The number of calls successfully connected from the originator point of view. This statistic is incremented whenever the originating side receives a <code>StartMediaTransmission</code> message.	Global
Received Calls	The number of received calls, not necessarily answered. This statistic is incremented whenever the <code>CallState TsRingIn</code> or the <code>CallState TsWaitCalling</code> message is received by the call terminating side.	Global
Answered Calls	The number of calls received, successfully answered and connected. This statistic is incremented whenever the receiving side receives a <code>StartMediaTransmission</code> message.	Global
Transferred Calls	The number of transferred calls. This statistic is updated whenever the <code>CallState TsOnHook</code> message is received, after the second <code>SkTransfer</code> softkey was sent, to complete the transfer.	Global

Statistic	Description	Per Channel/Global
Active Calls	Number of active calls at one time. This statistic is incremented when the <code>StartMediaTransmission</code> message is received, and is decremented when the <code>CallState TsOnHook</code> message is received or at the end of the loop.	Global
Busy Calls	The number of calls that were rejected with the party being busy as a cause. This statistic is incremented when the <code>CallState TsBusy</code> message is received.	Global
End Call Initiated	The number of initiated end call operations. This statistic is updated whenever the <code>skEndCall</code> softkey is sent.	Global
End Call Received	This statistic is updated whenever the <code>CallState TsOnHook</code> message is received, without the <code>skEndCall</code> softkey being previously sent.	Global
End Calls Completed	This statistic is updated whenever the <code>CallState TsOnHook</code> message is received after having sent a <code>SkEndCall</code> softkey.	Global
Answered Calls TX	The number of acknowledged calls (answered), but not necessarily connected. This statistic is incremented when the <code>CallState TsConnected</code> message is received by originating side.	Global
Attempted Answered Calls RX	The number of calls received and answered, but not necessarily completed. This statistic is incremented whenever the <code>SkAnswer</code> softkey event is sent by the receiving side.	Global

VoIPSkinny Call Rates



Statistic	Description	Per Channel/Global
Attempted Calls /s, Connected Calls /s, Received Calls /sec, Answered Calls /s, Rejected Calls /s, Calls with Authentication Required /s, Transferred Calls /s, Busy Calls /s, Redirected Calls /s	The per polling interval rates corresponding to some of the previous VoIPSkinny Calls statistics.	Global

VoIPSkinny Call Times

Call Setup Time TX Avg (ms)	The time it takes to setup a call and receive a call acceptance acknowledgement from the remote endpoint, including the post-dial delay and computed as the time between the <code>SkNewCall</code> and the <code>StartMediaTransmission</code> events.	Global
Call Setup Time RX Avg (ms)	The time from receiving the request for the call until receiving the final caller acknowledgment that the call setup has been successfully completed, computed as time between the <code>CallState TsRingIn</code> and the <code>StartMediaTransmission</code> events.	Global
Talk Time (Avg)	The active conversational between the <code>StartMediaTransmission</code> message until the <code>CloseReceiveChannel</code> message.	Global
End Call Time (Avg)	The time between the sending the <code>SkEndCall</code> softkey and receiving the <code>CallState TsOnHook</code> message.	Global
Total Call Duration (Avg)	The total call duration comprising the call setup, talk, and end call times.	Global

VolpSkinny Delays

Statistic	Description	Per Channel/Global
Post Dial Delay (Avg) [ms]	The per polling interval time elapsed between sending of the last dialed number digit and receiving of a <code>CallStateMessage</code> <code>TsRingOut</code> , <code>TsCongestion</code> , <code>TsBusy</code> , or <code>TsInvalidNumber</code> message.	Global
Media Delay TX (Avg) [ms], Media Delay TX (Max) [ms], Media Delay TX (Min) [ms]	The per polling interval average/min/max media delay, including both the call setup delay and the post dial delay, is delimited in time by the sending of the last dialed digit and the receiving of the first RTP packet at the call initiating endpoint.	Global
Media Delay RX (Avg) [ms], Media Delay RX (Max) [ms], Media Delay RX (Min) [ms]	The per polling interval average/min/max time elapsed between receiving the <code>CallState</code> <code>TsRingIn</code> message and receiving the first media packet. The media delay includes both the call setup delay and post-pickup delay.	Global
Post-Pickup Delay (Avg) [ms], Post-Pickup Delay (Max) [ms], Post-Pickup Delay (Min) [ms]	The per polling interval average/min/max time elapsed between sending the <code>SoftKeyEventMessage</code> (<code>SoftKeyEvent = Answer</code>) and receiving the first media packet.	Global
Dial Tone Delay (Avg) [ms]	The per polling interval time elapsed between sending the <code>OffHook</code> message or the <code>NewCall</code> softkey and receiving the <code>StationStartToneMessage</code> (<code>DtDialTone</code>).	Global
Busy Tone Delay (Avg) [ms]	The per polling interval time elapsed between sending the <code>NewCall</code> softkey included in the <code>OffHook</code> message and receiving the <code>StationStartToneMessage</code> (<code>DtLineBusyTone</code>).	Global

VoIPSkinny Registrations



Statistic	Description	Per Channel/Global
Attempted Registrations	<p>The cumulative count of attempted registrations, incremented when a phone is starting registration with the primary CCM. If the phone is already registered, the statistic is not incremented.</p> <p>Note: If a problem occurs with the primary CCM, the phone tries to re-register with the second CCM and the statistic is incremented.</p>	Global
Successful Registrations	The cumulative count of successful registrations, incremented when a registration completes, that is all the registration sequence messages have been sent and replies for them have been received from the primary CCM.	Global
Failed Registrations	The cumulative count of failed registrations, incremented whenever the CCM replies with the <code>RegisterReject</code> message, the timeout for the registration function expires, or in case of connection failure.	Global
Attempted De-Registrations	The cumulative count of attempted de-registrations, incremented when a phone is starting de-registration with the primary CCM by sending the <code>Unregister</code> message.	Global
Successful De-Registrations	The cumulative count of successful deregistrations, incremented when the phone receives the reply <code>UnregisterAck</code> message from the primary CCM.	Global
Failed De-Registrations	The cumulative count of failed deregistrations, incremented when the timeout for the Skinny Unregister Client script function expires.	Global
Registration Time (Avg) [ms]	The time it takes for the registration function to complete, including the time for the establishment of the primary and secondary CCM connections, the time spent to send all the registration sequence messages and to receive the replies from the primary CCM.	Global
DeRegistration Time (Avg) [ms]	The time it takes the phone to send the <code>Unregister</code> message and to wait for the reply <code>UnregisterAck</code> message from the primary CCM.	Global
VoIPSkinny Registration Rates		
Attempted Registrations /s	The per polling interval attempted registration rate.	Global

Statistic	Description	Per Channel/Global
Successful Registrations /s	The per polling interval successful registration rate.	Global
Attempted De-Registrations /s	The per polling interval attempted de-registration rate.	Global
Successful De-Registrations /s	The per polling interval successful de-registration rate.	Global
VoIPSkinny Errors		
Transport Errors	The number of Skinny transport errors, occurring when a Skinny message cannot be sent due to a socket error.	Global
Trigger Errors	The number of trigger errors.	Global
RTP Errors	The total number of RTP related errors, incremented when any RTP script function is failing or exiting on the Warning or Timeout outputs.	Global
Timeout Errors	The number of script functions timeout errors.	Global
Internal Errors	The total number of internal errors.	Global
VoIPSkinny Busy Hour Call Measurements		
BHCA	The Busy Hour Call Attempts rate that represents the number of calls initiated in one hour.	Global
BHCC	The Busy Hour Call Completions rate that represents the number of calls initiated and connected in one hour.	Global
VoIPSkinny Other		
Payload Bytes Received, Payload Bytes Received/s	The inbound RTP payload bytes number, inbound RTP payload bytes rate.	Both
Triggers Sent, Triggers Sent /s	The number of triggers sent, the rate of triggers sent.	Global



Statistic	Description	Per Channel/Global
Triggers Received, Triggers Received /s	The number of triggers received, the rate of triggers received	Global
Triggers Bytes Sent, Triggers Bytes Sent /s	The number of trigger bytes sent, the rate of trigger bytes sent.	Global
Triggers Bytes Received, Triggers Bytes Received /s	The number of trigger bytes received, the rate of trigger bytes received.	Global

Note: Statistics from this category are not displayed in any of the pre-defined views, but can be assigned to custom statistics views.

EXAMPLE

```
set Activity_VoIPSkinnyPeer1 [$myNetTraffic activityList.appendItem \  
    -protocolAndType          "VoIPSkinny Peer" ]  
  
set Timeline1 [::IxLoad new ixTimeline]  
$Timeline1 config \  
    -rampUpValue              1 \  
    -rampUpType               0 \  
    -offlineTime              0 \  
    -rampDownTime             20 \  
    -standbyTime              0 \  
    -iterations               1 \  
    -rampUpInterval           1 \  
    -sustainTime              20 \  
    -timelineType             0 \  
    -name                     "Timeline1"  
  
$Activity_VoIPSkinnyPeer1 config \  
    -enable                   1 \  
    -name                     "VoIPSkinnyPeer1" \  
    -enableConstraint         false \  
    -userObjectiveValue       100 \  
    -constraintValue          100 \  
    -userObjectiveType        "channels" \  
    -timeline                 $Timeline1  
  
$Activity_VoIPSkinnyPeer1 agent.config \  
    -enable                   1 \  
    -name                     "VoIPSkinnyPeer1"
```

SEE ALSO

config



Scenario Settings

VoIP Skinny Peer Scenario Settings

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.scenarioSettings.config \  
-option      value
```

DESCRIPTION

Scenario Settings specifies the test scenario file that will be used by the Tcl script.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOption` subcommands defined in the `ixConfig` command.

`scenarioFile`

The full path to the test scenario file for the activity.

`activeScenarioChannel`

Test scenario channel (0-based index) that is associated with the VoIP Skinny Peer activity. Default = 0.

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 agent.pm.scenarioSettings.config \  
-scenarioFile              "E:\\ScenarioTestFiles\\Skinny.tst" \  
-activeScenarioChannel     0
```

SEE ALSO

Execution Settings

VoIP Skinny Peer Execution Settings

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.executionSettings.config \  
-option value
```

DESCRIPTION

This object defines the execution settings for the VoIP Skinny Peer.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

`loopMode`

Defines how many loops are executed for every voice channel corresponding to this activity.

Value	Description
0 (default)	Loop for the entire test duration.
1	Execute a number of loops. Specify the number of loops in <code>loopCount</code> .

`loopCount`

If `loopMode` is 1, this option defines the number of loops that the test performs. Default="1".

`loopPreDelay`

Delay before first loop (ms). Default="0", min="0" max="3600000".

`loopMidDelay`

Delay between loops (ms). Default="0" min="0" max="3600000".

`aliases`



Number of aliases (phone numbers) per channel. Default="1", min="1" max="16000".

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 agent.pm.executionSettings.config \  
-loopMidDelay          0 \  
-loopCount             1 \  
-loopPreDelay         0 \  
-loopMode              0 \  
-aliases                1
```

SEE ALSO

Dial Plan

VoIP Skinny Peer Dial Plan

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.dialPlan.config \  
-option      value
```

DESCRIPTION

The Dial Plan object configures the registration names, phone numbers, and source, destination, and transfer addresses for the channels/phones emulated by the VoIP Skinny Peer activity.

SUBCOMMANDS

None.

OPTIONS

The options for this command are configured and read using the standard `config`, `cget`, and `getOption` subcommands defined in the `ixConfig` command.

Source options

`useSourcePhoneBook`

Method used to select phone number.

Value	Usage
0	Use the phone number specified by pattern.
1	Use the phone number specified by Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (`useSourcePhoneBook=1`). The generated Tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

`sourcePhoneSpecified`

If `useSourcePhoneBook` is 0, this option specifies the phone number. You can use sequence generators in this field to generate multiple phone numbers. See the sequence generator appendix. Default="160[00000000-]".

`sourcePhoneType`

Type of source phone number:

0 = Specified by `sourcePhoneSpecified` as digits (default).



1 = Specified by `sourcePhoneBook` as a file name.

`sourcePhoneBook`

If `useSourcePhoneBook` is 1, this option specifies the phone book entry name.

Default="<None>".

Destination options

`useDestPhoneBook`

Method used to select the phone number used to override destination phone number.

Value	Usage
0 (default)	Specify pattern.
1	Specify Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (`useDestPhoneBook=1`). The generated Tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

`destPhoneSpecified`

If `useDestPhoneBook` is 0, this option specifies the phone number.

Default="170[00000000-]".

`destPhoneType`

Type of destination phone number:

0 = Specified by `destPhoneSpecified` as digits (default).

1 = Specified by `destPhoneBook` as a file name.

`destPhoneBook`

If `useDestPhoneBook` is 1, this option specifies the phone book file name.

Default="<None>".

`symDestStr`

String identifying the VoIP Skinny Peer that is the destination for traffic from this VoIP Skinny Peer.

Default="None".

ovrDestPhone

Enables overriding of phone number from the destination VoIP Skinny Peer.

Value	Usage
0 (default)	Disabled
1	Enabled

Registration options

useSourceRegBook

Method used to select registration names.

Value	Usage
0	Use the phone number specified by pattern.
1	Use the phone number specified by Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (useSourcePhoneBook=1). The generated Tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

sourceRegSpecified

If useSourceRegBook is 0, this option specifies the phone number. Default="SEP0000000[15000-]".

sourceRegType

Type of registration names.

0 = Specified by sourceRegSpecified as digits (default).

1 = Specified by sourceRegBook as a file name.

sourceRegBook

If useSourceRegBook is 1, this option specifies the phone book file name. Default="<None>".

Transfer and Conference options

useTransferPhoneBook

Method used to select the phone number used to override transfer and conference phone number.



Value	Usage
0 (default)	Specify pattern.
1	Specify Phonebook entry.

Note: This options appears in the generated tcl code only if the test configuration contains a reference to a Phonebook entry (`transferPhoneBook`). The generated Tcl script will run only on the machine it has been generated on and only if the corresponding Phonebook entry has not yet been deleted since the generation of the Tcl code.

`transferPhoneSpecified`

If `useTransferPhoneBook` is 0, this option specifies the phone number. Default="180[00000000-]".

`transferPhoneType`

Type of transfer phone number type.

0 = Specified by `transferPhoneSpecified` as digits (default).

1 = Specified by `transferPhoneBook` as a file name.

`transferPhoneBook`

If `useTransferPhoneBook` is 1, this option specifies the phone book file name. Default="<None>".

`symTransferStr`

String identifying the VoIP Skinny Peer used for transfer and conference functions. Default="None".

`ovrTransferPhone`

Enables overriding of phone number from the transfer and conferencing VoIP Skinny Peer.

Value	Usage
0 (default)	Disabled
1	Enabled

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 agent.pm.dialPlan.config \  
-useSourcePhoneBook      0 \  
-sourcePhoneSpecified    "160\[00000000-\]" \  
-sourcePhoneType         0 \  
-sourcePhoneBook         "&lt;None&gt;" \  
-useDestPhoneBook        0 \  
-destPhoneSpecified      "170\[00000000-\]" \  
-destPhoneType           0 \  
-destPhoneBook           "&lt;None&gt;" \  
-symDestStr              "None" \  
-ovrDestPhone            false \  
-useSourceRegBook        0 \  
-sourceRegSpecified      "SEP0000000\[15000-\]" \  
-sourceRegType           0 \  
-sourceRegBook           "&lt;None&gt;" \  
-useTransferPhoneBook    0 \  
-transferPhoneSpecified  "180\[00000000-\]" \  
-transferPhoneType       0 \  
-transferPhoneBook       "&lt;None&gt;" \  
-symTransferStr          "None" \  
-ovrTransferPhone        false \  

```

SEE ALSO



Skinny Settings

VoIP Skinny Peer Signaling Settings

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.signalingSettings.config \  
-option value
```

DESCRIPTION

This object defines the VoIP Skinny Peer Skinny settings.

SUBCOMMANDS

None.

OPTIONS

`enableSkinny`

Enables use of Skinny signaling for the VoIP Skinny Peer.

0 = Skinny disabled

1 = Skinny enabled (default)

`skinny_enableTos`

Enables use of TOS/DSCP. Use the `skinny_tos` option to specify the TOS/DSCP value.

0 = TOS disabled (default)

1 = TOS enabled

`skinny_tos`

If `skinny_enableTos` is 1, this option sets the value of the TOS bits.

Value	Usage
0 (default)	Best Effort (0x00)
1	Class 1 (0x20)
2	Class 2 (0x40)
3	Class 3 (0x60)
4	Class 4 (0x80)
5	Express Forwarding (0xA0)

Value	Usage
6	Control (0xC0)

`seqRegistration`

Enables Sequential Registration. 0 = Disabled (default), 1= Enabled.

`failDirectly`

If `seqRegistration = 1`, this option controls the registration failure behavior enforced by the Cisco CallManager.

0 = Do not fail if previously failed (default),

1= Fail registration if previously failed.

`skinnyVersion`

Version of Skinny protocol used.

0 = Skinny version 4 (default)

1 = Skinny version 5

`ccm_number`

Cisco Call Manager number. (default = 0).

`cmVersion`

Cisco Call Manager version (default = 3.4). Note: This is a string value.

`secondaryKeepAlive`

Interval (in seconds) at which secondary keep alive messages are sent. (default = 60).

`primaryKeepAlive`

Interval (in seconds) at which primary keep alive messages are sent. (default = 60).

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 agent.pm.skinnySettings.config \  
-seqRegistration           false \  
-skinnyVersion            0 \  
-skinnyServer             false \  
-_gbSeqRegistration       false \  
-skinny_tos               0 \  
-skinny_enableTos        false \  
-_skinnyClient1          false \  
-ccm_number               0 \  
-failDirectly            false \  
-cmVersion                "3.4" \  
-secondaryKeepAlive      60 \  
-primaryKeepAlive        30 \  
-enableSkinny            true \  
-_enableSkinny1         false
```

SEE ALSO

Call Managers

Call Managers

List of VoIP Skinny Peer Call Managers

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.signalingSettings.appendItem \  
-option value
```

DESCRIPTION

This object contains the list of VoIP Skinny Peer Skinny Call Managers.

SUBCOMMANDS

The following subcommands are available to handle options. Except where noted, no value is returned; an exception is raised in the case of an error. In all cases where they are used the `option` must begin with a hyphen (-). The `value` must be of a type appropriate for the option.

`appendItem option value option value...`

The `appendItem` subcommand may be used to add an item to a list. Any number of options in the listed item may be set as part of the append.

`configItem index option value option value...`

The `configItem` subcommand may be used to configure a particular item in a list. Any number of options in the list item may be set. The `index` argument is used to indicate which item in the list is to be configured.

`clear`

The `clear` subcommand may be used to delete all listed items from a list.

`deleteItem index`

The `deleteItem` subcommand may be used to delete a listed item from a list. The `index` argument is used to indicate which item in the list is to be configured.

`getItem index`

The `getItem` subcommand may be used to retrieve an item from a list. The `index` argument is used to indicate which item in the list is to be retrieved. This subcommand returns the object from the list.

`indexCount`

The `indexCount` subcommand returns the number of objects in the list.

OPTIONS

`id`



Name of the Call Manager. Default="callManager".

cmPort

Call Manager port number. Default="2000"

cmAddress

Call Manager IP address. Default="127.0.0.1"

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 \ agent.pm.skinnySettings.callManagers.appendItem  
-id          "callManager" \  
-cmPort      "2000" \  
-cmAddress   "127.0.0.1"
```

SEE ALSO

Skiny Settings

Codec Settings

VoIP Skinny Peer Codec Settings

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.codecSettings.codecs.appendItem \  
-option value  
$Activity_VoIPSkinnyPeer1 agent.pm.codecSettings.dataCodecs.appendItem \  
-option value
```

DESCRIPTION

Codec Settings contains the list of codecs that will be used by the VoIP Skinny Peers in the test. Codec Settings is a list of one or more `codec` (audio codec) or `dataCodec` objects. To add `codec` or `dataCodec` objects, use the `appendItem` command. To clear the codec settings, use the `clear` subcommand.

SUBCOMMANDS

`clear`

Clears the list of codec settings. For example:

```
$Activity_VoIPSkinnyPeer1 agent.pm.codecSettings.codecs.clear
```

OPTIONS

None.

EXAMPLE

See the examples for `Data Codecs` and `Codecs`.

SEE ALSO

`Data Codecs`, `Codecs`

Data Codecs

VoIP Skinny Peer Data Codecs

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.codecSettings.dataCodecs.appendItem \  
-option value
```

DESCRIPTION

Data Codecs configures a data codec object, which is added to the `Codec Settings` list of codecs.

SUBCOMMANDS

None.

OPTIONS

`id`

Codec type. One of the following:

Codec	Description
Rtp2833Events	Named Events Payload format used for carrying DTMF digits and other line and trunk signals as events.
Rtp2833Tones	RTP Payload format that can represent tones consisting of one or more frequencies.

`dPayloadType`

Payload type used for RTP data packets. Default=(see table) min="96" max="127"

Codec	Default value for dPayloadType
Rtp2833Events	100
Rtp2833Tones	101

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 \ agent.pm.codecSettings.dataCodecs.clear
```

```
$Activity_VoIPSkinnyPeer1 \ agent.pm.codecSettings.dataCodecs.appendItem \  
-id          "Rtp2833Events" \  
-dPayloadType 100
```

```
$Activity_VoIPSkinnyPeer1 \ agent.pm.codecSettings.dataCodecs.appendItem \  
-id          "Rtp2833Tones" \  
-dPayloadType 101
```

SEE ALSO

Codec Settings

Codecs

VoIP Skinny Peer Audio Codec

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.codecSettings.codecs.appendItem \  
-option value
```

DESCRIPTION

Codecs configures an audio codec object, which is added to the `Codec Settings` list of codecs. To add a `codec` object, use the `appendItem` command.

SUBCOMMANDS

None.

OPTIONS

`id`

Codec type. One of the following:

Codec	Description
CodecAMR	Adaptive multi-rate codec
CodecG711u	G.711 mu-law codec
CodecG711a	G.711 A-law codec
CodecG723x153	G.723.1 codec @ 5.3 kbps
CodecG723x163	G.723.1 codec @ 6.3 kbps
CodecG726x16	G.726 codec @ 16 Kbps
CodecG726x24	G.726 codec @ 24 Kbps
CodecG726x32	G.726 codec @ 32 Kbps
CodecG729A	G.729 Annex-A codec

Options for CodecAMR

`dPayloadIn`

Incoming dynamic payload type. Default="98" min="0" max="127".

`dPayloadOut`

Outgoing dynamic payload type. Default="98" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 14. Default=14.

payloadFormat

Payload format.

Value	Usage
0 (default)	Bandwidth-efficient format
1	Octet-aligned format

mode

Codec bit rate. One of the following:

Mode	Description
0 (default)	4.75 kbps
1	5.15 kbps
2	5.90 kbps
3	6.70 kbps
4	7.40 kbps
5	7.95 kbps
6	10.20 kbps
7	12.20 kbps

Options for CodecG711u

dPayloadIn

Incoming dynamic payload type. Default="0" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="0" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG711a

dPayloadIn



Incoming dynamic payload type. Default="8" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="8" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 40, 80, 160, 240. Default=160.

Options for CodecG723x153

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 20. Default=20.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG723x163

dPayloadIn

Incoming dynamic payload type. Default="4" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="4" min="0" max="127".

frameSize

Bytes per frame. Must be one of the following: 24. Default=24.

Options for CodecG726x16

dPayloadIn

Incoming dynamic payload type. Default="102" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="102" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 20, 40, 60. Default=20.

Options for CodecG726x24

dPayloadIn

Incoming dynamic payload type. Default="103" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="103" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian



Option	Description
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 30, 60, 90. Default=30.

Options for CodecG726x32

dPayloadIn

Incoming dynamic payload type. Default="104" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="104" min="0" max="127".

byteOrder

Byte order.

Option	Description
0 (default)	Big Endian
1	Little Endian

frameSize

Bytes per frame. Must be one of the following: 40, 80, 120. Default=40.

Options for CodecG729

dPayloadIn

Incoming dynamic payload type. Default="18" min="0" max="127".

dPayloadOut

Outgoing dynamic payload type. Default="18" min="0" max="127".

cbxFrameSize

Bytes per frame. Must be one of the following: 10, 20, 30, 40, 50, Custom. Default=10.

customFrameSize

If `cbxFrameSize` is Custom, this option configures the custom frame size. Default="120"
min="10" max="200".

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 agent.pm.codecSettings.codecs.clear

$Activity_VoIPSkinnyPeer1 \ agent.pm.codecSettings.codecs.appendItem \
-id          "CodecG711u" \
-dPayloadOut 0 \
-dPayloadIn  0 \
-frameSize  160

$Activity_VoIPSkinnyPeer1 \ agent.pm.codecSettings.codecs.appendItem \
-id          "CodecG711a" \
-dPayloadOut 8 \
-dPayloadIn  8 \
-frameSize  160
```

SEE ALSO

Codec Settings



RTP Settings

VoIPSkinny Peer RTP Settings

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.rtpSettings.config \  
-option value
```

DESCRIPTION

RTP Settings configures the VoIPSIPPeer RTP transport settings.

SUBCOMMANDS

None.

OPTIONS

enableRTP

Enables use of RTP to transport the media traffic.

0 = disabled (default)

1 = enabled

rtpPort

RTP port number. Default="10000".

Note: Valid port numbers are between 1000 and 65534.

enableRTCP

Enables the sending and receiving of RTCP packets.

chEnableHwAcc

If true, enables hardware acceleration for RTP traffic. Default=false.

enableAdvStatCalc

Enables the computation of advanced RTP statistics.

enablePerStream

Enables computation of per-stream statistics.

```
enableMDI
```

Enables the Media Delay Index.

```
enableNBExec
```

If `true`, all RTP functions from a scenario execute in a non-blocking mode, i.e the current function from a channel executes in the background, allowing the execution to continue on that channel with the next script function. Default= `False`.

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 agent.pm.rtpSettings.config \  
-enableRTP                true \  
-enableRTCP               false \  
-enableMDI                false \  
-chEnableHwAcc            true \  
-chDisableHwAcc           false \  
-enableAdvStatCalc        false \  
-enablePerStream          false \  
-rtpPort                  "[10000-65535,4]" \  
-enableNBExec              false
```

SEE ALSO

Audio Settings

VoIPSkinny Peer audio settings

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.audioSettings.config
```

DESCRIPTION

The Audio Settings configure the VoIPSkinny Peer audio RTP settings.

SUBCOMMANDS

None.

OPTIONS

`enableAudio`

If selected, audio script functions are executed, otherwise they are skipped.

`audioClip`

The played audio clip file.

`playTypeAudio`

The mode in which the clip is played.

Value	Usage
0 (default)	The clip is played for clip duration or for the duration of the Talk Time parameter in the case of BHCA/CPS/LPS objectives.
1	The clip is played for a user-defined duration.

`audioDurationUnit`

The play duration unit, which can be milliseconds (0), seconds (1), minutes (2), or hours (3).

`outputLevel`

The output level of the played clip.

`enableTosRtp`

Enables use of TOS/DSCP. Use the `rtpTos` option to specify the TOS/DSCP value. Default= False

rtpTosVal

The Type of Service (TOS/DSCP) byte setting in the sent RTP packets has one of the following values:

- Best Effort (0x00): Routine service
- Class 1 (0x20): Priority service, Assured Forwarding class 1
- Class 2 (0x40): Immediate service, Assured Forwarding class 2
- Class 3 (0x60): Flash, Assured Forwarding class 3
- Class 4 (0x80): Flash-override, Assured Forwarding class 4
- Express Forwarding (0xA0): Critical-ecp
- Control (0xC0): Internet-control
- Custom: A user-specified value.

useMOS

Enables the computation of MOS scores. Default= False.

enableAudioOWD

If true, IxLoad computes the One-way Delay metric, a network measurement specifying the amount of time (in ms) that a packet has spent on the network before it was received on the destination side.
Default= False

useJitter

If true, enables use of a jitter buffer. Default= False.

jitMs

If useJitter is 1, this option configures the size of the jitter buffer, in milliseconds. Default="20"
min="1" max="3000".

useJitComp

If true, enables dynamic modification of the jitter buffer size. Default= False.

jitCMs

If useJitComp is 1, this option configures the maximum size in of the jitter buffer, in milliseconds.
Default="1000" min="0" max="3000".

jitCMaxDrop

If useJitComp is 1, this option configures the condition - a maximum number of consecutive packets dropped - that determines the jitter buffer size to be increased.



enableQoV

If true, this enables QoV P.862 PESQ and P.56 QoV computation. Default= False.

channelTypeQoV

When enableQoV is true, this specifies the objective type as either of the following:

- Number of channels (0)
- Percentage (1)

valueQoV

When enableQoV is true, this specifies the number of channels for which PESQ and P.56 QoV metrics are computed (when channelTypeQoV is 0). Alternatively this represents the percentage of channels for which PESQ and P.56 QoV metrics are computed (when channelTypeQoV is 1).

unitsQoV

The channels selection mode, which can be any of the following:

- First channels (0)
- Last channels (1)
- Evenly-spaced channels (2)
- Random (3)

metricsQoV

When enableQoV is true, this specifies the metric that is calculated by the Zion card. Available options are:

- PESQ and P.56 (0)
- PESQ (1)
- P56 (2)

useSilence

If true, RTP packets containing artificial background noise are sent when no other media (DTMF, MF, real payload, and so on) is sent over the communication channel. Default= False.

silenceMode

If useSilence is 1, this option configures the silence mode.

Value	Usage
0	Null data encoded

Value	Usage
1 (default)	Comfort noise.

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 agent.pm.audioSettings.config \  
-enableAudio true \  
-audioClip "US_042.wav" \  
-playTypeAudio 0 \  
-audioDurationUnit 1 \  
-audioDuration 10 \  
-outputLevel -20 \  
-enableAudioOWD false \  
-enableTosRtp false \  
-rtpTosVal 32 \  
-useMos false \  
-useJitter false \  
-jitMs 20 \  
-useJitComp false \  
-jitCms 1000 \  
-jitCMaxDrop 7 \  
-enableQoV false \  
-channelTypeQoV 0 \  
-valueQoV 100 \  
-unitsQoV 0 \  
-metricsQoV 0 \  
-useSilence false \  
-silenceMode 1 \  

```

SEE ALSO

Other Settings

VoIPSkinny Peer Other Settings

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.otherSettings.config \  
-option value
```

DESCRIPTION

This object configures the VoIP Skinny Peer activity's miscellaneous options.

SUBCOMMANDS

None.

OPTIONS

VOIP_Var0

The VOIP_Var1...VOIP_Var5 and VOIP_IPAddr1...VOIP_IPAddr5 string-type variables supporting generator expressions enable you to generate 10 series of global variables whose values are used at runtime by the simulated Skinny phones/channels. `Default= ""`.

Use the VOIP_Var1...VOIP_Var5 variables to represent phone numbers, and the VOIP_IPAddr1...VOIP_IPAddr5 to represent IP addresses.

VOIP_Var1

See VOIP_Var0.

VOIP_Var2

See VOIP_Var0.

VOIP_Var3

See VOIP_Var0.

VOIP_Var4

See VOIP_Var0.

VOIP_IPAddress0

See VOIP_Var0.

VOIP_IPAddress1

See VOIP_Var0.

VOIP_IPAddress2

See VOIP_Var0.

VOIP_IPAddress3

See VOIP_Var0.

VOIP_IPAddress4

See VOIP_Var0.

ipPreference

Type of addressing to be used on the subnet that the VOIP Skinny Peer runs on.

Value	Usage
0 (default)	IPv4
1	IPv6

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 agent.pm.otherSettings.config \  
-ipPreference 0 \  
-VOIP_Var1 " \  
-VOIP_Var0 " \  
-VOIP_Var3 " \  
-VOIP_Var2 " \  
-VOIP_Var4 " \  
-VOIP_IPAddress4 " \  
-VOIP_IPAddress1 " \  
-VOIP_IPAddress0 " \  
-VOIP_IPAddress3 " \  
-VOIP_IPAddress2 "
```

SEE ALSO



Custom Activity Link Settings

VoIP Skinny Peer CustomActivityLinkSettings

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 agent.pm.customActivityLinkSettings.config \  
-option      value
```

DESCRIPTION

CustomActivityLinkSettings configures the settings for the BHCA objective for VoIPSkinny Peer activities. This options in this object correspond to the controls on the Custom Parameters tab for a NetTraffic/ActivityLink in the Timeline and Objective branch of the Test Configuration tree in the IxLoad GUI.

Note: The CustomActivityLinkSettings class has to be configured alongside the CustomParameters class that implements the same functionality.

SUBCOMMANDS

None.

OPTIONS

talkTime

If bhcaType is 0, this option specifies the Talk Time that will be used to attain the BHCA test objective. Default="40000".

interCallDuration

Inter-call duration. Default="4000".

bhcaType

Determines how the BHCA objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	BHCA will be met by specifying the talk time. Specify the talk time in talkTime.
1	BHCA will be met by specifying the number of channels. Specify the number of channels in channelsNo.

channelsNo



If `bhcaType` is 1, this option specifies the number of channels that will be used to attain the BHCA test objective. Default="100".

`callSetupTime`

Estimated call setup time. Default="500".

`callTeardownTime`

Estimated call teardown time. Default="500".

`bhcaObjectiveValue`

BHCA objective value. Default="80000".

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 \ agent.pm.customActivityLinkSettings.config\  
-talkTime                40000 \  
-channelsNo              100 \  
-bhcaType                0 \  
-callTeardownTime        500 \  
-interCallDuration       4000 \  
-bhcaObjectiveValue      80000 \  
-callSetupTime           500
```

SEE ALSO

Custom Parameters

VoIPSkinny Peer CustomParameters

SYNOPSIS

```
$Activity_VoIPSkinnyPeer1 customParameters.config \  
-option value
```

DESCRIPTION

CustomParameters configures the settings for the BHCA objective for VoIPSkinny Peer activities. This options in this object correspond to the controls on the Custom Parameters tab for a NetTraffic/ActivityLink in the Timeline and Objective branch of the Test Configuration tree in the GUI.

Note: The CustomParameters class has to be configured alongside the CustomActivityLinkSettings class that implements the same functionality.

SUBCOMMANDS

None.

OPTIONS

talkTime

If bhcaType is 0, this option specifies the Talk Time that will be used to attain the BHCA test objective. Default="40000".

interCallDuration

Inter-call duration. Default="4000".

bhcaType

Determines how the BHCA objective will be met: by specifying the talk time or the number of channels.

Value	Usage
0 (default)	BHCA will be met by specifying the talk time. Specify the talk time in talkTime.
1	BHCA will be met by specifying the number of channels. Specify the number of channels in channelsNo.

channelsNo

If bhcaType is 1, this option specifies the number of channels that will be used to attain the BHCA test objective. Default="100".



callSetupTime

Estimated call setup time. Default="500".

callTeardownTime

Estimated call teardown time. Default="500".

bhcaObjectiveValue

BHCA objective value. Default="80000".

EXAMPLE

```
$Activity_VoIPSkinnyPeer1 customParameters.config\  
-talkTime                40000 \  
-channelsNo              100 \  
-bhcaType                 0 \  
-callTeardownTime        500 \  
-interCallDuration       4000 \  
-bhcaObjectiveValue      80000 \  
-callSetupTime           500
```

SEE ALSO



38

Nessus (Vulnerability)

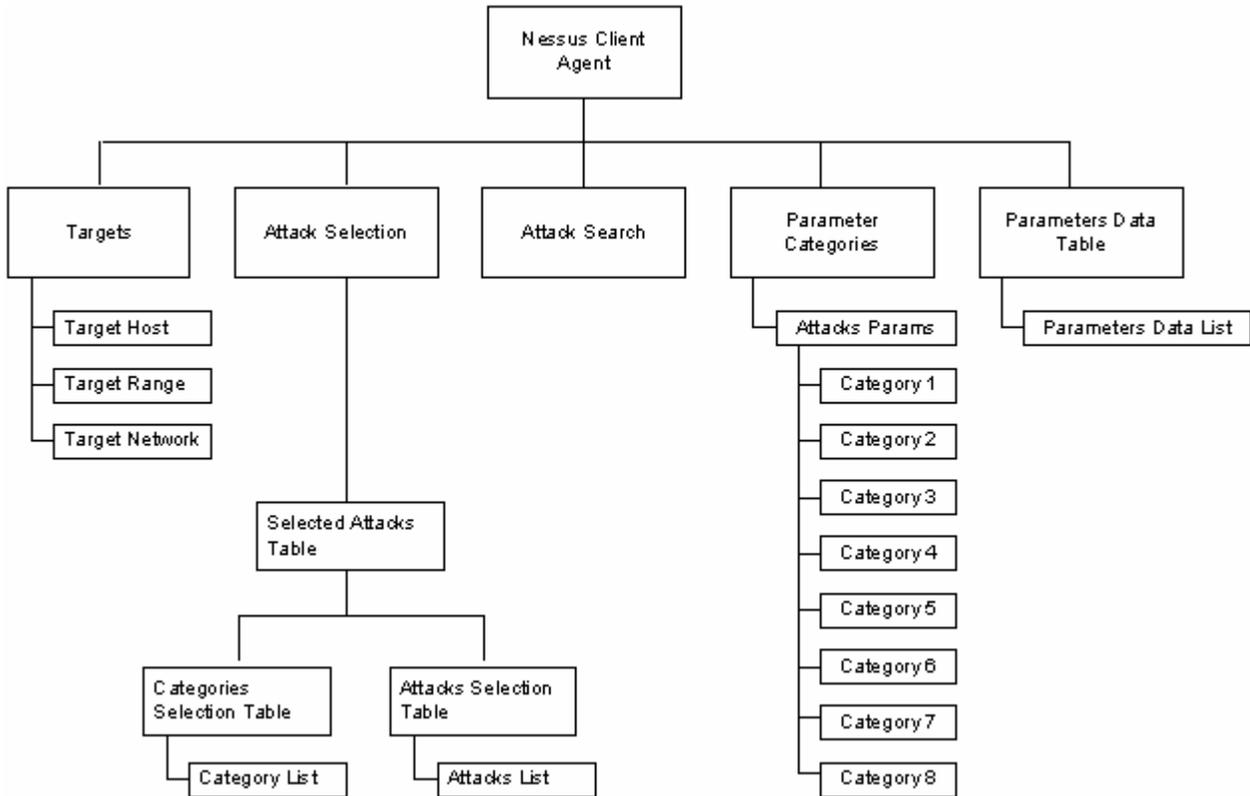
This section describes the Nessus (Vulnerability) Tcl API objects.

In this chapter:

Overview	2048
Nessus Client Agent	2052
Attacks Search	2086
Attacks Select	2087
Parameters Data Table	2091
Category List	2094
Statistics	2095

Overview

The IxLoad Nessus API consists of a client agent and its commands.



Objectives

The objectives (userObjective) you can set for Nessus are listed below. Test objectives are set in the ixTimeline object.

- simulatedUsers

Nessus Client Commands

This section describes the Nessus client commands.

Nessus Client Agent

The Nessus Client Agent command defines a client running one or more attacks against one or more hosts. Refer to `Nessus Client Agent` on page 21-6 for a full description of this command. The most significant options of this command are listed below.

Option	Description
enable	Enables the use of this client agent.
name	The name associated with this object, which must be set at object creation time.
protocol	Protocol used by the client agent.
type	Defines the agent as either a client or server.

Parameter Categories

The Parameter Categories command configures the list of parameters for vulnerattacks. Refer to `Parameter Categories` on page 21-8 for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	Parameter Category to be added. This is an item of type Attacks Params.

Attacks Params

The Attacks Params command creates a list of one or more categories of attack parameters. Refer to `Attacks Params` on page 21-11 for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	Parameter category to be added. The choices are: cat1, cat2, cat3, cat4, cat5, cat6, cat7, cat8, or cat9. Refer to the section that describes each category.

Targets

The Targets command configures the target of an attack. Refer to `Targets` on page 21-23 for a full description of this command. The most significant options of this command are listed below.

Option	Description
id	This is an item of one of the following types: Target Host, TarRange, or Target Network. (Default = "{}").



Category List

The Category List command configures a list of attack categories. Refer to *Category List* on page 21-27 for a full description of this command. The most significant options of this command are listed below.

Option	Description
categorySelected	If true, the category is selected.
categoryID	Category ID. A value for this parameter must be one of the defined attack parameter categories, such as cat1, cat2, and so on.
categoryName	Name of the category specified for categoryID.

Attacks List

The Attacks List command configures a list of attacks. Refer to *Attacks List* on page 21-28 for a full description of this command. The most significant options of this command are listed below.

Option	Description
attackSelected	If true, the attack is selected (enabled) and will be run.
attackID	Nessus ID of the attack.
attackCategoryID	Nessus ID of the category of the attack.
attackName	Name of the attack.

Attacks Select

The Attacks Select command selects attacks from a list of attack categories. Refer to *Attacks Select* on page 21-29 for a full description of this command. The most significant options of this command are listed below.

Option	Description
categoriesSelectionTable	List of categories of attacks.
attackSelectionTable	List of attacks available within an attack category.
selectedAttacksTable	Table of selected attacks.
attackDescription	Description of the selected attack.

Attacks Search

The Attacks Search command object searches the attack descriptions for a text string and returns the results in a table. Refer to `Attacks Search` on page 21-30 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>searchText</code>	Text string that will be searched for in the attack descrip
<code>attackSearchResultsTable</code>	Name of the table of that will hold the search results.

Parameters Data List

The Parameters Data List command configures a parameter and value to be used in an attack. Refer to `Parameters Data List` on page 21-31 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>paramName</code>	Name of the parameter to be added to the list.
<code>paramValue</code>	Value for <code>paramName</code> .

Parameters Data Table

The Parameters Data Table command configures a list of Parameter Data List items. Refer to `Parameters Data Table` on page 21-32 for a full description of this command. The most significant options of this command are listed below.

Option	Description
<code>paramName</code>	Name of the parameter to be added to the list.
<code>paramValue</code>	Value for <code>paramName</code> .



Nessus Client Agent

Nessus Client Agent - create a Nessus client

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.config
```

DESCRIPTION

A Nessus client agent is added to the `activityList` object. The `activityList` object is added to the `ixNetTraffic` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`enable`

Enables the use of this client agent. (Default = true).

`name`

The name associated with this object, which must be set at object creation time.

STATISTICS

EXAMPLE

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
#####
# Activity VulnerabilityClient1 of NetTraffic Traffic1@Network1
#####
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
\
    -protocolAndType                "nessus Client" ]
```

```
$Activity_VulnerabilityClient1 agent.config \  
-enable true \  
-name "VulnerabilityClient1"
```

SEE ALSO

`ixNetTraffic`



Parameter Categories

Parameter Categories—Configures the list of parameters for vulnerability attacks.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

A parameter category is added to the list of Parameter Categories object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

Parameter Category to be added. This is an item of type `Attacks Params`.

EXAMPLE

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \
```

```
-id                "cat1" \
-pref93            "yes" \
-pref100           "default" \
-pref99            "yes" \
-pref97            "5" \
-pref95            "320"
```

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \
```

```
-id                "cat2" \
-pref9             "user=%USER%&pass=%PASS%" \
-pref10            "" \
```

```
-pref11          "/"
```

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \
```

```
-id              "cat3" \
```

```
-pref12         "no" \
```

```
-pref13         "yes" \
```

```
-pref16         "Nessus <listme@listme.dsbl.org>" \
```

```
-pref14         "7" \
```

```
-pref15         "f\[a-z\]\\\.tests?"
```

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \
```

```
-id              "cat4" \
```

```
-pref19         "Avoid false alarms" \
```

```
-pref17         "0" \
```

```
-pref23         "Private LAN" \
```

```
-pref22         "no" \
```

```
-pref18         "Quiet" \
```

```
-pref20         "Quiet" \
```

```
-pref24         "yes" \
```

```
-pref21         "no"
```

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \
```

```
-id              "cat5" \
```

```
-pref29         "" \
```

```
-pref28         "" \
```

```
-pref38         "anonymous" \
```

```
-pref39         "" \
```

```
-pref34         "" \
```



```
-pref27      "" \
-pref36      "/incoming" \
-pref37      "nessus@nessus.org" \
-pref30      "" \
-pref26      "yes" \
-pref25      "no" \
-pref31      "" \
-pref32      "" \
-pref35      "" \
-pref33      "" \
-pref41      "" \
-pref40      "" \
-pref42      ""
```

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \
```

```
-id          "cat6" \
-pref63      "SYN scan" \
-pref55      "no" \
-pref54      "" \
-pref62      "no" \
-pref61      "no" \
-pref60      "no" \
-pref52      "" \
-pref59      "no" \
-pref56      "no" \
-pref49      "" \
-pref48      "" \
-pref58      "no" \
```

```
-pref57          "no" \  
-pref45          "" \  
-pref44          "no" \  
-pref47          "" \  
-pref46          "" \  
-pref50          "" \  
-pref53          "Normal" \  
-pref43          "no" \  
-pref51          ""
```

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \  
-
```

```
-id              "cat7" \  
-pref64          "Trust the permissions (drwxrwx--  
-)"
```

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \  
-
```

```
-id              "cat8" \  
-pref67          "" \  
-pref66          "" \  
-pref65          "All" \  
-pref69          "" \  
-pref68          "" \  
-pref70          "2" \  
-pref71          "5" \  
-pref72          "5" \  
-pref73          "6"
```

```
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem \  
-
```

```
-id              "cat9" \  
-
```



```
-pref78      "root" \  
-pref74      " " \  
-pref75      " " \  
-pref76      " " \  
-pref77      " "
```

SEE ALSO

Parameter Categories

Attacks Params

Attacks Params—Creates a list of one or more categories of attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

A list of attack parameters is created by modifying the options of the `pm.Attackoption` of the `paramCategories` object using `appendItem`.

SUBCOMMANDS

None.

OPTIONS

`id`

Parameter category to be added. The choices are: `cat1`, `cat2`, `cat3`, `cat4`, `cat5`, `cat6`, `cat7`, `cat8`, or `cat9`. Refer to the section that describes each category.

EXAMPLE

See `Parameter Categories`.

SEE ALSO

`Parameter Categories`



Category 1 (cat1)

Category 1—Configures the list of category 1 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

To configure a Category 1 object, you use the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

None.

OPTIONS

pref93

`auto_enable_dependencies`. (Default = "yes").

pref95

`plugins_timeout`. (Default = "320").

pref97

`checks_read_timeout`. (Default = "5").

pref99

`optimize_test`. (Default = "yes").

pref100

`port_range`. (Default = "default").

EXAMPLE

See `Parameter Categories`.

SEE ALSO

`Parameter Categories`



Category 2 (cat2)

Category 2—Configures the list of category 2 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

Configures a Category 2 object (see example below).

SUBCOMMANDS

None.

OPTIONS

pref9

HTTP login page[entry]:Login form fields. (Default = "user=%USER%&pass=%PASS%").

pref10

HTTP login page[entry]:Login form :

pref11

HTTP login page[entry]:Login page. (Default = "/").

EXAMPLE

```
set ClientTraffic [::IxLoad new ixClientTraffic options...] \
$ClientTraffic agentList.appendItem \
$ClientTraffic agentList(0).pm.paramCategories. \
attacksParams.cat2.config \
    -pref9      "user=%ATTACKER%&pass=%PASSWORD%"
```

SEE ALSO

Parameter Categories



Category 3 (cat3)

Category 3—Configures the list of category 3 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

Configures a Category 3 object..

SUBCOMMANDS

None.

OPTIONS

pref12

No archive. Choices = yes, no. (Default = "no").

pref13

Local distribution Choices = yes, no. (Default = "yes").

pref14

Misc information on News server[entry]:Max crosspost. (Default = "7").

pref15

Misc information on News server[entry]:Test group name regex. (Default = "[a-z]\.tests?").

pref16

Misc information on News server[entry]:From address. (Default = "Nessus <listme@listme.dsbl.org>").

EXAMPLE

```
set ClientTraffic [::IxLoad new ixClientTraffic options...] \
$ClientTraffic agentList.appendItem \
$ClientTraffic agentList(0).pm.paramCategories. \
attacksParams.cat3.config \
```

-pref12 "yes"

SEE ALSO

Parameter Categories



Category 4 (cat4)

Category 4—Configures the list of category 4 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

Configures a Category 4 object.

SUBCOMMANDS

None.

OPTIONS

pref17

Global variable settings[entry]:Debug level. (Default = "0").

pref18

Log verbosity. Choices = "Normal," "Quiet," "Verbose." (Default = "Quiet").

pref19

Report paranoia. Choices = "Normal," "Avoid false alarms." (Default = "Avoid false alarms").

pref20

Report verbosity Choices = "Normal," "Quiet." (Default = "Quiet").

pref21

Thorough tests (slow). Choices = "yes," "no." (Default = "no").

pref22

Enable experimental scripts. Choices = "yes," "no." (Default = "no").

pref23

Network type. Choices = "Mixed (use RFC 1918)," "Private LAN" (Default = "Private LAN")

pref24

Enable CGI scanning. Choices = "yes," "no." (Default = "yes").

EXAMPLE

```
set ClientTraffic [::IxLoad new ixClientTraffic options...] \  
$ClientTraffic agentList.appendItem \  
$ClientTraffic agentList(0).pm.paramCategories. \  
attacksParams.cat4.config \  
-pref17      "1"
```

SEE ALSO

Parameter Categories



Category 5 (cat5)

Category 5—Configures the list of category 5 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

Configures a Category 5 object.

SUBCOMMANDS

None.

OPTIONS

pref25

Only use NTLMv2. Choices = "yes," "no." (Default = "no").

pref26

Never send SMB credentials in clear text. Choices = "yes," "no." (Default = "yes").

pref27

Login configurations[entry]:SMB domain (optional). (Default = "{}").

pref28

Login configurations[password]:SMB password. (Default = "{}").

pref29

Login configurations[entry]:SMB account. (Default = "{}").

pref30

Login configurations[password]:IMAP password (sent in clear). (Default = "{}").

pref31

Login configurations[entry]:IMAP account. (Default = "{}").

pref32

Login configurations[password]:POP3 password (sent in clear). (Default = "{}").

pref33

Login configurations[entry]:POP3 account. (Default = "{}").

pref34

Login configurations[password]:POP2 password (sent in clear). (Default = "{}").

pref35

Login configurations[entry]:POP2 account. (Default = "{}").

pref36

Login configurations[entry]:FTP writeable directory. (Default = "/incoming").

pref37

Login configurations[password]:FTP password (sent in clear). (Default = "nes

pref38

Login configurations[entry]:FTP account. (Default = "anonymous").

pref39

Login configurations[password]:NNTP password (sent in clear). (Default = "{}").

pref40

Login configurations[entry]:NNTP account. (Default = "{}").

pref41

Login configurations[password]:HTTP password (sent in clear). (Default = "{}").

pref42

Login configurations[entry]:HTTP account. (Default = "{}").

EXAMPLE

```
set ClientTraffic [::IxLoad new ixClientTraffic options...] \  
$ClientTraffic agentList.appendItem \  
$ClientTraffic agentList(0).pm.paramCategories. \  
attacksParams.cat5.config \  
-pref25      "yes"
```

SEE ALSO

Parameter Categories

Category 6 (cat6)

Category 6—Configures the list of category 6 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

Configures a Category 6 object.

SUBCOMMANDS

None.

OPTIONS

pref43

Run dangerous port scans even if safe checks are set. Choices = "yes," "no." (Default = "no").

pref44

Do not scan targets not in the file. Choices = "yes," "no." (Default = "no").

pref45

Nmap (NASL wrapper)[file]:File containing grepable results. (Default = "{}").

pref46

Nmap (NASL wrapper)[entry]:Minimum wait between probes (ms)</Descrip (Default = "{}").

pref47

Nmap (NASL wrapper)[entry]:Ports scanned in parallel (min). (Default = "{}").

pref48

Nmap (NASL wrapper)[entry]:Ports scanned in parallel (max). (Default = "{}").

pref49

Nmap (NASL wrapper)[entry]:Initial RTT timeout (ms). (Default = "{}").



pref50

Nmap (NASL wrapper)[entry]:Max RTT Timeout (ms). (Default = "{}").

pref51

Nmap (NASL wrapper)[entry]:Min RTT Timeout (ms). (Default = "{}").

pref52

Nmap (NASL wrapper)[entry]:Host Timeout (ms). (Default = "{}").

pref53

Timing policy : Choices:

Value	Description
"Auto (nessus specific!)"	Auto (nessus specific!)
"Normal"	(default) Normal
"Insane"	Insane
"Aggressive"	Aggressive
"Polite"	Polite
"Sneaky"	Sneaky
"Paranoid"	Paranoid

pref54

Nmap (NASL wrapper)[entry]:Source port. (Default = "{}").

pref55

Do not randomize the order in which ports are scanned. Choices = "yes," "no." (Default = "no").

pref56

Get Identd info. Choices = "yes," "no." (Default = "no").

pref57

Fragment IP packets (bypasses firewalls) Choices = "yes," "no." (Default = "no").

pref58

Use hidden option to identify the remote OS. Choices = "yes," "no." (Default = "no").

pref59

Identify the remote OS. Choices = "yes," "no." (Default = "no").

pref60

RPC port scan Choices = "yes," "no." (Default = "no").

pref61

Service scan Choices = "yes," "no." (Default = "no").

pref62

UDP port scan. Choices = "yes," "no." (Default = "no").

pref63

TCP scanning technique : Choices:

Value	Description
"connect()"	connect()
"SYN scan"	(default) SYN scan
"FIN scan"	FIN scan
"Xmas Tree scan"	Xmas Tree scan

EXAMPLE

```
set ClientTraffic [::IxLoad new ixClientTraffic options...] \  
$ClientTraffic agentList.appendItem \  
$ClientTraffic agentList(0).pm.paramCategories. \  
attacksParams.cat6.config \  
-pref43 "yes"
```

SEE ALSO

Parameter Categories



Category 7 (cat7)

Category 7—Configures the list of category 7 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

Configures a Category 7 object.

SUBCOMMANDS

None.

OPTIONS

pref64

How to check if directories are writeable :: Choices: = "Trust the permissions (drwxrwx---)"
(default="Trust the permissions (drwxrwx---)")

EXAMPLE

```
set ClientTraffic [::IxLoad new ixClientTraffic options...] \
$ClientTraffic agentList.appendItem \
$ClientTraffic agentList(0).pm.paramCategories. \
attacksParams.cat7.config \
    -pref64      "default="Trust the permissions (drwxrwx---)"
```

SEE ALSO

Parameter Categories



Category 8 (cat8)

Category 8—Configures the list of category 8 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

Configures a Category 8 object.

SUBCOMMANDS

None.

OPTIONS

pref65

Test SSL based services. Choices = "Known SSL ports," "All." (Default = "All").

pref66

Services[file]:CA file. (Default = "{}").

pref67

Services[password]:PEM password. (Default = "{}").

pref68

Services[file]:SSL private key. (Default = "{}").

pref69

Services[file]:SSL certificate. (Default = "{}").

pref70

Services[entry]:Wrapped service read timeout. (Default = "2").

pref71

Services[entry]:Network read/write timeout. (Default = "5").

pref72

Services[entry]:Network connection timeout. (Default = "5").

pref73

Services[entry]:Number of connections done in parallel. (Default = "6").

EXAMPLE

```
set ClientTraffic [::IxLoad new ixClientTraffic options...] \  
$ClientTraffic agentList.appendItem \  
$ClientTraffic agentList(0).pm.paramCategories. \  
attacksParams.cat8.config \  
    -pref65    "default="All"
```

SEE ALSO

Parameter Categories



Category 9 (cat9)

Category 9—Configures the list of category 9 attack parameters.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.paramCategories.appendItem
```

DESCRIPTION

Configures a Category 9 object.

SUBCOMMANDS

None.

OPTIONS

pref74

SSH settings[password]:Passphrase for SSH key. (Default = {}).

pref75

SSH settings[file]:SSH private key to use. (Default = {}).

pref76

SSH settings[file]:SSH public key to use. (Default = {}).

pref77

SSH settings[password]:SSH password (unsafe!). (Default = {}).

pref78

SSH settings[entry]:SSH user name. (Default = "root").

EXAMPLE

```
set ClientTraffic [::IxLoad new ixClientTraffic options...] \
$ClientTraffic agentList.appendItem \
$ClientTraffic agentList(0).pm.paramCategories. \
attacksParams.cat9.config \
```

-pref74 "passphrase"

SEE ALSO

Parameter Categories



Targets

Targets—Configures the target of an attack.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.targets.appendItem
```

DESCRIPTION

A Targets object is added to the Nessus Client Agent object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`id`

This is an item of one of the following types: `Target Host`, `Target Range`, or `TarNetwork`. (Default = "{}").

EXAMPLE

```
$Activity_VulnerabilityClient1 agent.pm.targets.appendItem \  
-id "TargetHost" \  
-destinationHost "198.18.0.101"
```

SEE ALSO

Target Host, Target Range, Target Network.

Target Host

Target Host—Configures a single IP address to be the target of an attack.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.targets.appendItem
```

DESCRIPTION

A `Target Host` defines a single IP address that will be the subject of an attack. To specify a `Target Host` that will be attacked, you add it to the list of `Targets`. The list of `Targets` is added to the `Nessus Client Agent` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`destinationHost`

IP address of host to be attacked. (Default = "198.18.0.101").

EXAMPLE

```
$Activity_VulnerabilityClient1 agent.pm.targets.appendItem \  
-id "TargetHost" \  
-destinationHost "198.18.0.101"
```

SEE ALSO

`Targets`, `Target Range`, `Target Network`.



Target Range

Target Range—Configures a range of IP addresses to be the target of an attack.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.targets.appendItem
```

DESCRIPTION

A `Target Range` defines a range of IP addresses that will be the subject of an attack. To specify a `Target Range` that will be attacked, you add it to the list of `Targets`. The list of `Targets` is added to the `Nessus Client Agent` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`destinationHostsFrom`

First IP address of the range. (Default = "198.18.0.101").

`destinationHostsTo`

Last IP address of the range. (Default = "198.18.0.108").

EXAMPLE

```
$Activity_VulnerabilityClient1 agent.pm.targets.appendItem \  
-id "TargetRange" \  
-destinationHostsTo "198.18.0.108" \  
-destinationHostsFrom "198.18.0.101"
```

SEE ALSO

`Targets`, `Target Host`, `Target Network`.



Target Network

Target Network—Configures a network to be the target of an attack.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.targets.appendItem
```

DESCRIPTION

A `Target Network` defines all hosts on a particular network to be the subject of an attack. To specify a `Target Network` that will be attacked, you add it to the list of `Targets`. The list of `Targets` is added to the `Nessus Client Agent` object using the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`destinationNetwork`

IP address of the network. (Default = "198.18.0.0").

`destinationMask`

Subnet mask applied to `destinationNetwork` address to select hosts to be attacked. (Default = "255.255.0.0").

EXAMPLE

```
$Activity_VulnerabilityClient1 agent.pm.targets.appendItem \  
-id "TargetNetwork" \  
-destinationNetwork "198.18.0.0" \  
-destinationMask "255.255.0.0"
```

SEE ALSO

`Targets`, `Target Host`, `Target Range`.

Attacks Search

Attacks Search—Searches for an attack.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.attackSearch.config
```

DESCRIPTION

An Attacks Search object searches the attack descriptions for a text string and returns the results in a table. To add an Attack Search object, use the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`searchText`

Text string that will be searched for in the attack descriptions. (Default = "{}").

`attackSearchResults`
Table

Name of the table of that will hold the search results. This table contains `Attacks List` items. (Default = "{}").

EXAMPLE

```
$Activity_VulnerabilityClient1 agent.pm.attackSearch.config
    searchText                "Trojan"
    attackSearchResultsTable  "searchResults_trojans"
```

Attacks Select

Attacks Select—Selects attacks from a list of attack categories.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.attackSelection.config
```

DESCRIPTION

An `Attacks Select` object selects an attack from a list of attack categories. An `Attacks Select` object is part of an `Attacks Selection` object. To add an `Attack Select` object, use the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`categoriesSelectionTable`

List of categories of attacks. This table contains `Category List` items. If `dontSave = "1,"` the items in this list are not persistent. (Default = "{}").

`attackSelectionTable`

List of attacks available within an attack category. This table contains `Attacks List` items. If `dontSave = "1,"` the items in this list are not persistent. (Default = "{}").

`selectedAttacksTable`

Table of selected attacks. This table contains `Attacks List` items. (Default = "{}").

`attackDescription`

Description of the selected attack. (Default = "{}").

EXAMPLE

```
$Activity_VulnerabilityClient1 agent.pm.attackSelection.config
    categoriesSelectionTable        "Backdoors"
```



attackSelectionTable	"FTP Writeable Directories"
selectedAttacksTable	"FTP Writeable Directories"
attackDescription	"Write access to FTP dirs"

Attacks List

Attacks List—Configures a list of attacks.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1 agent.pm.attackSelection.config
$Activity_VulnerabilityClient1 agent.pm.attackSelection.selectedAttack
```

DESCRIPTION

An `Attacks List` is a list of attacks. Attacks lists are added to a Selected Attacks Table, and a Selected Attacks Table is part of an Attacks Selection object. To add an attack to a Attacks List, use the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`attackSelected`

If `true`, the attack is selected (enabled) and will be run. (Default = "0").

`attackID`

Nessus ID of the attack. For a list of attack IDs, see the Nessus Web site:

<http://www.nessus.org/plugins/index.php> <http://www.nessus.org/plugins/index.php>.

(Default = "{}").

`attackCategoryID`

Nessus ID of the category of the attack. (Default = "{}").

`attackName`

Name of the attack. (Default = "{}").

EXAMPLE

```
$Activity_VulnerabilityClient1 agent.pm.attackSelection.select\
-attackSelected 1 \
```



```
-attackID      14259 \  
-attackName    "Nmap (NASL wrapper)"
```

Parameters Data Table

Parameters Data Table—Table of Parameter Data List objects to be used in an attack.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
set Activity_VulnerabilityClient1(0).pm.ParametersDataTable.appendItem
options...
```

DESCRIPTION

A Parameters Data Table is a list of Parameter Data List objects. The parameters and their values in the Parameters Data Table can be used in an attack. To add a Parameters Data Table object, use the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`paramName`

Name of the parameter to be added to the list. (Default = "{}").

`paramValue`

Value for `paramName`. (Default = "{}").

EXAMPLE

```
set Activity_VulnerabilityClient1(0).pm.ParametersDataT\
  -paramName "SMTP Timeout"\
  -paramValue "30"
```



Parameters Data List

Parameters Data List—Parameters to be used in an attack, and a value for the parameter.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]

set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]

set
Activity_VulnerabilityClient1(0).pm.ParametersDataTable.ParametersDataLoptions...
```

DESCRIPTION

A Parameters Data List is an item in a Parameter Data Table object. To add a Parameters Data Table object, use the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`paramName`

Name of the parameter to be added to the list. (Default = "{}").

`paramValue`

Value for `paramName`. (Default = "{}").

EXAMPLE

```
set Activity_VulnerabilityClient1(0).pm.ParametersDataT\
-paramName "SMTP Timeout"\
-paramValue "30"
```


Category List

Category List—Configures a list of attack categories.

SYNOPSIS

```
set Traffic1_Network1 [::IxLoad new ixNetTraffic]
set Activity_VulnerabilityClient1 [$Traffic1_Network1 activityList.appendItem
options...]
$Activity_VulnerabilityClient1(0).pm.categorySelection. \
categoriesSelectionTable.appendItem options...
```

DESCRIPTION

A `Category List` is a list of categories of attacks; every attack belongs to a `cate`Category lists are added to a `Categories Selection Table`, and a `Categories Selection Table` is added to an `Attacks Selection` object. To add a category to a `Category List`, use the `appendItem` subcommand from the `ixConfigSequenceContainer` command.

SUBCOMMANDS

The options for this command are configured and read using the standard `config`, `cget`, and `getOptions` subcommands defined in the `ixConfig` command.

OPTIONS

`categorySelected`

If true, the category is selected. (Default = "0").

`categoryID`

Category ID. A value for this parameter must be one of the defined attack parameter categories, such as `cat1`, `cat2`, and so on. (Default = "{}").

`categoryName`

Name of the category specified for `categoryID`. (Default = "{}").

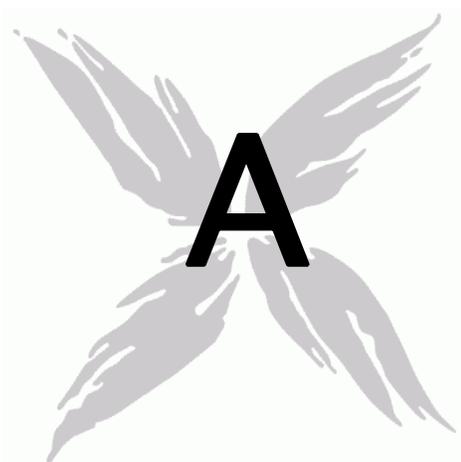
EXAMPLE

```
$clientTraffic agentList(0).pm.attackSelection. \
categoriesSelectionTable.appendItem \
    -categorySelected 1 \
    -categoryID "cat1" \
    -categoryName "Nessus Global"
```

Statistics

The table below describes the Vulnerability Attacks statistics.

Statistic	Description
Attacks Executed	Percentage of the configured attacks that the port was instructed to execute.
Attacks Total	Total number of vulnerability attacks run during the test.
Scans Executed	Percentage of scans run.
Scans Total	Percentage of the configured scans that the port was instructed to execute.



IP, TCP, Run State, and Curve Segment Statistics

Statistics in the results files and reports are averaged over all ports. If a statistic for an interval is missing, IxLoad interpolates it from the statistic immediately prior to it and the statistic after it.

For the per-Interface and TCP statistics, see [Per-Interface and TCP Statistics](#) (on page 2097).

For the Run State statistics, see [Run State Statistics](#) (on page 2104).

For the Curve Segment statistics, see [Curve Segment Statistics](#) (on page 2105).



Notes:

- IxLoad increments its TCP statistics at the time it causes a TCP packet to be generated. If a lower layer process in the TCP stack fails to transmit a packet, IxLoad does not update its statistics accordingly.
- If a process sends a SYN to the server port to which there is no corresponding listening socket, the Ixia port stack generates an RST, and the IxLoad will be unaware of the RST.

In this appendix:

Per-Interface and TCP Statistics	2097
Run State Statistics	2104
Curve Segment Statistics	2105

Per-Interface and TCP Statistics

The TCP statistics are displayed by most of IxLoad's protocols in their statistics views in StatViewer. In the first table below, *Caption* is the label shown in StatViewer for a statistic. *Name* is the name of the statistic as it appears in the Stats Catalog. To make queries from the API, you must use a statistic's name.

For the per-Interface and TCP statistics see the following:

Per-Interface Statistics (on page 2104)

TCP Statistics (on page 2097)

TCP Statistics

The following TCP statistics are available for some, but not all, protocols and are published on a per-protocol basis, not on a per-interface basis. At the time of printing, the following protocols do not support TCP statistics:

Radius	WAP	DDoS
TFTP	Trace File Replay	Vulnerability (Nessus)
DHCP	Packet Monitor	Application Test (Verify)

To confirm that TCP statistics are available, configure a test in the GUI, refresh the stat views, and then display the list of statistics in the view editor. If TCP statistics are listed, the protocol supports them.

FTP displays TCP statistics separately for the control and data connections. For the list of TCP statistics captions displayed in the FTP statistics views, see FTP Captions for TCP Statistics (on page 2100).

The table below describes the TCP statistics.

Statistic	Description
TCP Simulated Users	Number of simulated users.
Physical Rx Drops	Number of incoming packets dropped due to buffer overflow.
Physical Tx Drops	Number of outgoing packets dropped due to buffer overflow. Typically, this is caused by stopping a large test or configuring a Ramp Down time that is too short.
TCP Connection Lifetime	Amount of time elapsed between the time the first SYN in the TCP connection handshake is received and the last FIN or ACK sent or the TCP connection. This statistic measures the total lifetime of a connection through all three major stages of the connection: handshake duration + data transfer duration + close duration.



SYNs Sent (caption) TCP SYN Sent (name)	Number of connection requests (SYNs) sent. Only the initial SYN sent is counted in this statistic; retried SYNs are not included. Retried SYNs are counted in the TCP Retries statistic..
SYNs Received (caption) TCP SYN Received (name)	Number of connection requests (SYNs) received.
SYN/SYN-ACKs Received (caption) TCP SYN_SYN-ACK Received (name)	Number of connection requests (SYNs) and connection request acknowledgements (SYN and ACK flags set) received.
TCP SYN Failed	Number of connection requests (SYNs) sent for which a reset (RST) was received.
SYN-ACKs Sent (caption) TCP SYN-ACK Sent (name)	Number of connection request acknowledgements (SYN and ACK flags set) sent.
Connection Requests Failed (caption) TCP Connection Requests Failed (name)	Number of attempts to establish connections which did not result in connections being created.
TCP Connections Established	Number of connections established. Note: For a peer-to-peer protocol, this statistic counts the aggregate number of connections established, not the number established from the point of view of one side or the other.
FINs Sent (caption) TCP FIN Sent (name)	Number of connection termination requests (FINs) sent. Only the initial FIN sent is counted in this statistic; retried FINs are not included. Retried FINs are counted in the TCP Retries statistic.
FINs Received (caption) TCP FIN Received (name)	Number of connection termination requests (FINs) received.
FIN-ACKs Sent (caption) TCP FIN-ACK Sent (name)	Number of connection termination acknowledgements (FIN-ACK) sent. Only the initial FIN-ACK sent is counted in this statistic; retried FINs are not included. Retried FINs are counted in the TCP Retries statistic..
FIN-ACKs Received (caption) TCP FIN-ACK Received (name)	Number of connection termination acknowledgements (FIN-ACK) received.
Resets Sent (caption) TCP Resets Sent (name)	Number of Resets (RST) sent for any reason.
Resets Received (caption) TCP Resets Received (name)	Number of Resets (RST) received. Includes RSTs received as responses to SYNs and for any other reasons.
TCP Bytes Sent	Number of bytes sent in TCP packets.

TCP Bytes Received	Number of bytes received in TCP packets.
Retries (caption) TCP Retries (name)	Total number of retries attempted for all segments.
Timeouts (caption) TCP Timeouts (name)	Total number of timeouts that occurred for all segments.
Accept Queue Entries (caption) TCP Accept Queue Entries (name)	Number of entries in the listening socket's queue of connections awaiting acceptance.
Listen Queue Drops (caption) TCP Listen Queue Drops (name)	<p>Number of incoming SYN packets dropped from the socket listen queue.</p> <p>An incoming SYN packet is held in a listen queue while the host replies with a SYN+ACK and waits for the confirming ACK (the three-way handshake). A listen queue in this state is called a half-open connection.</p> <p>The QuickHTTP and QuickTCP activities do not use a Listen Queue, so for those activities, this statistic is always 0 (zero).</p>
TCP Connections in ESTABLISHED State	<p>Number of TCP connections in the ESTABLISHED state.</p> <p>A connection in the ESTABLISHED state can transfer data between the two ends in both directions.</p>
TCP Connections in SYN-SENT State	<p>Number of TCP connections in the SYN-SENT state.</p> <p>A client enters the SYN-SENT state after it has sent a SYN segment to the server to open a connection.</p>
TCP Connections in SYN-RECEIVED State	<p>Number of TCP connections in the SYN-RECEIVED state.</p> <p>A server enters the SYN-RECEIVED state after it receives a SYN from a client, requesting a connection. The server replies with a SYN+ACK segment.</p>
TCP Connections in FIN-WAIT-1 State	<p>Number of TCP connections in the FIN-WAIT-1 state.</p> <p>Sockets in the FIN-WAIT-1 state are closed and tearing down the connection.</p>
TCP Connections in FIN-WAIT-2 State	<p>Number of TCP connections in the FIN-WAIT-2 state.</p> <p>A connection in the FIN-WAIT-2 state has closed the local socket and is waiting for shutdown from the remote socket.</p>
TCP Connections in TIME-WAIT State	<p>Number of TCP connections in the TIME-WAIT state.</p> <p>A connection in the TIME-WAIT state has closed the local socket and is waiting for remote shutdown retransmission.</p>
TCP Connections in CLOSE State	<p>Number of TCP connections in the CLOSE state.</p> <p>A connection in the CLOSE state is closed.</p>
TCP Connections in CLOSE-WAIT State	<p>Number of TCP connections in the CLOSE-WAIT state.</p> <p>A connection in the CLOSE-WAIT state is waiting for the local socket to close after a remote shut down.</p>



TCP Connections in LAST-ACK State	Number of TCP connections in the LAST-ACK state. A connection in the LAST-ACK state is performing a remote shutdown; it will close the connection and wait for the acknowledgement.
TCP Connections in LISTENING State	Number of TCP connections in the LISTENING state. A socket in the LISTENING state is listening for an incoming connection.
TCP Connections in CLOSING State	Number of TCP connections in the CLOSING state. A socket in the CLOSING state is closed, has performed a remote shutdown, and is waiting for the acknowledgement.

FTP Captions for TCP Statistics

FTP displays TCP statistics separately for the control and data connections. For the list of TCP statistics captions displayed in the FTP statistics views, see the table below.

Statistic	Description
TCP Statistics for Control Connections	
Control SYNs Sent	Number of SYNs sent on control connections. See <i>SYNs Sent</i> in the TCP Statistics (on page 2097) table.
Control SYNs Received	Number of connection requests (SYNs) received on control connections. See <i>SYNs Received</i> in the TCP Statistics (on page 2097) table.
Control SYN/SYN-ACKs Received	Number of connection requests (SYNs) and connection request acknowledgements (SYN and ACK flags set) received on control connections. See <i>SYN/SYN-ACKs Received</i> in the TCP Statistics (on page 2097) table.
Control SYN-ACKs Sent	Number of connection request acknowledgements (SYN and ACK flags set) sent on control connections. See <i>SYN-ACKs Sent</i> in the TCP Statistics (on page 2097) table.
Control Connection Requests Failed	Number of attempts to establish control connections which did not result in connections being created. See <i>Connection Requests Failed</i> in the TCP Statistics (on page 2097) table.
Control FINs Sent	Number of connection termination requests (FINs) sent on control connections. See <i>FINs Sent</i> in the TCP Statistics (on page 2097) table.

Control FINs Received	Number of connection termination requests (FINs) received on control connections. See <i>FINs Received</i> in the TCP Statistics (on page 2097) table.
Control FIN-ACKs Sent	Number of connection termination acknowledgements (FIN-ACK) sent on control connections. See <i>FIN-ACKs Sent</i> in the TCP Statistics (on page 2097) table.
Control FIN-ACKs Received	Number of connection termination acknowledgements (FIN-ACK) received on control connections. See <i>FIN-ACKs Received</i> in the TCP Statistics (on page 2097) table.
Control Resets Sent	Number of Resets (RST) sent on control connections. See <i>Resets Sent</i> in the TCP Statistics (on page 2097) table.
Control Resets Received	Number of Resets (RST) received on control connections. See <i>Resets Received</i> in the TCP Statistics (on page 2097) table.
Control Retries	Total number of retries attempted on control connections for all segments. See <i>Retries</i> in the TCP Statistics (on page 2097) table.
Control Timeouts	Total number of timeouts that occurred on control connections for all segments. See <i>Timeouts</i> in the TCP Statistics (on page 2097) table.
Control Accept Queue Entries	Number of entries in the listening socket's queue of control connections awaiting acceptance. See <i>Accept Queue Entries</i> in the TCP Statistics (on page 2097) table.
Control Listen Queue Drops	Number of incoming SYN packets on control connections dropped from the socket listen queue. See <i>Listen Queue Entries</i> in the TCP Statistics (on page 2097) table.
TCP Statistics for Data Connections	
Data SYNs Sent	Number of SYNs sent on data connections. See <i>SYNs Sent</i> in the TCP Statistics (on page 2097) table.



Data SYNs Received	Number of connection requests (SYNs) received on data connections. See <i>SYNs Received</i> in the TCP Statistics (on page 2097) table.
Data SYN/SYN-ACKs Received	Number of connection requests (SYNs) and connection request acknowledgements (SYN and ACK flags set) received on data connections. See <i>SYN/SYN-ACKs Received</i> in the TCP Statistics (on page 2097) table.
Data SYN-ACKs Sent	Number of connection request acknowledgements (SYN and ACK flags set) sent on data connections. See <i>SYN-ACKs Sent</i> in the TCP Statistics (on page 2097) table.
Data Connection Requests Failed	Number of attempts to establish data connections which did not result in connections being created. See <i>Connection Requests Failed</i> in the TCP Statistics (on page 2097) table.
Data FINs Sent	Number of connection termination requests (FINs) sent on data connections. See <i>FINs Sent</i> in the TCP Statistics (on page 2097) table.
Data FINs Received	Number of connection termination requests (FINs) received on data connections. See <i>FINs Received</i> in the TCP Statistics (on page 2097) table.
Data FIN-ACKs Sent	Number of connection termination acknowledgements (FIN-ACK) sent on data connections. See <i>FIN-ACKs Sent</i> in the TCP Statistics (on page 2097) table.
Data FIN-ACKs Received	Number of connection termination acknowledgements (FIN-ACK) received on data connections. See <i>FIN-ACKs Received</i> in the TCP Statistics (on page 2097) table.
Data Resets Sent	Number of Resets (RST) sent on data connections. See <i>Resets Sent</i> in the TCP Statistics (on page 2097) table.
Data Resets Received	Number of Resets (RST) received on data connections. See <i>Resets Received</i> in the TCP Statistics (on page 2097) table.

Data Retries	Total number of retries attempted on data connections for all segments. See <i>Retries</i> in the TCP Statistics (on page 2097) table.
Data Timeouts	Total number of timeouts that occurred on data connections for all segments. See <i>Timeouts</i> in the TCP Statistics (on page 2097) table.
Data Accept Queue Entries	Number of entries in the listening socket's queue of data connections awaiting acceptance. See <i>Accept Queue Entries</i> in the TCP Statistics (on page 2097) table.
Data Listen Queue Drops	Number of incoming SYN packets on data connections dropped from the socket listen queue. See <i>Listen Queue Entries</i> in the TCP Statistics (on page 2097) table.

Advanced TCP Statistics

The following TCP statistics are available if you check the `ENABLE TCP ADVANCED STATS` option on the Test Options window. (see Test Options). These statistics will be present in the CSV when the option is enabled, and you can create a custom view in the Statistics Viewer with these statistics, if required.

The table below describes the Advanced TCP statistics.

Statistic	Description
TCP Lost Retransmits	Retransmissions for segments that have not been acknowledged.
TCP Fast Retransmits	Retransmissions that occurred before the retransmission timer expired because the other side sent three ACKs for the same segment.
TCP Forward Retransmits	Number of segments retransmitted even though there was no indication that they were actually lost. Retransmission stopped when either of the following occurs: <ul style="list-style-type: none"> The maximum time to wait for a remote response is reached. This timeout occurs when the total time of all retransmission intervals exceeds the maximum time to wait for a remote response. The number of retransmissions configured in maximum retransmissions per packet is reached. Forward Retransmits occur only on SACK-negotiated connections.
TCP Slow Start Retransmits	Retransmissions during the Slow Start phase.
TCP Local Advertisement Window	Window size advertised by the local side.



TCP Remote Advertisement Window	Window size advertised by the remote side.
TCP Syn-SynAck Time	Average time elapsed between the time the SYN was sent and the SYN-ACK was received.
TCP Syn-SynAck Time Squared	Variation in the TCP Syn-SynAck Time.

Per-Interface Statistics

The table below describes the per-Interface statistics.

Statistic	Description
Packets Sent	Number of IP packets sent.
Packets Received	Number of IP packets received.
Bytes Sent	Number of bytes sent in IP packets.
Bytes Received	Number of bytes received in IP packets.
Fragments Received	Number of IP packet fragments received.
Reassembly Timeouts	Number of fragmented IP packets that could not be reassembled within the timeout period.

Run State Statistics

The Run State statistics (see the table below) identify the phase that the test is in at a given time. The Run State statistics are stored in the CSV files, and can be retrieved using the IxLoad Tcl API (see the `runstatestats.tcl` sample script). There are different run states for Basic timelines and Advanced timelines.

Run State	Description
Basic Timeline	
ID	Idle
RU	Ramp Up
SU	Sustain
RD	Ramp Down
Advanced Timeline	
ID	Idle
LR	Linear segment
LU	Linear segment, Upwards
LD	Linear segment, Downwards
LI	Linear segment, Idle
ST	Steps segment, Upwards

SD	Steps segment, Downwards
BU	Bursts segment
BR	Bursts segment, Right skew
BL	Bursts segment, Left skew
PU	Pulses segment
PO	Poisson segment

Curve Segment Statistics

In a test that uses an Advanced timeline, the Curve Segment statistics identify the segment that is active at a given time. The Curve Segment statistics are stored in the CSV files, and can be retrieved using the IxLoad Tcl API.

Curve Segments are numbered starting with 0 (zero), and continuing through the *n*th segment. Segment 0 is the segment during which test initialization occurs; no traffic is sent during segment 0. For a Basic timeline, the Curve Segment is always 0.

Connection Latency Statistics

The table below describes the connection latency statistics.

Statistic	Description
Connection Latency 0 - 10 μ sec	Number of connections established after a delay of 0 to 10 microseconds.
Connection Latency 10 - 20 μ sec	Number of connections established after a delay of 10 to 20 microseconds.
Connection Latency 20 - 30 μ sec	Number of connections established after a delay of 20 to 30 microseconds.
Connection Latency 30 - 40 μ sec	Number of connections established after a delay of 30 to 40 microseconds.
Connection Latency 40 - 50 μ sec	Number of connections established after a delay of 40 to 50 microseconds.
Connection Latency 50 - 60 μ sec	Number of connections established after a delay of 50 to 60 microseconds.
Connection Latency 60 - 70 μ sec	Number of connections established after a delay of 60 to 70 microseconds.

Connection Latency 70 - 80 μ sec	Number of connections established after a delay of 70 to 80 microseconds.
Connection Latency 90 - 100 μ sec	Number of connections established after a delay of 90 to 100 microseconds.
Connection Latency 100 - 200 μ sec	Number of connections established after a delay of 100 to 200 microseconds.
Connection Latency 200 - 300 μ sec	Number of connections established after a delay of 200 to 300 microseconds.
Connection Latency 300 - 400 μ sec	Number of connections established after a delay of 300 to 400 microseconds.
Connection Latency 400 - 500 μ sec	Number of connections established after a delay of 400 to 500 microseconds.
Connection Latency 500 - 600 μ sec	Number of connections established after a delay of 500 to 600 microseconds.
Connection Latency 600 - 700 μ sec	Number of connections established after a delay of 600 to 700 microseconds.
Connection Latency 700 - 800 μ sec	Number of connections established after a delay of 700 to 800 microseconds.
Connection Latency 800 - 900 μ sec	Number of connections established after a delay of 800 to 900 microseconds.
Connection Latency 900 - 1000 μ sec	Number of connections established after a delay of 900 to 1000 microseconds.
Connection Latency > 1000 μ sec	Number of connections established after a delay of over 1000 microseconds.

IxServer Layer 2-3 Statistics

The table below describes the IxServer statistics displayed by IxLoad. IxServer statistics are displayed in separate views for client/peer and server ports. The views appear automatically in the top-level statistics views. Each view column (except for the Link State and Line Speed statistics) has a summary footer value showing the cumulative values for all ports in the view.



Notes:

- If you aggregate 1G and 10G ports, no ARP stats are displayed.
- If you aggregate 10G ports, only statistics from port 13 are displayed.

Statistic	Description
Bits Received Rate (Kb/s)	Rate at which bits are being received.
Bits Sent Rate (Kb/s)	Rate at which bits are being transmitted.
Bytes Received	Total number of bytes received.
Bytes Sent	Total number of bytes transmitted.
Frames Sent	Number of frames successfully transmitted. This statistic does not include frames retransmitted due to collisions.
Frames Sent Rate	Rate at which frames are being transmitted. This statistic does not include frames retransmitted due to collisions.
Line Speed	For Ethernet load modules, this statistic indicates the speed, in Mbps, negotiated on the link. For POS modules, this statistic indicates the POS level: OC-3, OC-12, or OC-48
Link State	Connectivity on the link. This statistic can be one of the following values: Up: A link is established with another device. Loopback: The port has loopback enabled. Down: There is no connection to another device.
Receive Arp Reply	Number of ARP replies received.
Receive Arp Request	Number of ARP requests received.
Transmit Arp Reply	Number of ARP replies sent.
Transmit Arp Request	Number of ARP requests sent.
Valid Frames Received	Number of valid frames received. A valid frame is a frame that is 64 bytes to 1518 bytes long, including the FCS but excluding the preamble and SFD. The frame length must be an integer number of octets. Only frames that have a valid FCS are counted by this statistic. VLAN-tagged frames that are larger than 1518 bytes but less than 1522 bytes are also included in this statistic.
Valid Frames Received Rate	Rate at which valid frames are being received. See <i>Valid Frames Received</i> (above) for a description of what constitutes a valid frame.

Index

:
:
IxLoad • 109

A

About this Guide • xix
Access Attribute Set List • 1484
AccessAttribSetList • 1502
Accounting Attribute Set List • 1485
AcctngAttribSetList • 1504
AcOptionsChoices • 585
activityList • 75, 177
Adding an IxVerify Activity • 810
advanced • 1655
Advanced • 1230, 1258
Advanced Options • 800, 991, 1004, 1234, 1294, 1667, 1679
Advanced Options tab • 1620
Advanced Settings • 1979
Advanced TCP Statistics • 2103
All Mail Messages • 1183, 1207
Alternative Capability • 1813
Alternative Capability List • 1811
Alternative Capability Value Set List • 1795
API Description • 68
API Objects • 1884
API Overview • 38, 816, 978, 1179, 1578, 1757
Application Replay Peer Agent • 782
Application Test • 808
AppReplay • 780
ArpProtocolFilter • 1400
Attachment • 1556, 1568
Attachment List • 1182, 1216
Attacks List • 2050, 2089
Attacks Params • 2049, 2059
Attacks Search • 2051, 2086
Attacks Select • 2050, 2087
Attribute • 1335, 1357
Attribute List • 1499
Attribute Type and Values • 1335, 1359
Audio Clips Pool • 821, 848, 885, 887, 918
Audio Settings • 1747, 1785, 1876, 1938, 2036
Authentication Extension Plugins • 378
Automatic Settings (CA) • 1850
Automatic Settings (GW) • 1841
availableTosList • 798, 1657

B

Background Reading • 24
Backward Compatibility • 29
Bandwidth Conditional Statistics • 1325

BandwidthUnitsChoices • 501
BandwidthUnitsInChoices • 502
BearerCapabilityChoices • 625
BearerTypeChoices • 626
Building an IxLoad Test • 46
Bulk MGCP • 814
Bulk MGCP Client Statistics • 869
Bulk MGCP Server Statistics • 874
Bulk MGCP Statistics • 869
Bulk SIP • 878
Bulk SIP Client Statistics • 950
Bulk SIP Server Statistics • 963
Bulk SIP Statistics • 950

C

Call Managers • 2023
Capability List • 1797
Category 1 (cat1) • 2060
Category 2 (cat2) • 2062
Category 3 (cat3) • 2064
Category 4 (cat4) • 2066
Category 5 (cat5) • 2068
Category 6 (cat6) • 2071
Category 7 (cat7) • 2075
Category 8 (cat8) • 2076
Category 9 (cat9) • 2078
Category List • 2050, 2094
CertFormatChoices • 473
CertInfo • 469
Channel View • 1271
Cloud Rules • 1985
Cloud Servers • 1981
Codec Settings • 1729, 1771, 1916, 2025
Codecs • 1733, 1772, 1867, 1919, 2028
Command List • 1334, 1341
Commands • 821, 850, 1230, 1243
Configuring a Network Address on the IxLoad Development Station • 26
Configuring a Permanent Route to Ixia Ports • 27
Configuring a Telnet Client • 1620
Configuring a Telnet Server • 1618
Configuring the IxAppOptions.ini file • 25
Connection Latency Statistics • 2105
Constants • 43
Content • 1515, 1542
Content of Messages • 882, 887, 902, 941
Control • 1335, 1353
Conventions • xx
Copyrights and Trademarks • 2
CountryChoices • 474
Creating a Telnet Test • 1617
Creating the Ranges • 247
Creating the Stack • 245
Curve Segment Statistics • 2105
Custom Activity Link Settings • 1799, 1859, 1960, 2043
Custom Endpoint Names • 821, 825, 854, 868
Custom Parameters • 1822, 1976, 2045
CustomFlowIP • 791
CustomFlowTCP • 786
CustomFlowUDP • 788
customPayloadObject • 1098
CustomPayloadObject • 1139

D

- Data Codecs • 1731, 1778, 1865, 1917, 2026
- DataLinkChoices • 589
- DataSegment • 1376
- DDoS • 976
- DDoS Client Agent • 978, 979
- DDoS Command • 978, 981
- DDoS Statistics • 987
- Debugging • 78
- Deprecated Commands • 30
- DHCP • 988
- DHCP Client Agent • 990, 994
- DHCP Client and Server • 318
- DHCP Client Example • 355, 356, 357, 358, 359, 360, 361, 362
- DHCP Command List • 990, 996
- DHCP Server Example • 372, 373, 374, 375, 376, 377
- DHCP Statistics • 1026
- Dhcp6DuidTypeChoices • 352
- Dhcp6IaTypeChoices • 353
- DHCPOptionSet • 345
- DHCPOptionSetList • 344
- DHCPOptionTLV • 348
- DHCPOptionTLVList • 347
- DHCPPlugin • 319
- DHCPPortGroupData • 327
- DHCPRange • 331
- DHCPRangeList • 330
- DHCPServerPlugin • 363
- DHCPServerRange • 366
- DHCPServerRangeList • 365
- DHCPServerSessionData • 371
- DHCPSessionData • 321
- Dial Plan • 1814, 1969, 2015
- DNPluginBase • 379
- DNPortGroupBase • 380
- DNRRangeBase • 384
- DNS • 1035
- DNS Client Advanced Options • 1037, 1045
- DNS Client Agent • 1036, 1040
- DNS Client Query • 1037, 1043
- DNS Client Statistics • 1060
- Dns Plugin Example • 709
- DNS Record • 818, 824, 835, 863
- DNS Server Advanced Options • 1038, 1055
- DNS Server Agent • 1037, 1047
- DNS Server Resource Record • 1057
- DNS Server Resource Records • 1039
- DNS Server Statistics • 1063
- DNS Server Zone Configuration • 1038, 1054
- DNS Server Zone Management • 1038, 1051
- DNS Statistics • 1060
- DNS Updates • 824, 861
- DNSessionBase • 382
- DnsHost • 707
- DnsHostList • 706
- DnsNameServer • 703
- DnsNameServerList • 702
- DnsPlugin • 700
- DnsSearch • 705
- DnsSearchList • 704
- Dot1x Plugin Example • 422, 423, 424, 425, 426, 427, 428

- Dot1xPlugin • 408
- Dot1xPortGroupData • 410
- Dot1xRange • 419
- Dot1xRangeList • 418
- Dot1xSessionData • 412
- DUT Commands • 74
- DutTestModeChoices • 417

E

- Edit Contact • 1934
- Editing the setup_simple.tcl script • 33, 35
- Effect of Options on DHCP Packet Size • 1034
- elpValidationOptions • 303
- eMacValidationOptions • 275
- Emulated Router Example • 295, 296, 297, 298, 299
- Emulated Router Plugin • 288
- EmulatedRouterRange • 291
- EmulatedRouterRangeList • 290
- Enable Filter • 1674
- Encaps1Choices • 586
- Encaps2Choices • 587, 588
- Endpoint Names • 819, 824, 837, 865
- Endpoints • 1843, 1852
- EthernetELMPlugin • 264
- EthernetFilter • 1396
- eValueType • 350
- Events • 1718
- Example 1 • 1150
- Example 2 • 1150
- Example 3 • 1150
- Example Program • 83
- Execution Settings • 1751, 1801, 1857, 1963, 2013
- ExpectedSystemTokenChoices • 440
- ExpectTcpFlagsChoices • 504

F

- FastInnerMethodChoices • 388
- FastProvisionModeChoices • 386
- FastStatelessResumeChoices • 389
- fileList • 1653
- Filter List • 1665, 1672
- Filter Plugin Example • 696
- FilterPlugin • 693
- Flow Definition • 784
- FlowDefinition • 1372
- Frame Jitter Conditional Statistics • 1326
- FramingCapabilityChoices • 630
- FTP • 1066
- FTP Captions for TCP Statistics • 2100
- FTP Client Action • 1068, 1076
- FTP Client Agent • 1067, 1070
- FTP Client Statistics • 1088
- FTP Server Agent • 1068, 1084
- FTP Server Statistics • 1090
- FTP Statistics • 1087

G

- GatewayIncrementModeChoices • 312
- Gateways • 1854
- General Settings • 881, 887, 897, 939
- General Tcl Conventions • 41
- Global Config • 1483, 1494, 1579, 1596

- Global Options • 1334, 1350
- Global Plugins • 692
- Global Statistics • 801
- Global Stream Statistics • 1299
- Global Video Quality Statistics • 1317
- GratArp Plugin Example • 699
- GratArpPlugin • 697
- GTP Plugin • 641
- GTPG Example • 681
- GTPGPDUFilter • 691
- GTPGPDUFilterList • 690
- GTPGPlugin • 680
- GTPGPluginPortGroupData • 688
- GTPGPluginSessionData • 685
- GTPGRange • 683
- GTPGRangeList • 682
- GTPS Plugin Example • 643
- GTPSActivity • 650
- GTPSPDUFilter • 657
- GTPSPDUFilterList • 656
- GTPSPDUFilterList2 • 658
- GTPSPlugin • 642
- GTPSPluginPortGroupData • 647
- GTPSPluginSessionData • 644
- GTPSRange • 660
- GTPSRangeList • 659
- GTPSUEMobilePathEntry • 679
- GTPSUEMobilePathEntryList • 678
- GTPSUERange • 672
- GTPSUERangeList • 671

H

- H248 Settings • 1726
- H248 TermGroups • 1703
- H323 Settings • 1804
- Header • 1230, 1263, 1556, 1567
- Header List • 1182, 1214
- HostAuthModeChoices • 422
- HTTP • 1092
- HTTP Client Action • 1096, 1113
- HTTP Client Agent • 1094, 1100
- HTTP Client Conditional Statistics • 1164
- HTTP Client per-URL Statistics • 1172
- HTTP Client Profile • 1110
- HTTP Client Statistics • 1158
- HTTP Server Agent • 1096, 1118
- HTTP Server Conditional View Statistics • 1154
- HTTP Server per-URL Statistics • 1157
- HTTP Server Statistics • 1151
- HTTP/QuickHTTP Client Statistics • 1459
- HTTP/QuickHTTP Server Statistics • 1463

I

- IcmpProtocol Filter • 1406
- IGMP and MLD Conditional Statistics • 1310
- IMAP • 1178
- IMAP Client Advanced Options • 1180, 1195
- IMAP Client Agent • 1180, 1185
- IMAP Client Commands • 1179
- IMAP Client Statistics • 1219
- IMAP Commands • 1180, 1188
- IMAP Server Advanced Options • 1181, 1202
- IMAP Server Agent • 1181, 1197

- IMAP Server Commands • 1180
- IMAP Server Config • 1181, 1204
- IMAP Server Statistics • 1222
- IMAP Statistics • 1219
- Impair Plugin • 482
- Impair Plugin Example • 507, 508, 509, 510, 511, 512, 513
- ImpairOrderChoices • 505
- ImpairProfile • 491
- ImpairProfileList • 490
- ImpairRange • 485
- ImpairRangeList • 484
- ImpairSessionData • 506
- InbuiltFlow • 1374
- Initial Overhead • 809
- Installing IxLoad Tcl • 35
- Installing IxOS Tcl • 35
- Internal Commands • 235
- Introduction • 23
- IP Address • 1023
- IP Plugin • 300
- IP Plugin Example • 313, 314, 315, 316, 317
- IP, TCP, Run State, and Curve Segment Statistics • 2096
- IPAddressRange • 1410
- IPProtocolFilter • 1398
- IpRangePortGroupData • 301
- IPSec Example • 557, 558, 559, 560, 561, 562, 563
- IPSec Plugin • 514
- IPSecCertificates • 553
- IPSecPortGroupData • 534
- IPSecRange • 517
- IPSecRangeList • 516
- IPSecSessionData • 537
- IPSecTunnelSetup • 540
- IpSessionData • 302
- IPTV • 1227
- IPTV / Video Client Statistics • 1299
- IPTV / Video Server Statistics • 1328
- IPTV / Video Statistics • 1299
- IPTV Conditional Statistics • 1314
- IPTV Global Statistics • 1311
- IPTV Mode Server and Client • 1227
- IPTV Options • 1228, 1232, 1272
- IPTV/ Video • 1225
- IpTypeChoices • 311
- IpV4V6Plugin • 304
- IpV4V6Range • 307
- IpV4V6RangeList • 306
- ixChassisChain • 71, 112
- ixClientNetwork • 199
- ixClientTraffic • 204
- ixClientTrafficNetworkMapping • 207
- ixConfig • 236
- ixConfigSequenceContainer • 238
- ixConfigSortedNamedItemList • 241
- ixCookieContent • 1124
- ixCookieContents • 1099
- ixCookieObject • 1127
- ixCustomPortMap • 115
- ixDns • 72
- ixDnsServerItem • 73
- ixDnsSuffixList • 73
- ixDut • 74, 184



- ixDutConfigSLB • 195
- ixDutConfigVip • 194
- ixDutConfigVirtual • 186
- ixDutNetworkRange • 189
- ixDutProtocolPortRange • 192
- ixEmulatedRouterIpAddressRange • 72, 73
- ixIntRange • 125
- IxLoad Settings Plugin Example • 730
- IxLoad Statistics Interpolation • 1176
- IxLoad Tcl API Commands • 106
- IxLoad Telnet Client and Server • 1614
- IxLoadSettingsPlugin • 728
- ixNetTraffic • 74, 172
- ixNetworkGroup • 70, 182
- ixNetworkRange • 217
- ixPort • 119
- ixRepository • 127
- ixResponseHeader • 1098, 1130
- ixScriptGen • 160
- IxServer Layer 2-3 Statistics • 2106
- ixServerNetwork • 221
- ixServerTraffic • 226
- ixServerTrafficNetworkMapping • 230
- ixStatCatalogItem • 131
- ixStatFilter • 133
- ixStatSpec • 135
- ixSubmapRange • 124
- ixSubscriberNetTraffic • 170
- ixTcpParameters • 73
- ixTest • 75, 137
- ixTestController • 76, 141
- ixTestControllerMonitor • 77, 148
- ixTimeline • 75, 163
- ixTrafficColumn • 75, 180
- ixTrafficFlow • 75, 179
- IxVerify VAAT in IxLoad • 809
- ixView • 75, 197
- ixWebPageObject • 1098

L

- L1EthernetPlugin • 259
- L2EthernetPlugin • 271
- L2EthernetPortGroupData • 273
- L2TP Plugin • 602
- L2tp Plugin Example • 634, 635, 636, 637, 638, 639, 640
- L2tpLnsIpAddress • 632
- L2tpLnsIpAddressList • 631
- L2tpPortGroupData • 604
- L2tpRange • 612
- L2tpRangeList • 611
- L2tpSessionData • 609
- LacToLNSMappingChoices • 628
- Layer 2 Example • 284, 285, 286, 287
- Layer 2 Protocols • 270
- LDAP • 1332
- LDAP Client Agent • 1334, 1337
- LDAP Client Commands • 1333
- LDAP Statistics • 1360
- License Server Parameters • 25
- Limitations • 1688, 1758, 1826, 1883, 1893, 1993
- List Programming in the Network Stack • 246
- Lists of Objects • 42

- LOD_IPSecAssociate • 551
- LOD_IPSecEapAkaTuple • 544
- LOD_IPSecEapAkaTupleList • 543
- LOD_IPSecEapSimTuple • 546
- LOD_IPSecEapSimTupleList • 545
- LOD_IPSecEapTuple • 543
- LOD_IPSecManualKeyingOpts • 547
- LOD_THY_IP Plugin Example • 313
- LOD_THY_TrackList • 1293
- Log File Locations • 79
- Log File Parameters • 79
- Logging Levels • 78
- LoopBeginCommand • 795
- LoopEndCommand • 796
- Low Level Parameters • 833, 860
- Low-Level Parameters • 818, 823

M

- MACAddressRange • 1408
- MacRange • 277
- MacRangeList • 276
- MacSessionData • 274
- Mail Message Instance List • 1182, 1183, 1209
- MailBoxItem • 1416, 1430
- MailMessage • 1555, 1563
- Mails • 1181, 1205
- Mandatory Objects to Complete a Script • 40
- Matching the TEARDOWN Statistics to Other Statistics • 1551
- Maximum Numbers of Scripts That Can Be Run • 67
- Media Settings • 819, 841, 884, 887, 910, 944
- MediumChoices • 263
- MGC Automatic • 1708
- MGCP CA Agent • 1845
- MGCP Client Agent • 817, 828
- MGCP Client API • 817
- MGCP GW Agent • 1830
- MGCP Server Agent • 823, 856
- MGCP Server API • 823
- MGCP Settings (CA) • 1847
- MGCP Settings (GW) • 1838
- MGW Automatic • 1705
- Mobile Subscribers Example • 771, 772, 773, 774, 775, 776, 777, 778, 779
- Mobile Subscribers Plugins • 730
- MobileSubscribersPlugin • 731
- MobileSubscribersPortGroupData • 751
- MobileSubscribersRange • 734
- MobileSubscribersRangeList • 733
- MobileSubscribersSessionData • 749
- Modification • 1335, 1355
- Modifying Older Scripts • 68
- Monitoring Status and Retrieving Results • 34, 37
- Multi Version Support • 41
- Multicast and VoD Conditional Statistics • 1309
- Multicast and VoD Global Statistics • 1308

N

- Nacl3 (EAPoUDP) Plugin Example • 462, 463, 464, 465, 466, 467, 468
- Nacl3Plugin • 429
- Nacl3PortGroupData • 443
- Nacl3Range • 436

- NacI3RangeList • 435
- NacI3SessionData • 431
- NacPosture • 449
- NacPostureList • 447
- NacPostureList2 • 448
- NacSequence • 446
- NacSequenceList • 445
- NacSettings • 441
- NacTLV • 453
- NacTlvAppCode • 461
- NacTlvAppCodeList • 460
- NacTlvAppType • 459
- NacTlvAppTypeList • 458
- NacTLVList • 451
- NacTLVList2 • 452
- NacTlvVendorId • 457
- NacTlvVendorIdList • 456
- NegotiationTypeChoices • 266
- Nessus (Vulnerability) • 2047
- Nessus Client Agent • 2049, 2052
- Nessus Client Commands • 2048
- Network- and Session-Specific Data • 248
- Network Commands • 69
- Network Setup • 25
- Network Stack API • 244
- Network Stack Configuration in 4.10 and Later Releases • 251
- Network Stack Configuration in pre-4.10 Releases • 250
- Network Stack Object Model • 253
- Network Stack Object Model - Base Classes beginning with 'A-DH' • 253
- Network Stack Object Model - Base Classes beginning with 'DN' • 253
- Network Stack Object Model - Base Classes beginning with 'I' • 253
- Network Stack Object Model - Base Classes Beginning with 'P' • 254
- Network Stack Object Model - Base Classes Beginning with 'R' • 255
- Network Stack Object Model - Base Classes Beginning with 'Se' • 256
- Network Stack Object Model - Base Classes Beginning with 'St' • 256
- Network Stack Object Model - Base Classes Beginning with 'St' (cont.) • 257

O

- Object Creation • 41
- Object Structure • 46
- Objectives • 781, 808, 817, 880, 978, 990, 1036, 1067, 1094, 1179, 1228, 1333, 1368, 1415, 1439, 1466, 1482, 1513, 1554, 1578, 1601, 1640, 1664, 2048
- Option • 992, 1010
- Option Choices • 1021
- Option Set • 992, 1017, 1579, 1591
- Option Set Manager • 993, 1020, 1579, 1593
- Options • 1665, 1670
- Other Settings • 1740, 1780, 1880, 1926, 2040
- Overview • 880, 990, 1036, 1067, 1094, 1227, 1333, 1368, 1383, 1415, 1439, 1466, 1482, 1513, 1554, 1640, 1664, 2048

P

- Packages • 1716
- Packet Jitter Conditional Statistics • 1327
- Packet Monitor • 1381
- Packet Monitor Server Agent • 1384
- Packet Transport Conditional Statistics • 1320
- PacketFilter • 1389
- PacketMonitor Server Agent • 1385
- PageObject • 1135
- Parameter Categories • 2049, 2054
- Parameters • 818, 826, 830, 858
- Parameters Data List • 2051, 2092
- Parameters Data Table • 2051, 2091
- Peer-to-Peer Application • 1366
- Peer-to-Peer Application Agent • 1368, 1370
- Peer-to-Peer Application Agent Commands • 1368
- Peer-to-peer Global Statistics • 1378
- Per-Interface and TCP Statistics • 2097
- Per-Interface Statistics • 2104
- Per-Stream Statistics • 1303
- Per-stream Video Quality Statistics • 1319
- Physical Layer • 258
- Physical Layer Example • 267, 268, 269
- PLAssociate • 633
- PLSessionDataBase • 570, 607
- PluginProperties • 1387
- POP3 • 1413
- POP3 Client Agent • 1415, 1418
- POP3 Client Statistics • 1433
- POP3 Server Agent • 1416, 1426
- POP3 Server Statistics • 1436
- POP3 Statistics • 1433
- Pop3Command • 1416, 1422
- PortRange • 1412
- PPPoX Plugin • 564
- Pppox Plugin Example • 596, 597, 598, 599, 600, 601
- PppoxAcMac • 594
- PppoxAcMacList • 593
- PppoxAcName • 591
- PppoxAcNameList • 590
- PppoxPortGroupData • 566
- PppoxRange • 575
- PppoxRangeList • 574
- PppoxSessionData • 572
- PresentationItem • 1515, 1540
- Profiles • 1231, 1268, 1714
- Properties • 1720
- ProtocolChoices • 725
- ProtocolFilter • 1395

Q

- Quick Start • 31
- QuickHTTP • 1438
- QuickHTTP Client Advanced • 1449
- QuickHTTP Client Advanced Options • 1440
- QuickHTTP Client Agent • 1439, 1442
- QuickHTTP Client Command • 1451
- QuickHTTP Client Commands • 1440
- QuickHTTP Client HTTP • 1446
- QuickHTTP Client HTTP Options • 1439
- QuickHTTP Server Advanced • 1458
- QuickHTTP Server Advanced Options • 1441



- QuickHTTP Server Agent • 1440, 1453
- QuickHTTP Server HTTP • 1456
- QuickHTTP Server HTTP Options • 1441
- QuickHTTP Statistics • 1459
- QuickTCP • 1465
- QuickTCP Client Advanced • 1474
- QuickTCP Client Agent • 1466, 1468
- QuickTCP Command • 1466, 1471
- QuickTCP Server Agent • 1467, 1476
- QuickTCP Server HTTP Options • 1467
- QuickTCP Server TCP • 1478
- QuickTCP Statistics • 1479

R

- Radius • 1480
- Radius Client Agent • 1483, 1486
- RADIUS Client Statistics • 1505
- Radius Command List • 1483, 1488
- Radius Plugin • 754
- RadiusMessage • 764
- RadiusMessageList • 763
- RadiusOptionSet • 761
- RadiusOptionSetList • 760
- RadiusOptionTLV • 767
- RadiusOptionTLVList • 766
- RadiusRange • 757
- RadiusRangeList • 756
- RadiusSessionData • 770
- realFileList • 1068, 1079
- Related Documentation • xxii
- Relay Agent • 991, 1007
- ResponseTypeChoices • 439
- RoleChoices • 569
- Route • 723
- RouteList • 722
- Routes Plugin Example • 727
- RoutesPlugin • 721
- RTP Settings • 1745, 1783, 1874, 1936, 2034
- RTSP • 1511
- RTSP Client Agent • 1513, 1517
- RTSP Client Statistics • 1546
- RTSP Conditional Statistics • 1315
- RTSP Server Agent • 1515, 1535
- RTSP Server Statistics • 1551
- RTSP Statistics • 1545
- RtspCommand • 1514, 1524
- RtspgetParamOptionList • 1533
- RtspHeader • 1514, 1529
- RtspHeaders • 1514, 1527
- RtspsetParamOptionList • 1531
- RuleData • 1987
- Rules • 883, 887, 905
- Run State Statistics • 2104
- Running an IxLoad Tcl Script • 65
- Running the sample scripts • 33, 36

S

- Sample Scripts Shipped with IxLoad • 79
- Scenario Settings • 1753, 1820, 1856, 1968, 2012
- Scenarios • 886, 888, 922, 945
- Scene Analysis Conditional Statistics • 1328
- SDP Settings • 1743, 1872
- SelectTcpFlagsChoices • 503

- Sequence Generators • 1626
- Server Network List • 1667, 1680
- Server Rules • 1983
- ServiceOptionsChoices • 584
- SessionAllocMethodChoices • 629
- Setting Ixia Chassis Base Addresses • 29
- Setting the Global Plug-ins List • 245
- Settings • 1887
- Signaling • 1231, 1264
- Signaling Settings • 1929
- Signals • 1722
- Simulated Endpoints • 1863
- Simulated MGC • 1699
- Simulated MGW • 1701
- Simultaneous Capability • 1803
- Simultaneous Capability List • 1818
- Simultaneous Capability Value Set List • 1809
- SIP Client Agent • 881, 889
- SIP Client Commands • 881
- SIP Server Agent • 886, 932
- SIP Server Commands • 886
- SIP Server List • 1888
- Skinny Settings • 2020
- SMTP • 1553
- SMTP Client Agent • 1555, 1557
- SMTP Client Statistics • 1573
- SMTP Server Agent • 1557, 1570
- SMTP Server Statistics • 1575
- SMTP Statistics • 1573
- SmtppCommand • 1555, 1561
- Specific Secrets • 1484, 1496
- SpeedChoices • 262
- S RTP Settings • 1957
- SSH • 1577
- SSH Client Agent • 1579, 1581
- SSH Client Statistics • 1597
- SSH Command List • 1579, 1583
- statCollectorUtils • 77, 129, 150, 233
- State Machine • 883, 887, 907, 942
- Stateless Peer • 1600
- Stateless Peer Advanced Options • 1603
- Stateless Peer Agent • 1602
- Stateless Peer Available TOS List • 1610
- Stateless Peer Commands • 1601
- Stateless Peer Overview • 1601
- Stateless Peer Payload Header List • 1608
- Stateless Peer Protocol Flows • 1604
- Statistics • 810, 1151, 1681, 1724, 2095
- Stats • 1232, 1274
- Step 1
 - Initial Overhead • 47
- Step 10
 - Prepare to Run the Test • 58
- Step 11
 - Start the Test • 64
- Step 2
 - Define the TrafficFlow • 49
- Step 3
 - Define the TrafficColumn • 49
- Step 4
 - Define the NetTraffic • 50
- Step 5
 - Define ixSubscriberNetTraffic • 52
- Step 6

- Define the NetworkGroup • 52
- Step 7
 - Define the NetworkGroup • 53
- Step 8
 - Define the NetworkRange • 56
- Step 9
 - Define the ixTimeline • 57
- Steps for Custom Traffic Mapping • 115
- Stopping a Test by Pressing Enter • 65
- Stream • 1234, 1287, 1516, 1544
- Strings and Numbers • 43
- Submap • 122
- Subobjects • 42
- Supported Ciphers • 1142

T

- T.30 Settings • 1950
- T.38 Settings • 1946
- Target Host • 2081
- Target Network • 2084
- Target Range • 2082
- Targets • 2049, 2080
- TargetTypeChoices • 726
- TCL API Internal Overview • 43
- Tcl API Structure • 39
- TCP Plugin Example • 720
- TCP Reset Statistics • 1175
- TCP Statistics • 2097
- TCPPlugin • 710
- TCPProtocol Filter • 1402
- Technical Support • xvii
- Telnet • 1612
- Telnet Client • 1616
- Telnet Client Loop Begin Command • 1626
- Telnet Client Loop End Command • 1626
- Telnet Client Telnet Session Command • 1622
- Telnet Client Think Command • 1625
- Telnet Client Advanced Options Tab • 1627
- Telnet Client Commands • 1621
- Telnet Client EXIT Command • 1625
- Telnet Client LOGIN Command • 1623
- Telnet Client OPEN Command • 1623
- Telnet Client PASSWORD Command • 1624
- Telnet Client SEND Command • 1624
- Telnet Client Statistics • 1629
- Telnet RFCs • 1614
- Telnet Server • 1614
- Telnet Server Statistics • 1634
- Telnet Statistics • 1628
- Telnet tab • 1621
- Telnet Tab • 1618
- Terminal Capability Set • 1817
- Test Operation Commands • 76
- Test Structure Commands • 75
- Testing the Development Station's Routing • 26
- TFTP • 1638
- TFTP Client Advanced • 1649
- TFTP Client Advanced Options • 1641
- TFTP Client Agent • 1640, 1642
- TFTP Client Statistics • 1658
- TFTP Command List • 1641, 1644
- TFTP Server Agent • 1651
- TFTP Server Statistics • 1660

- Think • 797
- Timer Settings • 1955
- TLS Cyphers • 1975
- TLS Settings • 1972
- Trace File Options • 1666, 1677
- Trace File Replay • 1662
- Trace File Replay Client Agent • 1664, 1668
- Trace File Replay Client Commands • 1664
- Trace File Replay Client Statistics • 1682
- Trace File Replay Server Agent • 1666, 1675
- Trace File Replay Server Commands • 1666
- Trace File Replay Server Statistics • 1684
- Traffic Commands • 74
- Transfer Address • 1966
- TunnelAuthenticationChoices • 627

U

- UdpProtocol Filter • 1404
- Unix / Linux • 66
- Unix Overview • 45
- Unix/Linux • 34
- Using a License Server • 24
- Using Auto-Generated Strings • 1218, 1432
- Using Sequence Generators in HTTP Client
 - Commands and Server Header Name=Value Fields • 1148
- Using System Variables • 1149
- Using The Sample Tcl Scripts • 32
- Using Variables in DHCP Fields • 1025
- Using Variables in SIP Fields • 948
- Using Your Own Web Pages In IxLoad • 1146

V

- Vendor List • 1484, 1498
- Video • 1227
- Video Client • 1228
- Video Client Agent • 1229, 1236
- Video Client API Structure • 1229
- Video Client Commands • 1229
- Video Client Data Conditional Statistics • 1306
- Video Config • 1298
- Video Description Conditional Statistics • 1321
- Video Frame Conditional Statistics • 1323
- Video Perceptual Quality Conditional Statistics • 1321
- Video Properties • 1233, 1283
- Video Quality Statistics • 1317
- Video Server • 1227
- Video Server Agent • 1233, 1277
- Video Server Commands • 1232
- Video Settings • 885, 920, 1789, 1942
- VlanIdRange • 280
- VlanIdRangeList • 279
- VoIP H.248 Peer • 1686
- VoIP H.323 Peer • 1755
- VoIP H248 MGC/MGW Peer API Objects • 1690
- VoIP H248 Peer Agent • 1693
- VoIP H248 Peer API Commands • 1689
- VoIP H248 TermGroup Peer API Objects • 1691
- VoIP H323 Peer Agent • 1761
- VoIP H323 Peer API Commands • 1759
- VoIP H323 Peer API Objects • 1759
- VoIP MGCP • 1824
- VoIP MGCP CA/MGW Peer API Objects • 1828



VoIP MGCP Endpoint Peer API Objects • 1829
VoIP MGCP Peer API Commands • 1827
VoIP SIP Cloud • 1882
VoIP SIP Cloud API Commands • 1884
VoIP SIP Peer • 1890
VoIP SIP Peer Agent • 1897
VoIP SIP Peer API Commands • 1894
VoIP SIP Peer API Objects • 1894
VoIP Skinny API Objects • 1994
VoIP Skinny Peer • 1991
VoIP Skinny Peer Agent • 1996
VoIP Skinny Peer API Commands • 1994
VoIPSIP Cloud Agent • 1885

W

WebAuth Example • 401, 402, 403, 404, 405, 406, 407
WebAuthPlugin • 390
WebAuthPortGroupData • 391
WebAuthRange • 400
WebAuthRangeList • 399
WebAuthSessionData • 393
Windows • 31, 66
Windows Overview • 43