

BIG-IP® Local Traffic Manager™: Concepts

Version 11.5.1



Table of Contents

Legal Notices and Acknowledgments.....	13
Legal Notices.....	13
Acknowledgments.....	14
 Introduction to Local Traffic Manager.....	 19
What is BIG-IP Local Traffic Manager?.....	19
Timeout settings for connections and sessions.....	19
Connection reaping.....	19
Idle timeout options.....	20
Idle timeout settings that affect connection reaping.....	20
Other timeout settings.....	20
Idle timeout settings that do not affect connection reaping.....	21
About the network map.....	21
The filtering mechanism.....	21
Object summary.....	22
The network map display.....	22
 Virtual Servers.....	 25
Introduction to virtual servers.....	25
About virtual server settings.....	25
Types of virtual servers.....	25
About source and destination addresses.....	27
About destination service ports.....	29
Status notification to virtual addresses.....	29
About profiles for traffic types.....	30
About VLAN and tunnel assignment.....	30
About source address translation (SNATs).....	30
About bandwidth control.....	30
About traffic classes.....	31
About connection and rate limits.....	31
About connection and persistence mirroring.....	31
About destination address and port translation.....	31
About source port preservation.....	32
About clone pools.....	32
About auto last hop.....	32
About NAT64.....	33
Virtual server resources.....	33
About virtual address settings.....	33
About automatic deletion.....	33

About traffic groups.....	33
About route advertisement.....	34
About ARP and virtual addresses.....	34
About ICMP echo responses.....	34
Virtual server and virtual address status.....	35
Clustered multiprocessing.....	35
Local Traffic Policies.....	37
About local traffic policy matching.....	37
About strategies for local traffic policy matching.....	37
Local traffic policy matching Requires profile settings.....	38
Local traffic policy matching Controls settings.....	38
About rules for local traffic policy matching.....	38
About conditions for local traffic policy matching.....	38
Local traffic policy matching Conditions operands	39
About actions for a local traffic policy rule	41
Local traffic policy matching Actions operands	41
Nodes.....	45
About nodes.....	45
About the node address setting.....	45
About health monitor association.....	45
About monitors and automatic node creation.....	46
About monitors and explicit node creation.....	46
About monitor removal.....	46
About node availability.....	46
About the ratio weight setting.....	47
About the connection rate limit setting.....	47
About node state.....	47
About node status.....	47
Pools.....	49
Introduction to pools.....	49
About load balancing pools.....	49
Pool features.....	49
About health monitor association.....	50
Pool member availability.....	50
Secure network address translations (SNATs) and network address translations (NATs).....	51
Action when a service becomes unavailable.....	51
Slow ramp time.....	51
Type of Service (ToS) level.....	51
Quality of Service (QoS) level.....	52
Number of reselect tries.....	52

About TCP request queue.....	52
About load balancing methods.....	53
About priority-based member activation.....	55
Pool member features.....	56
About pool member state.....	58
Pool and pool member status.....	58
Profiles.....	59
Introduction to profiles.....	59
Profile types.....	59
Default profiles.....	59
Custom and parent profiles.....	60
The default profile as the parent profile.....	61
The custom profile as the parent profile.....	61
Profiles and virtual servers.....	61
HTTP Profiles.....	63
Introduction to HTTP profiles.....	63
General HTTP properties.....	63
Proxy mode.....	63
Parent profile.....	64
HTTP settings.....	64
Basic Auth Realm.....	64
Fallback host.....	64
Fallback error codes.....	65
Headers in HTTP requests.....	65
Content erasure from HTTP headers.....	65
Headers in an HTTP response.....	65
Response chunking.....	65
OneConnect transformations.....	66
Rewrites of HTTP redirections.....	67
Cookie encryption and decryption.....	68
X-Forwarded-For header insertion.....	68
Maximum columns for linear white space.....	68
Linear white space separators.....	68
Maximum number of requests.....	68
Proxy Via headers.....	68
X-Forwarded-For header acceptance.....	70
Alternate X-Forwarded-For headers.....	70
Server agent name.....	70
Enforcement settings.....	70
Allow truncated redirects.....	71
Maximum header size.....	71
Oversize client headers.....	71

Oversize server headers.....	71
Maximum header count.....	71
Excess client headers.....	71
Excess server headers.....	72
Support for pipelining.....	72
Unknown methods.....	72
Explicit proxy settings.....	72
DNS Resolver.....	72
Route Domain.....	72
Tunnel Name.....	73
Host Names.....	73
Default Connect Handling.....	73
Connection Failed Message.....	73
DNS Lookup Failed Message.....	73
Bad Request Message.....	74
Bad Response Message.....	74
sFlow settings.....	74
Polling intervals.....	74
Sampling rates.....	74
About HTTP compression profiles.....	74
HTTP Compression profile options.....	75
URI compression.....	75
Content compression.....	75
Preferred compression methods.....	76
Minimum content length for compression.....	76
Compression buffer size.....	76
About the Vary header.....	76
Compression for HTTP/1.0 requests.....	77
About the Accept-Encoding header.....	77
Browser workarounds.....	77
About Web Acceleration profiles.....	78
Web Acceleration profile settings.....	78
Other Application-Layer Profiles.....	81
Overview of other application-layer profiles.....	81
About HTTP compression profiles.....	81
HTTP Compression profile options.....	82
About Web Acceleration profiles.....	82
Web Acceleration profile settings.....	82
Web Acceleration Profile statistics description.....	83
FTP profiles.....	84
DNS profiles.....	85
RTSP profiles.....	85
ICAP profiles.....	86

Request Adapt and Response Adapt profiles.....	86
Diameter profiles.....	86
RADIUS profiles.....	87
SIP profiles.....	87
SMTP profiles.....	88
SMTPS profiles.....	89
About iSession profiles.....	89
Screen capture showing compression settings.....	89
Rewrite profiles.....	90
About URI translation.....	91
Rules for matching requests to URI rules.....	91
About URI Rules.....	92
About Set-Cookie header translation.....	92
XML profiles.....	92
SPDY profiles.....	93
SPDY profile settings.....	94
SOCKS profiles.....	95
FIX profiles.....	96
About FIX profile tag substitution.....	96
About steering traffic using the FIX profile.....	96
About validating FIX messages.....	96
About using SSL encryption for FIX messages.....	98
About logging FIX messages.....	98
Video Quality of Experience profiles.....	98
About the video Quality of Experience profile.....	98
About mean opinion score.....	99
Content Profiles.....	101
Introduction to HTML content modification.....	101
About content selection types.....	101
Types of HTML rules.....	101
Sample HTML rules configuration.....	102
Session Persistence Profiles.....	103
Introduction to session persistence profiles.....	103
Persistence profile types.....	103
Session persistence and iRules.....	104
The OneConnect profile and session persistence.....	104
HTTP parsing with and without a OneConnect profile.....	105
Criteria for session persistence.....	105
The Match Across Services setting.....	105
The Match Across Virtual Servers setting.....	106
The Match Across Pools setting.....	106
Cookie persistence.....	107

HTTP Cookie Insert method.....	107
HTTP Cookie Rewrite method.....	107
HTTP Cookie Passive method.....	108
Cookie hash method.....	108
IPv4 IP address encoding.....	108
Port encoding.....	109
Destination address affinity persistence.....	110
Hash persistence.....	110
Microsoft Remote Desktop Protocol persistence.....	111
Benefits of Microsoft Remote Desktop Protocol persistence.....	111
Microsoft Remote Desktop Protocol persistence server platform issues.....	111
SIP persistence.....	112
Source address affinity persistence.....	112
SSL persistence.....	112
Universal persistence.....	113
Protocol Profiles.....	115
About protocol profiles.....	115
The Fast L4 profile type.....	115
PVA hardware acceleration.....	115
The Server Sack, Server Timestamp, and Receive Window settings.....	116
The Fast HTTP profile type.....	116
About TCP profiles.....	117
About tcp-lan-optimized profile settings.....	118
About tcp-wan-optimized profile settings.....	118
About tcp-mobile-optimized profile settings.....	118
About mptcp-mobile-optimized profile settings.....	119
The UDP profile type.....	120
The SCTP profile type.....	120
The Any IP profile type.....	120
Remote Server Authentication Profiles.....	123
Introduction to authentication profiles.....	123
BIG-IP system authentication modules.....	123
The LDAP authentication module.....	124
The RADIUS authentication module.....	124
The TACACS+ authentication module.....	124
The SSL client certificate LDAP authentication module.....	124
Search results and corresponding authorization status.....	125
SSL client certificate authorization.....	125
SSL certificates for LDAP authorization.....	125
Groups and roles for LDAP authorization.....	126
The SSL OCSP authentication module.....	126
The CRLDP authentication module.....	128

The Kerberos Delegation authentication module.....	128
Other Profiles.....	129
Introduction to other profiles.....	129
About OneConnect profiles.....	129
OneConnect and HTTP profiles.....	130
OneConnect and NTLM profiles.....	131
OneConnect and SNATs.....	131
About NTLM profiles.....	131
The Statistics profile type.....	132
The Stream profile type.....	133
The Request Logging profile type.....	133
The DNS Logging profile type.....	133
Health and Performance Monitoring.....	135
Introduction to health and performance monitoring.....	135
Comparison of monitoring methods.....	135
About monitor settings.....	136
Overview of monitor implementation.....	136
Monitor destinations.....	138
Transparent and Reverse modes.....	139
Monitors that contain the Transparent or Reverse settings.....	140
The Manual Resume feature.....	140
Resumption of connections.....	140
The Time Until Up feature.....	141
Dynamic ratio load balancing.....	141
Monitor plug-ins and corresponding monitor templates.....	141
Monitor association with pools and nodes.....	142
Monitor instances.....	142
NATs.....	143
Introduction to NATs.....	143
NATs for inbound connections.....	143
NATs for outbound connections.....	145
SNATs.....	147
About source address translation (SNATs).....	147
Comparison of NATs and SNATs.....	147
SNATs for client-initiated (inbound) connections.....	148
SNATs for server-initiated (outbound) connections.....	149
SNAT implementation.....	150
SNAT types.....	151
About translation addresses.....	151

Original IP addresses.....	152
VLAN traffic.....	152
Traffic Classes.....	153
About traffic classes.....	153
iRules.....	155
Introduction to iRules.....	155
Basic iRule elements.....	155
Event declarations.....	156
Operators.....	156
iRule commands.....	156
The pool command.....	157
The node command.....	157
Commands that select a pool of cache servers.....	157
The HTTP::redirect command.....	158
The snat and snatpool commands.....	158
iRules and administrative partitions.....	159
iRule evaluation.....	159
Event types.....	159
iRule context.....	159
iRules assignment to a virtual server.....	160
iRule command types.....	160
iRules and profiles.....	161
The profile command.....	161
Commands that override profile settings.....	162
Data groups.....	162
About the class match command.....	162
Storage options.....	163
iFiles.....	163
iFile commands.....	164
Dynamic Ratio Load Balancing.....	165
Introduction to dynamic ratio load balancing.....	165
Monitor plug-ins and corresponding monitor templates.....	165
Overview of implementing a RealServer monitor.....	165
Installing the monitor plug-in on a RealSystem server system (Windows version).....	166
Installing and compiling a Linux or UNIX RealSystem server monitor plug-in....	166
Overview of implementing a WMI monitor.....	166
IIS version support for the data gathering agent files.....	167
Installing the Data Gathering Agent f5isapi.dll or f5isapi64.dll on an IIS 5.0 server.....	167

Installing the Data Gathering Agent f5isapi.dll or f5isapi64.dll on an IIS 6.0 server.....	168
Installing the Data Gathering Agent F5.IsHandler.dll on an IIS 6.0 server.....	168
Installing the Data Gathering Agent F5.IsHandler.dll on an IIS 7.0 server.....	170
Installing the Data Gathering Agent F5.IsHandler.dll on an IIS 7.5 server.....	171

Legal Notices and Acknowledgments

Legal Notices

Publication Date

This document was published on February 13, 2017.

Publication Number

MAN-0377-07

Copyright

Copyright © 2013-2017, F5 Networks, Inc. All rights reserved.

F5 Networks, Inc. (F5) believes the information it furnishes to be accurate and reliable. However, F5 assumes no responsibility for the use of this information, nor any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent, copyright, or other intellectual property right of F5 except as specifically described by applicable user licenses. F5 reserves the right to change specifications at any time without notice.

Trademarks

AAM, Access Policy Manager, Advanced Client Authentication, Advanced Firewall Manager, Advanced Routing, AFM, APM, Application Acceleration Manager, Application Security Manager, ARX, AskF5, ASM, BIG-IP, BIG-IQ, Cloud Extender, CloudFucious, Cloud Manager, Clustered Multiprocessing, CMP, COHESION, Data Manager, DevCentral, DevCentral [DESIGN], DNS Express, DSC, DSI, Edge Client, Edge Gateway, Edge Portal, ELEVATE, EM, Enterprise Manager, ENGAGE, F5, F5 [DESIGN], F5 Certified [DESIGN], F5 Networks, F5 SalesXchange [DESIGN], F5 Synthesis, f5 Synthesis, F5 Synthesis [DESIGN], F5 TechXchange [DESIGN], Fast Application Proxy, Fast Cache, FirePass, Global Traffic Manager, GTM, GUARDIAN, iApps, IBR, Intelligent Browser Referencing, Intelligent Compression, IPv6 Gateway, iControl, iHealth, iQuery, iRules, iRules OnDemand, iSession, L7 Rate Shaping, LC, Link Controller, Local Traffic Manager, LTM, LineRate, LineRate Systems [DESIGN], LROS, LTM, Message Security Manager, MSM, OneConnect, Packet Velocity, PEM, Policy Enforcement Manager, Protocol Security Manager, PSM, Real Traffic Policy Builder, SalesXchange, ScaleN, Signalling Delivery Controller, SDC, SSL Acceleration, software designed applications services, SDAC (except in Japan), StrongBox, SuperVIP, SYN Check, TCP Express, TDR, TechXchange, TMOS, TotALL, Traffic Management Operating System, Trafix Systems, Trafix Systems (DESIGN), Transparent Data Reduction, UNITY, VAULT, vCMP, VE F5 [DESIGN], Versafe, Versafe [DESIGN], VIPRION, Virtual Clustered Multiprocessing, WebSafe, and ZoneRunner, are trademarks or service marks of F5 Networks, Inc., in the U.S. and other countries, and may not be used without F5's express written consent.

All other product and company names herein may be trademarks of their respective owners.

Patents

This product may be protected by one or more patents indicated at:

<http://www.f5.com/about/guidelines-policies/patents>

Export Regulation Notice

This product may include cryptographic software. Under the Export Administration Act, the United States government may consider it a criminal offense to export this product from the United States.

RF Interference Warning

This is a Class A product. In a domestic environment this product may cause radio interference, in which case the user may be required to take adequate measures.

FCC Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This unit generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Any modifications to this device, unless expressly approved by the manufacturer, can void the user's authority to operate this equipment under part 15 of the FCC rules.

Canadian Regulatory Compliance

This Class A digital apparatus complies with Canadian ICES-003.

Standards Compliance

This product conforms to the IEC, European Union, ANSI/UL and Canadian CSA standards applicable to Information Technology products at the time of manufacture.

Acknowledgments

This product includes software developed by Bill Paul.

This product includes software developed by Jonathan Stone.

This product includes software developed by Manuel Bouyer.

This product includes software developed by Paul Richards.

This product includes software developed by the NetBSD Foundation, Inc. and its contributors.

This product includes software developed by the Politecnico di Torino, and its contributors.

This product includes software developed by the Swedish Institute of Computer Science and its contributors.

This product includes software developed by the University of California, Berkeley and its contributors.

This product includes software developed by the Computer Systems Engineering Group at the Lawrence Berkeley Laboratory.

This product includes software developed by Christopher G. Demetriou for the NetBSD Project.

This product includes software developed by Adam Glass.

This product includes software developed by Christian E. Hopps.

This product includes software developed by Dean Huxley.

This product includes software developed by John Kohl.

This product includes software developed by Paul Kranenburg.

This product includes software developed by Terrence R. Lambert.

This product includes software developed by Philip A. Nelson.

This product includes software developed by Herb Peyerl.

This product includes software developed by Jochen Pohl for the NetBSD Project.

This product includes software developed by Chris Provenzano.

This product includes software developed by Theo de Raadt.

This product includes software developed by David Muir Sharnoff.

This product includes software developed by SigmaSoft, Th. Lockert.

This product includes software developed for the NetBSD Project by Jason R. Thorpe.

This product includes software developed by Jason R. Thorpe for And Communications, <http://www.and.com>.

This product includes software developed for the NetBSD Project by Frank Van der Linden.

This product includes software developed for the NetBSD Project by John M. Vinopal.

This product includes software developed by Christos Zoulas.

This product includes software developed by the University of Vermont and State Agricultural College and Garrett A. Wollman.

This product includes software developed by Balazs Scheidler (bazsi@balabit.hu), which is protected under the GNU Public License.

This product includes software developed by Niels Mueller (nisse@lysator.liu.se), which is protected under the GNU Public License.

In the following statement, *This software* refers to the Mitsumi CD-ROM driver: This software was developed by Holger Veit and Brian Moore for use with 386BSD and similar operating systems. *Similar operating systems* includes mainly non-profit oriented systems for research and education, including but not restricted to NetBSD, FreeBSD, Mach (by CMU).

This product includes software developed by the Apache Group for use in the Apache HTTP server project (<http://www.apache.org/>).

This product includes software licensed from Richard H. Porter under the GNU Library General Public License (© 1998, Red Hat Software), www.gnu.org/copyleft/lgpl.html.

This product includes the standard version of Perl software licensed under the Perl Artistic License (© 1997, 1998 Tom Christiansen and Nathan Torkington). All rights reserved. You may find the most current standard version of Perl at <http://www.perl.com>.

This product includes software developed by Jared Minch.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product contains software based on oprofile, which is protected under the GNU Public License.

This product includes RRDtool software developed by Tobi Oetiker (<http://www.rrdtool.com/index.html>) and licensed under the GNU General Public License.

This product contains software licensed from Dr. Brian Gladman under the GNU General Public License (GPL).

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes Hypersonic SQL.

This product contains software developed by the Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, and others.

This product includes software developed by the Internet Software Consortium.

This product includes software developed by Nominum, Inc. (<http://www.nominum.com>).

This product contains software developed by Broadcom Corporation, which is protected under the GNU Public License.

This product contains software developed by MaxMind LLC, and is protected under the GNU Lesser General Public License, as published by the Free Software Foundation.

This product includes unbound software from NLnetLabs. Copyright ©2007. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of NLnetLabs nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes Intel QuickAssist kernel module, library, and headers software licensed under the GNU General Public License (GPL).

This product includes software licensed from Gerald Combs (gerald@wireshark.org) under the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or any later version. Copyright ©1998 Gerald Combs.

This product includes software developed by Thomas Williams and Colin Kelley. Copyright ©1986 - 1993, 1998, 2004, 2007

Permission to use, copy, and distribute this software and its documentation for any purpose with or without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Permission to modify the software is granted, but not the right to distribute the complete modified source code. Modifications are to be distributed as patches to the released version. Permission to distribute binaries produced by compiling modified sources is granted, provided you

1. distribute the corresponding source modifications from the released version in the form of a patch file along with the binaries,
2. add special version identification to distinguish your version in addition to the base release version number,
3. provide your name and address as the primary contact for the support of your modified version, and
4. retain our contact information in regard to use of the base software.

Permission to distribute the released version of the source code along with corresponding source modifications in the form of a patch file is granted with same provisions 2 through 4 for binary distributions. This software is provided "as is" without express or implied warranty to the extent permitted by applicable law.

This product includes software developed by Brian Gladman, Worcester, UK Copyright ©1998-2010. All rights reserved. The redistribution and use of this software (with or without changes) is allowed without the payment of fees or royalties provided that:

- source code distributions include the above copyright notice, this list of conditions and the following disclaimer;
- binary distributions include the above copyright notice, this list of conditions and the following disclaimer in their documentation.

This software is provided 'as is' with no explicit or implied warranties in respect of its operation, including, but not limited to, correctness and fitness for purpose.

This product includes software developed by the Computer Systems Engineering Group at Lawrence Berkeley Laboratory. Copyright ©1990-1994 Regents of the University of California. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: This product includes software developed by the Computer Systems Engineering Group at Lawrence Berkeley Laboratory.
4. Neither the name of the University nor of the Laboratory may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes software developed by Sony Computer Science Laboratories Inc. Copyright © 1997-2003 Sony Computer Science Laboratories Inc. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY SONY CSL AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL SONY CSL OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY

OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains software developed by Google, Inc. Copyright ©2011 Google, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This product includes software developed by Ian Gulliver ©2006, which is protected under the GNU General Public License, as published by the Free Software Foundation.

This product includes software developed by Jeremy Ashkenas and DocumentCloud, and distributed under the MIT license. Copyright © 2010-2013 Jeremy Ashkenas, DocumentCloud.

This product includes gson software, distributed under the Apache License version 2.0. Copyright © 2008-2011 Google Inc.

This product includes ec2-tools software, copyright © 2008, Amazon Web Services, and licensed under the Amazon Software License. A copy of the License is located at <http://aws.amazon.com/asl/>.

This product includes crypto.js software, copyright © 2009-2013, Jeff Mott, and distributed under the BSD New license.

Introduction to Local Traffic Manager

What is BIG-IP Local Traffic Manager?

BIG-IP® Local Traffic Manager™ controls network traffic that comes into or goes out of a local area network (LAN), including an intranet.

A commonly-used feature of Local Traffic Manager is its ability to intercept and redirect incoming network traffic, for the purpose of intelligently tuning the load on network servers. However, tuning server load is not the only type of local traffic management.

Local Traffic Manager includes a variety of features that perform functions such as inspecting and transforming header and content data, managing SSL certificate-based authentication, and compressing HTTP responses. In so doing, the BIG-IP system not only directs traffic to the appropriate server resource, but also enhances network security and frees up server resources by performing tasks that web servers typically perform.

Note: *BIG-IP Local Traffic Manager is one of several products that constitute the BIG-IP product family. All products in the BIG-IP product family run on the powerful Traffic Management Operating System, commonly referred to as TMOS®.*

Timeout settings for connections and sessions

Local Traffic Manager has a number of time-outs that can be set to promote active connection management. The system manages each connection explicitly by keeping track of a connection in the connection table while the connection is still active. The connection table contains state information about client-side and server-side connections, as well as the relationships between them.

Each connection in the connection table consumes system resources to maintain the table entry and monitor connection status. Local Traffic Manager must determine when a connection is no longer active and then retire the connection to avoid exhausting critical system resources. Resources such as memory and processor cycles are at risk if the connection table grows and remains unchecked.

You can also manage the duration of entries in the persistence table when using session persistence.

Connection reaping

Connections that close or reset in a normal way are retired from the connection table automatically. A significant number of connections, however, often remain idle without closing normally, for any number of reasons. Consequently, Local Traffic Manager must reap these connections once they have been determined to be inactive. Reaping is the process of retiring or recycling connections that would otherwise remain idle.

Since you can configure timeout settings in multiple places, it is important to understand that sometimes more than one timeout setting affects the same connection. The optimal timeout configuration is one that

retains idle connections for an appropriate amount of time (variable by application) before deciding that the connections are inactive and should be retired, to conserve system resources.

Idle timeout options

Idle connections can be timed out by protocol profiles or SNATs associated with the virtual server handling the connection. Connections that a virtual server does not manage can be timed out based on SNAT automap or VLAN group settings.

The shortest timeout value that applies to a connection is the value that always takes effect. In some cases, however, you might want to change this behavior.

For example, you might have configured a forwarding virtual server that is intended to carry long-standing connections, and these connections might become idle for long periods of time (such as SSH sessions). In this case, you can configure a long idle timeout value on the related protocol profile (in this case, TCP).

Idle timeout settings that affect connection reaping

A list of objects containing idle connection timeout settings that affect reaping. For each object type, the table lists the default value and whether that value is user-configurable.

Configuration Object Types	Default in Seconds	User-configured?
Fast L4, Fast HTTP, TCP, and Sctp profiles	300	Yes
UDP profiles	60	Yes
SNAT automap	Indefinite	No
VLAN group	300	No

Other timeout settings

Local Traffic Manager[™] includes two other idle timeout settings, but these settings do not affect connection reaping. These settings appear in the OneConnect[™] and persistence profile types.

The OneConnect timeout value controls the length of time that an idle server-side connection is available for re-use; that is, this timeout value might cause the system to close a server-side connection after becoming idle for a certain period of time. In this case, since that connection was never actively in use, no active client-side connections are affected, and the system transparently selects or establishes another server-side connection for new connections. The OneConnect timeout setting need not be coordinated with the idle timeout settings of other profiles.

Persistence timeout settings are actually idle timeout settings for a session, rather than for a single connection. Thus, persistence timeout settings should typically be set to a value slightly larger than the applicable connection idle timeout settings, to allow sessions to continue even if a connection within the session has expired.

Idle timeout settings that do not affect connection reaping

Local Traffic Manager includes two other idle timeout settings, but these settings do not affect connection reaping. These settings appear in the OneConnect™ and persistence profile types. This table shows the default values for these settings and whether the settings are user-configurable.

Configuration Object Type	Default in Seconds	User-configured?
OneConnect™ profiles	Disabled	Yes
Cookie Hash, Destination Address Affinity, Hash, SIP, Source Address Affinity, and Universal persistence profiles	180	Yes
MSRDP and SSL persistence profiles	300	Yes

About the network map

The BIG-IP Configuration utility includes a feature known as the network map. The *network map* shows a summary of local traffic objects, as well as a visual map of the virtual servers, pools, and pool members on the BIG-IP® system. For both the local traffic summary and the network map, you can customize the display using a search mechanism that filters the information that you want to display, according to criteria that you specify. The system highlights in blue all matches from a search operation.

The filtering mechanism

You can filter the results of the network map feature by using the Type and Status lists in the filter bar, as well as a Search box. With the Search box, you can optionally type a specific string. Figure 1.1 shows the filtering options on the Network Map screen.

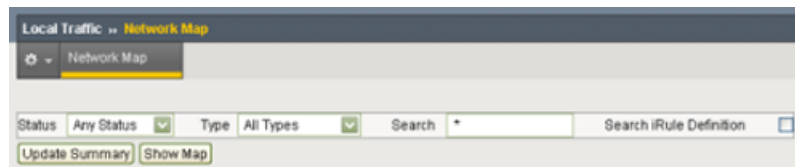


Figure 1: Filtering options on the Network Map screen

When using the Search box, you can specify a text string that you want the system to use in a search operation. The default is asterisk (*). The settings of the Status and Type fields determine the scope of the search. The system uses the specified search string to filter the results that the system displays on the screen.

For example, if you constrain the search to include only unavailable nodes whose IP address includes 10.10, the operation returns those nodes, along with the members of the pool, the pool itself, the associated virtual server, and any iRules® that you explicitly applied to that virtual server. The system sorts results alphabetically, by virtual server name.

The system supports searching on names, IP address, and IP address:port combinations, in both IPv4 and IPv6 address formats. The system processes the string as if an asterisk wildcard character surrounds the string. For example, you specify 10, the system effectively searches as if you had typed *10*. You can also

specifically include the asterisk wildcard character. For example, you can use the following search strings: 10.10.10.*:80, 10.10*, and *:80. If you specifically include a wildcard character, the system treats the string accordingly. For example, if you specify 10*, the system assumes you want to search for objects whose IP addresses begin with 10.

Tip: Browsers have limits as to how much data they can render before they become sluggish and halt processing. Mapping large configurations might approach those limits; therefore, memory constraints might prevent the system from producing a network map of the whole configuration. If this might happen, the system posts an alert indicating that you can use the Network Map summary screen to determine the complexity of the configuration, which can give you an indication of the size of the resulting map. You can modify the search criteria to return fewer results, producing a map that does not encounter those limits.

Object summary

When you first open the Network Map screen, the screen displays a summary of local traffic objects. This summary includes the type of objects specified with the search mechanism, the number of each type of object, and, for each object type, the number of objects with a given status.

The summary displays data for these object types:

- Virtual servers
- Pools
- Pool members
- Nodes
- iRules

Note: A local traffic summary includes only those objects that are referenced by a virtual server. For example, if you have configured a pool on the system but there is no virtual server that references that pool, the local traffic summary does not include that pool, its members, or the associated nodes in the summary.

This figure shows an example of a network map screen that summarizes local traffic objects on the system.

Object Type	Total	Available	Unavailable	Offline	Unknown
Virtual Servers	1	0	0	0	1
Pools	1	1	0	0	0
Pool Members	1	1	0	0	0
Nodes	1	1	0	0	0
iRules	0	-	-	-	-

Figure 2: Local Traffic summary

The network map display

The network map presents a visual hierarchy of the names and status of virtual servers, pools, pool members, nodes, and iRules[®] defined on the system. The map shows all objects in context, starting with the virtual servers at the top. The Status, Type, and Search settings at the top of the screen determine the objects that the map includes.

When you position the cursor over an object on the map, the system presents hover text containing information about the object. When you position the cursor over the status icon accompanying an object, the system presents hover text containing information about the object's status, text which also appears on the pool's Properties screen.

The system arranges objects in alphabetic order, and organizes the dependent objects in a hierarchy.

Due to the way that a network map presents objects in context, the updated screen also shows objects of other statuses, types, and names that relate to those objects. This is because a network map always shows objects in context with the objects that depend on them, and the objects they depend on.

For example, if you have an available virtual server with an available pool and two pool members, one available and one offline, then selecting Offline from the Status list causes the system to show the offline pool member in context with the available virtual server and the available pool. This is because the available virtual server and the available pool depend on the offline pool member.

Virtual Servers

Introduction to virtual servers

Virtual servers and virtual addresses are two of the most important components of any BIG-IP[®] Local Traffic Manager[™] configuration:

- A *virtual server* is a traffic-management object on the BIG-IP system that is represented by an IP address and a service. Clients on an external network can send application traffic to a virtual server, which then directs the traffic according to your configuration instructions. Virtual servers typically direct traffic to a pool of servers on an internal network, by translating the destination IP address in each packet to a pool member address. Overall, virtual servers increase the availability of resources for processing client requests.
- A *virtual address* is the IP address component of a virtual server. For example, if a virtual server's destination IP address and service are 10.10.10.2:80, then the IP address 10.10.10.2 is a virtual address. You do not explicitly create virtual addresses; instead, the BIG-IP system creates a virtual address when you create a virtual server and specify the destination IP address.

You can create a many-to-one relationship between virtual servers and a virtual address. For example, you can create the three virtual servers 10.10.10.2:80, 10.10.10.2:443, and 10.10.10.2:161 for the same virtual address, 10.10.10.2.

You can enable and disable a virtual address. When you disable a virtual address, none of the virtual servers associated with that address can receive incoming network traffic.

About virtual server settings

A virtual server has several settings that you can configure to affect the way that a virtual server manages traffic. You can also assign certain resources to a virtual server, such as a load balancing pool and various policies. Together, these properties, settings, and resources represent the definition of a virtual server, and most have default values. When you create a virtual server, you can either retain the default values or adjust them to suit your needs.

If you have created a virtual server that is a standard type of virtual server, one of the resources you typically assign to the virtual server is a default pool. A default pool is the server pool to which Local Traffic Manager[™] sends traffic if no iRule or policy exists that specifies a different pool. Note that if you plan on using an iRule or policy to direct traffic to a pool, you must assign the iRule or policy as a resource to the virtual server.

Types of virtual servers

There are several different types of virtual servers that you can create.

Table 1: Types of virtual servers

Type	Description
Standard	A <i>Standard</i> virtual server (also known as a <i>load balancing</i> virtual server) directs client traffic to a load balancing pool and is the most basic type of virtual server. When you first create the virtual server, you assign an existing default pool to it. From then on, the virtual server automatically directs traffic to that default pool.
Forwarding (Layer 2)	You can set up a <i>Forwarding (Layer 2)</i> virtual server to share the same IP address as a node in an associated VLAN. To do this, you must perform some additional configuration tasks. These tasks consist of: creating a VLAN group that includes the VLAN in which the node resides, assigning a self-IP address to the VLAN group, and disabling the virtual server on the relevant VLAN.
Forwarding (IP)	A <i>Forwarding (IP)</i> virtual server is just like other virtual servers, except that a forwarding virtual server has no pool members to load balance. The virtual server simply forwards the packet directly to the destination IP address specified in the client request. When you use a forwarding virtual server to direct a request to its originally-specified destination IP address, Local Traffic Manager adds, tracks, and reaps these connections just as with other virtual servers. You can also view statistics for a forwarding virtual servers.
Performance (HTTP)	A <i>Performance (HTTP)</i> virtual server is a virtual server with which you associate a Fast HTTP profile. Together, the virtual server and profile increase the speed at which the virtual server processes HTTP requests.
Performance (Layer 4)	A <i>Performance (Layer 4)</i> virtual server is a virtual server with which you associate a Fast L4 profile. Together, the virtual server and profile increase the speed at which the virtual server processes Layer 4 requests.
Stateless	A <i>stateless</i> virtual server prevents the BIG-IP system from putting connections into the connection table for wildcard and forwarding destination IP addresses. When creating a stateless virtual server, you cannot configure SNAT automap, iRules, or port translation, and you must configure a default load balancing pool. Note that this type of virtual server applies to UDP traffic only.
Reject	A <i>Reject</i> virtual server specifies that the BIG-IP system rejects any traffic destined for the virtual server IP address.
DHCP Relay	A <i>DHCP Relay</i> virtual server relays Dynamic Host Control Protocol (DHCP) messages between clients and servers residing on different IP networks. Known as a <i>DHCP relay agent</i> , a BIG-IP system with a DHCP Relay type of virtual server listens for DHCP client messages being broadcast on the subnet and then relays those messages to the DHCP server. The DHCP server then uses the BIG-IP system to send the responses back to the DHCP client. Configuring a DHCP Relay virtual server on the BIG-IP system relieves you of the tasks of installing and running a separate DHCP server on each subnet.
Internal	An <i>internal virtual server</i> is one that can send traffic to an intermediary server for specialized processing before the standard virtual server sends the traffic to its final destination. For example, if you want the BIG-IP system to perform content adaptation on HTTP requests or responses, you can create an internal virtual server that load balances those requests or responses to a pool of ICAP servers before sending the traffic back to the standard virtual server.

About source and destination addresses

There are two distinct types of virtual servers that you can create: virtual servers that listen for a host destination address and virtual servers that listen for a network destination address. For both types of virtual servers, you can also specify a source IP address.

About source addresses

When configuring a virtual sever, you can specify an IP address or network from which the virtual server will accept traffic. For this setting to function properly, you must specify a value other than 0.0.0.0/0 or ::/0 (that is, any/0, any6/0). To maximize utility of this setting, specify the most specific address prefixes spanning all customer addresses and no others.

About host destination addresses

A *host virtual server* represents a specific site, such as an Internet web site or an FTP site, and the virtual server load balances traffic targeted to content servers that are members of a pool. A host virtual server provides a level of security, similar to an access control list (ACL), because its destination address includes a port specification, causing the virtual server to accept only traffic destined for that port.

The IP address that you assign to a host virtual server should match the IP address that Domain Name System (DNS) associates with the site's domain name. When the BIG-IP® system receives a connection request for that site, Local Traffic Manager™ recognizes that the client's destination IP address matches the IP address of the virtual server, and subsequently forwards the client request to one of the content servers that the virtual server load balances.

About network destination addresses

A *network virtual server* is a virtual server whose IP address has no bits set in the host portion of the IP address (that is, the host portion of its IP address is 0). There are two kinds of network virtual servers: those that direct client traffic based on a range of destination IP addresses, and those that direct client traffic based on specific destination IP addresses that the BIG-IP system does not recognize. A network virtual server provides a level of security because its destination network address includes a port specification, causing the virtual server to accept only traffic destined for that port on the specified network .

When you have a range of destination IP addresses

With an IP address whose host bit is set to 0, a virtual server can direct client connections that are destined for an entire range of IP addresses, rather than for a single destination IP address (as is the case for a host virtual server). Thus, when any client connection targets a destination IP address that is in the network specified by the virtual server IP address, Local Traffic Manager (LTM®) can direct that connection to one or more pools associated with the network virtual server.

For example, the virtual server can direct client traffic that is destined for any of the nodes on the 192.168.1.0 network to a specific load balancing pool such as ingress-firewalls. Or, a virtual server could direct a web connection destined to any address within the subnet 192.168.1.0/24, to the pool default_webservers.

When you have transparent devices (wildcard virtual servers)

Besides directing client connections that are destined for a specific network or subnet, a network virtual server can also direct client connections that have a specific destination IP address that the virtual server

does not recognize, such as a transparent device. This type of network virtual server is known as a wildcard virtual server.

Wildcard virtual servers are a special type of network virtual server designed to manage network traffic that is targeted to transparent network devices. Examples of transparent devices are firewalls, routers, proxy servers, and cache servers. A wildcard virtual server manages network traffic that has a destination IP address unknown to the BIG-IP system.

Unrecognized client IP addresses

A host-type of virtual server typically manages traffic for a specific site. When receiving a connection request for that site, Local Traffic Manager forwards the client to one of the content servers that the virtual server load balances.

However, when load balancing transparent nodes, the BIG-IP system might not recognize a client's destination IP address. The client might be connecting to an IP address on the other side of the firewall, router, or proxy server. In this situation, Local Traffic Manager cannot match the client's destination IP address to a virtual server IP address.

Wildcard network virtual servers solve this problem by not translating the incoming IP address at the virtual server level on the BIG-IP system. For example, when Local Traffic Manager does not find a specific virtual server match for a client's destination IP address, LTM matches the client's destination IP address to a wildcard virtual server, designated by an IP address of 0.0.0.0. Local Traffic Manager then forwards the client's packet to one of the firewalls or routers that the wildcard virtual server load balances, which in turn forwards the client's packet to the actual destination IP address.

Default and port-specific wildcard servers

There are two kinds of wildcard virtual servers that you can create:

Default wildcard virtual servers

A *default wildcard virtual server* is a wildcard virtual server that uses port 0 and handles traffic for all services. A wildcard virtual server is enabled for all VLANs by default. However, you can specifically disable any VLANs that you do not want the default wildcard virtual server to support. Disabling VLANs for the default wildcard virtual server is done by creating a VLAN disabled list. Note that a VLAN disabled list applies to default wildcard virtual servers only. You cannot create a VLAN disabled list for a wildcard virtual server that is associated with one VLAN only.

Port-specific wildcard virtual servers

A *port-specific wildcard virtual server* handles traffic only for a particular service, and you define it using a service name or a port number. You can use port-specific wildcard virtual servers for tracking statistics for a particular type of network traffic, or for routing outgoing traffic, such as HTTP traffic, directly to a cache server rather than a firewall or router.

If you use both a default wildcard virtual server and port-specific wildcard virtual servers, any traffic that does not match either a standard virtual server or one of the port-specific wildcard virtual servers is handled by the default wildcard virtual server.

We recommend that when you define transparent nodes that need to handle more than one type of service, such as a firewall or a router, you specify an actual port for the node and turn off port translation for the virtual server.

Multiple wildcard servers

You can define multiple wildcard virtual servers that run simultaneously. Each wildcard virtual server must be assigned to an individual VLAN, and therefore can handle packets for that VLAN only.

In some configurations, you need to set up a wildcard virtual server on one side of the BIG-IP system to load balance connections across transparent devices. You can create another wildcard virtual server on the

other side of the BIG-IP system to forward packets to virtual servers receiving connections from the transparent devices and forwarding them to their destination.

About route domain IDs

Whenever you configure the **Source** and **Destination** settings on a virtual server, the BIG-IP system requires that the route domain IDs match, if route domain IDs are specified. To ensure that this requirement is met, the BIG-IP system enforces specific rules, which vary depending on whether you are modifying an existing virtual server or creating a new virtual server.

Table 2: Modifying an existing virtual server

User action	Result
In the destination address, you change an existing route domain ID.	The system automatically changes the route domain ID on the source address to match the new destination route domain ID.
In the source address, you change an existing route domain ID.	If the new route domain ID does not match the route domain ID in the destination address, the system displays an error message stating that the two route domain IDs must match.

Table 3: Creating a new virtual server

User action	Result
You specify a destination IP address only, with a route domain ID, and do not specify a source IP address.	The source IP address defaults to 0.0.0.0 and inherits the route domain ID from the destination IP address.
You specify both source and destination addresses but no route domain IDs.	The BIG-IP system uses the default route domain.
You specify both source and destination addresses and a route domain ID on each of the IP addresses.	The BIG-IP system verifies that both route domain IDs match. Otherwise, the system displays an error message.
You specify both source and destination addresses and a route domain ID on one of the addresses, but exclude an ID from the other address.	The system verifies that the specified route domain ID matches the ID of the default route domain. Specifically, when one address lacks an ID, the only valid configuration is one in which the ID specified on the other address is the ID of a default route domain. Otherwise, the system displays an error message.

About destination service ports

Status notification to virtual addresses

You can configure a virtual server so that the status of the virtual server contributes to the associated virtual address status. When disabled, the status of the virtual server does not contribute to the associated virtual

address status. This status, in turn, affects the behavior of the system when you enable route advertisement of virtual addresses.

About profiles for traffic types

Not only do virtual servers distribute traffic across multiple servers, they also treat varying types of traffic differently, depending on your traffic-management needs. For example, a virtual server can enable compression on HTTP request data as it passes through the BIG-IP system, or decrypt and re-encrypt SSL connections and verify SSL certificates. For each type of traffic destined for a specific virtual server, the virtual server can apply an entire group of settings (known as a *profile*) to affect the way that the BIG-IP system manages that traffic type.

In addition to compression and SSL profiles, you can configure a virtual server to apply profiles such as TCP, UDP, SPDY, SIP, FTP, and many more.

About VLAN and tunnel assignment

When you configure a virtual server, you can specify one or more VLANs, tunnels, or both, using the `VLAN` and `Tunnel Traffic` and **VLANs and Tunnels** settings. Configuring this feature specifies the VLANs or tunnels from which the virtual server will accept traffic. In a common configuration, the VLANs and tunnels selected reside on the external network.

About source address translation (SNATs)

When the default route on the servers does not route responses back through the BIG-IP system, you can create a secure network address translation (SNAT). A *secure network address translation (SNAT)* ensures that server responses always return through the BIG-IP® system. You can also use a SNAT to hide the source addresses of server-initiated requests from external devices.

For inbound connections from a client, a SNAT translates the source IP address within packets to a BIG-IP system IP address that you or the BIG-IP system defines. The destination node then uses that new source address as its destination address when responding to the request.

For outbound connections, SNATs ensure that the internal IP address of the server node remains hidden to an external host when the server initiates a connection to that host.

If you want the system to choose a SNAT translation address for you, you can select the Auto Map feature. If you prefer to define your own address, you can create a SNAT pool and assign it to the virtual server.

Important: *F5 recommends that before implementing a SNAT, you understand network address translation (NAT).*

About bandwidth control

You can specify an existing static bandwidth control policy for the system to use to enforce a throughput policy for incoming network traffic. A *static* bandwidth control policy controls the aggregate rate for a group of applications or a network path. The bandwidth control policy enforces the total amount of bandwidth that can be used, specified as the maximum rate of the resource you are managing. The rate can be the total bandwidth of the BIG-IP® device, or it might be a group of traffic flows.

About traffic classes

When you create or modify a virtual server, you can assign one or more existing traffic classes to the virtual server. A *traffic class* allows you to classify traffic according to a set of criteria that you define, such as source and destination IP addresses. Traffic classes define not only classification criteria, but also a classification ID. Once you have defined the traffic class and assigned the class to a virtual server, the BIG-IP system associates the *classification ID* to each traffic flow. In this way, the BIG-IP system can regulate the flow of traffic based on that classification.

When attempting to match traffic flows to a traffic class, the BIG-IP system uses the most specific match possible.

About connection and rate limits

A virtual server, pool member, or node can prevent an excessive number of connection requests, such as during a Denial of Service (DoS) attack or during a high-demand shopping event. To ensure the availability of a virtual server, pool member, or node, you can use the BIG-IP® Local Traffic Manager™ to manage the total number of connections and the rate at which connections are made.

When you specify a connection limit, the system prevents the total number of concurrent connections to the virtual server, pool member, or node from exceeding the specified number.

When you specify a connection rate limit, the system controls the number of allowed new connections per second, thus providing a manageable increase in connections without compromising availability.

About connection and persistence mirroring

BIG-IP® system redundancy includes the ability for a device to mirror connection and persistence information to another device, to prevent interruption in service during failover. The BIG-IP system mirrors connection and persistence data over TCP port 1028 with every packet or flow state update.

Important: *Connection mirroring only functions between devices that reside on identical hardware platforms.*

Connection mirroring operates at the traffic group level. That is, each device in a device group has a specific mirroring peer device for each traffic group. The mirroring peer device is the traffic group's next-active device.

For example, if device `Bigip_A` is active for traffic group `traffic-group-1`, and the next-active device for that traffic group is `Bigip_C`, then the traffic group on the active device mirrors its in-process connections to `traffic-group-1` on `Bigip_C`.

If `Bigip_A` becomes unavailable and failover occurs, `traffic-group-1` on `Bigip_C` becomes active and continues the processing of any current connections.

Note: *The BIG-IP system can mirror connections for as many as 15 active traffic groups simultaneously.*

About destination address and port translation

When you enable address translation on a virtual server, the BIG-IP system translates the destination address of the virtual server to the node address of a pool member. When you disable address translation, the system uses the virtual server destination address without translation. The default is enabled.

When you enable port translation on a virtual server, the BIG-IP system translates the port of the virtual server. When you disable port translation, the system uses the port without translation. Turning off port translation for a virtual server is useful if you want to use the virtual server to load balance connections to any service. The default is enabled.

About source port preservation

On a virtual server, you can specify whether the BIG-IP system preserves the source port of the connection. You can instruct the BIG-IP system to either preserve the source port in certain or all cases, or change the source port for all connections. The default behavior is to attempt to preserve the source port but use a different port if the source port from a particular SNAT is already in use.

Alternatively, you can instruct the system to always preserve the source port. In this case, if the port is in use, the system does not process the connection. F5 Networks recommends that you restrict use of this setting to cases that meet at least one of the following conditions:

- The port is configured for UDP traffic.
- The system is configured for nPath routing or is running in transparent mode (that is, there is no translation of any other Layer 3 or Layer 4 field).
- There is a one-to-one relationship between virtual IP addresses and node addresses, or clustered multi-processing (CMP) is disabled.

Instructing the system to change instead of preserve the source port of the connection is useful for obfuscating internal network addresses.

About clone pools

You use a *clone pool* when you want to configure the BIG-IP system to send traffic to a pool of intrusion detection systems (IDSs). An *intrusion detection system* (IDS) is a device that monitors inbound and outbound network traffic and identifies suspicious patterns that might indicate malicious activities or a network attack. You can use the clone pool feature of a BIG-IP system to copy traffic to a dedicated IDS or a sniffer device.

Important: *A clone pool receives all of the same traffic that the load-balancing pool receives.*

To configure a clone pool, you first create the clone pool of IDS or sniffer devices and then assign the clone pool to a virtual server. The clone pool feature is the recommended method for copying production traffic to IDS systems or sniffer devices. Note that when you create the clone pool, the service port that you assign to each node is irrelevant; you can choose any service port. Also, when you add a clone pool to a virtual server, the system copies only new connections; existing connections are not copied.

You can configure a virtual server to copy client-side traffic, server-side traffic, or both:

- A *client-side clone pool* causes the virtual server to replicate client-side traffic (prior to address translation) to the specified clone pool.
- A *server-side clone pool* causes the virtual server to replicate server-side traffic (after address translation) to the specified clone pool.

You can configure an unlimited number of clone pools on the BIG-IP system.

About auto last hop

When you enable the **Auto Last Hop** setting, the BIG-IP system can send any return traffic to the MAC address that transmitted the request, even if the routing table points to a different network or interface. As

a result, the system can send return traffic to clients even when there is no matching route, such as when the system does not have a default route configured and the client is located on a remote network.

This setting is also useful when the system is load balancing transparent devices that do not modify the source IP address of the packet. Without the **Auto Last Hop** setting enabled, the system could return connections to a different transparent node, resulting in asymmetric routing.

You can configure this setting globally and on an object level. You set the global **Auto Last Hop** value on the System >> Configuration >> Local Traffic >> General screen. In this case, users typically retain the default setting, **Enabled**. When you configure **Auto Last Hop** at the object level with a value other than **Default**, the value you configure takes precedence over the global setting. This enables you to configure **Auto Last Hop** on a per-pool member basis. The default value for the virtual server **Auto Last Hop** setting is **Default**, which causes the system to use the global **Auto Last Hop** setting to send back the request.

About NAT64

You can instruct the BIG-IP system to allow IPv6 hosts to communicate with IPv4 servers. This setting is disabled by default.

Virtual server resources

When you create a virtual server, one of the resources that you can specify for a virtual server to use is a default server pool that you want to serve as the destination for any traffic coming from that virtual server. The system uses this pool, unless you have specified a different pool in another configuration object such as an iRule.

You can also assign other resources to a virtual server, such as iRules, policies, and persistence profiles.

About virtual address settings

A virtual address has settings that you can configure to affect the way the BIG-IP system manages traffic destined for that virtual address. When the system creates a virtual address, you can either retain the default values or adjust them to suit your needs.

About automatic deletion

You can enable an **Auto Delete** setting on a virtual address so that BIG-IP system automatically deletes the virtual address last associated virtual server is deleted. If you disable this setting, the system retains the virtual address, even when all associated virtual servers have been deleted. The default value is enabled.

About traffic groups

If you want the virtual address to be a floating IP address, that is, an address shared between two or more BIG-IP devices in a device group, you can assign a floating traffic group to the virtual address. A floating traffic group causes the virtual address to become a floating self IP address. A floating virtual address ensures that application traffic reaches its destination when the relevant BIG-IP device becomes unavailable.

If you want the virtual address to be a static (non-floating) IP address (used mostly for standalone devices), you can assign a non-floating traffic group to the virtual address. A non-floating traffic group causes the virtual address to become a non-floating self IP address.

About route advertisement

You can enable route advertisement for a specific virtual address. When you enable route advertisement, the BIG-IP system advertises routes to the virtual address for the purpose of dynamic routing. The system can advertise a route to the virtual address under any one of these conditions:

- When any virtual server is available. Additionally, when the **ICMP Echo** setting is set to **Selective**, the BIG-IP system sends an ICMP echo response for a request sent to the virtual address, if one or more virtual servers associated with the virtual address is in an Up or Unknown state.
- When all virtual servers are available. Additionally, when the **ICMP Echo** setting is set to **Selective**, the BIG-IP system always sends an ICMP echo response for a request sent to the virtual address, but only when all virtual servers are available.
- Always advertises the route regardless of the virtual servers available. Additionally, when the **ICMP Echo** setting is set to **Selective**, the BIG-IP system always sends an ICMP echo response for a request sent to the virtual address, regardless of the state of any virtual servers associated with the virtual address.

About ARP and virtual addresses

Whenever the system creates a virtual address, Local Traffic Manager™ internally associates the virtual address with a MAC address. This in turn causes the BIG-IP® system to respond to Address Resolution Protocol (ARP) requests for the virtual address, and to send gratuitous ARP requests and responses with respect to the virtual address. As an option, you can disable ARP activity for virtual addresses, in the rare case that ARP activity affects system performance. This most likely occurs only when you have a large number of virtual addresses defined on the system.

About ICMP echo responses

You can control whether the BIG-IP system sends responses to Internet Control Message Protocol (ICMP) echo requests, on a per-virtual address basis. Specifically, you can:

- Disable ICMP echo responses. This causes the BIG-IP system to never send an ICMP echo response for ICMP request packets sent to the virtual address, regardless of the state of any virtual servers associated with the virtual address.
- Enable ICMP echo responses. This causes the BIG-IP system to always send an ICMP echo response for ICMP request packets sent to the virtual address, regardless of the state of any virtual servers associated with the virtual address.
- Selectively enable ICMP echo responses. This causes the BIG-IP system to internally enable or disable ICMP responses for the virtual address based on node status for any associated virtual servers. This value affects the behavior of the system in different ways, depending on the value of the **Advertise Route** setting.

Virtual server and virtual address status

At any time, you can determine the status of a virtual server or virtual address, using the BIG-IP Configuration utility. You can find this information by displaying the list of virtual servers or virtual addresses and viewing the Status column, or by viewing the **Availability** property of the object.

The BIG-IP Configuration utility indicates status by displaying one of several icons, distinguished by shape and color:

- The shape of the icon indicates the status that the monitor has reported for that node.
- The color of the icon indicates the actual status of the node.

Clustered multiprocessing

The BIG-IP® system includes a performance feature known as Clustered Multiprocessing™, or CMP®. CMP is a traffic acceleration feature that creates a separate instance of the Traffic Management Microkernel (TMM) service for each central processing unit (CPU) on the system. When CMP is enabled, the workload is shared equally among all CPUs.

Whenever you create a virtual server, the BIG-IP system automatically enables the CMP feature. When CMP is enabled, all instances of the TMM service process application traffic.

When you view standard performance graphs using the BIG-IP Configuration utility, you can see multiple instances of the TMM service (`tmm0`, `tmm1`, and so on).

When CMP is enabled, be aware that:

- While displaying some statistics individually for each TMM instance, the BIG-IP system displays other statistics as the combined total of all TMM instances.
- Connection limits for a virtual server with CMP enabled are distributed evenly across all instances of the TMM service.

Note: F5 recommends that you disable the CMP feature if you set a small connection limit on pool members (for example, a connection limit of 2 for the 8400 platform or 4 for the 8800 platform).

You can enable or disable CMP for a virtual server, or you can enable CMP for a specific CPU.

Local Traffic Policies

About local traffic policy matching

BIG-IP® *local traffic policies* comprise a prioritized list of rules that match defined conditions and run specific actions, which you can assign to a virtual server that directs traffic accordingly. For example, you might create a policy that determines whether a client's browser is a Chrome browser and adds an `Alternative-Protocols` attribute to the header, so that subsequent requests from the Chrome browser are directed to a SPDY virtual server. Or you might create a policy that determines whether a client is using a mobile device, and then redirects its requests to the applicable mobile web site's URL.

About strategies for local traffic policy matching

Each BIG-IP® local traffic matching policy requires a matching strategy to determine the rule that applies if more than one rule matches.

The BIG-IP policies provide three policy matching strategies: a first-match, best-match, and all-match strategy. Each policy matching strategy prioritizes rules according to the rule's position within the Rules list.

Note: A rule without conditions becomes the default rule in a best-match or first-match strategy, when the rule is the last entry in the Rules list.

Table 4: Policy matching strategies

Matching strategy	Description
First-match strategy	A <i>first-match strategy</i> starts the actions for the first rule in the Rules list that matches.
Best-match strategy	<p>A <i>best-match strategy</i> selects and starts the actions of the rule in the Rules list with the best match, as determined by the following factors.</p> <ul style="list-style-type: none">• The number of conditions and operands that match the rule.• The length of the matched value for the rule.• The priority of the operands for the rule. <hr/> <p>Note: In a best-match strategy, when multiple rules match and specify an action, conflicting or otherwise, only the action of the best-match rule is implemented. A best-match rule can be the lowest ordinal, the highest priority, or the first rule that matches in the Rules list.</p> <hr/>
All-match strategy	<p>An <i>all-match strategy</i> starts the actions for all rules in the Rules list that match.</p> <hr/> <p>Note: In an all-match strategy, when multiple rules match, but specify conflicting actions, only the action of the best-match rule is implemented. A best-match rule can be the lowest ordinal, the highest priority, or the first rule that matches in the Rules list.</p> <hr/>

Local traffic policy matching Requires profile settings

This table summarizes the profile settings that are required for local traffic policy matching.

Requires Setting	Description
http	Specifies that the policy matching requires an HTTP profile.
ssl	Specifies that the policy matching requires a Client SSL profile.
tcp	Specifies that the policy matching requires a TCP profile.

Local traffic policy matching Controls settings

This table summarizes the controls settings that are required for local traffic policy matching.

Controls Setting	Description
acceleration	Provides controls associated with acceleration functionality.
caching	Provides controls associated with caching functionality.
classification	Provides controls associated with classification.
compression	Provides controls associated with HTTP compression.
forwarding	Provides controls associated with forwarding functionality.
request-adaptation	Provides controls associated with request-adaptation functionality.
response-adaptation	Provides controls associated with response-adaptation functionality.
server-ssl	Provides controls associated with server-ssl functionality.

About rules for local traffic policy matching

BIG-IP® local traffic policy *rules* match defined conditions and start specific actions. You can create a policy with rules that are as simple or complex as necessary, based on the passing traffic. For example, a rule might simply determine that a client's browser is a Chrome browser that is not on an administrator network. Or a rule might determine that a request URL starts with `/video`, that the client is a mobile device, and that the client's subnet does not match `172.27.56.0/24`.

About conditions for local traffic policy matching

The *conditions* for a local traffic policy rule define the necessary criteria that must be met in order for the rule's actions to be applied. For example, a policy might include the following conditions, which, when met by a request, would allow the rule's specified actions to be applied.

Condition	Setting
Operand	http-host
Event	request

Condition	Setting
Selector	all
Condition	equals
Values	www.siterequest.com

Local traffic policy matching Conditions operands

This table summarizes the operands for each condition used in policy matching.

Operand	Type	Valid Events	Selectors and Parameters	Description
client-ssl	string/number	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • cipher • cipher-bits • protocol 	Requires a Client SSL profile for policy matching.
http-basic-auth	string	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • password • username 	Returns <username>: <password> or parts of it.
http-cookie	string	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • all • name 	Returns the value of a particular cookie or cookie attribute.
http-header	string	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • all • name (required) 	Returns the value of a particular header.
http-host	string/number	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • all • host • port 	Provides all or part of the HTTP Host header.
http-method	string	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • all 	Provides the HTTP method.
http-referer	string/number	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • all • extension • host • path • path-segment • index (required) • port • query-parameter • name (required) • query-string • scheme • unnamed-query-parameter • index (required) 	Provides all or part of the HTTP Referer header.

Operand	Type	Valid Events	Selectors and Parameters	Description
http-set-cookie	string	<ul style="list-style-type: none"> response 	<ul style="list-style-type: none"> domain <ul style="list-style-type: none"> name (required) expiry <ul style="list-style-type: none"> name (required) path <ul style="list-style-type: none"> name (required) value <ul style="list-style-type: none"> name (required) version <ul style="list-style-type: none"> name (required) 	Sets the selected setting of a particular cookie or cookie attribute.
http-status	string/number	<ul style="list-style-type: none"> response 	<ul style="list-style-type: none"> all code text 	Returns the HTTP status line or part of it.
http-uri	string/number	<ul style="list-style-type: none"> request 	<ul style="list-style-type: none"> all extension host path path-segment <ul style="list-style-type: none"> index (required) port query-parameter <ul style="list-style-type: none"> name (required) query-string scheme unnamed-query-parameter <ul style="list-style-type: none"> index (required) 	Provides all or part of the request URI.
http-version	string/number	<ul style="list-style-type: none"> request response 	<ul style="list-style-type: none"> response <ul style="list-style-type: none"> all major minor protocol 	Provides HTTP/1.1 a number.
tcp	number	<ul style="list-style-type: none"> request response 	<ul style="list-style-type: none"> mss internal true port <ul style="list-style-type: none"> internal true local true 	Requires a TCP profile for policy matching.

Operand	Type	Valid Events	Selectors and Parameters	Description
			<ul style="list-style-type: none"> route-domain internal true rtt internal true vlan internal true vlan-id internal true 	

About actions for a local traffic policy rule

The *actions* for a local traffic policy rule determine how traffic is handled. For example, actions for a rule could include the following ways of handling traffic.

- Blocking traffic
- Rewriting a URL
- Logging traffic
- Adding a specific header
- Redirecting traffic to a different pool member
- Selecting a specific Web Application policy

Local traffic policy matching Actions operands

This table summarizes the actions associated with the conditions of the rule used in policy matching.

Target	Type	Valid Events	Action
acceleration	string/number	<ul style="list-style-type: none"> request 	<ul style="list-style-type: none"> disable enable
cache	string	<ul style="list-style-type: none"> request response 	<ul style="list-style-type: none"> disable enable pin true
compress	string	<ul style="list-style-type: none"> request response 	<ul style="list-style-type: none"> disable enable
decompress	string	<ul style="list-style-type: none"> request response 	<ul style="list-style-type: none"> disable enable
forward	string	<ul style="list-style-type: none"> request 	<ul style="list-style-type: none"> reset select clone-pool

Target	Type	Valid Events	Action
			<ul style="list-style-type: none"> • member • nexthop • node • pool • rateclass • snat • snatpool • vlan • vlan-id
http-cookie	string	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • insert <ul style="list-style-type: none"> • name (required) • value (required) • remove <ul style="list-style-type: none"> • name (required)
http-header	string/number	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • insert <ul style="list-style-type: none"> • name (required) • value (required) • remove <ul style="list-style-type: none"> • name (required) • replace <ul style="list-style-type: none"> • name (required) • value (required)
http-host	string	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • replace <ul style="list-style-type: none"> • value
http-referer	string	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • insert <ul style="list-style-type: none"> • value (required) • remove • replace <ul style="list-style-type: none"> • value
http-reply	string	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • redirect <ul style="list-style-type: none"> • location (required)
http-set-cookie	string/number	<ul style="list-style-type: none"> • response 	<ul style="list-style-type: none"> • insert <ul style="list-style-type: none"> • name (required) • domain • path • value (required) • remove

Target	Type	Valid Events	Action
			<ul style="list-style-type: none"> • name (required)
http-uri	string/number	<ul style="list-style-type: none"> • response 	<ul style="list-style-type: none"> • replace <ul style="list-style-type: none"> • path • query-string • value
log	string/number	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • write <ul style="list-style-type: none"> • message (required)
pem	string/number	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • classify <ul style="list-style-type: none"> • application • category • defer • protocol
request-adapt	string/number	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • disable • enable
response-adapt	string/number	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • disable • enable
server-ssl	string/number	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • disable • enable
tcl	string/number	<ul style="list-style-type: none"> • request • response 	<ul style="list-style-type: none"> • set-variable <ul style="list-style-type: none"> • name (required) • expression (required)
tcp-nagle	string/number	<ul style="list-style-type: none"> • request 	<ul style="list-style-type: none"> • disable • enable

Nodes

About nodes

A *node* is a logical object on the BIG-IP® Local Traffic Manager™ system that identifies the IP address of a physical resource on the network. You can explicitly create a node, or you can instruct Local Traffic Manager (LTM®) to automatically create one when you add a pool member to a load balancing pool.

The difference between a node and a pool member is that a node is designated by the device's IP address only (10.10.10.10), while designation of a pool member includes an IP address and a service (such as 10.10.10.8).

A primary feature of nodes is their association with health monitors. Like pool members, nodes can be associated with health monitors as a way to determine server status. However, a health monitor for a pool member reports the status of a service running on the device, whereas a health monitor associated with a node reports status of the device itself.

Nodes are the basis for creating a load balancing pool. For any server that you want to be part of a load balancing pool, you must first create a node, that is, designate that server as a node. After designating the server as node, you can add the node to a pool as a pool member. You can also associate a health monitor with the node, to report the status of that server.

About the node address setting

This setting specifies the IP address of the node. If you are using a route domain other than route domain 0, you can append a route domain ID to this node address. For example, if the node address applies to route domain 1, then you can specify a node address of 10.10.10.10.:%1.

About health monitor association

Using Local Traffic Manager™, you can monitor the health or performance of your nodes by associating monitors with those nodes. This is similar to associating a monitor with a load balancing pool, except that in the case of nodes, you are monitoring the IP address, whereas with pools, you are monitoring the services that are active on the pool members.

Local Traffic Manager (LTM®) contains many different pre-configured monitors that you can associate with nodes, depending on the type of traffic you want to monitor. You can also create your own custom monitors and associate them with nodes. The only pre-configured monitors that are not available for associating with nodes are monitors that are specifically designed to monitor pools or pool members rather than nodes.

Note: Any monitor that you associate with a node must reside either in partition *Common* or in the partition that contains the node.

There are two ways that you can associate a monitor with a node: by assigning the same monitor (that is, a default monitor) to multiple nodes at the same time, or by explicitly associating a monitor with each node as you create it.

About monitors and automatic node creation

If you create a pool member without first creating the parent node, Local Traffic Manager™ automatically creates the parent node for you. Fortunately, you can configure Local Traffic Manager (LTM®) to automatically associate one or more monitor types with every node that LTM creates. This eliminates the task of having to explicitly choose monitors for each node.

Keep the following in mind when working with default monitors:

- If a user with permission to manage objects in partition `Common` disables a monitor that is designated as the default monitor for nodes (such as the `icmp` monitor), this affects all nodes on the system. Ensure that the default monitor for nodes always resides in partition `Common`.
- To specify default monitors, you must have the `Administrator` user role assigned to your user account.
- If all nodes reside in the same partition, the default monitor must reside in that partition or in partition `Common`. If nodes reside in separate partitions, then the default monitor must reside in partition `Common`.

About monitors and explicit node creation

Sometimes, you might want to explicitly create a node, rather than having Local Traffic Manager™ create the node automatically. In this case, when you create the node, you can either associate non-default monitors with the node, or associate the default monitors with the node.

About monitor removal

You can remove a monitor that is explicitly associated with a specific node. When removing a monitor associated with a specific node, you can either remove the monitor association altogether, or change it so that only the default monitor is associated with the node.

Alternatively, you can remove any default monitors, that is, monitors that Local Traffic Manager™ automatically associates with any node that you create.

About node availability

You can specify the minimum number of health monitors that must report a node as being available to receive traffic before Local Traffic Manager™ reports that node as being in an up state.

About the ratio weight setting

When you are using the Ratio load balancing method, you can assign a ratio weight to each node in a pool. LTM® uses this ratio weight to determine the correct node for load balancing.

Note that at least one node in the pool must have a ratio value greater than 1. Otherwise, the effect equals that of the Round Robin load balancing method.

About the connection rate limit setting

The connection rate limit setting specifies the maximum rate of new connections allowed for the node. When you specify a connection rate limit, the system controls the number of allowed new connections per second, thus providing a manageable increase in connections without compromising availability. The default value of 0 specifies that there is no limit on the number of connections allowed per second.

About node state

A node must be enabled in order to accept traffic. When a node is disabled, Local Traffic Manager™ allows existing connections to time out or end normally. In this case, the node can accept new connections only if the connections belong to an existing persistence session. (In this way, a disabled node differs from a node that is set to down. The down node allows existing connections to time out, but accepts no new connections whatsoever.)

About node status

At any time, you can determine the status of a node, using the BIG-IP Configuration utility. You can find this information by displaying the list of nodes and viewing the Status column, or by viewing the Availability property of a node.

The BIG-IP Configuration utility indicates status by displaying one of several icons, distinguished by shape and color:

- The shape of the icon indicates the status that the monitor has reported for that node.
- The color of the icon indicates the actual status of the node.

Tip: You can manually set the availability of a node with the *Manual Resume* attribute of the associated health monitor.

Pools

Introduction to pools

In a typical client-server scenario, a client request goes to the destination IP address specified in the header of the request. For sites with a large amount of incoming traffic, the destination server can quickly become overloaded as it tries to service a large number of requests. To solve this problem, BIG-IP® Local Traffic Manager™ distributes client requests to multiple servers instead of to the specified destination IP address only. You configure Local Traffic Manager to do this when you create a load balancing pool.

About load balancing pools

A *load balancing pool* is a logical set of devices, such as web servers, that you group together to receive and process traffic. Instead of sending client traffic to the destination IP address specified in the client request, Local Traffic Manager™ sends the request to any of the servers that are members of that pool. This helps to efficiently distribute the load on your server resources.

When you create a pool, you assign pool members to the pool. A *pool member* is a logical object that represents a physical node (server), on the network. You then associate the pool with a virtual server on the BIG-IP® system. Once you have assigned a pool to a virtual server, Local Traffic Manager (LTM®) directs traffic coming into the virtual server to a member of that pool. An individual pool member can belong to one or multiple pools, depending on how you want to manage your network traffic.

The specific pool member to which Local Traffic Manager chooses to send the request is determined by the load balancing method that you have assigned to that pool. A *load balancing method* is an algorithm that LTM uses to select a pool member for processing a request. For example, the default load balancing method is *Round Robin*, which causes Local Traffic Manager to send each incoming request to the next available member of the pool, thereby distributing requests evenly across the servers in the pool.

Pool features

You can configure Local Traffic Manager™ (LTM) to perform a number of different operations for a pool. For example, you can:

- Associate health monitors with pools and pool members
- Enable or disable SNAT connections
- Rebind a connection to a different pool member if the originally-targeted pool member becomes unavailable
- Specify a load balancing algorithm for a pool
- Set the Quality of Service or Type of Service level within a packet
- Assign pool members to priority groups within a pool

You use the BIG-IP Configuration utility to create a load balancing pool, or to modify a pool and its members. When you create a pool, LTM® automatically assigns a group of default settings to that pool and its members.

You can retain these default settings or modify them. Also, you can modify the settings at a later time, after you have created the pool.

About health monitor association

Health monitors are a key feature of Local Traffic Manager™. Health monitors help to ensure that a server is in an up state and able to receive traffic. When you want to associate a monitor with an entire pool of servers, you do not need to explicitly associate that monitor with each individual server. Instead, you can simply assign the monitor to the pool itself. Local Traffic Manager then automatically monitors each member of the pool.

Local Traffic Manager contains many different pre-configured monitors that you can associate with pools, depending on the type of traffic you want to monitor. You can also create your own custom monitors and associate them with pools. The only monitor types that are not available for associating with pools are monitors that are specifically designed to monitor nodes and not pools or pool members. That is, the destination address in the monitor specifies an IP address only, rather than an IP address and a service port. These monitor types are:

- ICMP
- TCP Echo
- Real Server
- SNMP DCA
- SNMP DCA Base
- WMI

With Local Traffic Manager, you can configure your monitor associations in many useful ways:

- You can associate a health monitor with an entire pool instead of an individual server. In this case, Local Traffic Manager automatically associates that monitor with all pool members, including those that you add later. Similarly, when you remove a member from a pool, Local Traffic Manager no longer monitors that server.
- When a server that is designated as a pool member allows multiple processes to exist on the same IP address and port, you can check the health or status of each process. To do this, you can add the server to multiple pools, and then within each pool, associate a monitor with the that server. The monitor you associate with each server checks the health of the process running on that server.
- When associating a monitor with an entire pool, you can exclude an individual pool member from being associated with that monitor. In this case, you can associate a different monitor for that particular pool member, or you can exclude that pool member from health monitoring altogether. For example, you can associate pool members A, B, and D with the `http` monitor, while you associate pool member C with the `https` monitor.
- You can associate multiple monitors with the same pool. For instance, you can associate both the `http` and `https` monitors with the same pool.

Pool member availability

You can specify a minimum number of health monitors. Before Local Traffic Manager™ can report the pool member as being in an up state, this number of monitors, at a minimum, must report a pool member as being available to receive traffic.

Secure network address translations (SNATs) and network address translations (NATs)

When configuring a pool, you can specifically disable any secure network address translations (SNATs) or network address translations (NATs) for any connections that use that pool. By default, these settings are enabled. You can change this setting on an existing pool by displaying the Properties screen for that pool.

One case in which you might want to configure a pool to disable SNAT or NAT connections is when you want the pool to disable SNAT or NAT connections for a specific service. In this case, you could create a separate pool to handle all connections for that service, and then disable the SNAT or NAT for that pool.

Action when a service becomes unavailable

You can specify the action that you want Local Traffic Manager™ to take when the service on a pool member becomes unavailable.

Possible actions are:

- None. This is the default action.
- The BIG-IP® system sends an RST (TCP-only) or ICMP message.
- Local Traffic Manager simply cleans up the connection.
- Local Traffic Manager selects a different node.

You should configure the system to select a different node in certain cases only, such as:

- When the relevant virtual server is a Performance (Layer 4) virtual server with address translation disabled.
- When the relevant virtual server's Protocol setting is set to UDP.
- When the pool is a gateway pool (that is, a pool of routers)

Slow ramp time

When you take a pool member offline, and then bring it back online, the pool member can become overloaded with connection requests, depending on the load balancing method for the pool. For example, if you use the Least Connections load balancing method, the system sends all new connections to the newly-enabled pool member (because, technically, that member has the least amount of connections).

With the slow ramp time feature, you can specify the number of seconds that the system waits before sending traffic to the newly-enabled pool member. The amount of traffic is based on the ratio of how long the pool member is available compared to the slow ramp time, in seconds. Once the pool member is online for a time greater than the slow ramp time, the pool member receives a full proportion of the incoming traffic.

Type of Service (ToS) level

Another pool feature is the Type of Service (ToS) level. The *ToS* level is one means by which network equipment can identify and treat traffic differently based on an identifier.

As traffic enters the site, Local Traffic Manager™ can set the ToS level on a packet. Using the IP ToS to Server ToS level that you define for the pool to which the packet is sent. Local Traffic Manager can apply an iRule and send the traffic to different pools of servers based on that ToS level.

Local Traffic Manager can also tag outbound traffic (that is, the return packets based on an HTTP GET) based on the IP ToS to Client ToS value set in the pool. That value is then inspected by upstream devices and given appropriate priority.

For example, to configure a pool so that a ToS level is set for a packet sent to that pool, you can set both the IP ToS to Client level and the IP ToS to Server levels to 16. In this case, the ToS level is set to 16 when sending packets to the client and when sending packets to the server.

Note: *If you change the ToS level on a pool for a client or a server, existing connections continue to use the previous setting.*

Quality of Service (QoS) level

Another setting for a pool is the Quality of Service (QoS) level. In addition to the ToS level, the QoS level is a means by which network equipment can identify and treat traffic differently based on an identifier. Essentially, the QoS level specified in a packet enforces a throughput policy for that packet.

As traffic enters the site, Local Traffic Manager™ can set the QoS level on a packet. Using the Link QoS to Server QoS level that you define for the pool to which the packet is sent, Local Traffic Manager can apply an iRule that sends the traffic to different pools of servers based on that QoS level.

Local Traffic Manager can also tag outbound traffic (that is, the return packets based on an HTTP GET) based on the Link QoS to Client QoS value set in the pool. That value is then inspected by upstream devices and given appropriate priority.

For example, to configure a pool so that a QoS level is set for a packet sent to that pool, you can set the Link QoS to Client level to 3 and the Link QoS to Server level to 4. In this case, the QoS level is set to 3 when sending packets to the client, and set to 4 when sending packets to the server.

Number of reselect tries

You can specify the number of times that the system tries to contact a new pool member after a passive failure. A *passive failure* consists of a server-connect failure or a failure to receive a data response within a user-specified interval. The default value of 0 indicates no reselects.

Note: *This setting is for use primarily with TCP profiles. Using this setting with a Fast L4 profile is not recommended.*

About TCP request queue

TCP request queuing provides the ability to queue connection requests that exceed the capacity of connections for a pool, pool member, or node, as determined by the connection limit. Consequently, instead of dropping connection requests that exceed the capacity of a pool, pool member, or node, TCP request queuing enables those connection requests to reside within a queue in accordance with defined conditions until capacity becomes available.

When using session persistence, a request becomes queued when the pool member connection limit is reached.

Without session persistence, when all pool members have a specified connection limit, a request becomes queued when the total number of connection limits for all pool members is reached.

Conditions for queuing connection requests include:

- The maximum number of connection requests within the queue, which equates to the maximum number of connections within the pool, pool member, or node. Specifically, the maximum number of connection requests within the queue cannot exceed the cumulative total number of connections for each pool member or node. Any connection requests that exceed the capacity of the request queue are dropped.
- The availability of server connections for reuse. When a server connection becomes available for reuse, the next available connection request in the queue becomes dequeued, thus allowing additional connection requests to be queued.
- The expiration rate of connection requests within the queue. As queue entries expire, they are removed from the queue, thus allowing additional connection requests to be queued.

Connection requests within the queue become dequeued when:

- The connection limit of the pool is increased.
- A pool member's slow ramp time limit permits a new connection to be made.
- The number of concurrent connections to the virtual server decreases below the connection limit.
- The connection request within the queue expires.

About load balancing methods

Load balancing is an integral part of the BIG-IP® system. Configuring load balancing on a BIG-IP system means determining your load balancing scenario, that is, which pool member should receive a connection hosted by a particular virtual server. Once you have decided on a load balancing scenario, you can specify the appropriate load balancing method for that scenario.

A *load balancing method* is an algorithm or formula that the BIG-IP system uses to determine the server to which traffic will be sent. Individual load balancing methods take into account one or more dynamic factors, such as current connection count. Because each application of the BIG-IP system is unique, and server performance depends on a number of different factors, we recommend that you experiment with different load balancing methods, and select the one that offers the best performance in your particular environment.

Default load balancing method

The default load balancing method for the BIG-IP system is Round Robin, which simply passes each new connection request to the next server in line. All other load balancing methods take server capacity and/or status into consideration.

If the equipment that you are load balancing is roughly equal in processing speed and memory, Round Robin mode works well in most configurations. If you want to use the Round Robin method, you can skip the remainder of this section, and begin configuring other pool settings that you want to add to the basic pool configuration.

Local Traffic Manager load balancing methods

There are several load balancing methods available within the BIG-IP system for load balancing traffic to pool members.

Method	Description	When to use
Round Robin	This is the default load balancing method. Round Robin mode passes each new connection request to the next server in line, eventually distributing connections evenly across the array of machines being load balanced.	Round Robin mode works well in most configurations, especially if the equipment that you are load balancing is roughly equal in processing speed and memory.

Method	Description	When to use
Ratio (member) Ratio (node)	Local Traffic Manager distributes connections among pool members or nodes in a static rotation according to ratio weights that you define. In this case, the number of connections that each system receives over time is proportionate to the ratio weight you defined for each pool member or node. You set a ratio weight when you create each pool member or node.	These are static load balancing methods, basing distribution on user-specified ratio weights that are proportional to the capacity of the servers.
Dynamic Ratio (member) Dynamic Ratio (node)	The Dynamic Ratio methods select a server based on various aspects of real-time server performance analysis. These methods are similar to the Ratio methods, except that with Dynamic Ratio methods, the ratio weights are system-generated, and the values of the ratio weights are not static. These methods are based on continuous monitoring of the servers, and the ratio weights are therefore continually changing. <i>Note: To implement Dynamic Ratio load balancing, you must first install and configure the necessary server software for these systems, and then install the appropriate performance monitor.</i>	The Dynamic Ratio methods are used specifically for load balancing traffic to RealNetworks® RealSystem® Server platforms, Windows® platforms equipped with Windows Management Instrumentation (WMI), or any server equipped with an SNMP agent such as the UC Davis SNMP agent or Windows 2000 Server SNMP agent.
Fastest (node) Fastest (application)	The Fastest methods select a server based on the least number of current sessions. These methods require that you assign both a Layer 7 and a TCP type of profile to the virtual server. <i>Note: If the OneConnect™ feature is enabled, the Least Connections methods do not include idle connections in the calculations when selecting a pool member or node. The Least Connections methods use only active connections in their calculations.</i>	The Fastest methods are useful in environments where nodes are distributed across separate logical networks.
Least Connections (member) Least Connections (node)	The Least Connections methods are relatively simple in that Local Traffic Manager passes a new connection to the pool member or node that has the least number of active connections. <i>Note: If the OneConnect feature is enabled, the Least Connections methods do not include idle connections in the calculations when selecting a pool member or node. The Least Connections methods use only active connections in their calculations.</i>	The Least Connections methods function best in environments where the servers have similar capabilities. Otherwise, some amount of latency can occur. For example, consider the case where a pool has two servers of differing capacities, A and B. Server A has 95 active connections with a connection limit of 100, while server B has 96 active connections with a much larger connection limit of 500. In this case, the Least Connections method selects server A, the server with the lowest number of active connections, even though the server is close to reaching capacity. If you have servers with varying capacities, consider using the Weighted Least Connections methods instead.
Weighted Least Connections (member) Weighted Least	Like the Least Connections methods, these load balancing methods select pool members or nodes based on the number of active connections. However, the Weighted Least Connections methods also base their selections on server capacity. The Weighted Least Connections (member) method specifies that the system uses the value you specify in Connection Limit to establish a proportional algorithm for each pool member. The system bases the load	Weighted Least Connections methods work best in environments where the servers have differing capacities. For example, if two servers have the same number of active connections but one server has more capacity than the other, Local Traffic Manager calculates the

Method	Description	When to use
Connections (node)	<p>balancing decision on that proportion and the number of current connections to that pool member. For example, member_a has 20 connections and its connection limit is 100, so it is at 20% of capacity. Similarly, member_b has 20 connections and its connection limit is 200, so it is at 10% of capacity. In this case, the system select selects member_b. This algorithm requires all pool members to have a non-zero connection limit specified. The Weighted Least Connections (node) method specifies that the system uses the value you specify in the node's Connection Limit setting and the number of current connections to a node to establish a proportional algorithm. This algorithm requires all nodes used by pool members to have a non-zero connection limit specified. If all servers have equal capacity, these load balancing methods behave in the same way as the Least Connections methods.</p> <hr/> <p>Note: <i>If the OneConnect feature is enabled, the Weighted Least Connections methods do not include idle connections in the calculations when selecting a pool member or node. The Weighted Least Connections methods use only active connections in their calculations.</i></p> <hr/>	percentage of capacity being used on each server and uses that percentage in its calculations.
Observed (member) Observed (node)	With the Observed methods, nodes are ranked based on the number of connections. The Observed methods track the number of Layer 4 connections to each node over time and create a ratio for load balancing.	The need for the Observed methods is rare, and they are not recommended for large pools.
Predictive (member) Predictive (node)	The Predictive methods use the ranking methods used by the Observed methods, where servers are rated according to the number of current connections. However, with the Predictive methods, Local Traffic Manager analyzes the trend of the ranking over time, determining whether a node's performance is currently improving or declining. The servers with performance rankings that are currently improving, rather than declining, receive a higher proportion of the connections.	The need for the Predictive methods is rare, and they are not recommend for large pools.
Least Sessions	<p>The Least Sessions method selects the server that currently has the least number of entries in the persistence table. Use of this load balancing method requires that the virtual server reference a type of profile that tracks persistence connections, such as the Source Address Affinity or Universal profile type.</p> <hr/> <p>Note: <i>The Least Sessions methods are incompatible with cookie persistence.</i></p> <hr/>	The Least Sessions method works best in environments where the servers or other equipment that you are load balancing have similar capabilities.
Ratio Least Connections	The Ratio Least Connections methods cause the system to select the pool member according to the ratio of the number of connections that each pool member has active.	

About priority-based member activation

Priority-based member activation is a feature that allows you to categorize pool members into priority groups, so that pool members in higher priority groups accept traffic before pool members in lower priority groups. The priority-based member activation feature has two configuration settings:

Priority group activation

For the priority group activation setting, you specify the minimum number of members that must remain available in each priority group in order for traffic to remain confined to that group. The allowed value for this setting ranges from 0 to 65535. Setting this value to **0** disables the feature (equivalent to using the default value of **Disabled**).

Priority group

When you enable priority group activation, you also specify a priority group for each member when you add that member to the pool. Retaining the default priority group value of 0 for a pool member means that the pool member is in the lowest priority group and only receives traffic when all pool members in higher priority groups are unavailable.

If the number of available members assigned to the highest priority group drops below the number that you specify, the BIG-IP® system distributes traffic to the next highest priority group, and so on.

For example, this configuration has three priority groups, 3, 2, and 1, with the priority group activation value (shown here as `min active members`) set to 2.

```
pool my_pool {
  lb_mode fastest
  min active members 2
  member 10.12.10.7:80 priority 3
  member 10.12.10.8:80 priority 3
  member 10.12.10.9:80 priority 3
  member 10.12.10.4:80 priority 2
  member 10.12.10.5:80 priority 2
  member 10.12.10.6:80 priority 2
  member 10.12.10.1:80 priority 1
  member 10.12.10.2:80 priority 1
  member 10.12.10.3:80 priority 1
}
```

Connections are first distributed to all pool members with priority 3 (the highest priority group). If fewer than two priority 3 members are available, traffic is directed to the priority 2 members as well. If both the priority 3 group and the priority 2 group have fewer than two members available, traffic is directed to the priority 1 group. The BIG-IP system continuously monitors the priority groups, and whenever a higher priority group once again has the minimum number of available members, the BIG-IP system limits traffic to that group.

Pool member features

A pool member consists of a server's IP address and service port number. An example of a pool member is `10.10.10.1:80`. Pool members have a number of features that you can configure when you create the pool.

Note: By design, a pool and its members always reside in the same administrative partition.

Ratio weights for pool members

When using a ratio-based load balancing method for distributing traffic to servers within a pool, you can assign a ratio weight to the corresponding pool members. The ratio weight determines the amount of traffic that the server receives. The ratio-based load balancing methods are: Ratio (node, member, and sessions), Dynamic Ratio (node and member), and Ratio Least Connections (node and member).

Priority group number

The Priority Group feature assigns a priority number to the pool member. Within the pool, traffic is then load balanced according to the priority number assigned to the pool member. For example, pool members assigned to group 3, instead of pool members in group 2 or group 1, normally receive all traffic. Thus, members that are assigned a high priority receive all traffic until the load reaches a certain level or some number of members in the group become unavailable. If either of these events occurs, some of the traffic goes to members assigned to the next higher priority group. This setting is used in tandem with the pool feature known as Priority Group Activation. You use the Priority Group Activation feature to configure the minimum number of members that must be available before Local Traffic Manager™ begins directing traffic to members in a lower priority group.

Connection limit

You can specify the maximum number of concurrent connections allowed for a pool member. Note that the default value of 0 (zero) means that there is no limit to the number of concurrent connections that the pool member can receive.

Connection rate limit

The maximum rate of new connections allowed for the pool member. When you specify a connection rate limit, the system controls the number of allowed new connections per second, thus providing a manageable increase in connections without compromising availability. The default value of 0 specifies that there is no limit on the number of connections allowed per second. The optimal value to specify for a pool member is between 300 and 5000 connections. The maximum value allowed is 100000.

Explicit monitor associations

Once you have associated a monitor with a pool, Local Traffic Manager automatically associates that monitor with every pool member, including those members that you add to the pool later. However, in some cases you might want the monitor for a specific pool member to be different from that assigned to the pool. In this case, you must specify that you want to explicitly associate a specific monitor with the individual pool member. You can also prevent Local Traffic Manager from associating any monitor with that pool member.

Explicit monitor association for a pool member

Local Traffic Manager contains many different monitors that you can associate with a pool member, depending on the type of traffic you want to monitor. You can also create your own custom monitors and associate them with pool members. The only monitor types that are not available for associating with pool members are monitors that are specifically designed to monitor nodes and not pools or pool members. These monitor types are:

- ICMP
- TCP Echo
- Real Server
- SNMP DCA
- SNMP DCA Base
- WMI

Multiple monitor association for a pool member

Local Traffic Manager allows you to associate more than one monitor with the same server. You can:

- Associate more than one monitor with a member of a single pool. For example, you can create monitors `http1`, `http2`, and `http3`, where each monitor is configured differently, and associate all three monitors with the same pool member. In this case, the pool member is marked as down if any of the checks is unsuccessful.
- Assign one IP address and service to be a member of multiple pools. Then, within each pool, you can associate a different monitor with that pool member. For example, suppose you assign the server `10.10.10:80` to three separate pools: `my_pool1`, `my_pool2`, and `my_pool3`. You can then associate all three custom HTTP monitors to that same server (one monitor per pool). The result is that Local Traffic Manager uses the `http1` monitor to check the health of server `10.10.10:80` in pool

`my_pool1`, the `http2` monitor to check the health of server `10.10.10.:80` in pool `my_pool2`, and the `http3` monitor to check the health of server `10.10.10:80` in pool `my_pool3`.

You can make multiple-monitor associations either at the time you add the pool member to each pool, or by later modifying a pool member's properties.

Availability requirement

You can specify a minimum number of health monitors. Before Local Traffic Manager can report the pool member as being in an `up` state, this number of monitors, at a minimum, must report a pool member as being available to receive traffic.

About pool member state

You can enable or disable individual pool members. When you enable or disable a pool member, you indirectly set the value of the pool member's `State` property, in the following way:

- `Enable` sets the `State` property of the pool member to `Enabled`.
- `Disable` sets the `State` property of the pool member to `Disabled`.

Note that the difference between a disabled pool member and a pool member that a monitor reports as `down` is that a disabled pool member continues to process persistent and active connections. Conversely, a pool member reported as `down` processes no connections whatsoever.

The status icons on the pool-member list screen and properties screen indicate whether a pool member is currently enabled or disabled.

Pool and pool member status

An important part of managing pools and pool members is viewing and understanding the status of a pool or pool member at any given time. The BIG-IP Configuration utility indicates status by displaying one of several icons, distinguished by shape and color, for each pool or pool member:

- The shape of the icon indicates the status that the monitor has reported for that pool or pool member. For example, a circle-shaped icon indicates that the monitor has reported the pool member as being `up`, whereas a diamond-shaped icon indicates that the monitor has reported the pool member as being `down`.
- The color of the icon indicates the actual status of the node itself. For example, a green shape indicates that the node is `up`, whereas a red shape indicates that the node is `down`. A black shape indicates that user-intervention is required.

At any time, you can determine the status of a pool. The status of a pool is based solely on the status of its members. Using the BIG-IP Configuration utility, you can find this information by viewing the `Availability` property of the pool. You can also find this information by displaying the list of pools and checking the `Status` column.

Profiles

Introduction to profiles

Profiles are a configuration tool that you can use to affect the behavior of certain types of network traffic. More specifically, a *profile* is an object that contains settings with values, for controlling the behavior of a particular type of network traffic, such as HTTP connections. Profiles also provide a way for you to enable connection and session persistence, and to manage client application authentication.

By default, Local Traffic Manager™ provides you with a set of profiles that you can use as is. These default profiles contain various settings with default values that define the behavior of different types of traffic. If you want to change those values to better suit the needs of your network environment, you can create a custom profile. A *custom profile* is a profile derived from a default profile and contains values that you specify.

You can use profiles in the following ways:

- You can use the default profiles, which means that you do not need to actively configure any profile settings. Local Traffic Manager uses them to automatically direct the corresponding traffic types according to the values specified in the those profiles.
- You can create a custom profile, using the default profile as the parent profile, modifying some or all of the values defined in that profile.
- You can create a custom profile to use as a parent profile for other custom profiles.

After configuring a profile, you associate the profile with a virtual server. The virtual server then processes traffic according to the values specified in the profile. Using profiles enhances your control over managing network traffic, and makes traffic-management tasks easier and more efficient.

You can associate multiple profiles with a single virtual server. For example, you can associate a TCP profile, an SSL profile, and an HTTP profile with the same virtual server.

Profile types

Local Traffic Manager™ provides several types of profiles. While some profile types correspond to specific application services, such as HTTP, SSL, and FTP, other profiles pertain to traffic behaviors applicable to basic protocols such as TCP and UDP, and authentication protocols such as LDAP, RADIUS, and Kerberos. Also included are profiles specifically for different types of session persistence.

Default profiles

Local Traffic Manager™ includes one or more default profiles for each profile type. A *default profile* is a system-supplied profile that contains default values for its settings. An example of a default profile is the `http default` profile. You can use a default profile in several ways:

- You can use a default profile as is. You simply configure your virtual server to reference the default profile.
- You can modify the default profile settings (not recommended). When you modify a default profile, you lose the original default profile settings. Thus, any custom profiles you create in the future that are based on that default profile inherit the modified settings.
- You can create a custom profile, based on the default profile (recommended). This allows you to preserve the default profile, and instead configure personalized settings in the custom profile. Custom profiles inherit some of the setting values of a parent profile that you specify. After creating a custom profile, you can configure your virtual server to reference the custom profile instead of the default profile.

Note: *You can modify a default profile, but you cannot create or delete a default profile.*

Local Traffic Manager provides a default profile that you can use as is for each type of traffic. A *default profile* includes default values for any of the properties and settings related to managing that type of traffic. To implement a default profile, you simply assign the profile to a virtual server. You are not required to configure the setting values.

Custom and parent profiles

A *custom profile* is a profile that is derived from a parent profile that you specify. A *parent profile* is a profile from which your custom profile inherits its settings and their default values.

When creating a custom profile, you have the option of changing one or more setting values that the profile inherited from the parent profile. In this way, you can pick and choose which setting values you would like to change and which ones you would like to retain. An advantage to creating a custom profile is that by doing so, you preserve the setting values of the parent profile.

Note: *If you do not specify a parent profile when you create a custom profile, Local Traffic Manager™ automatically assigns a related default profile as the parent profile. For example, if you create a custom HTTP type of profile, the default parent profile is the default profile `http`.*

If you do not want to use a default profile as is or change its settings, you can create a custom profile. Creating a custom profile and associating it with a virtual server allows you to implement your own specific set of traffic-management policies.

When you create a custom profile, the profile is a child profile and automatically inherits the setting values of a parent profile that you specify. However, you can change any of the values in the child profile to better suit your needs.

If you do not specify a parent profile, Local Traffic Manager uses the default profile that matches the type of profile you are creating.

Important: *When you create a custom profile, the BIG-IP® system places the profile into your current administrative partition.*

Important: *Within the BIG-IP Configuration utility, each profile creation screen contains a check box to the right of each profile setting. When you check a box for a setting and then specify a value for that setting, the profile then retains that value, even if you change the corresponding value in the parent profile later. Thus, checking the box for a setting ensures that the parent profile never overwrites that value through inheritance.*

Once you have created a custom profile, you can adjust the settings of your custom profile later if necessary. If you have already associated the profile with a virtual server, you do not need to perform that task again.

The default profile as the parent profile

A typical profile that you can specify as a parent profile when you create a custom profile is a default profile. For example, if you create a custom TCP-type profile called `my_tcp_profile`, you can use the default profile `tcp` as the parent profile. In this case, Local Traffic Manager™ automatically creates the profile `my_tcp_profile` so that it contains the same settings and default values as the default profile `tcp`. The new custom profile thus inherits its settings and values from its parent profile. You can then retain or change the inherited setting values in the custom profile to suit your needs.

The custom profile as the parent profile

When creating a custom profile, you can specify another custom profile, rather than the default profile, as the parent profile. The only restriction is that the custom profile that you specify as the parent must be of the same profile type as the profile you are deriving from the parent. Once you have created the new custom profile, its settings and default values are automatically inherited from the custom profile that you specified as the parent.

For example, if you create a profile called `my_tcp_profile2`, you can specify the custom profile `my_tcp_profile` as its parent. The result is that the default setting values of profile `my_tcp_profile2` are those of its parent profile `my_tcp_profile`.

If you subsequently modify the settings of the parent profile (`my_tcp_profile`), Local Traffic Manager™ automatically propagates those changes to the new custom profile.

For example, if you create the custom profile `my_tcp_profile` and use it as a parent profile to create the custom profile `my_tcp_profile2`, any changes you make later to the parent profile `my_tcp_profile` are automatically propagated to profile `my_tcp_profile2`. Conversely, if you modify any of the settings in the new custom profile (in our example, `my_tcp_profile2`), the new custom profile does not inherit values from the parent profile for those particular settings that you modified.

Profiles and virtual servers

Once you have created a profile for a specific type of traffic, you implement the profile by associating that profile with one or more virtual servers.

You associate a profile with a virtual server by configuring the virtual server to reference the profile. Whenever the virtual server receives that type of traffic, Local Traffic Manager™ applies the profile settings to that traffic, thereby controlling its behavior. Thus, profiles not only define capabilities per network traffic type, but also ensure that those capabilities are available for a virtual server.

Because certain kinds of traffic use multiple protocols and services, users often create multiple profiles and associate them with a single virtual server.

For example, a client application might use the TCP, SSL, and HTTP protocols and services to send a request. This type of traffic would therefore require three profiles, based on the three profile types TCP, Client SSL, and HTTP.

Each virtual server lists the names of the profiles currently associated with that virtual server. You can add or remove profiles from the profile list, using the BIG-IP Configuration utility. Note that Local Traffic

Manager (LTM[®]) has specific requirements regarding the combinations of profile types allowed for a given virtual server.

In directing traffic, if a virtual server requires a specific type of profile that does not appear in its profile list, Local Traffic Manager uses the relevant default profile, automatically adding the profile to the profile list. For example, if a client application sends traffic over TCP, SSL, and HTTP, and you have assigned SSL and HTTP profiles only, LTM automatically adds the default profile `tcp` to its profile list.

At a minimum, a virtual server must reference a profile, and that profile must be associated with a UDP, FastL4, Fast HTTP, or TCP profile type. Thus, if you have not associated a profile with the virtual server, Local Traffic Manager adds a `udp`, `fastl4`, `fasthttp`, or `tcp` default profile to the profile list.

The default profile that Local Traffic Manager chooses depends on the configuration of the virtual server's protocol setting. For example, if the protocol setting is set to UDP, Local Traffic Manager adds the `udp` profile to its profile list.

HTTP Profiles

Introduction to HTTP profiles

BIG-IP® Local Traffic Manager™ offers several features that you can use to intelligently control your application layer traffic. Examples of these features are the insertion of headers into HTTP requests and the compression of HTTP server responses.

These features are available through various configuration profiles. A *profile* is a group of settings, with values, that correspond to HTTP traffic. A profile defines the way that you want the BIG-IP system to manage HTTP traffic.

You can configure an HTTP profile to ensure that HTTP traffic management suits your specific needs. You can configure the profile settings either when you create a profile or after you create the profile by modifying the profile's settings. For all profile settings, you can specify values where none exist, or modify any default values to suit your needs. The BIG-IP system also includes default profiles that you can use as is, if you do not want to create a custom profile.

To manage HTTP traffic, you can use any of these profile types:

- HTTP (Hypertext Transfer Protocol)
- HTTP Compression
- Web Acceleration

In addition to the HTTP profiles, Local Traffic Manager includes other features to help you manage your application traffic, such as health monitors for checking the health of HTTP and HTTPS services, and iRules® for querying or manipulating header or content data.

General HTTP properties

There are a few general settings that you can configure to create a basic HTTP type of profile that uses most of the default settings.

Proxy mode

The HTTP profile provides three proxy modes: Reverse, Explicit, and Transparent. You can configure a custom HTTP profile that uses a specific proxy mode, and assign the custom HTTP profile to a virtual server to manage proxying of HTTP traffic, as necessary.

Proxy Mode	Description
Reverse	Default. You can specify the Reverse Proxy Mode to enable the BIG-IP® system to manage responses from multiple servers.
Explicit	The Explicit Proxy Mode enables the BIG-IP system to handle HTTP proxy requests and function as a gateway. By configuring browser traffic to use the proxy, you can control whether

Proxy Mode	Description
	to allow or deny a requested connection, based on configured policies. The Explicit Proxy Mode requires a DNS resolver, specified in the Explicit Proxy area of the screen.
Transparent	The Transparent Proxy Mode enables the BIG-IP system to forward invalid HTTP traffic to a specified server, instead of dropping the connection. By configuring an HTTP profile to forward invalid HTTP traffic, you can manage various atypical service provider scenarios, such as HTTP traffic from non-browser clients that function as web browsers.

Parent profile

Every profile that you create is derived from a parent profile. You can use the default `http` profile as the parent profile, or you can use another HTTP profile that you have already created.

HTTP settings

There are several general settings that you can configure to create an HTTP type of profile.

Basic Auth Realm

The Basic Auth Realm setting provides a quoted string for the basic authentication realm. The BIG-IP® system sends this string to a client whenever authorization fails.

Fallback host

Another feature that you can configure within an HTTP profile is HTTP redirection. HTTP redirection allows you to redirect HTTP traffic to another protocol identifier, host name, port number, or URI path.

Redirection to a fallback host occurs if all members of the targeted pool are unavailable, or if a selected pool member is unavailable. (The term *unavailable* refers to a member being disabled, marked as down, or having exceeded its connection limit.) When one or more pool members are unavailable, Local Traffic Manager™ can redirect the HTTP request to the fallback host, with the HTTP reply `Status Code 302 Found`.

Although HTTP redirection often occurs when the system generates an `LB_FAILED` iRule event, redirection can also occur without the occurrence of this event, such as when:

- The selected node sends an `RST` after a `TCP 3WHS` has completed, but before the node has sent at least a full response header.
- Local Traffic Manager finds the selected node to be unreachable while receiving the body portion of a request or a pipelined request.

When configuring Local Traffic Manager to redirect HTTP traffic to a fallback host, you can specify an IP address or a fully-qualified domain name (FQDN). The value that you specify becomes the value of the `Location` header that the server sends in the response. For example, you can specify a redirection as `http://redirector.siterequest.com`.

Fallback error codes

In addition to redirecting traffic when a target server becomes unavailable, you can also specify the HTTP error codes from server responses that should trigger a redirection to the fallback host. Typical error codes to specify are 500, 501, and 502.

Headers in HTTP requests

You can insert headers into HTTP requests. The HTTP header being inserted can include a client IP address. Including a client IP address in an HTTP header is useful when a connection goes through a secure network address translation (SNAT) and you need to preserve the original client IP address.

The format of the header insertion that you specify is generally a quoted string. Alternatively, however, you can insert a Tools Command Language (Tcl) expression into a header that dynamically resolves to the preferred value. When you assign the configured HTTP profile to a virtual server, Local Traffic Manager™ then inserts the header specified in the profile into any HTTP request that the BIG-IP® system sends to a pool or pool member.

Note: In addition to inserting a string such as a client IP address into an HTTP request, you can configure Local Traffic Manager to insert SSL-related headers into HTTP requests. Examples are: client certificates, cipher specifications, and client session IDs. To insert these types of headers, you must create an iRule.

Content erasure from HTTP headers

You can configure a profile to erase the contents of a header from an HTTP request that is being sent from a client to a server. With this feature, you can erase header content from HTTP requests before forwarding the requests over the network. Such headers might contain sensitive information, such as user IDs or telephone numbers, that must be erased before the information is forwarded.

When you use this setting, Local Traffic Manager™ erases the contents of the specified header and replaces that content with blank spaces. The header itself is retained.

Note: This feature does not apply to HTTP responses being sent from a server to a client.

The client header with the contents to be erased must be specified as a quoted string.

Headers in an HTTP response

You can specify any headers within an HTTP response that you want the BIG-IP® system to allow. If you are specifying more than one header, separate the headers with a blank space. For example, if you type the string `Content-Type Set-Cookie Location`, the BIG-IP system then allows the headers `Content-Type`, `Set-Cookie`, and `Location`.

Response chunking

Sometimes, you might want to inspect and/or modify HTTP application data, such as compressing the content of an HTTP response. Such inspections or modifications require that the response be *unchunked*,

that is, not in chunked encoding. Using the Response Chunking feature, Local Traffic Manager™ can unchunk a chunked response before performing an action on that response.

Possible chunking behaviors of Local Traffic Manager

The BIG-IP® system takes specific action on a response depending on whether the response is chunked or unchunked.

Action	Original response is chunked	Original response is unchunked
Unchunk	Local Traffic Manager™ unchunks the response and processes the HTTP content, and passes the response on as unchunked. The connection closes when all data is sent to the client as indicated by the Connection: Close header.	Local Traffic Manager processes the HTTP content and passes the response on untouched.
Rechunk	Local Traffic Manager unchunks the response, processes the HTTP content, re-adds the chunk trailer headers, and then passes the response on as chunked. Any chunk extensions are lost.	Local Traffic Manager adds transfer encoding and chunking headers on egress.
Selective	Same as Rechunk.	Local Traffic Manager processes the HTTP content and then passes the response on untouched.
Preserve	Local Traffic Manager leaves the response chunked, processes the HTTP content, and passes the response on untouched. Note that if HTTP compression is enabled, Local Traffic Manager does not compress the response.	Local Traffic Manager processes the HTTP content and then passes the response on untouched.

OneConnect transformations

You can enable or disable part of the OneConnect™ feature, for HTTP/1.0 connections only. When this setting is enabled and a OneConnect profile is assigned to the virtual server, the setting performs Connection header transformations, for the purpose of keeping a client connection open. More specifically:

1. A client sends an HTTP/1.0 request.
2. The server sends a response, which initially includes a Connection: Close header.
3. Local Traffic Manager™ transforms the Connection: Close header to Connection: Keep-Alive.
4. Through use of the OneConnect profile, the server-side connection detaches, goes into the pool of available server-side connections used for servicing other requests, and eventually closes. This process is hidden from the client.
5. The client-side connection remains open, operating under the assumption that the server-side connection is still open and therefore able to accept additional requests from that client.

Note: For this feature to take effect, you must also configure a OneConnect™ profile, which enables connection pooling.

Rewrites of HTTP redirections

Sometimes, a client request is redirected from the HTTPS protocol to the HTTP protocol, which is a non-secure channel. If you want to ensure that the request remains on a secure channel, you can cause the redirection to be rewritten so that it is redirected back to the HTTPS protocol.

To enable Local Traffic Manager™ to rewrite HTTP redirections, you use the Rewrite Redirections setting to specify the way that you want the system to handle URIs during the rewrite.

Note that the rewriting of any redirection takes place only in the HTTP `Location` header of the redirection response, and not in any content of the redirection.

Possible values

When configuring Local Traffic Manager to rewrite HTTP redirections, you specify one of these values:

None

The system does not rewrite any redirections. This is the default value.

All

The system rewrites the URI in all HTTP redirect responses. In this case, the system rewrites those URIs as if they matched the originally-requested URIs.

Matching

The system rewrites the URI in any HTTP redirect responses that match the request URI (minus an optional trailing slash).

Nodes

The system rewrites the hidden node IP address to a virtual server address, and rewrites the port number. You choose this value when the virtual server is not configured with a Client SSL profile (that is, when the virtual server is configured to process plain HTTP traffic only).

Note: For values *All*, *Matching*, and *Nodes*, the system always hides the node IP address. Also, the system hides the node IP address independently of the protocol rewrite, with no regard to the protocol in the original redirection.

Examples of rewriting HTTP redirections with the system listening on port 443

This table shows examples of how redirections of client requests are transformed when the BIG-IP system is listening on port 443, and the Rewrite Redirections setting is enabled.

Original Redirection	Rewrite of Redirection
http://www.myweb.com/myapp/	https://www.myweb.com/myapp/
http://www.myweb.com:8080/myapp/	https://www.myweb.com/myapp/

Examples of rewriting HTTP redirections with the system listening on port 4443

This table shows examples of how redirections of client requests are transformed when the BIG-IP system is listening on port 443, and the Rewrite Redirections setting is enabled.

Original Redirection	Rewrite of Redirection
http://www.myweb.com/myapp/	https://www.myweb.com:4443/myapp/
http://www.myweb.com:8080/myapp/	https://www.myweb.com:4443/myapp/

Cookie encryption and decryption

You can use the BIG-IP Configuration utility to encrypt one or more cookies that the BIG-IP[®] system sends to a client system. When the client sends the encrypted cookie back to the BIG-IP system, the system decrypts the cookie.

X-Forwarded-For header insertion

When using connection pooling, which allows clients to make use of existing server-side connections, you can insert the `XForwarded For` header into a request. When you configure Local Traffic Manager[™] to insert this header, the target server can identify the request as coming from a client other than the client that initiated the connection. The default setting is `Disabled`.

Maximum columns for linear white space

You can specify the maximum number of columns allowed for a header that is inserted into an HTTP request.

Linear white space separators

You can specify the separator that Local Traffic Manager[™] should use between HTTP headers when a header exceeds the maximum width specified by the LWS Maximum Columns feature.

Maximum number of requests

You can specify the maximum number of requests that the system allows for a single Keep-Alive connection. When the specified limit is reached, the final response contains a `Connection: close` header, which is followed by the closing of the connection. The default setting is 0, which in this case means that the system allows an infinite number of requests per Keep-Alive connection.

Proxy Via headers

You can configure the BIG-IP[®] system to remove, preserve, or append `Via` headers in HTTP client requests, HTTP server responses, or both.

Overview: Using Via headers

`Via` headers provide useful information about intermediate routers that can be used in network analysis and troubleshooting.

About using `Via` headers in requests and responses

The `Via` header, configured in an HTTP profile, provides information about each intermediate router that forwards a message. Intermediate routers between a client and an origin web server use the `Via` header to indicate intermediate protocols and recipients. This information can be used for the following tasks:

- Identifying the intermediate routers that forward messages.
- Identifying the protocols for intermediate routers.

About identifying intermediate routers with a Via header

The `Via` header, configured in an HTTP profile, concatenates information for each router in a response or request, separated by commas. For example, the following `Via` header includes two routers, with each router comprising the required protocol and address:

```
Via: 1.1 wa.www.siterequest1.com, 1.1 wa.www.siterequest2.com
```

When a client initiates a request with a `Via` header to an origin web server, the origin web server returns a response with a `Via` header often following a similar path. For example, a `Via` header router sequence for the request would be 1, 2, 3, and the router sequence for the client's response would be 3, 2, 1.

The inverse is true when an origin web server initiates a response with a `Via` header to a client. For example, a `Via` header router sequence for a response would be 1, 2, 3, and the router sequence for the client's request would be 3, 2, 1.

About identifying protocols for intermediate routers with a Via header

You can identify specific protocols and versions of protocols for intermediate routers by using a `Via` header, configured in an HTTP profile. When a client sends a request to an origin web server, the header information is concatenated for each intermediate router, including the protocol type (if different from HTTP) and version.

The `Via` header includes both required and optional protocol information about each router, as follows:

- The HTTP protocol name is optional; however, other protocol names are required.
- The protocol version of the message is required, which for HTTP is 1.0, 1.1, and so on.
- The host name is required. For privacy purposes, however, an alias can replace the actual host name.
- The port number associated with the host name is optional. When the port number is omitted, the default port applies.
- A comment describing the router is optional, and includes whatever string you specify in the **Send Proxy Via Header Host Name** field, by selecting **Append** in the list for **Send Proxy Via Header In Request** or **Send Proxy Via Header In Response**.

***Note:** If you prefer to replace the host name with another string, instead of appending a string to the `Via` header, you must use an `iRule` or the command line.*

Because the `Via` header includes the protocol name and version, applications are able to acquire this information for the various intermediate routers and use it, as necessary.

Via Header settings

This table describes controls and strings for **Via Header** settings in an HTTP profile.

Control	Default	Description
Send Proxy Via Header In Request	Remove	<p>Specifies whether to Remove, Preserve, or Append <code>Via</code> headers included in a client request to an origin web server.</p> <ul style="list-style-type: none"> • Remove. The BIG-IP® system deletes the <code>Via</code> header from the client request. • Preserve. The BIG-IP system includes the <code>Via</code> header in the client request to the origin web server.

Control	Default	Description
		<ul style="list-style-type: none"> • Append. The BIG-IP system appends the string specified in the Send Proxy Via Header In Host Name field to the Via header in the client request to the origin web server.
Send Proxy Via Header In Response	Remove	<p>Specifies whether to Remove, Preserve, or Append <code>Via</code> headers included in an origin web server response to a client.</p> <ul style="list-style-type: none"> • Remove. The BIG-IP system deletes the <code>Via</code> header from the origin web server response. • Preserve. The BIG-IP system includes the <code>Via</code> header in the origin web server response to the client. • Append. The BIG-IP system appends the string specified in the Send Proxy Via Header In Host Name field to the Via header in the origin web server response to the client.
Send Proxy Via Header Host Name	None	<p>Specifies a string to append as a comment when sending a <code>Via</code> header in a request to an origin web server or in a response to a client.</p> <hr/> <p><i>Note: If you prefer to replace the host name with another string, instead of appending a string to the <code>Via</code> header, you must use an <code>iRule</code> or the <code>command line</code>.</i></p> <hr/>

X-Forwarded-For header acceptance

This setting enables or disables trusting the client IP address, and statistics from the client IP address, based on the request's X-Forwarded-For (XFF) headers, if they exist.

Alternate X-Forwarded-For headers

Specifies alternative XFF headers instead of the default X-Forwarded-For header. If you are specifying more than one alternative XFF header, separate the alternative XFF headers with a blank space, such as `client1 proxyserver 129.78.138.66`.

Server agent name

When you create an HTTP profile, you can specify the string used as the server name in traffic generated by the BIG-IP® system. The default value is `BigIP`.

Enforcement settings

There are some settings related to enforcement that you can configure to create an HTTP type of profile.

Allow truncated redirects

The Allow Truncated Redirect setting determines the way in which the BIG-IP® system passes through traffic, when a redirect that lacks the trailing carriage-return and line-feed pair at the end of the headers is parsed. The default is Disabled, which silently drops the invalid HTTP request.

Maximum header size

This setting specifies the maximum size in bytes that the BIG-IP® system allows for all HTTP request headers combined, including the request line. If the combined headers length in bytes in a client request exceeds this value, the system stops parsing the headers and resets the TCP connection. The default value is 32,768 bytes.

Oversize client headers

The **Oversize Client Headers** setting determines the way in which the BIG-IP® system passes through HTTP traffic when the **Maximum Header Size** value is exceeded by the client. The default is disabled, which rejects the connection.

Note: This feature is only available on the HTTP profile when you set the proxy mode feature to **Transparent**.

Oversize server headers

The **Oversize Server Headers** setting determines the way in which the BIG-IP® system passes through HTTP traffic when the **Maximum Header Size** value is exceeded by the server. The default is disabled, which rejects the connection.

Note: This feature is only available on the HTTP profile when you set the proxy mode feature to **Transparent**.

Maximum header count

The Maximum Header Count setting determines the maximum number of headers in an HTTP request or response that the BIG-IP® system accepts. If a client or server sends a request or response with the number of headers exceeding the specified value, then the connection is dropped. The default value is 64.

Excess client headers

The **Excess Client Headers** setting specifies the way in which the BIG-IP® system passes through HTTP traffic when the **Maximum Header Count** value is exceeded by the client. The default is disabled, which rejects the connection.

Note: This feature is only available on the HTTP profile when you set the proxy mode feature to **Transparent**.

Excess server headers

The **Excess Server Headers** setting specifies the way in which the BIG-IP® system passes through HTTP traffic when the **Maximum Header Count** value is exceeded by the server. The default is disabled, which rejects the connection.

***Note:** This feature is only available on the HTTP profile when you set the proxy mode feature to **Transparent**.*

Support for pipelining

Normally, a client cannot initiate a request until the previous request has received a response. HTTP/1.1 pipelining allows clients to initiate multiple requests even when prior requests have not received a response. Note, however, that each initiated request is still processed sequentially; that is, a request in the queue is not processed until the previous request has received a response.

By enabling support for pipelining on the BIG-IP® system, you remove the need to enable pipelining on the destination server itself. By default, this feature is enabled.

Unknown methods

The **Unknown Method** setting determines the way in which the BIG-IP® system manages HTTP traffic when an unknown HTTP method is parsed. You can configure the **Unknown Method** setting to allow, reject, or pass through the HTTP traffic. The default is to allow unknown methods.

Explicit proxy settings

when you set the proxy mode to **Explicit**, you must also configure the settings in the Explicit Proxy area of the HTTP profile.

DNS Resolver

The **DNS Resolver** setting specifies the DNS resolver to use for DNS inquiries handled by the virtual servers associated with the HTTP explicit forward proxy profile you are creating.

***Note:** This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**, in which case the setting is required. If no DNS resolver exists on the system, you can create one at **DNS > Caches > Cache List > Create**.*

Route Domain

You can configure an HTTP profile to specify the route domain that is used for outbound connect requests for the explicit forward proxy feature. The default route domain is 0.

Note: This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**.

Tunnel Name

The **Tunnel Name** setting specifies the tunnel that is used for outbound connect requests when the explicit forward proxy feature is used. Specifying a tunnel enables other virtual servers to receive connections initiated by the proxy service.

Note: This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**.

Host Names

The **Host Name** setting specifies the name of hosts that should not be proxied when an explicit forward proxy is used.

Note: This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**.

Default Connect Handling

The **Default Connect Handling** setting specifies the behavior of the forward explicit proxy service when handling outbound requests. By default, this setting is disabled.

- Enabled (checked) indicates that outbound requests are delivered directly, regardless of the presence of listening virtual servers.
- Disabled (check box cleared) indicates that outbound requests are delivered only if another virtual server is listening on the tunnel for the requested outbound connection. With this setting, virtual servers are required, and the system processes the outbound traffic before it leaves the device.

Note: This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**.

Connection Failed Message

You can configure an http explicit forward proxy profile to specify the message that appears when a connection failure occurs. You can include TCL expressions.

Note: This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**.

DNS Lookup Failed Message

You can configure an http explicit forward proxy profile to specify the message that appears when a DNS lookup failure occurs. You can include TCL expressions.

Note: This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**.

Bad Request Message

You can configure an http explicit forward proxy profile to specify the message that appears when a bad request occurs. You can include TCL expressions.

Note: This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**.

Bad Response Message

You can configure an http explicit forward proxy profile to specify the message that appears when a bad response occurs. You can include TCL expressions.

Note: This setting is available on the HTTP profile only when you set the proxy mode feature to **Explicit**.

sFlow settings

You can configure the HTTP profile to use sFlow technology to monitor traffic passing through the BIG-IP system.

Polling intervals

You can configure an HTTP profile to specify the maximum interval in seconds between two pollings. The default value is **Default**, which represents the value set on the System :: sFlow :: Global Settings :: http :: Properties screen. The initial default value is 10 seconds.

Sampling rates

You can configure an HTTP profile to specify the ratio of packets observed to the samples generated. For example, a sampling rate of 2000 specifies that the system randomly generates 1 sample for every 2000 packets observed. The default value is **Default**, which represents the value set on the System :: sFlow :: Global Settings :: http :: Properties screen. The initial default value is 1024 packets.

About HTTP compression profiles

HTTP compression reduces the amount of data to be transmitted, thereby significantly reducing bandwidth usage. All of the tasks needed to configure HTTP compression on the BIG-IP® system, as well as the compression software itself, are centralized on the BIG-IP system. The tasks needed to configure HTTP compression for objects in an Application Acceleration Manager module policy node are available in the Application Acceleration Manager, but an HTTP compression profile must be enabled for them to function.

When configuring the BIG-IP system to compress data, you can:

- Configure the system to include or exclude certain types of data.

- Specify the levels of compression quality and speed that you want.

You can enable the HTTP compression option by setting the **URI Compression** or the **Content Compression** setting of the **HTTP Compression** profile to **URI List** or **Content List**, respectively. This causes the BIG-IP system to compress HTTP content for any responses in which the values that you specify in the **URI List** or **Content List** settings of an HTTP profile match the values of the `Request-URI` or `Content-Type` response headers.

Exclusion is useful because some URI or file types might already be compressed. Using CPU resources to compress already-compressed data is not recommended because the cost of compressing the data usually outweighs the benefits. Examples of regular expressions that you might want to specify for exclusion are `.*\.pdf`, `.*\.gif`, or `.*\.html`.

Note: The string that you specify in the **URI List** or the **Content List** setting can be either a pattern string or a regular expression. List types are case-sensitive for pattern strings. For example, the system treats the pattern string `www.f5.com` differently from the pattern string `www.F5.com`. You can override this case-sensitivity by using the Linux `regex` command.

HTTP Compression profile options

You can use an HTTP Compression profile alone, or with the BIG-IP® Application Acceleration Manager, to reduce the amount of data to be transmitted, thereby significantly reducing bandwidth usage. The tasks needed to configure HTTP compression for objects in an Application Acceleration Manager policy node are available in the Application Acceleration Manager, but an HTTP Compression profile must be enabled for them to function.

URI compression

If you enable compression, you probably do not want Local Traffic Manager™ to compress every kind of server response. Therefore, you can instruct Local Traffic Manager (LTM®) to include in compression, or exclude from compression, certain responses that are specified in the URIs of client requests.

More specifically, you can type regular expressions to specify the types of server responses that you want Local Traffic Manager to include in, or exclude from, compression. For example, you can specify that you want LTM to compress all `.htm` responses by typing the regular expression `.*.htm`. LTM then compares that response type to the URI specified within each client request, and if the system finds a match, takes some action.

Note: The string that you specify can be either a pattern string or a regular expression. Note that list types are case-sensitive for pattern strings. For example, the system treats the pattern string `www.f5.com` differently from the pattern string `www.F5.com`. You can override this case-sensitivity by using the Linux `regex` command.

Content compression

If you enable compression, you probably do not want Local Traffic Manager™ to compress every kind of server response. Therefore, you can instruct Local Traffic Manager (LTM®) to include in compression, or exclude from compression, certain responses that are specified in the `Content-Type` header of server responses.

More specifically, you can type regular expressions to specify the types of server responses that you want Local Traffic Manager to include in, or exclude from, compression. For example, you can specify that you want LTM to compress all `.htm` responses by typing the regular expression `.*.htm`. Local Traffic Manager then compares that response type to the value of the `Content-Type` header specified within each server response, and if the system finds a match, takes some action.

***Note:** The string that you specify can be either a pattern string or a regular expression. Note that list types are case-sensitive for pattern strings. You can override this case-sensitivity by using the Linux `regex` command.*

Preferred compression methods

You can specify the compression method that you want Local Traffic Manager™ to use when compressing responses. The two possible compression methods are `gzip` and `deflate`.

Minimum content length for compression

When compression is enabled, you can specify the minimum length of a server response in uncompressed bytes that Local Traffic Manager™ requires for compressing that response. Local Traffic Manager (LTM®) finds the content length of a server response in the `Content-Length` header of the server response. Thus, if the content length specified in the response header is below the value that you assign for minimum content length, LTM does not compress the response. The length in bytes applies to content length only, not headers.

For example, using the default value of 1024, Local Traffic Manager compresses only those responses with HTTP content containing at least 1024 bytes.

Sometimes the `Content-Length` header does not indicate the content length of the response. In such cases, LTM compresses the response, regardless of size.

Compression buffer size

When compression is enabled, you can specify the maximum number of compressed bytes that Local Traffic Manager™ buffers before deciding whether or not to preserve a `Keep-Alive` connection and rewrite the `Content-Length` header.

For example, using the default value of 4096, Local Traffic Manager (LTM®) buffers up to 4096 bytes of compressed data before deciding whether or not to preserve the connection and rewrite the `Content-Length` header.

The Local Traffic Manager decision to rewrite the `Content-Length` header depends on whether response chunking is enabled (using the **Response Chunking** profile setting).

About the Vary header

When compression is enabled, the **Vary Header** setting inserts the `Vary: Accept-Encoding` header into a compressed server response. If the `Vary` header already exists in the response, Local Traffic Manager™ appends the value `Accept-Encoding` to that header.

The reason for inserting the `Vary: Accept-Encoding` header into a server response is to follow a recommendation by RFC2616, which states that the `Vary` header should be inserted into any cacheable

response that is subject to server-driven negotiation. Server responses that are subject to HTTP compression fall into this category.

If the **Vary Header** setting is disabled, Local Traffic Manager does not insert the `Vary` header into a server response.

To disable the `Vary` header, locate the **Vary Header** setting and clear the **Enabled** box.

Compression for HTTP/1.0 requests

The **HTTP/1.0 Requests** setting is included for backward compatibility, allowing HTTP compression for responses to HTTP/1.0 client requests. The default value for this setting is Disabled.

If this setting is set to Enabled, Local Traffic Manager™ only compresses responses in either of the following cases:

- When the server responds with a `Connection: close` header
- When the response content is no greater than the value of the **Compression Buffer Size** setting

To enable compression for HTTP/1.0 requests, locate the **HTTP/1.0 Requests** setting and select the check box.

About the Accept-Encoding header

Normally, when you enable HTTP compression, Local Traffic Manager™ strips out the `Accept-Encoding` header from the HTTP request. This causes Local Traffic Manager, instead of the target server, to perform the HTTP compression.

By default, the **Keep Accept Encoding** setting is disabled. If you want to allow the target server instead of Local Traffic Manager to perform the HTTP compression, simply enable this setting.

Browser workarounds

When you enable the **Browser Workarounds** setting, the system uses built-in workarounds for several common browser issues that occur when compressing content. The default setting is disabled (cleared). More specifically, enabling this setting prevents the system from compressing server responses when any of these conditions exists:

- The client browser is Netscape® version 4.0x.
- The client browser is Netscape version 4.x (that is, versions 4.10 and higher), and the `Content-Type` header of the server response is not set to **text/html** or **text/plain**.
- The client browser is Microsoft® Internet Explorer® (any version), the `Content-Type` header of the server response is set to either **text/css** or **application/x-javascript**, and the client connection uses SSL.
- The client browser is Microsoft® Internet Explorer® (any version), the `Content-Type` header of the server response is set to either **text/css** or **application/x-javascript**, and the `Cache-Control` header of the server response is set to **no-cache**.

About Web Acceleration profiles

When used by BIG-IP® Local Traffic Manager™ without other provisioned modules, the Web Acceleration profile uses basic default acceleration.

Web Acceleration profile settings

This table describes the Web Acceleration profile configuration settings and default values.

Setting	Value	Description
Name	No default	Specifies the name of the profile.
Parent Profile	Selected predefined or user-defined profile	Specifies the selected predefined or user-defined profile.
Partition / Path	Common	Specifies the partition and path to the folder for the profile objects.
Cache Size	100	This setting specifies the maximum size in megabytes (MB) reserved for the cache. When the cache reaches the maximum size, the system starts removing the oldest entries.
Maximum Entries	10000	Specifies the maximum number of entries that can be in the cache.
Maximum Age	3600	Specifies how long in seconds that the system considers the cached content to be valid.
Minimum Object Size	500	Specifies the smallest object in bytes that the system considers eligible for caching.
Maximum Object Size	50000	Specifies the largest object in bytes that the system considers eligible for caching.
URI Caching	Not Configured	Specifies whether the system retains or excludes certain Uniform Resource Identifiers (URIs) in the cache. The process forces the system either to cache URIs that typically are ineligible for caching, or to not cache URIs that typically are eligible for caching.
URI List	No default value	Specifies the URIs that the system either includes in or excludes from caching. <ul style="list-style-type: none"> Pin List. Lists the URIs for responses that you want the system to store indefinitely in the cache. Include List. Lists the URIs that are typically ineligible for caching, but the system caches them. When you add URIs to the Include List, the system caches the GET methods and other methods, including non-HTTP methods. Exclude List. Lists the URIs that are typically eligible for caching, but the system does not cache them. Include Override List. Lists URIs to cache, though typically, they would not be cached due to defined constraints, for example, the Maximum Object Size setting. The default value is none. URIs in

Setting	Value	Description
		the Include Override List list are cacheable even if they are not specified in the Include List.
Ignore Headers	All	<p>Specifies how the system processes client-side <code>Cache-Control</code> headers when caching is enabled.</p> <ul style="list-style-type: none"> • None. Specifies that the system honors all <code>Cache-Control</code> headers. • <code>Cache-Control:max-age</code>. Specifies that the system disregards a <code>Cache-Control:max-age</code> request header that has a value of <code>max-age=0</code>. • All. Specifies that the system disregards all <code>Cache-Control</code> headers.
Insert Age Header	Enabled	Specifies, when enabled, that the system inserts <code>Date</code> and <code>Age</code> headers in the cached entry. The <code>Date</code> header contains the current date and time on the BIG-IP® system. The <code>Age</code> header contains the length of time that the content has been in the cache.
Aging Rate	9	Specifies how quickly the system ages a cache entry. The aging rate ranges from 0 (slowest aging) to 10 (fastest aging).
AM Applications	No default	Lists enabled Application Acceleration Manager applications in the Enabled field and available applications in the Available field.

Other Application-Layer Profiles

Overview of other application-layer profiles

BIG-IP® Local Traffic Manager™ offers several features that you can use to intelligently control your application layer traffic. These features are available through various configuration profiles.

A *profile* is a group of settings, with values, that correspond to a specific type of traffic, such as FTP traffic. A profile defines the way that you want the BIG-IP system to manage that traffic type. After you configure the type of profile you need, you assign it to a virtual server.

To manage application layer traffic, you can use any of these profile types:

- FTP (File Transfer Protocol)
- DNS (Domain Name System)
- RTSP (Real Time Streaming Protocol)
- ICAP (Internet Content Adaptation Protocol)
- Request Adapt and Response Adapt
- Diameter
- RADIUS (Remote Authentication Dial-In User Service)
- SIP (Session Initiation Protocol)
- SMTP
- SMTPS
- iSession
- Rewrite
- XML
- SPDY
- FIX
- video Quality of Experience (QOE)

In addition to these application layer profiles, Local Traffic Manager includes other features to help you manage your application traffic, such as health monitors for checking the health of an FTP service, and iRules® for querying or manipulating header or content data. Additional profiles may be available with other modules.

About HTTP compression profiles

HTTP compression reduces the amount of data to be transmitted, thereby significantly reducing bandwidth usage. All of the tasks needed to configure HTTP compression on the BIG-IP® system, as well as the compression software itself, are centralized on the BIG-IP system. The tasks needed to configure HTTP compression for objects in an Application Acceleration Manager module policy node are available in the Application Acceleration Manager, but an HTTP compression profile must be enabled for them to function.

When configuring the BIG-IP system to compress data, you can:

- Configure the system to include or exclude certain types of data.
- Specify the levels of compression quality and speed that you want.

You can enable the HTTP compression option by setting the **URI Compression** or the **Content Compression** setting of the **HTTP Compression** profile to **URI List** or **Content List**, respectively. This causes the BIG-IP system to compress HTTP content for any responses in which the values that you specify in the **URI List** or **Content List** settings of an HTTP profile match the values of the `Request-URI` or `Content-Type` response headers.

Exclusion is useful because some URI or file types might already be compressed. Using CPU resources to compress already-compressed data is not recommended because the cost of compressing the data usually outweighs the benefits. Examples of regular expressions that you might want to specify for exclusion are `.*\.pdf`, `.*\.gif`, or `.*\.html`.

Note: The string that you specify in the **URI List** or the **Content List** setting can be either a pattern string or a regular expression. List types are case-sensitive for pattern strings. For example, the system treats the pattern string `www.f5.com` differently from the pattern string `www.F5.com`. You can override this case-sensitivity by using the Linux `regex` command.

HTTP Compression profile options

You can use an HTTP Compression profile alone, or with the BIG-IP® Application Acceleration Manager, to reduce the amount of data to be transmitted, thereby significantly reducing bandwidth usage. The tasks needed to configure HTTP compression for objects in an Application Acceleration Manager policy node are available in the Application Acceleration Manager, but an HTTP Compression profile must be enabled for them to function.

About Web Acceleration profiles

When used by BIG-IP® Local Traffic Manager™ without other provisioned modules, the Web Acceleration profile uses basic default acceleration.

Web Acceleration profile settings

This table describes the Web Acceleration profile configuration settings and default values.

Setting	Value	Description
Name	No default	Specifies the name of the profile.
Parent Profile	Selected predefined or user-defined profile	Specifies the selected predefined or user-defined profile.
Partition / Path	Common	Specifies the partition and path to the folder for the profile objects.
Cache Size	100	This setting specifies the maximum size in megabytes (MB) reserved for the cache. When the cache reaches the maximum size, the system starts removing the oldest entries.
Maximum Entries	10000	Specifies the maximum number of entries that can be in the cache.

Setting	Value	Description
Maximum Age	3600	Specifies how long in seconds that the system considers the cached content to be valid.
Minimum Object Size	500	Specifies the smallest object in bytes that the system considers eligible for caching.
Maximum Object Size	50000	Specifies the largest object in bytes that the system considers eligible for caching.
URI Caching	Not Configured	Specifies whether the system retains or excludes certain Uniform Resource Identifiers (URIs) in the cache. The process forces the system either to cache URIs that typically are ineligible for caching, or to not cache URIs that typically are eligible for caching.
URI List	No default value	<p>Specifies the URIs that the system either includes in or excludes from caching.</p> <ul style="list-style-type: none"> • Pin List. Lists the URIs for responses that you want the system to store indefinitely in the cache. • Include List. Lists the URIs that are typically ineligible for caching, but the system caches them. When you add URIs to the Include List, the system caches the GET methods and other methods, including non-HTTP methods. • Exclude List. Lists the URIs that are typically eligible for caching, but the system does not cache them. • Include Override List. Lists URIs to cache, though typically, they would not be cached due to defined constraints, for example, the Maximum Object Size setting. The default value is none. URIs in the Include Override List are cacheable even if they are not specified in the Include List.
Ignore Headers	All	<p>Specifies how the system processes client-side Cache-Control headers when caching is enabled.</p> <ul style="list-style-type: none"> • None. Specifies that the system honors all Cache-Control headers. • Cache-Control:max-age. Specifies that the system disregards a Cache-Control:max-age request header that has a value of max-age=0. • All. Specifies that the system disregards all Cache-Control headers.
Insert Age Header	Enabled	Specifies, when enabled, that the system inserts Date and Age headers in the cached entry. The Date header contains the current date and time on the BIG-IP® system. The Age header contains the length of time that the content has been in the cache.
Aging Rate	9	Specifies how quickly the system ages a cache entry. The aging rate ranges from 0 (slowest aging) to 10 (fastest aging).
AM Applications	No default	Lists enabled Application Acceleration Manager applications in the Enabled field and available applications in the Available field.

Web Acceleration Profile statistics description

This topic provides a description of Web Acceleration Profile statistics produced in tmsh.

Viewing Web Acceleration profile statistics

Statistics for the Web Acceleration Profile can be viewed in tmsh by using the following command.

```
tmsh show /ltm profile web-acceleration <profile_name>
```

Each statistic is described in the following example, appended after the asterisk (*).

```
-----
Ltm::Web Acceleration Profile: wa-profile-rbk
-----
Virtual Server Name          N/A

Cache
  Cache Size (in Bytes)      *current cache size for this profile
  Total Cached Items         *items in local cache
  Total Evicted Items        *count of items that have been evicted
                             from the local cache
  Inter-Stripe Size (in Bytes) *all the remote tmms cache
  Inter-Stripe Cached Items  *remote tmms count of cached items
  Inter-Stripe Evicted Items *remote tmms count of cached items
                             that have been evicted

Cache Hits/Misses           Count  Bytes
  Hits                      0      0  *tmm local cache served
                             response
  Misses (Cacheable)        0      0  *tmm local cache items that
                             match a rule set but are
                             not in cache
  Misses (Total)            0      0  *tmm local cache items that
                             either match or do not match
                             rule set and are not served
                             from cache
  Inter-Stripe Hits         0      0  *successful local access of
                             remote caches
  Inter-Stripe Misses       0      -  *unsuccessful local access
                             of remote caches
  Remote Hits               0      0  *successful remote tmms
                             accessing local cache
  Remote Misses             0      -  *unsuccessful remote tmms
                             accessing local cache
```

FTP profiles

Local Traffic Manager™ includes a profile type that you can use to manage File Transfer Protocol (FTP) traffic. You can tailor FTP profile settings to your specific needs. For those settings that have default values, you can retain those default settings or modify them. You can modify any settings either when you create the profile, or at any time after you have created it.

The Translate Extended value

Because IP version 6 addresses are not limited to 32 bits (unlike IP version 4 addresses), compatibility issues can arise when using FTP in mixed IP-version configurations.

By default, Local Traffic Manager automatically translates FTP commands when a client-server configuration contains both IP version 4 (IPv4) and IP version 6 (IPv6) systems. For example, if a client system running IPv4 sends the FTP PASV command to a server running IPv6, Local Traffic Manager automatically translates the PASV command to the equivalent FTP command for IPv6 systems, EPSV.

Local Traffic Manager translates the FTP commands `EPRV` and `PORT` in the same way.

Inherit Parent Profile

When you configure the BIG-IP® system to process FTP traffic, the FTP virtual server fully proxies the control channel, allowing you to use the optimization settings of the client-side and server-side TCP profiles assigned to the virtual server.

However, the profile settings of the FTP control channel are not passed down to the FTP data channel by default. Instead, the FTP data channel uses a Fast L4 flow, which is fully accelerated by Packet Velocity ASIC to maximize performance (on applicable hardware platforms). A data channel using Fast L4 cannot use the same full-proxy TCP optimizations that exist for the control channel.

To take advantage of these optimizations for the FTP data channel, you can enable the Inherit Parent Profile setting of the FTP profile. Enabling this setting disables Fast L4 for the FTP data channel, and instead allows the data channel to use the same TCP profile settings that the control channel uses.

Data Port

The Data Port setting allows the FTP service to run on an alternate port. The default data port is 20.

Security for FTP traffic

When the BIG-IP system includes a license for the BIG-IP® Application Security Manager™, you can enable a security scan for FTP traffic.

DNS profiles

You can create a custom DNS profile to enable various features such as converting IPv6-formatted addresses to IPv4 format, enabling DNS Express™, and enabling DNSSEC.

RTSP profiles

Local Traffic Manager® includes a profile type that you can use to manage Real Time Streaming Protocol (RTSP) traffic. *Real Time Streaming Protocol (RTSP)* is a protocol used for streaming-media presentations. Using RTSP, a client system can control a remote streaming-media server and allow time-based access to files on a server.

The RTSP profile in Local Traffic Manager supports these features:

- The setup of streaming media over UDP. In this case, the control connection opens the required ports to allow data to flow through the BIG-IP® system.
- Interleaved data over the control connection, essentially streaming media over TCP.
- Real Networks tunneling of RTSP over HTTP, through the RTSP port (554).

A common configuration for the RTSP profile is one that includes RTSP clients and media servers, as well as RTSP proxies to manage accounting and authentication tasks. In this proxied configuration, you most likely want the streaming media from the servers to pass directly to the client, bypassing the RTSP proxy servers.

To implement this configuration, you configure Local Traffic Manager by creating two virtual servers, one for processing traffic to and from the external network, and one for processing traffic to and from the internal network. For each virtual server, you assign a separate RTSP profile.

With this configuration:

- The RTSP profile on the external virtual server passes client IP address information to the RTSP profile on the internal virtual server.
- The RTSP profile on the internal virtual server extracts the client IP address information from the request, processes the media server's response, and opens the specified ports on the BIG-IP system. Opening these ports allows the streaming media to bypass the RTSP proxy servers as the data travels from the server to the client.

The client IP address information is stored in the Proxy Header setting that you specify in the RTSP profile.

ICAP profiles

You can configure one or more Internet Content Adaptation Protocol (ICAP) profiles when you want to use the BIG-IP® content adaptation feature for adapting HTTP requests and responses. This feature allows a BIG-IP virtual server to conditionally forward HTTP requests and HTTP responses to a pool of ICAP servers for modification, before sending a request to a web server or returning a response to the client system.

In a typical configuration, you create two ICAP profiles:

- You assign one of the profiles to a virtual server of type Internal that sends HTTP requests to a pool of ICAP servers.
- You assign the other profile to a virtual server of type Internal that sends HTTP responses to a pool of ICAP servers.

For more information on content adaptation for HTTP traffic, see the guide titled *BIG-IP® Local Traffic Manager: Implementations*, available on the AskF5™ knowledge base at <http://support.f5.com>.

Request Adapt and Response Adapt profiles

You can configure a Request Adapt or Response Adapt profile when you want to use the BIG-IP® content adaptation feature for adapting HTTP requests and responses. A Request Adapt or Response Adapt profile instructs an HTTP virtual server to send a request or response to a named virtual server of type Internal, for possible modification by an Internet Content Adaptation Protocol (ICAP) server.

For more information on content adaptation for HTTP traffic, see the guide titled *BIG-IP® Local Traffic Manager: Implementations*, available on the AskF5™ knowledge base at <http://support.f5.com>.

Diameter profiles

Local Traffic Manager™ includes a profile type that you can use to manage Diameter traffic. The *Diameter protocol* is an enhanced version of the Remote Authentication Dial-In User Service (RADIUS) protocol.

When you configure a Diameter type of profile, the BIG-IP® system can send client-initiated Diameter messages to load balancing servers. The BIG-IP system can also ensure that those messages persist on the servers.

RADIUS profiles

The BIG-IP® system includes a profile type that you can use to load balance Remote Authentication Dial-In User Service (RADIUS) traffic.

When you configure a RADIUS type of profile, the BIG-IP system can send client-initiated RADIUS messages to load balancing servers. The BIG-IP system can also ensure that those messages are persisted on the servers.

SIP profiles

Local Traffic Manager™ includes a services profile that you can use to manage Session Initiation Protocol (SIP) traffic. *Session Initiation Protocol* is an application-layer protocol that manages sessions consisting of multiple participants, thus enabling real-time messaging, voice, data, and video. A session can be a simple two-way telephone call or Instant Message dialogue, or a complex, collaborative, multi-media conference call that includes voice, data, and video.

SIP sessions, which are application level sessions, run through one of three Layer 4 protocols: SCTP, TCP, or UDP. The SIP profile configures how the system handles SIP sessions. The specified Layer 4 protocol profile configures the virtual server to open the required port to allow data to flow through the BIG-IP® system. When you assign a SIP profile to a virtual server, you can also assign an SCTP, TCP, or UDP profile to the server. If you do not assign one of these protocol profiles to the server, Local Traffic Manager automatically assigns one for you.

The SIP profile automatically configures the BIG-IP system to handle persistence for SIP sessions using Call-ID. The Call-ID is a globally unique identifier that groups together a series of messages, which are sent between communicating applications. You can customize how the system handles persistence for SIP sessions.

Maximum message size

Local Traffic Manager accepts incoming SIP messages that are 65535 bytes or smaller. If a SIP message exceeds this value, the system drops the connection.

Dialog snooping

Local Traffic Manager can snoop SIP dialog information and automatically forward SIP messages to the known SIP dialog. To forward these messages, you can specify a SIP proxy functional group.

Community string

You can specify the name of a proxy functional group. You use this setting in the case where you need multiple virtual servers, each referencing a SIP-type profile, and you want more than one of those profiles to belong to the same proxy functional group.

Connection termination criteria

Local Traffic Manager terminates a SIP connection when either the application that initiated the session (client) or the application that answered the initiated session (server) issues a BYE transaction. This is appropriate when a SIP session is running on UDP. However, if a SIP session is running on a SCTP or TCP connection, you can prevent the system from terminating the SIP connection.

SIP headers

An optional feature in a SIP profile is header insertion. You can specify whether Local Traffic Manager inserts Via, Secure Via, and Record-Route headers into SIP requests. When you assign the configured SIP profile to a virtual server, Local Traffic Manager then inserts the header specified in the profile into any SIP request that Local Traffic Manager sends to a pool or pool member.

SIP OneConnect

The SIP OneConnect™ feature allows connection flow reuse between inbound and outbound virtual servers for UDP connections. This feature addresses common SIP client behavior where source and destination ports are both 5060.

SIP OneConnect features a built-in dialog-aware behavior that addresses scenarios where the BIG-IP is the intermediary between more than two parties, creating an ambiguity between source and destination for the dialog. For example, in scenarios where an internal client initiates an outbound call using the wildcard virtual server to an external client that already has an existing flow on the inbound virtual server, the SIP OneConnect dialog-aware behavior correctly routes the response traffic.

Note: The SIP OneConnect dialog-aware feature is independent of the **Dialog Aware** setting in the SIP profile and is activated as long as two SIP profiles have identical community strings.

Activating SIP OneConnect

To activate the SIP OneConnect feature, type identical community strings in both SIP profiles used for the two virtual servers responsible for inbound and outbound SIP connections.

To disable the SIP OneConnect dialog-aware behavior and re-enable the default dialog-aware behavior, check the **Dialog Aware** setting when both community strings are set.

SMTP profiles

You can create an SMTP profile to secure SMTP traffic coming into the BIG-IP system. When you create an SMTP profile, BIG-IP® Protocol Security Manager™ provides several security checks for requests sent to a protected SMTP server:

- Verifies SMTP protocol compliance as defined in RFC 2821.
- Validates incoming mail using several criteria.
- Inspects email and attachments for viruses.
- Applies rate limits to the number of messages.
- Validates DNS SPF records.
- Prevents directory harvesting attacks.
- Disallows or allows some of the SMTP methods, such as VRFY, EXPN, and ETRN, that spam senders typically use to attack mail servers.
- Rejects the first message from a sender, because legitimate senders retry sending the message, and spam senders typically do not. This process is known as *greylisting*. The system does not reject subsequent messages from the same sender to the same recipient.

With an SMTP profile configured, the system either generates an alarm for, or blocks, any requests that trigger the security check.

Note: The SMTP profile is only available for BIG-IP systems that are licensed for BIG-IP® Protocol Security Manager™.

SMTPS profiles

The SMTPS profile provides a way to add SSL encryption to SMTP traffic quickly and easily. *SMTPS* is a method for securing Simple Mail Transport Protocol (SMTP) connections at the transport layer.

Normally, SMTP traffic between SMTP servers and clients is unencrypted. This creates a privacy issue because SMTP traffic often passes through routers that the servers and clients do not trust, resulting in a third party potentially changing the communications between the server and client. Also, two SMTP systems do not normally authenticate each other. A more secure SMTP server might only allow communications from other known SMTP systems, or the server might act differently with unknown systems.

To mitigate these problems, the BIG-IP system includes an SMTPS profile that you can configure. When you configure an SMTPS profile, you can activate support for the industry-standard STARTTLS extension to the SMTP protocol, by instructing the BIG-IP system to either allow, disallow, or require STARTTLS activation for SMTP traffic. The STARTTLS extension effectively upgrades a plain-text connection to an encrypted connection on the same port, instead of using a separate port for encrypted communication.

This illustration shows a basic configuration of a BIG-IP system that uses SMTPS to secure SMTP traffic between the BIG-IP system and an SMTP mail server.

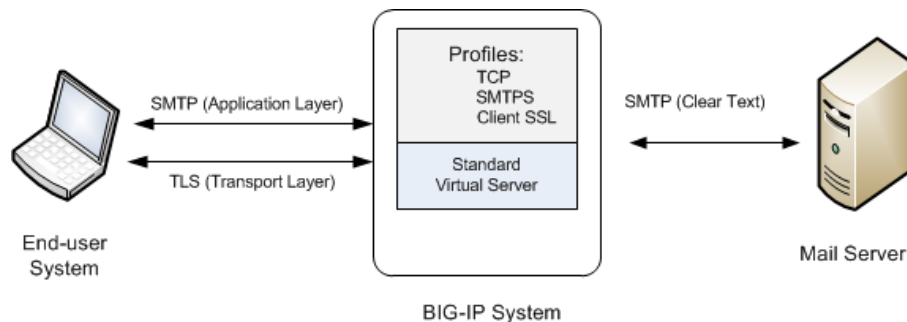


Figure 3: Sample BIG-IP configuration for SMTP traffic with STARTTLS activation

About iSession profiles

The iSession™ profile tells the system how to optimize traffic. Symmetric optimization requires an iSession profile at both ends of the iSession connection. The system-supplied parent iSession profile *isession*, is appropriate for all application traffic, and other iSession profiles have been pre-configured for specific applications. The name of each pre-configured iSession profile indicates the application for which it was configured, such as *isession-cifs*.

When you configure the iSession local endpoint on the Quick Start screen, the system automatically associates the system-supplied iSession profile *isession* with the iSession listener *isession-virtual* it creates for inbound traffic.

You must associate an iSession profile with any virtual server you create for a custom optimized application for outbound traffic, and with any iSession listener you create for inbound traffic.

Screen capture showing compression settings

The following screen capture shows the pertinent compression settings.

Note: If adaptive compression is disabled, you must manually select a compression codec for iSession[™] traffic. If you leave the other codecs enabled, the BIG-IP[®] system selects the bzip2 compression algorithm by default, and that might not be the algorithm you want.

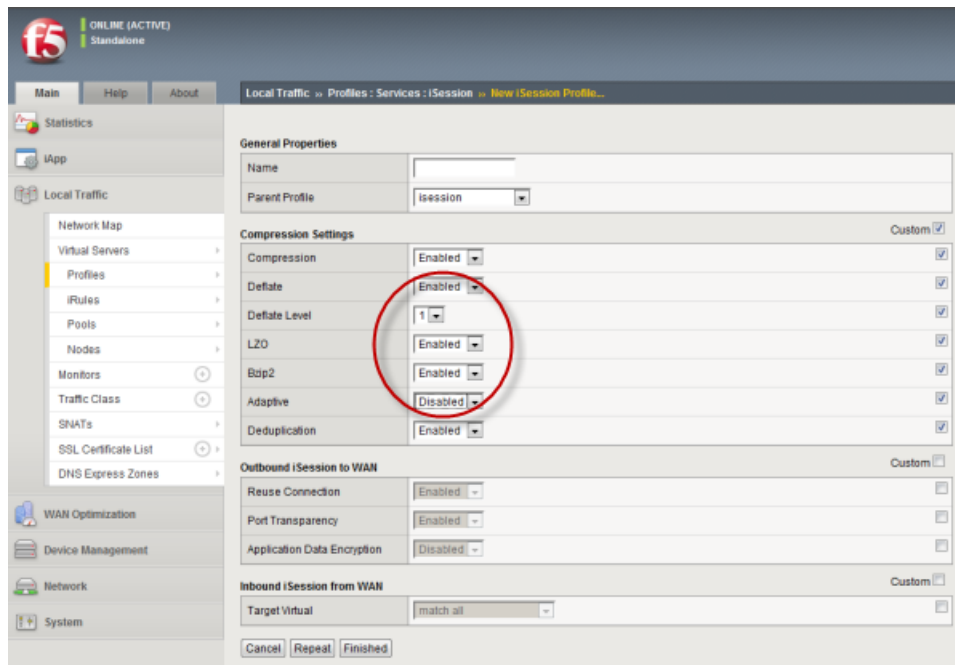


Figure 4: iSession profile screen with compression settings emphasized

Rewrite profiles

For environments that use web servers, you might want your websites to appear differently on the external network than on the internal network. For example, you might want the BIG-IP[®] system to send traffic destined for `www.company.com/usa/` to the internal server `usa.company.com` instead. Normally, this translation could cause some issues, such as the web server expecting to see a certain host name (such as for name-based virtual hosting) or the web server using the internal host name when sending a redirect to client systems.

You can solve these problems by configuring a *Rewrite profile*, which causes the BIG-IP system to act as a reverse proxy server. As a *reverse proxy server*, the BIG-IP system offloads the URI translation function from web servers enabled with features such as Apache's ProxyPass module. With a Rewrite profile, the BIG-IP system can perform URI scheme, host, port, and path modifications as HTTP traffic passes through the system. The feature also provides reverse translation for the Location, Content-Location, and URI headers in the server response to the client.

Important: The BIG-IP reverse proxy feature replaces the ProxyPass iRule available on the F5 Networks site <http://devcentral.f5.com>.

A typical use of a reverse proxy server is to grant Internet users access to application servers that are behind a firewall and therefore have private IP addresses and unregistered DNS entries.

About URI translation

You can configure the BIG-IP® system to perform URI translation on HTTP requests. Suppose that a company named `Siterequest` has a website `www.siterequest.com`, which has a public IP address and a registered DNS entry, and therefore can be accessed from anywhere on the Internet.

Furthermore, suppose that `Siterequest` has two application servers with private IP addresses and unregistered DNS entries, inside the company's firewall. The application servers are visible within the internal network as `appserver1.siterequest.com` and `appserver2.siterequest.com`.

Because these servers have no public DNS entries, any client system that tries to access one of these servers from outside the company network receives a `no such host` error.

As the illustration shows, you can prevent this problem by configuring the BIG-IP system to act as a reverse proxy server:

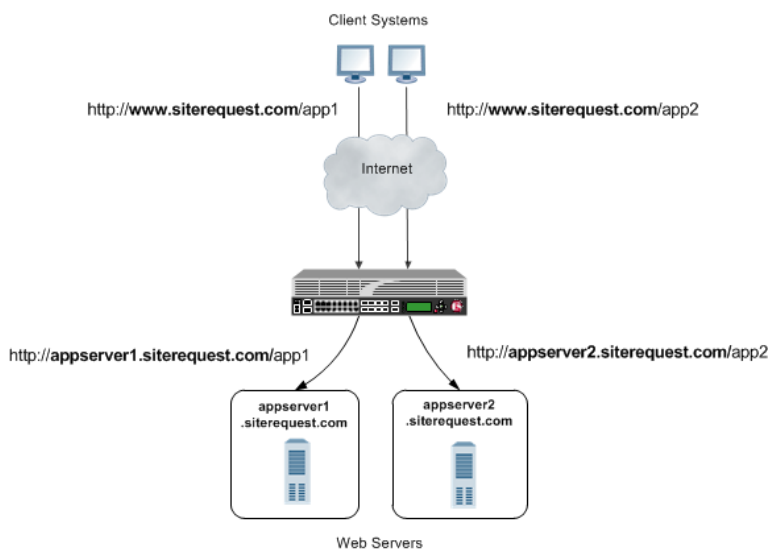


Figure 5: The BIG-IP system as a reverse proxy server for URI translation

In the example, the company `Siterequest` has decided to enable Web access to the internal application servers, without exposing them to the Internet directly. Instead, the company has integrated the servers with the web server `siterequest.com` so that `http://www.siterequest.com/sales` is mapped internally to `http://appserver1.siterequest.com/sales`, and `http://siterequest.com/marketing` is mapped internally to `http://appserver2.example.com/marketing`. This is a typical reverse-proxy configuration.

To configure the BIG-IP system to perform this translation, you create a Rewrite profile and configure one or more URI rules. A *URI rule* specifies the particular URI translation that you want the BIG-IP system to perform. Specifically, a URI rule translates the scheme, host, port, or path of any client URI, server URI, or both. A URI rule also translates any domain and path information in the `Set-Cookie` header of the response when that header information matches the information in the URI rule.

Rules for matching requests to URI rules

The BIG-IP® system follows these rules when attempting to match a request to a URI rule:

- A request does not need to match any entry. That is, if no entries match and there is no catch-all entry, then the Rewrite profile has no effect.
- Each request matches one entry only, which is the entry with the most specific host and path.
- If multiple entries match, then the BIG-IP system uses the entry with the deepest path name on the left side of the specified mapping.
- The BIG-IP system matches those requests that contain host names in URIs before matching requests that do not contain host names in URIs.
- The BIG-IP system processes the specified entries in the mapping from most-specific to least-specific, regardless of the order specified in the actual Rewrite profile.

About URI Rules

When creating a URI rule, you must specify the client and server URIs in these ways:

- When the URI is a path prefix only, the path must be preceded by and followed by a /, for example, `/sales/`.
- When the URI contains more than the path prefix (such as, a host), the URI must also contain a scheme and must be followed by a /, for example, `http://www.siterequest/sales/`.

About Set-Cookie header translation

A URI rule automatically performs translation on any domain and path information in the `Set-Cookie` header of a response when that header information matches the information in the URI rule.

When the `Set-Cookie` header information that you want the BIG-IP® system to translate does not match the information in an existing URI rule, you can create a separate *Set-Cookie rule* to perform this translation. You need to create a `Set-Cookie` rule only when the header information does not match the information specified in an existing URI rule.

The specific parts of the `Set-Cookie` header that you can specify for translation are:

- Client domain
- Client path
- Server domain
- Server path

You can specify that the BIG-IP system translate all of this information or a subset of this information, depending on your needs.

XML profiles

You can use the BIG-IP® system to perform XML content-based routing whereby the system routes requests to an appropriate pool, pool member, or virtual server based on specific content in an XML document. For example, if your company transfers information in XML format, you could use this feature to examine the XML content with the intent to route the information to the appropriate department.

You can configure content-based routing by creating an XML profile and associating it with a virtual server. In the XML profile, define the matching content to look for in the XML document. Next, specify how to route the traffic to a pool by writing simple iRules®. When the system discovers a match, it triggers an iRule event, and then you can configure the system to route traffic to a virtual server, a pool, or a node.

The following example shows a simple XML document that the system could use to perform content-based routing. It includes an element called `FinanceObject` used in this implementation.

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:eai="http://192.168.149.250/eai_enu/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soapenv:Header/>
  <soapenv:Body>
    <eai:SiebelEmployeeDelete
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <FinanceObject xsi:type="xsd:string">Route to
Financing</FinanceObject>
      <SiebelMessage xsi:type="ns:ListOfEmployeeInterfaceTopElmt"
xmlns:ns="http://www.siebel.com/xml">
        <ListOfEmployeeInterface
xsi:type="ns:ListOfEmployeeInterface">
          <SecretKey>123456789</SecretKey>
          <Employee>John</Employee>
          <Title>CEO</Title>
        </ListOfEmployeeInterface>
      </SiebelMessage>
    </eai:SiebelEmployeeDelete>
  </soapenv:Body>
</soapenv:Envelope>
```

SPDY profiles

You can use the BIG-IP® Local Traffic Manager™ SPDY (pronounced "speedy") profile to minimize latency of HTTP requests by multiplexing streams and compressing headers. When you assign a SPDY profile to an HTTP virtual server, the HTTP virtual server informs clients that a SPDY virtual server is available to respond to SPDY requests.

When a client sends an HTTP request, the HTTP virtual server manages the request as a standard HTTP request. It receives the request on port 80, and sends the request to the appropriate server. When the server provides a response, the BIG-IP system inserts an HTTP header into the response (to inform the client that a SPDY virtual server is available to handle SPDY requests), compresses and caches it, and sends the response to the client.

A client that is enabled to use the SPDY protocol sends a SPDY request to the BIG-IP system, the SPDY virtual server receives the request on port 443, converts the SPDY request into an HTTP request, and sends the request to the appropriate server. When the server provides a response, the BIG-IP system converts the HTTP response into a SPDY response, compresses and caches it, and sends the response to the client.

Summary of SPDY profile functionality

By using the SPDY profile, the BIG-IP Local Traffic Manager system provides the following functionality for SPDY requests.

Creating concurrent streams for each connection.

You can specify the maximum number of concurrent HTTP requests that are accepted on a SPDY connection. If this maximum number is exceeded, the system closes the connection.

Limiting the duration of idle connections.

You can specify the maximum duration for an idle SPDY connection. If this maximum duration is exceeded, the system closes the connection.

Enabling a virtual server to process SPDY requests.

You can configure the SPDY profile on the virtual server to receive both HTTP and SPDY traffic, or to receive only SPDY traffic, based in the activation mode you select. (Note that setting this to receive only SPDY traffic is primarily intended for troubleshooting.)

Inserting a header into the response.

You can insert a header with a specific name into the response. The default name for the header is X-SPDY.

Important: The SPDY protocol is incompatible with NTLM protocols. Do not use the SPDY protocol with NTLM protocols. For additional details regarding this limitation, please refer to the SPDY specification: <http://dev.chromium.org/spdy/spdy-authentication>.

SPDY profile settings

This table provides descriptions of the SPDY profile settings.

Setting	Default	Description
Name		Type the name of the SPDY profile.
Parent Profile	spdy	Specifies the profile that you want to use as the parent profile. Your new profile inherits all settings and values from the parent profile specified.
Concurrent Streams Per Connection	10	Specifies how many concurrent requests are allowed to be outstanding on a single SPDY connection.
Connection Idle Timeout	300	Specifies how many seconds a SPDY connection is left open idly before it is closed.
Activation Mode	NPN	Specifies how a connection is established as a SPDY connection. The value NPN specifies that the Transport Layer Security (TLS) Next Protocol Negotiation (NPN) extension determines whether the SPDY protocol is used. Clients that use TLS, but only support HTTP will work as if SPDY is not present. The value Always specifies that all connections must be SPDY connections, and that clients only supporting HTTP are not able to send requests. Selecting Always in the Activation Mode list is primarily intended for troubleshooting.
Insert Header	Disabled	Specifies whether an HTTP header that indicates the use of SPDY is inserted into the request sent to the origin web server.
Insert Header Name	X-SPDY	Specifies the name of the HTTP header controlled by the Insert Header Name setting.
Protocol Versions	All Versions Enabled	Used only with an Activation Mode selection of NPN , specifies the protocol and protocol version (http1.1 , spdy2 , spdy3 , or All Version Enabled) used in the SPDY profile. The order of the protocols in the Selected Versions Enabled list ranges from most preferred (first) to least preferred (last). Adding http1.1 to the Enabled list allows HTTP1.1 traffic to pass. If http1.1 is not added to the Enabled list, clients that do not support http1.1 are blocked. Clients typically use the first supported

Setting	Default	Description
		protocol. At least one SPDY version must be included in the Enabled list.
Priority Handling	Strict	Specifies how the SPDY profile handles priorities of concurrent streams within the same connection. Selecting Strict processes higher priority streams to completion before processing lower priority streams. Selecting Fair enables higher priority streams to use more bandwidth than lower priority streams, without completely blocking the lower priority streams.
Receive Window	32	Specifies the <i>receive window</i> , which is SPDY protocol functionality that controls flow, in KB. The receive window allows the SPDY protocol to stall individual upload streams when needed. This functionality is only available in SPDY3.
Frame Size	2048	Specifies the size of the data frames, in bytes, that the SPDY protocol sends to the client. Larger frame sizes improve network utilization, but can affect concurrency.
Write Size	16384	Specifies the total size of combined data frames, in bytes, that the SPDY protocol sends in a single write function. This setting controls the size of the TLS records when the SPDY protocol is used over Secure Sockets Layer (SSL). A large write size causes the SPDY protocol to buffer more data and improves network utilization.

SOCKS profiles

You can use the BIG-IP® Local Traffic Manager™ SOCKS profile to configure the BIG-IP system to handle proxy requests and function as a gateway. By configuring browser traffic to use the proxy, you can control whether to allow or deny a requested connection. To implement the profile, you must associate it with a virtual server.

Protocol Versions

You can specify one or more versions of SOCKS.

- **Socks4** indicates protocol support for SOCKS version 4.
- **Socks4A** indicates protocol support for SOCKS 4A, which adds host name support to version 4.
- **Socks5** specifies protocol support for SOCKS version 5, which includes host name and IPv6 support.

DNS Resolver

You must specify a DNS resolver to use for DNS inquiries handled by the virtual servers associated with this profile. If no DNS resolver exists on the system, you can create one at **DNS > Caches > Cache List > Create**.

Route Domain

You can specify a route domain to be used for outbound connect requests.

Tunnel Name

You must specify a tunnel that is used for outbound connect requests, enabling other virtual servers to receive connections initiated by the proxy service. A pre-configured tunnel `socks-tunnel` is available.

Default Connect Handling

You can specify the behavior of the proxy service when handling outbound requests.

- Enabled (checked) indicates that the proxy service delivers outbound requests directly, regardless of the presence of listening servers.
- Disabled (check box cleared) indicates that the proxy service delivers outbound requests only if another virtual server is listening on the tunnel for the requested outbound connection. With this setting, virtual servers are required, and the system processes the outbound traffic before it leaves the device.

FIX profiles

The BIG-IP® system Local Traffic Manager™ (LTM®) FIX profile provides you with the ability to use Financial Information eXchange (FIX) protocol messages in routing, load balancing, persisting, and logging connections. The BIG-IP system uses the FIX profile to examine the header, body, and footer of each FIX message, and then process each message according to the parameters that it contains.

The BIG-IP system supports FIX protocol versions 4.2, 4.4, and 5.0, and uses the key-value pair FIX message format.

Important: *You cannot configure or use the BIG-IP FIX Profile to provide low-latency electronic trading functionality. Instead, you must implement low-latency electronic trading functionality separately.*

About FIX profile tag substitution

The BIG-IP® system's FIX profile provides options for how the FIX messages should be parsed. Once configured, the BIG-IP system compares the FIX profile's Mapping List Sender value (SenderCompID) with the value received in the client message. If the values match, then the BIG-IP system provides tag substitution as defined by the data group definition in the corresponding mapping list.

Example

Two or more clients can define a FIX tag differently. On the BIG-IP server side, you can define a dictionary for each client that maps a client tag to a server tag. For example, a server might use 20001 for an analyst's average price target, and 20002 as a client twitter feed name. Then, in the dictionary for the first client, the tag 10001 is mapped to 20001, and, for the second client, the tag 30001 is mapped to 20001.

About steering traffic using the FIX profile

The BIG-IP® system's FIX profile can direct, or steer, FIX messages to a destination pool in accordance with the FIX login message that it receives, and the configured iRules®. Once a pool member is selected, which is only required one time for a connection, all messages in the same FIX session are forwarded, or persisted, to that pool member.

About validating FIX messages

The BIG-IP® system validates each Financial Information eXchange (FIX) protocol message, allowing and denying transmission accordingly. If a FIX message is valid, the BIG-IP system allows transmission, triggers the `FIX_MESSAGE` iRule event, and optionally logs the message. If a FIX message is invalid, the BIG-IP system logs the error, and either disallows transmission or drops the connection, as configured by the profile.

The BIG-IP system provides two types of parsing validation: full parsing validation and quick parsing validation.

Full Parsing Validation

When *full parsing validation* is applied, all fields are validated.

Quick Parsing Validation

When *quick parsing validation* is applied, the following fields are validated.

- The first three fields: 8 (BeginString), 9 (BodyLength), and 35 (MsgType).
- The last field.
- Field 49 (SenderCompID).
- Fields requested by an iRule tag query command.
- Fields in the message that precede the fields requested by an iRule tag query command.

For example, consider the following message: 8=FIX.4.2|9=100|35=A|600=X|700=Y|800=Z... In this example, the first three fields are always parsed: 8, 9, and 35. If the iRule command `FIX::tag 700` runs, then the fields preceding 700 in the example are parsed, specifically 600 (in addition to the first three fields).

The following table describes the different types of quick parsing validation that the BIG-IP system provides.

FIX Message	Description	Quick Parsing Validation	Example
Message sequence no <num> from <senderCompID> error: There is no = in the field starting at byte <byte offset of the field>	Field is not in the format of tag=val.	This error is partially checked when using quick parsing validation.	35=A;123xyz;. The second field is missing an = sign.
Message sequence no <num> from <senderCompID> error: the field starting at byte <byte offset of the field> has invalid tag	The tag is not an integer.	This error is partially checked when using quick parsing validation.	35=A;abc=xyz;. The tag abc in the second field is not an integer.
Message sequence no <num> from <senderCompID> error: there is no value found in the field starting at byte <byte offset of the field>	A value is missing.	This error is partially checked when using quick parsing validation.	35=A;50=;. The second field is missing a value.
The first (second, third) tag should be 8 (9, 35), but get <wrong value> from < senderCompID>	The first three tags are not 8, 9, and 35.	This error is fully checked when using quick parsing validation.	None.
Length mismatch: message sequence no <num> from <senderCompID> should be tag10 after <length> bytes, but encounter <val1 val2>	The length is mismatched.	This error is fully checked when using quick parsing validation.	None.

FIX Message	Description	Quick Parsing Validation	Example
Checksum mismatch: message sequence <num> from <senderCompID> declares checksum as <claimed value>, but calculated checksum from received data is <real value>	The checksum is mismatched.	This error is fully checked when using quick parsing validation.	None.
Message from <IP address> is longer than allowed	The message length is greater than 4MB.	This error is fully checked when using quick parsing validation.	None.

About using SSL encryption for FIX messages

You can configure a virtual server to use client and server SSL encryption with FIX protocol messages, as necessary, for transactions across the Internet, or for compliance purposes.

About logging FIX messages

The BIG-IP[®] system provides optional logging of each FIX message for auditing purposes. You can log events either locally on the BIG-IP system or remotely, using the BIG-IP system's high-speed logging mechanism. The recommended way to store logs is on a pool of remote logging servers.

For local logging, the high-speed logging mechanism stores the logs in either the Syslog or the MySQL database on the BIG-IP system, depending on a destination that you define. For remote logging, the high-speed logging mechanism sends log messages to a pool of logging servers that you define.

Video Quality of Experience profiles

The BIG-IP[®] system's video Quality of Experience (QoE) profile enables you to assess an audience's video session or overall video experience, providing an indication of customer satisfaction. The QoE profile uses static information, such as bitrate and duration of a video, and video metadata, such as URL and content type, in monitoring video streaming. Additionally, the QoE profile monitors dynamic information, such as the variable video downloading rate. By measuring the video playing rate and downloading rate, the user experience can be assessed and defined in terms of a single mean opinion score (MOS) of the video session, and a level of customer satisfaction can be derived. QoE scores are logged in the `ltm` log file, located in `/var/log`, which you can evaluate as necessary.

About the video Quality of Experience profile

The BIG-IP[®] system's video Quality of Experience (QoE) profile enables you to assess an audience's video session or overall video experience, providing an indication of customer satisfaction. The QoE profile uses static information, such as bitrate and duration of a video, and video metadata, such as URL and content

type, in monitoring video streaming. Additionally, the QoE profile monitors dynamic information, which reflects the real-time network condition.

By considering both the static video parameters and the dynamic network information, the user experience can be assessed and defined in terms of a single mean opinion score (MOS) of the video session, and a level of customer satisfaction can be derived. QoE scores are logged in the `ltm` log file, located in `/var/log`, which you can evaluate as necessary.

Note that for QoE to properly process video files, the video web servers must be compliant with supported video MIME types, for example, the following MIME types.

MIME Type	Suffix
video/mp4	.f4v
video/mp4	.mp4
video/x-flv	.flv
video/x-m4v	.m4v
video/quicktime	.m4v
application/x-mpegURL	.m3u8
video/mp2t	.ts

About mean opinion score

The video Quality of Experience (QoE) profile provides a mean opinion score (MOS), derived from static and dynamic parameters associated with a video stream. The following table summarizes the resultant values.

MOS	Quality	Description
5	Excellent	Indicates a superior level of quality, with imperceptible degradation in the video stream.
4	Good	Indicates an above-average level of quality, with perceptible degradation that is acceptable.
3	Fair	Indicates an average level of quality, with perceptible degradation that detracts from the video experience.
2	Poor	Indicates a below-average level of quality, with perceptible degradation that significantly detracts from the video experience.
1	Bad	Indicates a substandard level of quality, with perceptible degradation that proves to be significantly inferior and potentially unacceptable.

Content Profiles

Introduction to HTML content modification

When you configure an HTML profile on the BIG-IP® system, the system can modify HTML content that passes through the system, according to your specifications. For example, if you want the BIG-IP system to detect all content of type `text/html` and then remove all instances of the HTML `img` tag with the `src` attribute, you can configure an HTML profile accordingly, and assign it to the virtual server. The HTML profile ensures that the BIG-IP system removes those instances of the tag from any HTML content that passes through the virtual server.

Or, you can configure an HTML profile to match on a certain tag and attribute in HTML content when a particular iRule event is triggered, and then create an iRule that includes a command to replace the value of the matched attribute with a different attribute. The BIG-IP system includes several iRule commands that you can use when the `Raise Event on Comment` or `Raise Event on Tag` events are triggered. For more information on iRule commands related to HTML content modification, see the F5 Networks web site <http://devcentral.f5.com>.

HTML tag removal and replacement are just two of several HTML rules that you can configure to manipulate HTML content. An *HTML rule* defines the specific actions that you want the BIG-IP system to perform on a specified type HTML content.

About content selection types

When you create an HTML type of profile, you can specify the type of content that you want the BIG-IP® system to match on when determining which content to modify.

The types of content that you specify must be valid values for the `Content-Type` header of an HTTP response.

Typical `Content-Type` header values that you can specify are:

- `text/html`
- `text/xhtml`
- `text/xml`
- `text/javascript`

You must specify at least one valid content type if you want the BIG-IP system to match content based on an HTML rule that you create.

Types of HTML rules

You can create several different types of HTML rules for modifying HTML content.

HTML rule type	Description
Remove Comments	Removes comments within HTML content. There are no matching and no actions associated with this type of HTML rule.
Raise Event on Comments	Raises an <code>HTML_COMMENT_MATCHED</code> iRule event. There are no matching and no actions associated with this type of HTML rule.
Remove Attribute	Matches on the specified tag name, attribute name, and attribute value, and then removes the HTML tag attribute.
Append HTML	Matches on the specified tag name, attribute name, and attribute value, and then appends the specified HTML content to the tag delimiter.
Prepend HTML	Matches on the specified tag name, attribute name, and attribute value, and then prepends the specified HTML content to the tag delimiter.
Raise Event on Tag	Raises an <code>HTML_TAG_MATCHED</code> iRule event. With this type of rule, you can match on the tag name, attribute name, and attribute value.
Remove Tag	Matches on the specified tag name, attribute name, and attribute value, and then removes the HTML tag.

Sample HTML rules configuration

When you create an HTML rule, you can specify the actions that you want the BIG-IP® system to take based on that type of rule. The actions you specify vary depending on the type of HTML rule you are creating.

For example, suppose you want to replace this HTML content:

```

```

with:

```

```

To do this, you can create an HTML profile with a content selection type of `text/html` and the `Raise Event on Tag` rule.

In the example, the `Raise Event on Tag` rule specifies these match settings:

Tag name:	img
Attribute name:	src
Attribute value:	image/bigip8900.jpg

These settings instruct the BIG-IP system to match on any `` tag that includes the `src` attribute and the attribute value `image/bigip8900.jpg`.

After creating the HTML profile, you can write an iRule that specifies the `HTML_TAG_MATCHED` event, as well as the `HTML::tag attribute replace` command, which specifies the new attribute value. When the traffic reaches the virtual server, the BIG-IP system triggers the event, matches on the tag and attribute specified in the HTML rule, and replaces the old attribute value with the new value specified in the iRule.

Session Persistence Profiles

Introduction to session persistence profiles

Using BIG-IP® Local Traffic Manager™, you can configure session persistence. When you configure *session persistence*, Local Traffic Manager tracks and stores session data, such as the specific pool member that serviced a client request. The primary reason for tracking and storing session data is to ensure that client requests are directed to the same pool member throughout the life of a session or during subsequent sessions.

In addition, session persistence can track and store other types of information, such as user preferences or a user name and password.

Local Traffic Manager offers several types of session persistence, each one designed to accommodate a specific type of storage requirement for session data. The type of persistence that you implement depends on where and how you want to store client-specific information, such as items in a shopping cart or airline ticket reservations.

For example, you might store airline ticket reservation information in a back-end database that all servers can access, or on the specific server to which the client originally connected, or in a cookie on the client's machine. When you enable persistence, returning clients can bypass load balancing and instead connect to the server to which they last connected in order to access their saved information.

Local Traffic Manager keeps session data for a period of time that you specify.

The primary tool for configuring session persistence is to configure a persistence profile and assign it to a virtual server. If you want to enable persistence for specific types of traffic only, as opposed to all traffic passing through the virtual server, you can write an iRule.

A *persistence profile* is a pre-configured object that automatically enables persistence when you assign the profile to a virtual server. By using a persistence profile, you avoid having to write a program to implement a type of persistence.

Each type of persistence that Local Traffic Manager offers includes a corresponding default persistence profile. These persistence profiles each contain settings and setting values that define the behavior of the BIG-IP system for that type of persistence. You can either use the default profile or create a custom profile based on the default.

Persistence profile types

You can configure persistence profile settings to set up session persistence on the BIG-IP® system. You can configure these settings when you create a profile or after profile creation by modifying the profile's settings.

The persistence types that you can enable using a persistence profile are:

Cookie persistence

Cookie persistence uses an HTTP cookie stored on a client's computer to allow the client to reconnect to the same server previously visited at a web site.

Destination address affinity persistence

Also known as sticky persistence, destination address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the destination IP address of a packet.

Hash persistence

Hash persistence allows you to create a persistence hash based on an existing iRule.

Microsoft® Remote Desktop Protocol persistence

Microsoft® Remote Desktop Protocol (MSRDP) persistence tracks sessions between clients and servers running the Microsoft® Remote Desktop Protocol (RDP) service.

SIP persistence

SIP persistence is a type of persistence used for servers that receive Session Initiation Protocol (SIP) messages sent through UDP, SCTP, or TCP.

Source address affinity persistence

Also known as simple persistence, source address affinity persistence supports TCP and UDP protocols, and directs session requests to the same server based solely on the source IP address of a packet.

SSL persistence

SSL persistence is a type of persistence that tracks non-terminated SSL sessions, using the SSL session ID. Even when the client's IP address changes, Local Traffic Manager™ still recognizes the connection as being persistent based on the session ID. Note that the term non-terminated SSL sessions refers to sessions in which Local Traffic Manager does not perform the tasks of SSL certificate authentication and encryption/re-encryption.

Universal persistence

Universal persistence allows you to write an expression that defines what to persist on in a packet. The expression, written using the same expression syntax that you use in iRules®, defines some sequence of bytes to use as a session identifier.

Session persistence and iRules

Instead of configuring a persistence profile, which enables a persistence type for all sessions passing through the virtual server, you can write an iRule, which enables a persistence type for particular requests (for example, for HTTP traffic that includes a certain cookie version only).

You can also use an iRule to enable persistence for SSL-terminated requests, that is, requests that Local Traffic Manager™ terminates by performing decryption and re-encryption and by handling SSL certificate authentication. In iRules® of this type, you can use an HTTP header insertion iRule command to insert an SSL session ID as a header into an HTTP request.

The OneConnect profile and session persistence

When you configure session persistence, Local Traffic Manager™ tracks and stores session data, such as the pool member that serviced a client request. Configuring a persistence profile for a virtual server ensures that client requests are directed to the same pool member throughout the life of a session or during subsequent sessions.

The `Request-URI` header in an HTTP request stores certain session data. Occasionally, however, for Cookie and Universal persistence types specifically, Local Traffic Manager ignores the session data in this header, and sends requests to an unexpected node. For example, this issue can occur when clients send requests to a virtual server through an internet proxy device. You can prevent this problem by creating a OneConnect™ profile, and assigning both the OneConnect profile and the persistence profile to the virtual server.

HTTP parsing with and without a OneConnect profile

If the virtual server does not reference a OneConnect™ profile, Local Traffic Manager™ performs load balancing for each TCP connection. Once the TCP connection is load balanced, the system sends all requests that are part of the connection to the same pool member.

For example, if the virtual server does not reference a OneConnect profile, and Local Traffic Manager initially sends a client request to node A in pool A, the system inserts a cookie for node A. Then, within the same TCP connection, if Local Traffic Manager receives a subsequent request that contains a cookie for node B in pool B, the system ignores the cookie information and incorrectly sends the request to node A instead.

Using a OneConnect™ type of profile ensures that the BIG-IP system does not ignore session data. If the virtual server references a OneConnect profile, Local Traffic Manager™ can perform load balancing for each request within the TCP connection. That is, when an HTTP client sends multiple requests within a single connection, Local Traffic Manager is able to process each HTTP request individually. Local Traffic Manager sends the HTTP requests to different destination servers if necessary.

For example, if the virtual server references a OneConnect profile and the client request is initially sent to node A in pool A, Local Traffic Manager inserts a cookie for node A. Then, within the same TCP connection, if Local Traffic Manager receives a subsequent request that contains a cookie for node B in pool B, the system uses that cookie information and correctly sends the request to node B.

Criteria for session persistence

Regardless of the type of persistence you are implementing, you can specify the criteria that Local Traffic Manager™ uses to send all requests from a given client to the same pool member. These criteria are based on the virtual server or servers that are hosting the client connection. To specify these criteria, you use the **Match Across Services**, **Match Across Virtual Servers**, and **Match Across Pools** profile settings. Before configuring a persistence profile, it is helpful to understand these settings.

The Match Across Services setting

When you enable the **Match Across Services** profile setting, Local Traffic Manager™ attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node only when the virtual server hosting the connection has the same virtual address as the virtual server hosting the initial persistent connection. Connection requests from the client that go to other virtual servers with different virtual addresses, or those connection requests that do not use persistence, are load balanced according to the load balancing method defined for the pool.

For example, suppose you configure virtual server mappings where the virtual server `v1:http` has persistence enabled and references the `http_pool` (containing the nodes `n1:http` and `n2:http`), and the virtual server

`v1:ssl` has persistence enabled and references the pool `ssl_pool` (containing the nodes `n1:ssl` and `n2:ssl`).

Suppose the client makes an initial connection to `v1:http`, and the load balancing algorithm assigned to the pool `http_pool` chooses `n1:http` as the node. If the client subsequently connects to `v1:ssl`, Local Traffic Manager™ uses the persistence session established with the first connection to determine the pool member that should receive the connection request, rather than the load balancing method. Local Traffic Manager should then send the third connection request to `n1:ssl`, which uses the same node as the `n1:http` node that currently hosts the client's first connection with which it shares a persistent session.

If the same client then connects to a virtual server with a different virtual address (for example, `v2:ssl`), Local Traffic Manager starts tracking a new persistence session, using the load balancing method to determine which node should receive the connection request. The system starts a new persistence session because the requested virtual server uses a different virtual address (`v2`) than the virtual server hosting the first persistent connection request (`v1`).

Important: *In order for the Match Across Services setting to be effective, virtual servers that use the same virtual address, as well as those that use SSL persistence, should include the same node addresses in the virtual server mappings.*

Note: *With respect to Cookie profiles, this setting applies to the Cookie Hash method only.*

The Match Across Virtual Servers setting

You can set Local Traffic Manager™ to maintain persistence for all sessions requested by the same client, regardless of which virtual server hosts each individual connection initiated by the client. When you enable the **Match Across Virtual Servers** setting, Local Traffic Manager attempts to send all persistent connection requests received from the same client, within the persistence time limit, to the same node. Connection requests from the client that do not use persistence are load balanced according to the currently selected load balancing method.

Note: *With respect to Cookie profiles, this setting applies to the Cookie Hash method only.*

Warning: *In order for this setting to be effective, virtual servers that use pools with TCP or SSL persistence should include the same member addresses in the virtual server mappings.*

The Match Across Pools setting

When you enable the **Match Across Pools** setting, Local Traffic Manager™ can use any pool that contains a given persistence record. The default is disabled (cleared).

Warning: *Enabling this setting can cause Local Traffic Manager to direct client traffic to a pool other than that specified by the virtual server.*

With respect to Cookie profiles, this setting applies to the Cookie Hash method only.

Cookie persistence

You can set up Local Traffic Manager™ to use HTTP cookie persistence. *Cookie persistence* uses an HTTP cookie stored on a client's computer to allow the client to reconnect to the same pool member previously visited at a web site.

There are four methods of cookie persistence available:

- HTTP Cookie Insert method
- HTTP Cookie Rewrite method
- HTTP Cookie Passive method
- Cookie Hash method

The method you choose to use affects how Local Traffic Manager returns the cookie when returning the cookie to the client.

HTTP Cookie Insert method

If you specify the **HTTP Cookie Insert** method within the Cookie persistence profile, the information about the server to which the client connects is inserted in the header of the HTTP response from the server as a cookie. By default, the cookie is named `BIGipServer<pool_name>`, and it includes the encoded address and port of the server handling the connection. The expiration date for the cookie is set based on the **Expiration** setting in the Cookie persistence profile. **HTTP Cookie Insert** is the default value for the **Cookie Method** setting.

***Tip:** You can assign this type of profile to a Performance (HTTP) type of virtual server.*

HTTP Cookie Rewrite method

If you specify HTTP Cookie Rewrite method, the BIG-IP system intercepts a `Set-Cookie` header, named `BIGipCookie`, sent from the server to the client, and overwrites the name and value of the cookie. The new cookie is named `BIGipServer<pool_name>` and it includes the address and port of the server handling the connection.

***Important:** We recommend that you use this method instead of the HTTP Cookie Passive method whenever possible.*

The HTTP Cookie Rewrite method requires you to set up the cookie created by the server. For the HTTP Cookie Rewrite method to succeed, there needs to be a blank cookie coming from the web server for Local Traffic Manager to rewrite. With Apache variants, the cookie can be added to every web page header by adding the following entry to the `httpd.conf` file: `Header add Set-Cookie BIGipCookie=00000000000000000000000000000000...`

(The cookie must contain a total of 120 zeros.)

***Note:** For backward compatibility, the blank cookie can contain only 75 zeros. However, cookies of this size do not allow you to use iRules® and persistence together.*

HTTP Cookie Passive method

If you specify the HTTP Cookie Passive method, the BIG-IP[®] system does not insert or search for blank `Set-Cookie` headers in the response from the server. This method does not try to set up the cookie. With this method, the server provides the cookie, formatted with the correct server information and timeout.

Important: *We recommend that you use the HTTP Cookie Rewrite method instead of the HTTP Cookie Passive method whenever possible.*

For the HTTP Cookie Passive method to succeed, there needs to be a cookie coming from the web server with the appropriate server information in the cookie. Using the BIG-IP Configuration utility, you generate a template for the cookie string, with encoding automatically added, and then edit the template to create the actual cookie.

For example, the following string is a generated cookie template with the encoding automatically added, where `[pool name]` is the name of the pool that contains the server, 336260299 is the encoded server address, and 20480 is the encoded port:

```
Set-Cookie:BIGipServer[poolname]=336268299.20480.0000; expires=Sat, 01-Jan-2002
00:00:00 GMT; path=/
```

To create your cookie from this template, type the actual pool name and an expiration date and time. Alternatively, you can perform the encoding using the following equation for address (a.b.c.d): $d * (256^3) + c * (256^2) + b * 256 + a$

The way to encode the port is to take the two bytes that store the port and reverse them. Thus, port 80 becomes $80 * 256 + 0 = 20480$. Port 1433 (instead of $5 * 256 + 153$) becomes $153 * 256 + 5 = 39173$.

With Apache variants, the cookie can be added to every web page header by adding the following entry to the `httpd.conf` file: `Header add Set-Cookie: "BIGipServer my_pool=184658624.20480.000; expires=Sat, 19-Aug-2002 19:35:45 GMT; path=/"`

Note: *the profile settings **Mirror Persistence**, **Match Across Services**, **Match Across Virtual Servers**, and **Match Across Pools** do not apply to the HTTP Cookie Passive method. These settings apply to the Cookie Hash method only.*

Cookie hash method

If you specify the Cookie Hash method, the hash method consistently maps a cookie value to a specific node. When the client returns to the site, Local Traffic Manager[™] uses the cookie information to return the client to a given node. With this method, the web server must generate the cookie; Local Traffic Manager does not create the cookie automatically as it does when you use the HTTP Cookie Insert method.

IPv4 IP address encoding

For HTTP Cookie Insert and HTTP Cookie Rewrite methods only, the BIG-IP system encodes the server address and port specified in an HTTP cookie. To encode an IPv4 server address specified within an HTTP cookie, the BIG-IP system:

1. Converts each octet value to the equivalent 1-byte hexadecimal value.
2. Reverses the order of the hexadecimal bytes and concatenate to make one 4-byte hexadecimal value.

3. Converts the resulting 4-byte hexadecimal value to its decimal equivalent.

For example, if the IP address of the destination server is 10.1.1.100, the BIG-IP system encodes the IP address as follows:

```
10.1.1.100 = 0x0A . 0x01 . 0x01 . 0x64
Reverse byte order, concatenated = 0x6401010A
0x6401010A = 1677787402
```

The address encoding algorithm is performed algebraically, as follows, for address a.b.c.d:

$$a + b*256 + c*(256^2) + d*(256^3)$$

For example, if the IP address of the destination server is 10.1.1.100, the encoded address is derived as follows:

```
a=10; b=1; c=1; d=100
10 + 1*256 + 1*(256^2) + 100*(256^3) = 1677787402
```

The result is that BIG-IP system combines the encoded values for the server and port and inserts them into the persistence cookie. For example, using the IP address and port 10.1.1.100:8080, the persistence value that the BIG-IP LTM system encodes in the cookie is as follows:

```
1677787402.36895.0000
```

Note that the field following the port encoding is reserved for future use and always contains four zeros as placeholders.

Port encoding

For HTTP Cookie Insert and HTTP Cookie Rewrite methods only, the BIG-IP system encodes the server address and port specified in an HTTP cookie. To encode a port specified within an HTTP cookie, the BIG-IP system:

1. Converts the decimal port value to the equivalent 2-byte hexadecimal value.
2. Reverses the order of the two hexadecimal bytes.
3. Converts the resulting 2-byte hexadecimal value to its decimal equivalent.

For example, if the port of the destination server is 8080, the BIG-IP system encodes the port as follows:

```
8080 = 0x1F90
Reverse byte order = 0x901F
0x901F = 36895
```

If the port value is less than 256, the first byte in step 1 is 0x00. For example, if the port value is 80, the BIG-IP system encodes the port as follows:

```
80 = 0x0050
Reverse byte order = 0x5000
0x5000 = 20480
```

The result is that BIG-IP system combines the encoded values for the server and port and inserts them into the persistence cookie. For example, using the IP address and port 10.1.1.100:8080, the persistence value that the BIG-IP LTM system encodes in the cookie is as follows:

```
1677787402.36895.0000
```

Note that the field following the port encoding is reserved for future use and always contains four zeros as placeholders.

Destination address affinity persistence

You can optimize your server array with destination address affinity persistence. *Destination address affinity persistence*, also known as sticky persistence, directs requests for a certain destination IP address to the same server, regardless of which client made the request.

This type of persistence provides the most benefits when load balancing caching servers. A caching server intercepts web requests and returns a cached web page if it is available. In order to improve the efficiency of the cache on these servers, it is necessary to send similar requests to the same server repeatedly. You can use the destination address affinity persistence type to cache a given web page on one server instead of on every server in an array. This saves the other servers from having to duplicate the web page in their cache, wasting memory.

Hash persistence

Hash persistence allows you to create a persistence hash based on an existing iRule that uses the `persist` iRule command. Using hash persistence is the same as using universal persistence, except that with hash persistence, the resulting persistence key is a hash of the data, rather than the data itself.

An example of a iRule that implements hash persistence:

```
rule my_persist_irule { when HTTP_REQUEST { persist hash [HTTP::header myheader]
} }
```

Note that if you use hash persistence and Local Traffic Manager™ cannot find an entry in the persistence table for a connection, and the system has not yet chosen a pool member due to fallback persistence, then the system uses the hash value, rather than the specified load balancing method, to select the pool member.

For example, if the persistence table contains no entry for the hash value 2356372769, and the number of active nodes in the pool remains the same, then a session with that hash value for persistence is always persisted to node 10.10.10.190 (assuming that the node is active).

Microsoft Remote Desktop Protocol persistence

MSRDP persistence provides an efficient way of load balancing traffic and maintaining persistent sessions between Windows® clients and servers that are running the Microsoft® Remote Desktop Protocol (RDP) service. The recommended scenario for enabling MSRDP persistence feature is to create a load balancing pool that consists of members running Windows Server 2003 or Windows Server 2008, where all members belong to a Windows cluster and participate in a Windows session directory.

Benefits of Microsoft Remote Desktop Protocol persistence

Normally, Windows servers running Microsoft Terminal Services can use a session broker (known as Terminal Services Session Directory in Windows Server 2003 and TS Session Broker in Windows Server 2008) to ensure that user sessions are assigned to specific servers. If a client initiates a connection request to the wrong terminal server, that server redirects the client to the appropriate server.

When you have a BIG-IP® system, however, the incorrect server needs to redirect the client to the BIG-IP system virtual server, rather than to an individual server in the load balancing pool. To ensure that this happens, you can configure an MSRDP profile. With an MSRDP profile, Local Traffic Manager™ uses a token that the session broker provides to maintain persistence records. If a user initiates a session for which no session broker token exists, Local Traffic Manager makes load balancing decisions according to whichever load balancing method is configured for the pool.

In summary, using Local Traffic Manager with an MSRDP persistence profile, in conjunction with a session broker, allows for higher scalability and a greater range and flexibility of load balancing options than when using a session broker alone.

Microsoft Remote Desktop Protocol persistence server platform issues

By default, Local Traffic Manager™ with MSRDP persistence enabled load balances connections according to the way that the user has configured Local Traffic Manager for load balancing, as long as the session broker is configured on each server in the pool. Terminal Services Session Directory and TS Session Broker are features that are only available on Windows Server 2003 or Windows Server 2008 respectively. Therefore, each server in the pool must be a Windows Server 2003 or Windows Server 2008 server, if you want to use MSRDP persistence in default mode. Also, each client system must be running the remote desktop client software that is included with any Windows Server 2003 or Windows Server 2008 system.

If, however, you want to enable MSRDP persistence but your server platforms are running older versions of Windows (on which Session Directory or TS Session Broker is not available), you can enable MSRDP persistence in non-default mode. This causes Local Traffic Manager to connect a client to the same Windows server by way of the user name that the client provides. Note that enabling MSRDP persistence in non-default mode (that is, with no session broker available on the servers) is less preferable than the default mode, because it provides limited load-balancing and redirection capabilities.

Configuring MSRDP persistence with a session broker

To enable MSRDP persistence in the default mode, you must configure a session broker on each Windows server in your load balancing pool. In addition to configuring a session broker, you must perform other Windows configuration tasks on those servers. However, before you configure your Windows servers, you must configure Local Traffic Manager, by performing tasks such as creating a load-balancing pool and designating your Windows servers as members of that pool.

Configuring MSRDSP persistence without a session broker

When a server has no session broker, the server cannot share sessions with other servers, and therefore cannot perform any redirections when a connection to a server becomes disconnected. In lieu of session sharing, Windows clients provide data, in the form of a user name, to the BIG-IP® system to allow the BIG-IP system to consistently connect that client to the same server. Enabling MSRDSP persistence to behave in this way is the non-default mode.

SIP persistence

Session Initiation Protocol is an application-layer protocol that manages sessions consisting of multiple participants, thus enabling real-time messaging, voice, data, and video. A session can be a simple two-way telephone call or Instant Message dialogue, or a complex, collaborative, multi-media conference call that includes voice, data, and video. With SIP, applications can communicate with one another by exchanging messages through the SCTP, TCP or UDP protocols.

SIP persistence is a type of persistence available for server pools. You can configure SIP persistence for proxy servers that receive SIP messages sent through the UDP profile. Local Traffic Manager™ currently supports persistence for SIP messages sent through the UDP, TCP, or SCTP protocols.

Important: *For virtual servers processing UDP traffic, always check that the value of the SIP profile **Timeout** setting is at least as long (in seconds) as the value of the **Idle Timeout** setting of the UDP profile. Doing so ensures that SIP traffic is persisted properly.*

Source address affinity persistence

Source address affinity persistence, also known as simple persistence, tracks sessions based only on the source IP address. When a client requests a connection to a virtual server that supports source address affinity persistence, Local Traffic Manager™ checks to see if that client previously connected, and if so, returns the client to the same pool member.

You might want to use source address affinity persistence and SSL persistence together. In situations where an SSL session ID times out, or where a returning client does not provide a session ID, you might want Local Traffic Manager to direct the client to the original pool member based on the client's IP address. As long as the client's source address affinity persistence record has not timed out, Local Traffic Manager can successfully return the client to the appropriate pool member.

Persistence settings apply to all protocols. When the persistence timer is set to a value greater than 0, persistence is on. When the persistence timer is set to 0, persistence is off.

The persistence mask feature works only for virtual servers that implement source address affinity persistence. By adding a persistence mask, you identify a range of source IP addresses to manage together as a single source address affinity persistent connection when connecting to the pool.

SSL persistence

SSL persistence is a type of persistence that tracks SSL sessions using the SSL session ID, and it is a property of each individual pool. Using SSL persistence can be particularly important if your clients typically have

translated IP addresses or dynamic IP addresses, such as those that Internet service providers typically assign. Even when the client's IP address changes, Local Traffic Manager™ still recognizes the session as being persistent based on the session ID.

You might want to use SSL persistence and source address affinity persistence together. In situations where an SSL session ID times out, or where a returning client does not provide a session ID, you might want Local Traffic Manager to direct the client to the original node based on the client's IP address. As long as the client's simple persistence record has not timed out, Local Traffic Manager can successfully return the client to the appropriate node.

Universal persistence

Included in the Local Traffic Manager™ Universal Inspection Engine (UIE) is a set of functions that you can specify within BIG-IP® system iRules® to direct traffic in more granular ways. Using these iRule functions, you can write expressions that direct traffic based on content data, or direct traffic to a specific member of a pool.

Universal persistence takes this iRules feature one step further, by allowing you to use the iRule `persist uie` command to implement persistence for sessions based on content data, or based on connections to a specific member of a pool. Universal persistence does this by defining some sequence of bytes to use as a session identifier.

To use iRule expressions for persistence, a universal persistence profile includes a setting that specifies the name of the iRule containing the expression.

```
rule my_persist_irule { when HTTP_REQUEST { persist uie [HTTP::header myheader]
} }
```

Unlike hash persistence, which uses a hash of the data as the persistence key, universal persistence uses the data itself as the persistence key.

Note: F5 Networks® recommends that you configure a OneConnect™ profile in addition to the Universal profile, to ensure that Local Traffic Manager load balances HTTP requests correctly.

Protocol Profiles

About protocol profiles

Some of the BIG-IP® Local Traffic Manager™ profiles that you can configure are known as protocol profiles. The protocol profiles types are:

- Fast L4
- Fast HTTP
- UDP
- SCTP

For each protocol profile type, BIG-IP Local Traffic Manager provides a pre-configured profile with default settings. In most cases, you can use these default profiles as is. If you want to change these settings, you can configure protocol profile settings when you create a profile, or after profile creation by modifying the profile's settings.

To configure and manage protocol profiles, log in to the BIG-IP Configuration utility, and on the Main tab, expand **Local Traffic**, and click **Profiles**.

The Fast L4 profile type

The purpose of a Fast L4 profile is to help you manage Layer 4 traffic more efficiently. When you assign a Fast L4 profile to a virtual server, the Packet Velocity® ASIC (PVA) hardware acceleration within the BIG-IP® system (if supported) can process some or all of the Layer 4 traffic passing through the system. By offloading Layer 4 processing to the PVA hardware acceleration, the BIG-IP system can increase performance and throughput for basic routing functions (Layer 4) and application switching (Layer 7).

You can use a Fast L4 profile with these types of virtual servers: Performance (Layer 4), Forwarding (Layer 2), and Forwarding (IP).

PVA hardware acceleration

Once you implement a Fast L4 profile, Local Traffic Manager™ automatically selects the most efficient PVA hardware acceleration mode for Layer 4 traffic, if PCVA is supported on the specific BIG-IP® platform. Possible modes are Full, Assisted, and None.

The particular hardware acceleration mode that Local Traffic Manager selects depends on these factors:

The Fast L4 profile settings

The mode that the BIG-IP selects is influenced by the way that you configure the settings of the Fast L4 profile.

The virtual server configuration

The mode that Local Traffic Manager selects is influenced by the specific features that you assigned to the virtual server (such as pools, SNAT pools, and iRules®).

A monitor assigned to associated nodes

For full PVA acceleration, you must assign monitors to the relevant nodes.

The value of the PVA Acceleration setting

The PVA Acceleration setting in the Fast L4 profile defines the maximum amount of hardware acceleration that you want to allow, for Layer 4 traffic passing through the virtual server. Therefore, if you set the value to:

- **Full:** The system can set hardware acceleration to any of the three modes (**Full**, **Assisted**, or **None**), depending on the virtual server configuration. This is the default value.
- **Assisted:** The system can set hardware acceleration to either **Assisted** or **None** mode, depending on the virtual server configuration.
- **None:** The system does not perform hardware acceleration.

Depending on the current mode to which hardware acceleration is automatically set, Local Traffic Manager accelerates Layer 4 traffic

Important: If you have a *VLAN* group configured on the BIG-IP system and its **Transparency Mode** setting is set to **Translucent** or **Transparent**, Local Traffic Manager automatically sets the **PVA Acceleration** value to **None**.

The Server Sack, Server Timestamp, and Receive Window settings

The table shown describes three of the Fast L4 profile settings -- Server Sack, Server Timestamp, and Receive Window.

Setting	Description
Server Sack	Specifies whether the BIG-IP system processes Selective ACK (Sack) packets in cookie responses from the server. The default is disabled.
Server Timestamp	Specifies whether the BIG-IP system processes timestamp request packets in cookie responses from the server. The default is disabled.
Receive Window	Specifies the amount of data the BIG-IP system can accept without acknowledging the server. The default value is 0 (zero).

The Fast HTTP profile type

The Fast HTTP profile is a configuration tool designed to speed up certain types of HTTP connections. This profile combines selected features from the TCP Express, HTTP, and OneConnect™ profiles into a single profile that is optimized for the best possible network performance. When you associate this profile with a virtual server, the virtual server processes traffic packet-by-packet, and at a significantly higher speed.

You might consider using a Fast HTTP profile when:

- You do not need features such as remote server authentication, SSL traffic management, and TCP optimizations, nor HTTP features such as data compression, pipelining, and RAM Cache.
- You do not need to maintain source IP addresses.
- You want to reduce the number of connections that are opened to the destination servers.

- The destination servers support connection persistence, that is, HTTP/1.1, or HTTP/1.0 with `Keep-Alive` headers. Note that IIS servers support connection persistence by default.
- You need basic iRule support only (such as limited Layer 4 support and limited HTTP header operations). For example, you can use the iRule events `CLIENT_ACCEPTED`, `SERVER_CONNECTED`, and `HTTP_REQUEST`.

A significant benefit of using a Fast HTTP profile is the way in which the profile supports connection persistence. Using a Fast HTTP profile ensures that for client requests, Local Traffic Manager™ can transform or add an `HTTP Connection` header to keep connections open. Using the profile also ensures that Local Traffic Manager pools any open server-side connections. This support for connection persistence can greatly reduce the load on destination servers by removing much of the overhead caused by the opening and closing of connections.

Note: The Fast HTTP profile is incompatible with all other profile types. Also, you cannot use this profile type in conjunction with VLAN groups, or with the IPv6 address format.

When writing iRules®, you can specify a number of events and commands that the Fast HTTP profile supports.

You can use the default `fasthttp` profile as is, or create a custom Fast HTTP profile.

About TCP profiles

TCP profiles are configuration tools that help you to manage TCP network traffic. Many of the configuration settings of TCP profiles are standard SYSCTL types of settings, while others are unique to Local Traffic Manager™.

TCP profiles are important because they are required for implementing certain types of other profiles. For example, by implementing TCP, HTTP, Rewrite, HTML, and OneConnect™ profiles, along with a persistence profile, you can take advantage of various traffic management features, such as:

- Content spooling, to reduce server load
- OneConnect, to pool idle server-side connections
- Layer 7 session persistence, such as hash or cookie persistence
- iRules® for managing HTTP traffic
- HTTP data compression
- HTTP pipelining
- URI translation
- HTML content modification
- Rewriting of HTTP redirections

The BIG-IP® system includes several pre-configured TCP profiles that you can use as is. In addition to the default `tcp` profile, the system includes TCP profiles that are pre-configured to optimize LAN and WAN traffic, as well as traffic for mobile users. You can use the pre-configured profiles as is, or you can create a custom profile based on a pre-configured profile and then adjust the values of the settings in the profiles to best suit your particular network environment.

To access the full set of TCP profiles, log in to the BIG-IP® BIG-IP Configuration utility and navigate to **Acceleration > Profiles > TCP** or **Local Traffic > Profiles > Protocol > TCP**.

About tcp-lan-optimized profile settings

The `tcp-lan-optimized` profile is a pre-configured profile type that can be associated with a virtual server. In cases where the BIG-IP virtual server is load balancing LAN-based or interactive traffic, you can enhance the performance of your local-area TCP traffic by using the `tcp-lan-optimized` profile.

If the traffic profile is strictly LAN-based, or highly interactive, and a standard virtual server with a TCP profile is required, you can configure your virtual server to use the `tcp-lan-optimized` profile to enhance LAN-based or interactive traffic. For example, applications producing an interactive TCP data flow, such as SSH and TELNET, normally generate a TCP packet for each keystroke. A TCP profile setting such as **Slow Start** can introduce latency when this type of traffic is being processed. By configuring your virtual server to use the **tcp-lan-optimized** profile, you can ensure that the BIG-IP system delivers LAN-based or interactive traffic without delay.

A `tcp-lan-optimized` profile is similar to a TCP profile, except that the default values of certain settings vary, in order to optimize the system for LAN-based traffic.

You can use the `tcp-lan-optimized` profile as is, or you can create another custom profile, specifying the `tcp-lan-optimized` profile as the parent profile.

About tcp-wan-optimized profile settings

The `tcp-wan-optimized` profile is a pre-configured profile type. In cases where the BIG-IP system is load balancing traffic over a WAN link, you can enhance the performance of your wide-area TCP traffic by using the `tcp-wan-optimized` profile.

If the traffic profile is strictly WAN-based, and a standard virtual server with a TCP profile is required, you can configure your virtual server to use a `tcp-wan-optimized` profile to enhance WAN-based traffic. For example, in many cases, the client connects to the BIG-IP virtual server over a WAN link, which is generally slower than the connection between the BIG-IP system and the pool member servers. By configuring your virtual server to use the `tcp-wan-optimized` profile, the BIG-IP system can accept the data more quickly, allowing resources on the pool member servers to remain available. Also, use of this profile can increase the amount of data that the BIG-IP system buffers while waiting for a remote client to accept that data. Finally, you can increase network throughput by reducing the number of short TCP segments that the BIG-IP® system sends on the network.

A `tcp-wan-optimized` profile is similar to a TCP profile, except that the default values of certain settings vary, in order to optimize the system for WAN-based traffic.

You can use the `tcp-wan-optimized` profile as is, or you can create another custom profile, specifying the `tcp-wan-optimized` profile as the parent profile.

About tcp-mobile-optimized profile settings

The `tcp-mobile-optimized` profile is a pre-configured profile type, for which the default values are set to give better performance to service providers' 3G and 4G customers. Specific options in the pre-configured profile are set to optimize traffic for most mobile users, and you can tune these settings to fit your network. For files that are smaller than 1 MB, this profile is generally better than the `mptcp-mobile-optimized` profile. For a more conservative profile, you can start with the `tcp-mobile-optimized` profile, and adjust from there.

Note: Although the pre-configured settings produced the best results in the test lab, network conditions are extremely variable. For the best results, start with the default settings and then experiment to find out what works best in your network.

This list provides guidance for relevant settings

- Set the **Proxy Buffer Low** to the **Proxy Buffer High** value minus 64 KB. If the **Proxy Buffer High** is set to less than 64K, set this value at 32K.
- The size of the **Send Buffer** ranges from 64K to 350K, depending on network characteristics. If you enable the **Rate Pace** setting, the send buffer can handle over 128K, because rate pacing eliminates some of the burstiness that would otherwise exist. On a network with higher packet loss, smaller buffer sizes perform better than larger. The number of loss recoveries indicates whether this setting should be tuned higher or lower. Higher loss recoveries reduce the goodput.
- Setting the **Keep Alive Interval** depends on your fast dormancy goals. The default setting of 1800 seconds allows the phone to enter low power mode while keeping the flow alive on intermediary devices. To prevent the device from entering an idle state, lower this value to under 30 seconds.
- The **Congestion Control** setting includes delay-based and hybrid algorithms, which might better address TCP performance issues better than fully loss-based congestion control algorithms in mobile environments. The Illinois algorithm is more aggressive, and can perform better in some situations, particularly when object sizes are small. When objects are greater than 1 MB, goodput might decrease with Illinois. In a high loss network, Illinois produces lower goodput and higher retransmissions.
- For 4G LTE networks, specify the **Packet Loss Ignore Rate** as 0. For 3G networks, specify 2500. When the **Packet Loss Ignore Rate** is specified as more than 0, the number of retransmitted bytes and receives SACKs might increase dramatically.
- For the **Packet Loss Ignore Burst** setting, specify within the range of 6–12, if the **Packet Loss Ignore Rate** is set to a value greater than 0. A higher **Packet Loss Ignore Burst** value increases the chance of unnecessary retransmissions.
- For the **Initial Congestion Window Size** setting, round trips can be reduced when you increase the initial congestion window from 0 to 10 or 16.
- Enabling the **Rate Pace** setting can result in improved goodput. It reduces loss recovery across all congestion algorithms, except Illinois. The aggressive nature of Illinois results in multiple loss recoveries, even with rate pacing enabled.

A tcp-mobile-optimized profile is similar to a TCP profile, except that the default values of certain settings vary, in order to optimize the system for mobile traffic.

You can use the tcp-mobile-optimized profile as is, or you can create another custom profile, specifying the tcp-mobile-optimized profile as the parent profile.

About mptcp-mobile-optimized profile settings

The mptcp-mobile-optimized profile is a pre-configured profile type for use in reverse proxy and enterprise environments for mobile applications that are front-ended by a BIG-IP® system. This profile provides a more aggressive starting point than the tcp-mobile-optimized profile. It uses newer congestion control algorithms and a newer TCP stack, and is generally better for files that are larger than 1 MB. Specific options in the pre-configured profile are set to optimize traffic for most mobile users in this environment, and you can tune these settings to accommodate your network.

Note: Although the pre-configured settings produced the best results in the test lab, network conditions are extremely variable. For the best results, start with the default settings and then experiment to find out what works best in your network.

The enabled **Multipath TCP (MPTCP)** option provides more bandwidth and higher network utilization. It allows multiple client-side flows to connect to a single server-side flow. MPTCP automatically and quickly

adjusts to congestion in the network, moving traffic away from congested paths and toward uncongested paths.

The **Congestion Control** setting includes delay-based and hybrid algorithms, which may better address TCP performance issues better than fully loss-based congestion control algorithms in mobile environments. Refer to the online help descriptions for assistance in selecting the setting that corresponds to your network conditions.

The enabled **Rate Pace** option mitigates bursty behavior in mobile networks and other configurations. It can be useful on high latency or high BDP (bandwidth-delay product) links, where packet drop is likely to be a result of buffer overflow rather than congestion.

An `mptcp-mobile-optimized` profile is similar to a TCP profile, except that the default values of certain settings vary, in order to optimize the system for mobile traffic.

You can use the `mptcp-mobile-optimized` profile as is, or you can create another custom profile, specifying the `mptcp-mobile-optimized` profile as the parent profile.

The UDP profile type

The UDP profile is a configuration tool for managing UDP network traffic.

Because the BIG-IP[®] system supports the OpenSSL implementation of datagram Transport Layer Security (TLS), you can optionally assign both a UDP and a Client SSL profile to certain types of virtual servers.

The SCTP profile type

Local Traffic Manager[™] includes a profile type that you can use to manage Stream Control Transmission Protocol (SCTP) traffic. *Stream Control Transmission Protocol (SCTP)* is a general-purpose, industry-standard transport protocol, designed for message-oriented applications that transport signalling data. The design of SCTP includes appropriate congestion-avoidance behavior, as well as resistance to flooding and masquerade attacks.

Unlike TCP, SCTP includes the ability to support several streams within a connection. While a *TCP stream* refers to a sequence of bytes, an *SCTP stream* represents a sequence of messages.

You can use SCTP as the transport protocol for applications that require monitoring and detection of session loss. For such applications, the SCTP mechanisms to detect session failure actively monitor the connectivity of a session.

The Any IP profile type

With the Any IP profile, you can enforce an idle timeout value on IP traffic other than TCP and UDP traffic. You can use the BIG-IP Configuration utility to create, view details for, or delete Any IP profiles.

When you configure an idle timeout value, you specify the number of seconds for which a connection is idle before the connection is eligible for deletion. The default is 60 seconds. Possible values that you can configure are:

Specify

Specifies the number of seconds that the Any IP connection is to remain idle before it can be deleted. When you select **Specify**, you must also type a number in the box.

Immediate

Specifies that you do not want the connection to remain idle, and that it is therefore immediately eligible for deletion.

Indefinite

Specifies that Any IP connections can remain idle indefinitely.

Remote Server Authentication Profiles

Introduction to authentication profiles

A significant feature of BIG-IP® Local Traffic Manager™ is its ability to support Pluggable Authentication Module (PAM) technology. *PAM* technology allows you to choose from a number of different authentication and authorization schemes to use to authenticate or authorize network traffic.

The goal of PAM technology is to separate an application, such as the BIG-IP system, from its underlying authentication technology. This means that you can dictate the particular authentication/authorization technology that you want the BIG-IP system to use to authenticate application traffic coming into the BIG-IP system.

To this end, Local Traffic Manager offers several authentication schemes, known as authentication modules. These *authentication modules* allow you to use a remote system to authenticate or authorize application requests that pass through the BIG-IP system.

The BIG-IP system normally routes remote authentication traffic through a Traffic Management Microkernel (TMM) switch interface (that is, an interface associated with a VLAN and a self IP address), rather than through the management interface. Therefore, if the TMM service is stopped for any reason, remote authentication is not available until the service is running again.

BIG-IP system authentication modules

Local Traffic Manager™ authentication modules that you can implement for remote authentication are:

Lightweight Directory Access Protocol (LDAP)

Local Traffic Manager can authenticate or authorize network traffic using data stored on a remote LDAP server or a Microsoft® Windows® Active Directory® server. Client credentials are based on basic HTTP authentication (user name and password).

Remote Authentication Dial-In User Service (RADIUS)

Local Traffic Manager can authenticate network traffic using data stored on a remote RADIUS server. Client credentials are based on basic HTTP authentication (user name and password).

TACACS+

Local Traffic Manager can authenticate network traffic using data stored on a remote TACACS+ server. Client credentials are based on basic HTTP authentication (user name and password).

SSL client certificate LDAP

Local Traffic Manager can authorize network traffic using data stored on a remote LDAP server. Client credentials are based on SSL certificates, as well as defined user groups and roles.

Online Certificate Status Protocol (OCSP)

Local Traffic Manager can check on the revocation status of a client certificate using data stored on a remote OCSP server. Client credentials are based on SSL certificates.

Certificate Revocation List Distribution Point (CRLDP)

Local Traffic Manager can use CRL distribution points to determine revocation status.

Kerberos Delegation

Local Traffic Manager can authenticate application traffic when you are using Microsoft Windows Integrated Authentication.

Important: When you create remote authentication objects and profiles, the BIG-IP® system places them into your current administrative partition. Note that the default profile always resides in partition *Common*.

The LDAP authentication module

An *LDAP authentication module* is a mechanism for authenticating or authorizing client connections passing through a BIG-IP® system. This module is useful when your authentication or authorization data is stored on a remote LDAP server or a Microsoft Windows Active Directory server, and you want the client credentials to be based on basic HTTP authentication (that is, user name and password).

With the LDAP authentication module, Local Traffic Manager™ can indicate that the authentication was a success or failure, or that the LDAP server needs a credential of some sort.

Additionally, the system can take some action based on certain information that the server returns in the LDAP query response. For example, LDAP response information can indicate the user's group membership, or it can indicate that the user's password has expired. To configure Local Traffic Manager to return specific data in an LDAP response, you can write an iRule, using the commands `AUTH::subscribe`, `AUTH::unsubscribe`, and `AUTH::response_data`. For more information, see the F5 Networks DevCentral web site, <http://devcentral.f5.com>.

The RADIUS authentication module

A *RADIUS authentication module* is a mechanism for authenticating client connections passing through a BIG-IP® system. You use this module when your authentication data is stored on a remote RADIUS server. In this case, client credentials are based on basic HTTP authentication (that is, user name and password).

The TACACS+ authentication module

A *TACACS+ authentication module* is a mechanism for authenticating client connections passing through a BIG-IP® system. You use this module when your authentication data is stored on a remote TACACS+ server. In this case, client credentials are based on basic HTTP authentication (that is, user name and password).

The SSL client certificate LDAP authentication module

An *SSL client certificate LDAP authentication module* is a mechanism for authorizing client connections passing through a BIG-IP® system. With the SSL client certificate LDAP authentication module, you can use a remote LDAP server to impose access control on application traffic. The module bases this access control on SSL certificates, as well as user groups and roles that you specify.

With the SSL client certificate LDAP authentication module, Local Traffic Manager™ can indicate that the authorization was a success or failure, or that the LDAP server needs a credential of some sort.

Additionally, the system can take some action based on certain information that the server returns in the LDAP query response. For example, LDAP response information can indicate the user's group membership, or it can indicate that the user's password has expired. To configure Local Traffic Manager to return specific data in an LDAP response, you can write an iRule, using the commands `AUTH::subscribe`, `AUTH::unsubscribe`, and `AUTH::response_data`. For more information, see the F5 Networks DevCentral web site, <http://devcentral.f5.com>.

Search results and corresponding authorization status

This table shows the results of certificate-based authorization being performed.

Result of search	Authorization status
No records match	Authorization fails
One record matches	Authorization succeeds and is subject to groups and roles
Two or more records match	Authorization fails, due to invalid database entries

SSL client certificate authorization

Before you can implement an SSL client certificate LDAP module, you must understand the two different types of credentials that the BIG-IP® system uses to authorize application traffic using data on a remote LDAP server. These two types of credentials are:

- SSL certificates
- Groups and roles

With SSL client certificate LDAP authorization, Local Traffic Manager™ can authorize clients based on signed client certificates issued by trusted CAs. Then, to further enhance the ability of the system to authorize client requests, you can also specify groups and roles. Basing authorization on certificates as well as groups and roles provides the flexibility you need to control client access to system resources.

SSL certificates for LDAP authorization

During the process of authorizing a client, Local Traffic Manager™ must search the LDAP database. When using certificate-based authorization, the system can search the LDAP database in three ways:

User

If certificates are not stored in the LDAP database, you can configure the system to extract a user name from the certificate presented as part of the incoming client request. The system then checks to see if an entry for the user exists in the LDAP database. This scenario is a good choice for a company that acts as its own Certificate Authority, where the company is assured that if the certificate is verified, then the user is authorized.

Certificate Map

If you create an object and class that map certificates to users in the LDAP database, you can then configure the system to search for a certificate in the map, and retrieve a user from that map. The system then checks to ensure that the user in the LDAP database is a valid user.

Certificate

Many LDAP server environments already incorporate certificates into the user information stored in the LDAP database. One way of configuring authorization in LDAP server environments is to configure the system to compare an incoming certificate to the certificate stored in the LDAP database for the user associated with the client request. If the certificate is found in the user's LDAP profile, access is granted to the user, and the request is granted.

Groups and roles for LDAP authorization

In addition to enabling certificate-based authorization, you can also configure authorization based on groups and roles.

Groups

Because LDAP servers already have the concept and structure of groups built into them, Local Traffic Manager™ can include groups in its authorization feature. To enable the use of groups for authorization purposes, you must indicate the base and scope under which the system will search for groups in the LDAP database. Also, you must specify setting values for a group name and a member name. Once you have completed these tasks, the system can search through the list of valid groups until a group is found that has the current user as a member.

Roles

Unlike a group, a role is a setting directly associated with a user. Any role-based authorization that Local Traffic Manager (LTM®) performs depends on the LDAP database having the concept of roles built into it. To determine if a user should be granted access to a resource, LTM searches through the roles assigned to the user and attempts to match that role to a valid role defined by the administrator.

The SSL OCSP authentication module

An *SSL OCSP authentication module* is a mechanism for authenticating client connections passing through a BIG-IP® system. More specifically, an SSL OCSP authentication module checks the revocation status of an SSL certificate, as part of authenticating that certificate.

Online Certificate Status Protocol (OCSP) is a third-party software application and industry-standard protocol that offers an alternative to a certificate revocation list (CRL) when using public-key technology. A CRL is a list of revoked client certificates, which a server system can check during the process of verifying a client certificate.

You implement an SSL OCSP authentication module when you want to use OCSP instead of a CRL as the mechanism for checking the revocation status of SSL certificates.

Local Traffic Manager™ supports both CRLs and the OCSP protocol. If you want to use CRLs instead of OCSP, you configure an SSL profile.

For more information on OCSP, see RFC 2560 at URL <http://www.ietf.org>.

Using OCSP to check on the revocation status of client certificates offers distinct advantages over the use of a CRL.

The limitations of CRLs

When presented with a client certificate, Local Traffic Manager sometimes needs to assess the revocation state of that certificate before accepting the certificate and forwarding the connection to a target server. The standard method of assessing revocation status is a CRL, which is stored in a separate CRL file on each machine in your configuration. Although CRLs are considered to be a standard way of checking revocation

status of SSL certificates, a CRL is updated only at fixed intervals, thus presenting a risk that the information in the CRL is outdated at the time that the status check occurs.

Also, having to store a separate CRL file on each machine presents other limitations:

- All CRL files must be kept in sync.
- Having a separate CRL file on each machine poses a security risk.
- Multiple CRL files cannot be administered from a central location.

The benefits of OCSP

OCSP ensures that Local Traffic Manager always obtains real-time revocation status during the certificate verification process.

OCSP is based on a client/server model, where a client system requests revocation status of a certificate, and a server system sends the response. Thus, when you implement the SSL OCSP authentication module, the BIG-IP system acts as the OCSP client, and an external system, known as an OCSP responder, acts as the OCSP server. An *OCSP responder* is therefore an external server that sends certificate revocation status, upon request, to the BIG-IP system.

How OCSP works

In general, after receiving an SSL certificate from a client application, the BIG-IP system (acting as an OCSP client) requests certificate revocation status from an OCSP responder, and then blocks the connection until it receives status from that responder. If the status from the responder shows that the certificate is revoked, Local Traffic Manager rejects the certificate and denies the connection. If the status from the responder shows that the certificate is still valid, Local Traffic Manager continues with its normal certificate verification process to authenticate the client application.

More specifically, when an application client sends a certificate for authentication, the BIG-IP system follows this process:

- First, the BIG-IP system checks that the signer of the certificate is listed in the trusted CAs file.
- If the certificate is listed, the BIG-IP system then checks to see if the certificate is revoked. Without OCSP, if the CRL option is configured on the BIG-IP system, the BIG-IP system checks revocation status by reading the certificate revocation list (CRL). With OCSP, however, the BIG-IP system bypasses the CRL and prepares to send a revocation status request to the appropriate OCSP responder.
- Next, the BIG-IP system queries the first responder, even if the responder does not match the certificate authority that signed the client certificate. However, if the first responder fails with a connection error, the BIG-IP system queries the next responder.
- Next, the BIG-IP system attempts to match that CA with a CA listed in an SSL OCSP profile.
- If a match exists, the BIG-IP system checks the target URL within the client certificate's AuthorityInfoAccess (AIA) field (if the field exists), and uses that URL to send the request for certificate revocation status to the OCSP responder.
- If the `Ignore AIA` parameter is enabled within the SSL OCSP profile, then the BIG-IP system instead uses the URL specified in the `url` parameter of the matching SSL OCSP profile to send the request for certificate revocation status.
- If no match exists, the BIG-IP system checks the `calist` setting of another SSL OCSP profile defined on the system. If all SSL OCSP profiles are checked and no match is found, the certificate verification fails, and the BIG-IP system denies the original client request.
- Once the BIG-IP system has received certificate revocation status from a responder, the BIG-IP system, when configured to do so, inserts a certificate status header into the original client request. The name of the certificate status header is `SSLClientCertificateStatus`.

You must then assign the OCSP profile to a virtual server.

A single SSL OCSP profile can target a specific responder, or multiple SSL OCSP profiles can target the same responder. Each responder itself is associated with a certificate authority (CA), and multiple responders can be associated with the same CA.

Note: *Local Traffic Manager allows you to enable both the CRL and the OCSP options. Most users need to enable either one or the other, but not both. However, in the rare case that you want to enable both options, be aware that both the search of the CRL file, and the connection to the responder must be successful. Otherwise, Local Traffic Manager cannot obtain status.*

Note that an OCSP responder object is an object that you create that includes a URL for an external OCSP responder. You must create a separate OCSP responder object for each external OCSP responder.

When you subsequently create an OCSP configuration object, the configuration object contains a reference to any OCSP responder objects that you have created.

The CRLDP authentication module

A CRLDP authentication module is a mechanism for authenticating client connections passing through a BIG-IP® system. You implement a CRLDP authentication module when you want the BIG-IP system to use CRL distribution points as the mechanism for checking the revocation status of SSL certificates.

This CRLDP authentication feature is based on a technology known as *Certificate Revocation List Distribution Points (CRLDP)*. CRLDP is an industry-standard protocol that offers an alternative method for checking a standard certificate revocation list (CRL) to determine revocation status. CRLDP is also an alternative to using Online Certificate Status Protocol (OCSP).

A CRLDP authentication module uses CRL distribution points to check the revocation status of an SSL certificate, as part of authenticating that certificate. *CRL distribution points* are a mechanism used to distribute certificate revocation information across a network. More specifically, a distribution point is a Uniform Resource Identifier (URI) or directory name specified in a certificate that identifies how the server obtains CRL information. Distribution points can be used in conjunction with CRLs to configure certificate authorization using any number of LDAP servers.

The Kerberos Delegation authentication module

A *Kerberos Delegation authentication module* is a mechanism for authenticating client connections passing through a BIG-IP® system. You can use this module when you are using Microsoft Windows Integrated Authentication to authenticate application traffic.

The Kerberos Delegation module obtains delegated Kerberos credentials for the client principal, and then retrieves Kerberos credentials for the server-side principal. The Kerberos Delegation module essentially acts as a proxy for Kerberos credentials. That is, when connecting to a server that is inside its domain, the browser client fetches Kerberos credentials. These credentials, known as *delegated credentials*, are passed to the BIG-IP system, which in turn retrieves credentials for the real server that is on the back end, and passes those credentials back to the client.

Other Profiles

Introduction to other profiles

In addition to the profiles described in previous chapters, you can configure these BIG-IP® Local Traffic Manager™ profiles:

- OneConnect™
- NTLM
- Statistics
- Stream

For each profile type, Local Traffic Manager provides a pre-configured profile with default settings. In most cases, you can use these default profiles as is. If you want to change these settings, you can configure profile settings when you create a profile, or after profile creation by modifying the profile's settings.

About OneConnect profiles

The OneConnect profile type implements the BIG-IP® system's OneConnect feature. This feature can increase network throughput by efficiently managing connections created between the BIG-IP system and back-end pool members. You can use the OneConnect feature with any TCP-based protocol, such as HTTP or RTSP.

How does OneConnect work?

The *OneConnect* feature works with request headers to keep existing server-side connections open and available for reuse by other clients. When a client makes a new connection to a virtual server configured with a OneConnect profile, the BIG-IP system parses the request, selects a server using the load-balancing method defined in the pool, and creates a connection to that server. When the client's initial request is complete, the BIG-IP system temporarily holds the connection open and makes the idle TCP connection to the pool member available for reuse.

When another connection is subsequently initiated to the virtual server, if an existing server-side flow to the pool member is open and idle, the BIG-IP system applies the OneConnect source mask to the IP address in the request to determine whether the request is eligible to reuse the existing idle connection. If the request is eligible, the BIG-IP system marks the connection as non-idle and sends a client request over that connection. If the request is not eligible for reuse, or an idle server-side flow is not found, the BIG-IP system creates a new server-side TCP connection and sends client requests over the new connection.

Note: The BIG-IP system can pool server-side connections from multiple virtual servers if those virtual servers reference the same OneConnect profile and the same pool. Also, the re-use of idle connections can cause the BIG-IP system to appear as though the system is not load balancing traffic evenly across pool members.

About client source IP addresses

The standard address translation mechanism on the BIG-IP system translates only the destination IP address in a request and not the source IP address (that is, the client node's IP address). However, when the OneConnect feature is enabled, allowing multiple client nodes to re-use a server-side connection, the source IP address in the header of each client node's request is always the IP address of the client node that initially opened the server-side connection. Although this does not affect traffic flow, you might see evidence of this when viewing certain types of system output.

The OneConnect profile settings

When configuring a OneConnect profile, you specify this information:

Source mask

The mask applied to the source IP address to determine the connection's eligibility to reuse a server-side connection.

Maximum size of idle connections

The maximum number of idle server-side connections kept in the connection pool.

Maximum age before deletion from the pool

The maximum number of seconds that a server-side connection is allowed to remain before the connection is deleted from the connection pool.

Maximum reuse of a connection

The maximum number of requests to be sent over a server-side connection. This number should be slightly lower than the maximum number of HTTP *Keep-Alive* requests accepted by servers in order to prevent the server from initiating a connection close action and entering the `TIME_WAIT` state.

Idle timeout override

The maximum time that idle server-side connections are kept open. Lowering this value may result in a lower number of idle server-side connections, but may increase request latency and server-side connection rate.

OneConnect and HTTP profiles

Content switching for HTTP requests

When you assign both a OneConnect profile and an HTTP profile to a virtual server, and an HTTP client sends multiple requests within a single connection, the BIG-IP system can process each HTTP request individually. The BIG-IP system sends the HTTP requests to different destination servers as determined by the load balancing method. Without a OneConnect profile enabled for the HTTP virtual server, the BIG-IP system performs load-balancing only once for each TCP connection.

HTTP version considerations

For HTTP traffic to be eligible to use the OneConnect feature, the web server must support HTTP *Keep-Alive* connections. The version of the HTTP protocol you are using determines to what extent this support is available. The BIG-IP system therefore includes a *OneConnect transformations* feature within the HTTP profile, specifically designed for use with HTTP/1.0 which by default does not enable *Keep-Alive* connections. With the OneConnect transformations feature, the BIG-IP system can transform HTTP/1.0 connections into HTTP/1.1 requests on the server side, thus allowing those connections to remain open for reuse.

The two different versions of the HTTP protocol treat *Keep-Alive* connections in these ways:

HTTP/1.1 requests

HTTP Keep-Alive connections are enabled by default in HTTP/1.1. With HTTP/1.1 requests, the server does not close the connection when the content transfer is complete, unless the client sends a `Connection: close` header in the request. Instead, the connection remains active in anticipation of the client reusing the same connection to send additional requests. For HTTP/1.1 requests, you do not need to use the OneConnect transformations feature.

HTTP/1.0 requests

HTTP Keep-Alive connections are not enabled by default in HTTP/1.0. With HTTP/1.0 requests, the client typically sends a `Connection: close` header to close the TCP connection after sending the request. Both the server and client-side connections that contain the `Connection: close` header are closed once the response is sent. When you assign a OneConnect profile to a virtual server, the BIG-IP system transforms `Connection: close` headers in HTTP/1.0 client-side requests to `X-Connection: close` headers on the server side, thereby allowing a client to reuse an existing connection to send additional requests.

OneConnect and NTLM profiles

NT Lan Manager (NTLM) HTTP 401 responses prevent the BIG-IP® system from detaching the server-side connection. As a result, a late FIN from a previous client connection might be forwarded to a new client that re-used the connection, causing the client-side connection to close before the NTLM handshake completes. If you prefer NTLM authentication support when using the OneConnect feature, you should configure an NTLM profile in addition to the OneConnect profile.

OneConnect and SNATs

When a client makes a new connection to a virtual server that is configured with a OneConnect profile and a source network address translation (SNAT) object, the BIG-IP system parses the HTTP request, selects a server using the load-balancing method defined in the pool, translates the source IP address in the request to the SNAT IP address, and creates a connection to the server. When the client's initial HTTP request is complete, the BIG-IP system temporarily holds the connection open and makes the idle TCP connection to the pool member available for reuse. When a new connection is initiated to the virtual server, the BIG-IP system performs SNAT address translation on the source IP address and then applies the OneConnect source mask to the translated SNAT IP address to determine whether it is eligible to reuse an existing idle connection.

About NTLM profiles

NT LAN Manager (NTLM) is an industry-standard technology that uses an encrypted challenge/response protocol to authenticate a user without sending the user's password over the network. Instead, the system requesting authentication performs a calculation to prove that the system has access to the secured NTLM credentials. NTLM credentials are based on data such as the domain name and user name, obtained during the interactive login process.

The NTLM profile within BIG-IP® Local Traffic Manager™ optimizes network performance when the system is processing NT LAN Manager traffic. When both an NTLM profile and a OneConnect™ profile are associated with a virtual server, the local traffic management system can take advantage of server-side connection pooling for NTLM connections.

How does the NTLM profile work?

When the NTLM profile is associated with a virtual server and the server replies with the HTTP 401 Unauthorized HTTP response message, the NTLM profile inserts a cookie, along with additional profile options, into the HTTP response. The information is encrypted with a user-supplied passphrase and associated with the serverside flow. Further client requests are allowed to reuse this flow only if they present the NTLMConnPool cookie containing the matching information. By using a cookie in the NTLM profile, the BIG-IP system does not need to act as an NTLM proxy, and returning clients do not need to be re-authenticated.

The NTLM profile works by parsing the HTTP request containing the NTLM type 3 message and securely storing the following pieces of information (aside from those which are disabled in the profile):

- User name
- Workstation name
- Target server name
- Domain name
- Cookie previously set (cookie name supplied in the profile)
- Source IP address

With the information safely stored, the BIG-IP system can then use the data as a key when determining which clientside requests to associate with a particular serverside flow. You can configure this using the NTLM profile options. For example, if a server's resources can be openly shared by all users in that server's domain, then you can enable the Key By NTLM Domain setting, and all serverside flows from the users of the same domain can be pooled for connection reuse without further authentication. Or, if a server's resources can be openly shared by all users originating from a particular IP address, then you can enable the Key By Client IP Address setting and all serverside flows from the same source IP address can be pooled for connection reuse.

The Statistics profile type

The Statistics profile provides user-defined statistical counters. Each profile contains 32 settings (Field1 through Field32), which define named counters. Using a Tcl-based iRule command, you can use the names to manipulate the counters while processing traffic.

For example, you can create a profile named `my_stats`, which assigns the counters `tot_users`, `cur_users`, and `max_users` to the profile settings **Field1**, **Field2**, and **Field3** respectively. You can then write an iRule named `track_users`, and then assign the `my_stats` profile and the `track_users` iRule to a virtual server named `stats-1`.

In this example, the counter `tot_users` counts the total number of connections, the counter `cur_users` counts the current number of connections, and the counter `max_users` retains the largest value of the counter `cur_users`.

```
profile stats my_stats {
  defaults from stats
  field1 tot_users
  field2 cur_users
  field3 max_users
}

rule track_users {
  when CLIENT_ACCEPTED {
```

```

        STATS::incr my_stats tot_users
        STATS::setmax my_stats max_users [STATS::incr my_stats cur_users]
    }
}

virtual stats-1 {
    destination 10.10.55.66:http
    ip protocol tcp
    profile http my_stats tcp
    pool pool1
    rule track_users
}

```

The Stream profile type

You can use the *Stream profile* to search and replace strings within a data stream, such as a TCP connection.

Note that list types are case-sensitive for pattern strings. For example, the system treats the pattern string `www.f5.com` differently from the pattern string `www.F5.com`. You can override this case sensitivity by using the Linux `regex` command.

The Request Logging profile type

A *Request Logging profile* gives you the ability to configure data within a log file for HTTP requests and responses, according to parameters that you specify.

The DNS Logging profile type

A DNS logging profile gives you the ability to log DNS queries and responses, according to parameters that you specify.

Health and Performance Monitoring

Introduction to health and performance monitoring

BIG-IP® Local Traffic Manager™ can monitor the health or performance of either pool members or nodes. Local Traffic Manager supports these methods of monitoring:

Simple monitoring

Simple monitoring merely determines whether the status of a node is up or down. Simple monitors do not monitor pool members (and therefore, individual protocols, services, or applications on a node), but only the node itself. The system contains two types of simple monitors, ICMP and TCP_ECHO.

Active monitoring

Active monitoring checks the status of a pool member or node on an ongoing basis, at a set interval. If a pool member or node being checked does not respond within a specified timeout period, or the status of a node indicates that performance is degraded, Local Traffic Manager can redirect the traffic to another pool member or node. There are many types of active monitors. Each type of active monitor checks the status of a particular protocol, service, or application. For example, one type of monitor is HTTP. An HTTP type of monitor allows you to monitor the availability of the HTTP service on a pool, pool member, or node. A WMI type of monitor allows you to monitor the performance of a node that is running the Windows Management Instrumentation (WMI) software. Active monitors fall into two categories: Extended Content Verification (ECV) monitors, and Extended Application Verification (EAV) monitors.

Note: If you configure a performance monitor, such as the SNMP DCA or WMI monitor type, you should also configure a health monitor. Configuring a health monitor ensures that Local Traffic Manager reports accurate node availability status.

Passive monitoring

Passive monitoring occurs as part of a client request. This kind of monitoring checks the health of a pool member based on a specified number of connection attempts or data request attempts that occur within a specified time period. If, after the specified number of attempts within the defined interval, the system cannot either connect to the server or receive a response, or if the system receives a bad response, the system marks the pool member as down. There is only one type of passive monitor, called an *Inband* monitor.

Comparison of monitoring methods

In the short description, briefly describe the purpose and intent of the information contained in this topic. This element is an F5® requirement.

Monitoring Method	Benefits	Constraints
Simple	<ul style="list-style-type: none">• Works well when you only need to determine the up or down status of a node.	<ul style="list-style-type: none">• Can check the health of a node only, and not a pool member.

Monitoring Method	Benefits	Constraints
Active	<ul style="list-style-type: none"> • Can check for specific responses • Can run with or without client traffic 	<ul style="list-style-type: none"> • Creates additional network traffic beyond the client request and server response • Can be slow to mark a pool member as down
Passive	<ul style="list-style-type: none"> • Creates no additional network traffic beyond the client request and server response • Can mark a pool member as down quickly, as long as there is some amount of network traffic 	<ul style="list-style-type: none"> • Cannot check for specific responses • Can potentially be slow to mark a pool member as up

About monitor settings

Every monitor consists of settings with values. The settings and their values differ depending on the type of monitor. In some cases, Local Traffic Manager™ assigns default values. This example shows that an ICMP-type monitor has these settings and default values.

The settings specify that an ICMP type of monitor is configured to check the status of an IP address every five seconds, and to time out every 16 seconds. The destination IP address that the monitor checks is specified by the Alias Address setting, with the value * All Addresses. Thus, in the example, all IP addresses with which the monitor is associated are checked.

```
Name my_icmp
Type ICMP
Interval 5
Timeout 16
Transparent No
Alias Address * All Addresses
```

Overview of monitor implementation

You implement monitors using either the BIG-IP Configuration utility or a command line utility. The task of implementing a monitor varies depending on whether you are using a pre-configured monitor or creating a custom monitor. A *pre-configured monitor* is an existing monitor that Local Traffic Manager™ provides for you, with its settings already configured. A *custom monitor* is a monitor that you create based on one of the allowed monitor types.

If you want to implement a pre-configured monitor, you need only associate the monitor with a pool, pool member, or node, and then configure the virtual server to reference the relevant pool. If you want to implement a custom monitor, you must first create the custom monitor. Then you can associate the custom monitor with a pool, pool member, or node, and configure the virtual server to reference the pool.

Pre-configured monitors

For a subset of monitor types, Local Traffic Manager™ includes a set of pre-configured monitors. You cannot modify pre-configured monitor settings, as they are intended to be used as is. The purpose of a

pre-configured monitor is to eliminate the need for you to explicitly create a monitor. You use a pre-configured monitor when the values of the settings meet your needs as is.

The names of the pre-configured monitors that Local Traffic Manager includes are:

- gateway_icmp
- http
- https
- https_443
- icmp
- inband
- real_server
- snmp_dca
- tcp
- tcp_echo

An example of a pre-configured monitor is the `icmp` monitor. The example shows the `icmp` monitor, with values configured for its **Interval**, **Timeout**, and **Alias Address** settings. Note that the Interval value is 5, the Timeout value is 16, the Transparent value is No, and the Alias Address value is * All Addresses.

If the Interval, Timeout, Transparent, and Alias Address values meet your needs, you simply assign the `icmp` pre-configured monitor directly to a pool, pool member, or node, using the Pools or Nodes screens within the BIG-IP Configuration utility. In this case, you do not need to use the Monitors screens, unless you simply want to view the values of the pre-configured monitor settings.

```
Name icmp
Type ICMP
Interval 5
Timeout 16
Transparent No
Alias Address * All Addresses
```

Important: All pre-configured monitors reside in partition `Common`.

Custom monitors

You create a custom monitor when the values defined in a pre-configured monitor do not meet your needs, or no pre-configured monitor exists for the type of monitor you are creating.

When you create a custom monitor, you use the BIG-IP Configuration utility or a command line utility to: give the monitor a unique name, specify a monitor type, and, if a monitor of that type already exists, import settings and their values from the existing monitor. You can then change the values of any imported settings.

You must base each custom monitor on a monitor type. When you create a monitor, the BIG-IP Configuration utility displays a list of monitor types. To specify a monitor type, simply choose the one that corresponds to the service you want to check. For example, if you want to create a monitor that checks the health of the HTTP service on a pool, you choose HTTP as the monitor type.

If you want to check more than one service on a pool or pool member (for example HTTP and HTTPS), you can associate more than one monitor on that pool or pool member.

Checking services is not the only reason for implementing a monitor. If you want to verify only that the destination IP address is alive, or that the path to it through a transparent node is alive, use one of the simple monitors, `icmp` or `tcp_echo`. Or, if you want to verify TCP only, use the monitor `tcp`.

Importing settings from a pre-configured monitor

If a pre-configured monitor exists that corresponds to the type of custom monitor you are creating, you can import the settings and values of that pre-configured monitor into the custom monitor. You are then free to change those setting values to suit your needs. For example, if you create a custom monitor called `my_icmp`, the monitor can inherit the settings and values of the pre-configured monitor `icmp`. This ability to import existing setting values is useful when you want to retain some setting values for your new monitor but modify others.

The example shows a custom ICMP-type monitor called `my_icmp`, which is based on the pre-configured monitor `icmp`. Note that the Interval value is changed to 10, and the Timeout value is 20. The other settings retain the values defined in the pre-configured monitor.

```
Name my_icmp
Type ICMP
Interval 10
Timeout 20
Transparent No
Alias Address * All Addresses
```

Importing settings from a custom monitor

You can import settings from another custom monitor instead of from a pre-configured monitor. This is useful when you would rather use the setting values defined in another custom monitor, or when no pre-configured monitor exists for the type of monitor you are creating. For example, if you create a custom monitor called `my_oracle_server2`, you can import settings from an existing Oracle-type monitor such as `my_oracle_server1`. In this case, because Local Traffic Manager™ does not provide a pre-configured Oracle-type monitor, a custom monitor is the only kind of monitor from which you can import setting values.

Selecting a monitor is straightforward. Like `icmp`, each of the monitors has a Type setting based on the type of service it checks, for example, `http`, `https`, `ftp`, `pop3`, and takes that type as its name. (Exceptions are port-specific monitors, like the `external` monitor, which calls a user-supplied program.)

Monitor destinations

By default, the value for the **Alias Address** setting in the monitors is set to the wildcard `* Addresses`, and the **Alias Service Port** setting is set to the wildcard `* Ports`. This value causes the monitor instance created for a pool, pool member, or node to take that node's address or address and port as its destination. You can, however, replace either or both wildcard symbols with an explicit destination value, by creating a custom monitor. An explicit value for the **Alias Address** and/or **Alias Service Port** setting is used to force the instance destination to a specific address and/or port which might not be that of the pool, pool member, or node.

The ECV monitor types HTTP, HTTPS, and TCP include the settings **Send String** and **Receive String** for the send string and receive expression, respectively.

The most common **Send String** value is `GET /`, which retrieves a default HTML page for a web site. To retrieve a specific page from a web site, you can enter a **Send String** value that is a fully qualified path name:

```
"GET /www/support/customer_info_form.html"
```

The **Receive String** value is the text string that the monitor looks for in the returned resource. The most common **Receive String** values contain a text string that is included in a particular HTML page on your site. The text string can be regular text, HTML tags, or image names.

The sample **Receive String** value below searches for a standard HTML tag:

```
"<HEAD>"
```

You can also use the default null **Receive String** value [""]. In this case, any content retrieved is considered a match. If both the **Send String** and **Receive String** fields are left empty, only a simple connection check is performed.

For HTTP and FTP monitor types, you can use the special values `GET` or `hurl` in place of `Send String` and `Receive String` values. For FTP monitors specifically, the `GET` value should specify the full path to the file to retrieve.

Transparent and Reverse modes

The normal and default behavior for a monitor is to ping the destination pool, pool member, or node by an unspecified route, and to mark the node up if the test is successful. However, with certain monitor types, you can specify a route through which the monitor pings the destination server. You configure this by specifying the **Transparent** or **Reverse** setting within a custom monitor.

Transparent setting

Sometimes it is necessary to ping the aliased destination through a transparent pool, pool member, or node. When you create a custom monitor and set the **Transparent** setting to **Yes**, Local Traffic Manager™ forces the monitor to ping through the pool, pool member, or node with which it is associated (usually a firewall) to the pool, pool member, or node. (That is, if there are two firewalls in a load balancing pool, the destination pool, pool member, or node is always pinged through the pool, pool member, or node specified; not through the pool, pool member, or node selected by the load balancing method.) In this way, the transparent pool, pool member, or node is tested: if there is no response, the transparent pool, pool member, or node is marked as down.

Common examples are checking a router, or checking a mail or FTP server through a firewall. For example, you might want to check the router address `10.10.10.53:80` through a transparent firewall `10.10.10.101:80`. To do this, you create a monitor called `http_trans` in which you specify `10.10.10.53:80` as the monitor destination address, and set the **Transparent** setting to **Yes**. Then you associate the monitor `http_trans` with the transparent pool, pool member, or node.

This causes the monitor to check the address `10.10.10.53:80` through `10.10.10.101:80`. (In other words, the BIG-IP® system routes the check of `10.10.10.53:80` through `10.10.10.101:80`.) If the correct response is not received from `10.10.10.53:80`, then `10.10.10.101:80` is marked down.

Reverse setting

With the **Reverse** setting set to **Yes**, the monitor marks the pool, pool member, or node down when the test is successful. For example, if the content on your web site home page is dynamic and changes frequently, you may want to set up a reverse ECV service check that looks for the string `"Error"`. A match for this string means that the web server was down.

Monitors that contain the Transparent or Reverse settings

This table shows the monitors that contain either the Transparent setting or both the Reverse and Transparent settings.

Monitor Type	Settings
TCP	Transparent and Reverse
HTTP	Transparent and Reverse
HTTPS	Transparent and Reverse
TCP Echo	Transparent
TCP Half Open	Transparent
ICMP	Transparent

The Manual Resume feature

By default, when a monitor detects that a resource (that is, a node or a pool member) is unavailable, the BIG-IP® system marks the resource as down and routes traffic to the next appropriate resource as dictated by the active load balancing method. When the monitor next determines that the resource is available again, the BIG-IP system marks the resource as up and immediately considers the resource to be available for load balancing connection requests. While this process is appropriate for most resources, there are situations where you want to manually designate a resource as available, rather than allow the BIG-IP system to do that automatically. You can manually designate a resource as available by configuring the Manual Resume setting of the monitor.

For example, consider a monitor that you assigned to a resource to track the availability of an HTML file, `index.html`, for a web site. During the course of a business day, you decide that you need to restart the system that hosts the web site. The monitor detects the restart action and informs the BIG-IP system that the resource is now unavailable. When the system restarts, the monitor detects that the `index.html` file is available, and begins sending connection requests to the web site. However, the rest of the web site might not be ready to receive connection requests. Consequently, the BIG-IP system sends connection requests to the web site before the site can respond effectively.

To prevent this problem, you can configure the Manual Resume setting of the monitor. When you set the Manual Resume setting to Yes, you ensure that the BIG-IP system considers the resource to be unavailable until you manually enable that resource.

Resumption of connections

If you have a resource (such as a pool member or node) that a monitor marked as down, and the resource has subsequently become available again, you must manually re-enable that resource if the monitor's **Manual Resume** setting is set to Yes. Manually re-enabling the resource allows the BIG-IP® system to resume sending connections to that resource.

The procedure for manually re-enabling a resource varies depending on whether the resource is a pool, a pool member, or a node.

The Time Until Up feature

By default, the BIG-IP® system marks a pool member or node as up immediately upon receipt of the first correct response to a ping command.

The Time Until Up feature provides a way to adjust the default behavior. This feature allows the system to delay the marking of a pool member or node as up for some number of seconds after receipt of the first correct response. The purpose of this feature is to ensure that the monitor marks the pool member or node as up only after the pool member or node has consistently responded correctly to the BIG-IP system during the defined time period. With this feature, you ensure that a pool member or node that is available only momentarily, after sending one correct response, is not marked as up.

A Time Until Up value of 0 causes the default behavior. When the Time Until Up value is a non-0 value, the BIG-IP system marks a pool member or node as up only when all pool member or node responses during the Time Until Up period are correct.

Dynamic ratio load balancing

You can configure Dynamic Ratio load balancing for pools that consist of RealNetworks® RealServer™ servers, Microsoft® Windows® servers equipped with Windows Management Instrumentation (WMI), or any server equipped with an SNMP agent such as the UC Davis SNMP agent or Windows® 2000 Server SNMP agent.

To implement Dynamic Ratio load balancing for these types of servers, BIG-IP® Local Traffic Manager™ provides a special monitor plug-in file and a performance monitor for each type of server. The exception is a server equipped with an SNMP agent. In this case, Local Traffic Manager provides the monitor only; no special plug-in file is required for a server running an SNMP agent.

You must install the monitor plug-in on each server to be monitored, and you must create a performance monitor that resides on the BIG-IP system. Once you have created a monitor, the monitor communicates directly with the server plug-in.

Monitor plug-ins and corresponding monitor templates

For each server type, this table shows the required monitor plug-in and the corresponding performance monitor types.

Server Type	Monitor plug-in	Monitor Type
RealServer Windows server	F5RealMon.dll	Real Server
RealServer UNIX server	f5realmon.so	Real Server
Windows server with WMI	f5isapi.dll or F5Isapi64.dll or F5.IsHandler.dll	WMI
Windows 2000 Server server	SNMP agent	SNMP DCA and SNMP DCA Base
UNIX server	UC Davis SNMP agent	SNMP DCA and SNMP DCA Base

Monitor association with pools and nodes

You must associate a monitor with the server or servers to be monitored. The server or servers can be either a pool, a pool member, or a node, depending on the monitor type. You can associate a monitor with a server in any of these ways:

Monitor-to-pool association

This type of association associates a monitor with an entire load balancing pool. In this case, the monitor checks all members of the pool. For example, you can create an instance of the monitor `http` for every member of the pool `my_pool`, thus ensuring that all members of that pool are checked.

Monitor-to-pool member association

This type of association associates a monitor with an individual pool member, that is, an IP address and service. In this case, the monitor checks only that pool member and not any other members of the pool. For example, you can create an instance of the monitor `http` for pool member `10.10.10.10:80` of `my_pool`.

Monitor-to-node association

This type of association associates a monitor with a specific node. In this case, the monitor checks only the node itself, and not any services running on that node. For example, you can create an instance of the monitor `icmp` for node `10.10.10.10`. In this case, the monitor checks the specific node only, and not any services running on that node. You can designate a monitor as the default monitor that you want Local Traffic Manager to associate with one or more nodes. In this case, any node to which you have not specifically assigned a monitor inherits the default monitor.

Some monitor types are designed for association with nodes only, and not pools or pool members. Other monitor types are intended for association with pools and pool members only, and not nodes.

Node-only monitors specify a destination address in the format of an IP address with no service port (for example, `10.10.10.2`). Conversely, monitors that you can associate with nodes, pools, and pool members specify a destination address in the format of an IP address and service port (for example, `10.10.10.2:80`). Therefore, when you use the BIG-IP Configuration utility to associate a monitor with a pool, pool member, or node, the utility displays only those pre-configured monitors that are designed for association with that server.

For example, you cannot associate the monitor `icmp` with a pool or its members, since the `icmp` monitor is designed to check the status of a node itself and not any service running on that node.

Monitor instances

When you associate a monitor with a server, Local Traffic Manager™ automatically creates an *instance* of that monitor for that server. A monitor association thus creates an instance of a monitor for each server that you specify. This means that you can have multiple instances of the same monitor running on your servers.

Because instances of monitors are not partitioned objects, a user can enable or disable an instance of a monitor without having permission to manage the associated pool or pool member.

For example, a user with the Manager role, who can access partition `AppA` only, can enable or disable monitor instances for a pool that resides in partition `Common`. However, that user cannot perform operations on the pool or pool members that are associated with the monitor. Although this is correct functionality, the user might not expect this behavior. You can prevent this unexpected behavior by ensuring that all pools and pool members associated with monitor instances reside in the same partition.

NATs

Introduction to NATs

In some cases, you might want to allow a client on an external network to send a request directly to a specific internal node (thus bypassing the normal load balancing server selection). To send a request directly to an internal server, a client normally needs to know the internal node's IP address, which is typically a private class IP address. Because private class IP addresses are non-routable, you can instead create a network translation address (NAT). A *NAT* is a feature of BIG-IP® Local Traffic Manager™ that provides a routable IP address that an external node can use to send traffic to, or receive traffic from, an internal node.

More specifically, a NAT is an address translation object that instructs Local Traffic Manager to translate one IP address in a packet header to another IP address. A NAT consists of a one-to-one mapping of a public IP address to an internal private class IP address.

You can use a NAT in two different ways:

To translate a private class destination address to a public address

When an external node sends traffic to the public IP address defined in a NAT, Local Traffic Manager automatically translates that destination address to the associated private class IP address, which represents a specific node on the internal network. This translation is hidden from the external node that sent the traffic.

To translate a private class source address to a public address

You can also use a NAT to translate an internal node's private class source IP address to a public IP address. This translation is hidden from the external node that receives the traffic.

To summarize, a NAT provides a routable address for sending packets to or from a node that has a private class IP address.

When you create a NAT, you can map only one private class IP address to a specific public IP address. That is, a NAT always represents a one-to-one mapping between a private class IP address and a public IP address. If you want to map more than one private class IP address (that is, multiple internal nodes) to a single public IP address, you can create a SNAT instead.

Note: *NATs do not support port translation, and are not appropriate for protocols that embed IP addresses in the packet, such as FTP, NT Domain, or CORBA IIOP.*

Tip: *When you use a NAT to provide access to an internal node, all ports on that internal node are open. To mitigate this security risk, consider using a SNAT instead.*

Local Traffic Manager can apply a NAT to either an inbound or an outbound connection.

NATs for inbound connections

With respect to NATs, an *inbound* connection is a connection that is initiated by a node on an external network, and comes into the BIG-IP® system to a node on the internal network.

Without a NAT

Normally, traffic coming into the BIG-IP system is load balanced to a server in a pool, based on the load balancing method configured for that pool, in the following way:

- A client on an external network typically sends traffic to a virtual server on the BIG-IP system. The destination IP address in this case is the virtual server address.
- Upon receiving a packet, the virtual server typically translates that destination IP address to the IP address of a pool member, for the purpose of load balancing that packet.
- The pool member then sends its response back through the BIG-IP system, using a route specified in the server node's routing table (ideally, a floating IP address assigned to an internal VLAN). On receiving the response, Local Traffic Manager™ then performs the reverse translation; that is, the system translates the pool member's actual source address to the virtual server address. This results in the source address in the response to the client being the virtual server address, which is the source address that the client expects to see.

This typical load balancing scenario ensures that for load balanced traffic, the client system never sees the internal private class IP address of an internal node.

With a NAT

If the client system wants to bypass the load balancing mechanism to send packets directly to a specific node on the internal network, the client needs a routable IP address to use to send packets to that server node.

A NAT solves this problem by providing a routable address that a client can use to make a request to an internal server directly. In this way, a NAT performs the same type of address translation that a virtual server performs when load balancing connections to pool members. In the case of a NAT, however, no load balancing occurs, because the client is sending a request to a specific node. The NAT translates the public destination IP address in the request to the private class IP address of the internal node.

When the server node sends the response, Local Traffic Manager performs the reverse translation, in the same way that a virtual server behaves.

Note: Local Traffic Manager does not track NAT connections. Therefore, the public IP address that you define in a NAT cannot be the same address as a virtual address or SNAT address.

For example, suppose a node on the internal network (such as a load balancing server) has a private class IP address of 172.16.20.3. You can create a NAT designed to translate a public destination address of your choice (such as 207.10.1.103) to the private class address 172.16.20.3. Consequently, whenever a node on the external network initiates a connection to the address 207.10.1.103, Local Traffic Manager translates that public destination address to the private class address 172.16.20.3.

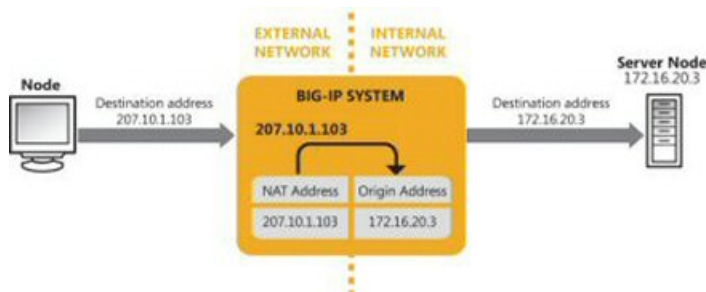


Figure 6: Sample NAT for an inbound connection

In this example, the NAT provides a routable address for an external node to initiate a connection to an internal node.

When you create a NAT, you must define two settings: NAT Address and Origin Address. In our example:

- The NAT address is 207.10.1.103, and the origin address is 172.16.20.3.
- The connection is an inbound connection, meaning that the connection is being initiated from the external network, through the BIG-IP system, to the internal network.
- Local Traffic Manager translates the NAT address to the origin address.
- The NAT address and the origin address are destination addresses.

NATs for outbound connections

The previous section summarized how a BIG-IP® system normally load balances incoming traffic, and translates the source IP address in a response back to the virtual address.

Sometimes, however, an internal node needs to initiate a connection, rather than simply respond to a request. When a node on an internal network initiates a connection, the connection is considered to be an *outbound* connection. In this case, because the outgoing packets do not represent a response to a load-balanced request, the packets do not pass through a virtual server, and therefore the system does not perform the usual source IP address translation.

Without a NAT, the source IP address is a non-routable address. With a NAT, however, Local Traffic Manager™ translates the internal node's private class IP address to a public IP address, to which the external node can then route its response.

For example, suppose an internal node (such as a mail server) has a private class IP address of 172.16.20.1. You can create a NAT designed to translate the private class address 172.16.20.1 to a public source address of your choice (such as 207.10.1.101). Consequently, whenever the internal node 172.16.20.1 initiates a connection destined for a node on the external network, the system translates that source address of 172.16.20.1 to its public address (207.10.1.101).

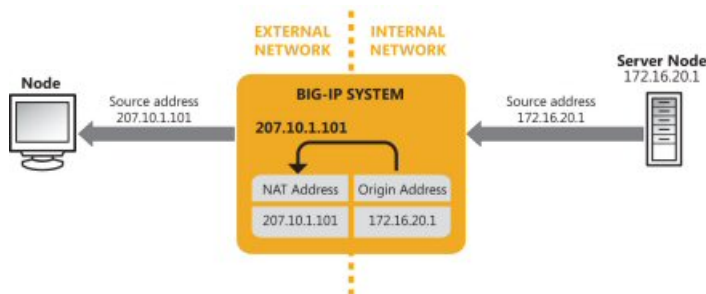


Figure 7: Sample NAT for an outbound connection

In this example, the NAT provides a way for an internal node to initiate a connection to a node on an external network, without showing a private class IP address as the source address.

A NAT has two settings; **NAT Address** and **Origin Address**. In this example:

- The NAT address is 207.10.1.101, and the origin address is 172.16.20.1.
- The connection is an outbound connection, meaning that the connection is being initiated from the internal network, through Local Traffic Manager, to the external network.
- Local Traffic Manager translates the origin address to the NAT address.
- The origin address and the NAT address are source addresses.

A NAT always represents a one-to-one mapping between a public address and a private class address. However, if you would like to map multiple internal nodes to a single public address, you can use a secure network translation address (SNAT) instead of a NAT. You can use SNATs for outbound connections only.

SNATs

About source address translation (SNATs)

When the default route on the servers does not route responses back through the BIG-IP system, you can create a secure network address translation (SNAT). A *secure network address translation (SNAT)* ensures that server responses always return through the BIG-IP® system. You can also use a SNAT to hide the source addresses of server-initiated requests from external devices.

For inbound connections from a client, a SNAT translates the source IP address within packets to a BIG-IP system IP address that you or the BIG-IP system defines. The destination node then uses that new source address as its destination address when responding to the request.

For outbound connections, SNATs ensure that the internal IP address of the server node remains hidden to an external host when the server initiates a connection to that host.

If you want the system to choose a SNAT translation address for you, you can select the Auto Map feature. If you prefer to define your own address, you can create a SNAT pool and assign it to the virtual server.

Important: *F5 recommends that before implementing a SNAT, you understand network address translation (NAT).*

Comparison of NATs and SNATs

A SNAT is similar to a NAT, except for some key differences listed in this table.

NATs	SNATs
You can map only one original address to a translation address.	You can map multiple original addresses to a single translation address. You can even map all node addresses on your network to a single public IP address, in a single SNAT object.
All ports on the internal node are open.	By default, SNATs support UDP and TCP only. This makes a SNAT more secure than a NAT.
Local Traffic Manager™ does not track NAT connections.	Local Traffic Manager tracks SNAT connections, which, in turn, allows SNATs and virtual servers to use the same public IP addresses.
You must explicitly enable a NAT on the internal VLAN where the internal node's traffic arrives on the BIG-IP® system.	By default, a SNAT that you create is enabled on all VLANs.

SNATs for client-initiated (inbound) connections

In the most common client-server network configuration, the Local Traffic Manager™ standard address translation mechanism ensures that server responses return to the client through the BIG-IP® system, thereby reversing the original destination IP address translation. This typical network configuration is as follows:

- The server nodes are on the same subnet as the BIG-IP system.
- The client nodes are on a different subnet from the server nodes.
- The BIG-IP system is the default gateway for the server subnet.

However, there are atypical network configurations in which the standard BIG-IP system address translation sequence by itself does not ensure that server responses use the required return path. Examples of these atypical configurations are:

When clients and servers are on the same network

If you want to load balance requests to server nodes that are on the same network as the client nodes, you can create a SNAT so that server responses are sent back through the virtual server, rather than directly from the server node to the client node. Otherwise, problems can occur such as the client rejecting the response because the source of the response does not match the destination of the request. Known as *virtual server bounceback*, this SNAT configuration causes the source of the response to match the destination of the request, thus ensuring that the client node accepts the response. You can use this kind of configuration when you want to load balance requests from web servers to application servers on the same network.

When the default gateway of the server node is not the BIG-IP system

For various reasons, the server node's default route cannot always be defined to be a route back through the BIG-IP system. Again, this can cause problems such as the client rejecting the response because the source of the response does not match the destination of the request. The solution is to create a SNAT. When Local Traffic Manager then translates the client node's source IP address in the request to the SNAT address, this causes the server node to use that SNAT address as its destination address when sending the response. This, in turn, forces the response to return to the client node through the BIG-IP system rather than through the server node's default gateway.

When using the OneConnect feature

Local Traffic Manager OneConnect™ feature allows client requests to re-use idle server-side connections. Without a SNAT, the source IP address in the server-side connection remains the address of the client node that initially established the connection, regardless of which other client nodes re-use the connection. Although this is not an issue for traffic routing, you might find it confusing when examining various types of system output. A SNAT solves this problem.

Note: Using a SNAT for inbound connections can impact the availability of ephemeral ports. This can lead to the SNAT being unable to process additional connections until some source ports become available.

This image shows a typical problem for client-initiated connections when Local Traffic Manager is not defined as the server's default gateway, and you have not configured a SNAT for inbound traffic.

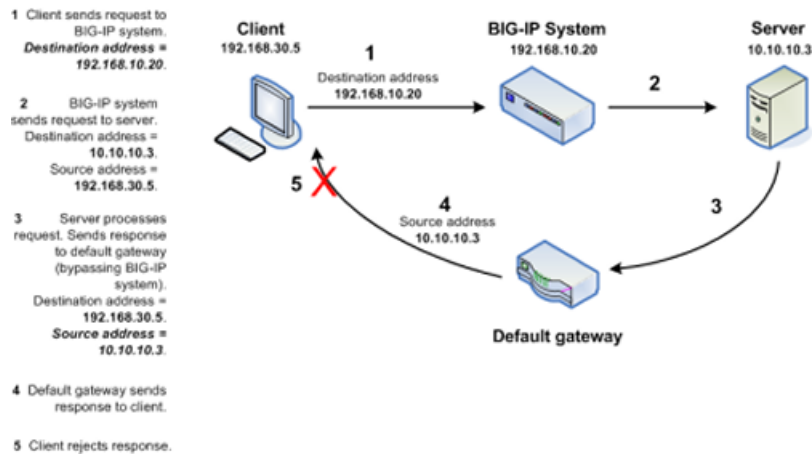


Figure 8: Client rejects response due to non-matching destination and source IP addresses

To prevent these problems, you can configure an inbound SNAT. An *inbound SNAT* translates the original client source IP address in a request to a BIG-IP system virtual server or BIG-IP system self IP address, forcing subsequent server response to return directly to Local Traffic Manager. When an inbound SNAT is configured on the system, Local Traffic Manager translates not only the destination IP address in the request (using the standard address translation mechanism), but also the source IP address in the request (using a SNAT).

The figure below shows that by configuring a SNAT, you ensure that the response returns through the BIG-IP system instead of through the default gateway, thus ensuring that the client can accept the server response.

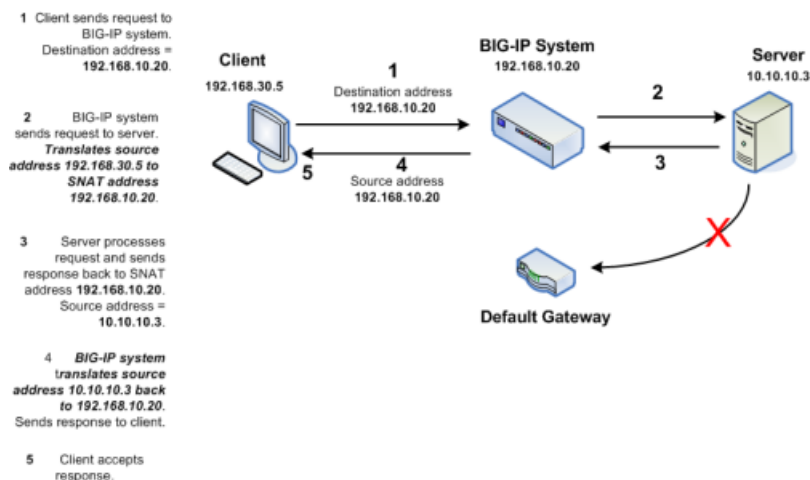


Figure 9: Client accepts response due to matching destination and source IP addresses

SNATs for server-initiated (outbound) connections

When an internal server initiates a connection to an external host, a SNAT can translate the private, source IP addresses of one or more servers within the outgoing connection to a single, publicly-routable address. The external destination host can then use this public address as a destination address when sending the response. In this way, the private class IP addresses of the internal nodes remain hidden from the external host.

More specifically, a SNAT for an outgoing connection works in the following way:

1. Local Traffic Manager™ receives a packet from an original IP address (that is, an internal server with a private IP address) and checks to see if that source address is defined in a SNAT.
2. If the original IP address is defined in a SNAT, Local Traffic Manager changes that source IP address to the translation address defined in the SNAT.
3. Local Traffic Manager then sends the packet, with the SNAT translation address as the source address, to the destination host.

In this example of an outgoing SNAT, Local Traffic Manager causes three internal nodes, with the IP addresses 172.16.20.4, 172.16.20.5, and 172.16.20.6, to advertise the public IP address 207.10.1.102 as the source IP address in the three outgoing connections.

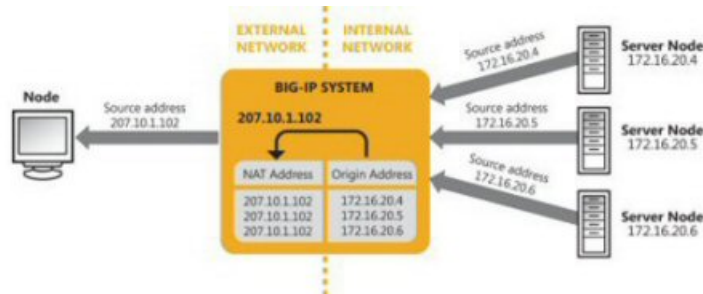


Figure 10: Sample SNAT for multiple outgoing connections

SNAT implementation

When you create a SNAT, you map an original IP address to a translation address in one of several ways, depending on your needs:

You can explicitly map one or more original IP addresses to a single translation address.

See figure

You can use the SNAT automap feature.

The SNAT automap feature automatically selects one of the system's self IP addresses (typically a floating self IP address of the egress VLAN), and maps it to the original IP address or addresses that you specify during SNAT creation. Note that if no floating self IP address is currently assigned to the egress VLAN, the system uses the floating IP address of a non-egress VLAN.

You can create a pool of translation addresses and map one or more original IP addresses to that SNAT pool.

This pool of addresses is known as a SNAT pool. You can map an original IP address to the SNAT pool by either creating a SNAT object or writing an iRule

You can create a SNAT pool and map all original IP addresses to that SNAT pool.

Yet another way to create a SNAT is to create a SNAT pool (using the New SNAT Pool screen of the BIG-IP Configuration utility) and directly assign it to a virtual server as a resource of that virtual server. Once you have assigned a SNAT pool to a virtual server, Local Traffic Manager™ automatically maps all original IP addresses coming through the virtual server to that SNAT pool.

SNAT types

The types of SNATs you can create are:

Standard SNAT

A standard SNAT is an object you create, using the BIG-IP Configuration utility, that specifies the mapping of one or more original IP addresses to a translation address. For this type of SNAT, the criteria that Local Traffic Manager™ uses to decide when to apply the translation address is based strictly on the original IP address. That is, if a packet arrives from the original IP address that you specified in the SNAT, then Local Traffic Manager translates that address to the specified translation address. There are three types of standard SNATs that you can create:

- A SNAT in which you specify a specific translation address
- A SNAT that uses the automap feature
- A SNAT in which you specify a SNAT pool as your translation address

Intelligent SNAT

Like a standard SNAT, an intelligent SNAT is the mapping of one or more original IP addresses to a translation address. However, you implement this type of SNAT mapping within an iRule instead of by creating a SNAT object. For this type of SNAT, the criteria that Local Traffic Manager uses to decide when to apply a translation address is based on any piece of data you specify within the iRule, such as an HTTP cookie or a server port.

SNAT pool assigned as a virtual server resource

This type of SNAT consists of just a SNAT pool that you directly assign as a resource to a virtual server. When you implement this type of SNAT, you create a SNAT pool only; you do not need to create a SNAT object or an iRule.

About translation addresses

You can specify the translation addresses that you want to map to your original IP addresses. A translation address can be in these three forms:

An IP Address

When creating a SNAT, you can specify a particular IP address that you want the SNAT to use as a translation address.

A SNAT pool

Specifying this value allows you to specify an existing SNAT pool to which you want to map your original IP address.

SNAT automap

Similar to a SNAT pool, the SNAT automap feature allows you to map one or more original IP addresses to a pool of translation addresses. With the SNAT automap feature, you do not need to create the pool. Instead, Local Traffic Manager™ effectively creates a pool for you, using self IP addresses as the translation addresses for the pool.

Original IP addresses

You can specify the original IP addresses that you want to map to translation addresses. You can specify one IP address or multiple IP addresses.

VLAN traffic

You can specify one or more VLANs to which you want the SNAT to apply.

Traffic Classes

About traffic classes

BIG-IP® Local Traffic Manager™ includes a feature known as traffic classes. A traffic class is a feature that you can use when implementing optimization profiles for modules such as the Application Acceleration Manager™.

A *traffic class* allows you to classify traffic according to a set of criteria that you define, such as source and destination IP addresses. In creating the traffic class, you define not only classification criteria, but also a classification ID. Once you have defined the traffic class and assigned the class to a virtual server, Local Traffic Manager associates the *classification ID* to each traffic flow. In this way, Local Traffic Manager can regulate the flow of traffic based on that classification.

When attempting to match traffic flows to a traffic class, Local Traffic Manager uses the most specific match possible.

To configure and manage traffic classes, log in to the BIG-IP Configuration utility, and on the Main tab, expand **Local Traffic**, and click **Traffic Classes**.

iRules

Introduction to iRules

An *iRule* is a powerful and flexible feature within BIG-IP® Local Traffic Manager™ that you can use to manage your network traffic. Using syntax based on the industry-standard Tools Command Language (Tcl), the iRules® feature not only allows you to select pools based on header data, but also allows you to direct traffic by searching on any type of content data that you define. Thus, the iRules feature significantly enhances your ability to customize your content switching to suit your exact needs.

Important: For complete and detailed information on iRules syntax, see the F5 Networks DevCentral web site, <http://devcentral.f5.com>. Note that iRules must conform to standard Tcl grammar rules; therefore, for more information on Tcl syntax, see <http://tmml.sourceforge.net/doc/tcl/index.html>.

An iRule is a script that you write if you want individual connections to target a pool other than the default pool defined for a virtual server. iRules allow you to more directly specify the destinations to which you want traffic to be directed. Using iRules, you can send traffic not only to pools, but also to individual pool members, ports, or URIs. The iRules you create can be simple or sophisticated, depending on your content-switching needs.

```
when CLIENT_ACCEPTED {  
  if { [IP::addr [IP::client_addr] equals 10.10.10.10] } {  
    pool my_pool  
  }  
}
```

This iRule is triggered when a client-side connection has been accepted, causing Local Traffic Manager to send the packet to the pool `my_pool`, if the client's address matches `10.10.10.10`.

Using a feature called the *Universal Inspection Engine*, you can write an iRule that searches either a header of a packet, or actual packet content, and then directs the packet based on the result of that search. iRules can also direct packets based on the result of a client authentication attempt.

iRules can direct traffic not only to specific pools, but also to individual pool members, including port numbers and URI paths, either to implement persistence or to meet specific load balancing requirements.

The syntax that you use to write iRules is based on the Tool Command Language (Tcl) programming standard. Thus, you can use many of the standard Tcl commands, plus a robust set of extensions that Local Traffic Manager provides to help you further increase load balancing efficiency.

Important: When referencing an object within an iRule, you must include the full path name of the object.

Basic iRule elements

iRules® are made up of these basic elements:

- Event declarations

- Operators
- iRule commands

Event declarations

iRules[®] are event-driven, which means that Local Traffic Manager[™] triggers an iRule based on an event that you specify in the iRule. An *event declaration* is the specification of an event within an iRule that causes Local Traffic Manager to trigger that iRule whenever that event occurs. Examples of event declarations that can trigger an iRule are `HTTP_REQUEST`, which triggers an iRule whenever the system receives an HTTP request, and `CLIENT_ACCEPTED`, which triggers an iRule when a client has established a connection.

```
when HTTP_REQUEST {
  if { [HTTP::uri] contains "aol" } {
    pool aol_pool
  } else {
    pool all_pool
  }
}
```

Operators

An iRule operator compares two operands in an expression.

For example, you can use the `contains` operator to compare a variable operand to a constant. You do this by creating an if statement that represents the following: "If the HTTP URI contains aol, send to pool aol_pool."

iRule commands

An *iRule command* within an iRule causes Local Traffic Manager[™] to take some action, such as querying for data, manipulating data, or specifying a traffic destination. The types of commands that you can include within iRules[®] are:

Statement commands

These commands cause actions such as selecting a traffic destination or assigning a SNAT translation address. An example of a statement command is `pool <name>`, which directs traffic to the named load balancing pool.

Commands that query or manipulate data

Some commands search for header and content data, while others perform data manipulation such as inserting headers into HTTP requests. An example of a query command is `IP::remote_addr`, which searches for and returns the remote IP address of a connection. An example of a data manipulation command is `HTTP::header remove <name>`, which removes the last occurrence of the named header from a request or response.

Utility commands

These commands are functions that are useful for parsing and manipulating content. An example of a utility command is `decode_uri <string>`, which decodes the named string using HTTP URI encoding and returns the result.

The pool command

Once you have specified a query within your iRule, you can use the `pool` command to select a load balancing pool to which you want Local Traffic Manager™ to send a request. Here is an example of this command.

```
when HTTP_REQUEST {
  set uri [HTTP::uri]
  if { $uri ends_with ".gif" } {
    pool my_pool
  } elseif { $uri ends_with ".jpg" } {
    pool your_pool
  }
}
```

The node command

As an alternative to the `pool` command, you can also write an iRule that directs traffic to a specific server. To do this, you use the `node` command.

```
when HTTP_REQUEST {
  if { [HTTP::uri] ends_with ".gif" } {
    node 10.1.2.200 80
  }
}
```

Commands that select a pool of cache servers

You can create an iRule that selects a server from a pool of cache servers. Here is an example of an iRule that selects a server from a pool of cache servers.

```
when HTTP_REQUEST
  # This line specifies the expressions that determine whether the BIG-IP
  # system sends requests to the cache pool:
  if { [HTTP::uri] ends_with "html" or [HTTP::uri] ends_with "gif" } {
    pool cache_pool
    set key [crc32 [concat [domain [HTTP::host] 2] [HTTP::uri]]]
    set cache_mbr [persist lookup hash $key node]
    if { $cache_mbr ne "" } {
      # This line verifies that the request is not coming from the cache:
      if { [IP::addr [IP::remote_addr] equals $cache_mbr] } {
        # This line sends the request from the cache to the origin pool:
        pool origin_pool
        return
      }
    }
  }
}
```

```

        # These lines ensure that the persistence record is added for this
host/URI:
    persist hash $key
    } else {
        pool origin_pool
    }
}

```

Note: Local Traffic Manager™ redirects URIs to a new cache member at the time that the BIG-IP® system receives a request for the URI, rather than when the pool member becomes unavailable.

The HTTP::redirect command

In addition to configuring an iRule to select a specific pool, you can also configure an iRule to redirect an HTTP request to a specific location, using the `HTTP::redirect` iRule command. The location can be either a host name or a URI.

This is an iRule that is configured to redirect an HTTP response.

```

when HTTP_RESPONSE {
  if { [HTTP::status] contains "404" } {
    HTTP::redirect "http://www.siterequest.com/"
  }
}

```

Here is an example of an iRule that redirects an HTTP request.

```

when HTTP_REQUEST {
  if { [HTTP::uri] contains "secure" } {
    HTTP::redirect "https://[HTTP::host][HTTP::uri]"
  }
}

```

The snat and snatpool commands

The iRules® feature includes the two statement commands `snat` and `snatpool`. Using the `snat` command, you can assign a specified translation address to an original IP address from within the iRule, instead of using the SNAT screens within the BIG-IP Configuration utility.

Using the `snatpool` command also assigns a translation address to an original IP address, although unlike the `snat` command, the `snatpool` command causes Local Traffic Manager™ to select the translation address from a specified SNAT pool that you previously created.

iRules and administrative partitions

You should be aware of certain iRule configuration concepts as they relate to administrative partitions:

- An iRule can reference any object, regardless of the partition in which the referenced object resides. For example, an iRule that resides in `partition_a` can contain a pool statement that specifies a pool residing in `partition_b`.
- You can remove iRule assignments only from virtual servers that reside in the current `Write` partition or in partition `Common`.
- Note that you can associate an iRule only with virtual servers that reside in the current `Write` partition or in partition `Common`.
- You can associate an existing iRule with multiple virtual servers. In this case, the iRule becomes the only iRule that is associated with each virtual server in the current `Write` partition. Because this command overwrites all previous iRule assignments, F5 does not recommend use of this command.

iRule evaluation

In a basic system configuration where no iRule exists, Local Traffic Manager™ directs incoming traffic to the default pool assigned to the virtual server that receives that traffic. However, you might want Local Traffic Manager to direct certain kinds of connections to other destinations. The way to do this is to write an iRule that directs traffic to that other destination, contingent on a certain type of event occurring. Otherwise, traffic continues to go to the default pool assigned to the virtual server.

iRules® are therefore evaluated whenever an event occurs that you have specified in the iRule. For example, if an iRule includes the event declaration `CLIENT_ACCEPTED`, then the iRule is triggered whenever Local Traffic Manager accepts a client connection. Local Traffic Manager then follows the directions in the remainder of the iRule to determine the destination of the packet.

Event types

The iRule command syntax includes several types of event declarations that you can specify within an iRule. For example:

- Global events, such as `CLIENT_ACCEPTED`
- HTTP events, such as `HTTP_REQUEST`
- SSL events, such as `CLIENTSSL_HANDSHAKE`
- Authentication events, such as `AUTH_SUCCESS`

For a complete list of iRule events and their descriptions, see the F5 Networks DevCentral web site, <http://devcentral.f5.com>.

iRule context

For every event that you specify within an iRule, you can also specify a context, denoted by the keywords `clientside` or `serverside`. Because each event has a default context associated with it, you need only declare a context if you want to change the context from the default.

The example shows `my_iRule1`, which includes the event declaration `CLIENT_ACCEPTED`, as well as the iRule command `IP::remote_addr`. In this case, the IP address that the iRule command returns is that of the client, because the default context of the event declaration `CLIENT_ACCEPTED` is `clientside`.

```
when CLIENT_ACCEPTED {
  if { [IP::addr [IP::remote_addr] equals 10.1.1.80] } {
    pool my_pool1
  }
}
```

Similarly, if you include the event declaration `SERVER_CONNECTED` in an iRule as well as the iRule command `IP::remote_addr`, the IP address that the iRule command returns is that of the server, because the default context of the event declaration `SERVER_CONNECTED` is `serverside`.

The preceding example shows what happens when you write an iRule that uses the default context when processing iRule commands. You can, however, explicitly specify the `clientside` and `serverside` keywords to alter the behavior of iRule commands.

Continuing with the previous example, the following example shows the event declaration `SERVER_CONNECTED` and explicitly specifies the `clientside` keyword for the iRule command `IP::remote_addr`. In this case, the IP address that the iRule command returns is that of the client, despite the server-side default context of the event declaration.

```
when SERVER_CONNECTED {
  if { [IP::addr [IP::addr [clientside {IP::remote_addr}] equals 10.1.1.80]
} {
  discard
}
}
```

Note: You make an event declaration in an iRule by using the *when* keyword, followed by the event name. The figure shows an example of an event declaration in an iRule.

iRules assignment to a virtual server

When you assign multiple iRules® as resources for a virtual server, it is important to consider the order in which you list them on the virtual server. This is because Local Traffic Manager™ processes duplicate iRule events in the order that the applicable iRules are listed. An iRule event can therefore terminate the triggering of events, thus preventing Local Traffic Manager from triggering subsequent events.

Note: If an iRule references a profile, Local Traffic Manager processes this type of iRule last, regardless of its order in the list of iRules assigned to a virtual server.

iRule command types

There are three types of iRule commands:

- Statement commands
- Query and manipulation commands
- Utility commands (also known as functions)

Statement commands

Some of the commands available for use within iRules are known as statement commands. *Statement commands* enable Local Traffic Manager™ to perform a variety of different actions. For example, some of these commands specify the pools or servers to which you want Local Traffic Manager to direct traffic. Other commands specify translation addresses for implementing SNAT connections. Still others specify objects such as data groups or a persistence profiles.

For a complete list of statement commands, see the F5 Networks DevCentral web site, <http://devcentral.f5.com>.

Query and manipulation commands

Using iRules® commands, you can query for specific data contained in the header or content of a request or response, or you can manipulate that data. Data manipulation refers to inserting, replacing, and removing data, as well as setting certain values found in headers and cookies.

For example, using the `IP::idle_timeout` command within an iRule, you can query for the current idle timeout value that is set in a packet header and then load balance the packet accordingly. You can also use the `IP::idle_timeout` command to set the idle timeout to a specific value of your choice.

iRule query and manipulation commands are grouped into categories called *namespaces*. Except for commands in the global namespace, each iRule query or manipulation command includes the namespace in its command name. For example, one of the commands in the IP namespace is `IP::idle_timeout`. One of the commands in the HTTP namespace is `HTTP::header`.

For a complete list of namespaces for iRules commands, see the F5 Networks DevCentral web site, <http://devcentral.f5.com>.

Utility commands

Local Traffic Manager includes a number of utility commands that you can use within iRules. You can use these commands to parse and retrieve content, encode data into ASCII format, verify data integrity, and retrieve information about active pools and pool members.

iRules and profiles

When you are writing an iRule, you might want that iRule to recognize the value of a particular profile setting so that it can make a more-informed traffic management decision. Fortunately, the iRules® feature includes a command that is specifically designed to read the value of profile settings that you specify within the iRule.

Not only can iRules read the values of profile settings, but they can also override values for certain settings. This means that you can apply configuration values to individual connections that differ from the values Local Traffic Manager™ applies to most connections passing through a virtual server.

The profile command

The iRules® feature includes a command called `PROFILE`. When you specify the `PROFILE` command in an iRule and name a profile type and setting, the iRule reads the value of that particular profile setting. To do this, the iRule finds the named profile type that is assigned to the virtual server and reads the value of the setting that you specified in the `PROFILE` command sequence. The iRule can then use this information to manage traffic.

For example, you can specify the command `PROFILE::tcp idle_timeout` within your iRule. Local Traffic Manager™ then finds the TCP profile that is assigned to the virtual server (for example, `my_tcp`) and queries for the value that you assigned to the Idle Timeout setting.

Commands that override profile settings

Some of the iRule commands for querying and manipulating header and content data have equivalent settings within various profiles. When you use those commands in an iRule, and an event triggers that iRule, Local Traffic Manager™ overrides the values of those profile settings, using the value specified within the iRule instead.

For example, an HTTP profile might specify a certain buffer size to use for compressing HTTP data, but you might want to specify a different buffer size for a particular type of HTTP connection. In this case, you can include the command `HTTP::compress_buffer_size` in your iRule, specifying a different value than the value in the profile.

Data groups

Data groups are useful when writing iRules®. A *data group* is simply a group of related elements, such as a set of IP addresses for AOL clients. When you specify a data group along with the `class match` command or the `contains` operator, you eliminate the need to list multiple values as arguments in an iRule expression.

You can define three types of data groups: address, integer, and string.

The BIG-IP® system includes three pre-configured data groups: `private_net`, `images`, and `aol`.

To understand the usefulness of data groups, it is helpful to first understand the `class match` command and the `contains` operator.

Note: *You can manage only those data groups that you have permission to manage, based on your user role and partition access assignment.*

Warning: *Do not attempt to modify or delete any of the three pre-configured data groups (`private_net`, `images`, and `aol`). Doing so can produce adverse results.*

About the class match command

The BIG-IP® system includes an iRule command called `class`, with a `match` option, which you can use to select a pool based on whether the command being used in the iRule represents a member of a specific data group. When you use the `class` command, the BIG-IP system knows that the string following the identifier is the name of a data group.

For example, using the `class` command, you can cause the BIG-IP system to load balance all incoming AOL connections to the pool `aol_pool`, if the value of the `IP::remote_addr` command is a member of the data group `AOL`. In this case, the `class match` command simply indicates that the object named `aol` is a collection of values (that is, a data group).

```
when CLIENT_ACCEPTED {
  if { [class match [IP::remote_addr] equals aol] } {
    pool aol_pool
  }
}
```

```

    } else {
        pool all_pool
    }
}

```

Storage options

With Local Traffic Manager™, you can store data groups in two ways, either in-line or externally.

In-line storage

When you create data groups, Local Traffic Manager automatically saves them in their entirety in the `bigip.conf` file. This type of storage is known as *in-line storage*.

In general, in-line storage uses additional system resources due to extensive searching requirements on large data groups. For this reason, Local Traffic Manager offers you the ability to store your data groups externally, that is, outside of the `bigip.conf` file.

External storage

You have the option to store data groups in another location on the BIG-IP® system, that is, outside of the `bigip.conf` file. Such data groups are called *external data groups*. Because the data group is stored externally in another location, the `bigip.conf` file itself contains only the filename and meta-data for the data group. The data in an externally-stored data group file is stored as a comma-separated list of values (CSV format).

Important: *If you attempt to load a `bigip.conf` file that contains external data group meta-data, and the file was created prior to BIG-IP system version 9.4, the system generates an error. The meta-data for the external data group contains the keyword `extern`, which generates an error during the load process. On BIG-IP systems running version 9.4 or later, the `extern` keyword is no longer needed in the `bigip.conf` file.*

To create an external data group, you first import a file from another location, using the **System** options of the BIG-IP Configuration utility. You then use the Local Traffic iRules® screens to create an external data group that is based on the imported file.

External data groups can scale to greater than 10,000,000 entries, depending on platform hardware and available memory (8 GB, or more, memory is recommended). Data groups with larger data items can be supported with fewer entries. Additionally, updates to external data groups are completely atomic: for example, the system updates a data group only after the new data successfully completes loading. You can use the command `[class exists xyz]` to check whether a data group has finished loading.

iFiles

Using the BIG-IP Configuration utility, you can create a special file called an iFile. An *iFile* is a file that is based on an external file that you previously imported to the BIG-IP® system from another system. You can reference an iFile from within an iRule, based on a specific iRule event.

To create an iFile and use it within an iRule, you start from the **Local Traffic** option on the Main tab.

Important: *Prior to creating an iFile, you must import a file to the BIG-IP system from another system.*

iFile commands

With these iRule commands, you can reference the new iFile from within an iRule:

- [ifile get IFILENAME]
- [ifile listall]
- [ifile attributes IFILENAME]
- [ifile size IFILENAME]
- [ifile last_updated_by IFILENAME]
- [ifile last_update_time IFILENAME]
- [ifile revision IFILENAME]
- [ifile checksum IFILENAME]
- array set [file attributes IFILENAME]

```
ltm rule ifile_rule {
    when HTTP_RESPONSE {
        # return a list of iFiles in all partitions
        set listifiles [ifile listall]
        log local0. "list of ifiles: $listifiles"

        # return the attributes of an iFile specified
        array set array_attributes [ifile attributes "/Common/iframeURL"]

        foreach {array attr} [array get array_attributes ] {
            log local0. "$array : $attr"
        }

        # serve an iFile when http status is 404.
        set file [ifile get "Common/iframeURL"]
        log local0. "file: $file"
        if { [HTTP::status] equals "404" } {
            HTTP::Respond 200 ifile "/Common/iframeURL"
        }
    }
}
```

Dynamic Ratio Load Balancing

Introduction to dynamic ratio load balancing

You can configure Dynamic Ratio load balancing for pools that consist of RealNetworks[®] RealServer[™] servers, Microsoft[®] Windows[®] servers equipped with Windows Management Instrumentation (WMI), or any server equipped with an SNMP agent such as the UC Davis SNMP agent or Windows[®] 2000 Server SNMP agent.

To implement Dynamic Ratio load balancing for these types of servers, BIG-IP[®] Local Traffic Manager[™] provides a special monitor plug-in file and a performance monitor for each type of server. The exception is a server equipped with an SNMP agent. In this case, the BIG-IP system provides the monitor only; no special plug-in file is required for a server running an SNMP agent.

Monitor plug-ins and corresponding monitor templates

Shows the required monitor plug-in and the corresponding performance monitor types.

Server Type	Monitor plug-in	Monitor Type
RealServer [™] Windows [®] server	F5RealMon.dll	Real Server
RealServer UNIX server	f5realmon.so	Real Server
Windows server with WMI	f5isapi.dll or F5Isapi64.dll or F5.IsHandler.dll	WMI
Windows 2000 Server server	SNMP agent	SNMP DCA and SNMP DCA Base
UNIX server	UC Davis SNMP agent	SNMP DCA and SNMP DCA Base

Overview of implementing a RealServer monitor

For RealSystem[®] Server systems, the BIG-IP[®] system provides a monitor plug-in that gathers the necessary metrics when you have installed the plug-in on the RealSystem Server system. Configuring a RealSystem Server for Dynamic Ratio load balancing consists of four tasks:

- Installing the monitor plug-in on the RealSystem Server system
- Configuring a Real Server monitor on the BIG-IP system
- Associating the monitor with the server to gather the metrics
- Creating or modifying the server pool to use Dynamic Ratio load balancing

Installing the monitor plug-in on a RealSystem server system (Windows version)

Perform this task to install the monitor plug-in on a RealSystem Server system (Windows version).

1. Download the monitor plug-in `F5RealServerPlugin.dll` from the BIG-IP® system. The plug-in is located in the folder `/usr/local/www/docs/agents`.
2. Copy `F5RealServerPlugin.dll` to the RealServer plug-ins directory. (For example, `C:\Program Files\RealServer\plug-ins`.)
3. If the RealSystem Server process is running, restart it.

Once the plug-in is installed and compiled, you must configure a Real Server monitor, associate the configured monitor with the node (a RealSystem Server server), and set the load balancing method to Dynamic Ratio.

Installing and compiling a Linux or UNIX RealSystem server monitor plug-in

Perform this task to install and compile a Linux or UNIX RealSystem Server monitor plug-in.

1. Using the **.iso** image, burn a CD-ROM of the BIG-IP® system software.
2. On the CD, navigate to the directory `/downloads/rsplug-ins`.
3. Copy the file `F5RealMon.src.tar.gz` to the directory `/var/tmp` on the BIG-IP system.
4. On the BIG-IP system, change to the directory `/var/tmp`:
`cd /var/tmp`
5. Use the UNIX **tar** command to uncompress the file `F5RealMon.src.tar.gz`:
`tar -xvzf F5RealMon.src.tar`
6. Change to the **F5RealMon.src** directory:
`cd F5RealMon.src`
7. Type the `ls` command to view the directory contents.
8. To compile the source, use the instructions in the file `build_unix_note`.
9. Start RealSystem Server.

Once the plug-in is installed and compiled, you must configure a Real Server monitor, associate the configured monitor with the node (a RealSystem Server server), and set the load balancing method to Dynamic Ratio.

Overview of implementing a WMI monitor

For Windows running Windows Management Instrumentation (WMI), the BIG-IP® system provides a Data Gathering Agent for the IIS server. Configuring a Windows platform for Dynamic Ratio load balancing consists of these tasks:

- Installing the Data Gathering Agent on the IIS server.
- Configuring a WMI monitor on the BIG-IP system.
- Associating the monitor with the server to gather the metrics.
- Creating or modifying the server pool to use the Dynamic Ratio load balancing method.

Important: To enable a user to access WMI metrics on a Windows server, you must configure the WMI monitor on the BIG-IP system correctly.

The procedure for installing the Data Gathering Agent on an IIS server differs depending on whether the server is running IIS version 5.0, 6.0, or 7.0, and whether the Data Gathering Agent is the file `f5isapi.dll` (or `f5isapi64.dll`) or the file `F5.IsHandler.dll`.

Tip: F5 Networks® recommends that you install only the Data Gathering Agent file that pertains to your specific configuration. Installing multiple Data Gathering Agent files could result in unwanted behavior.

IIS version support for the data gathering agent files

The procedure for installing the Data Gathering Agent on an IIS server differs depending on whether the server is running IIS version 5.0, 6.0, or 7.0, and whether the Data Gathering Agent is the file `f5isapi.dll` (or `f5isapi64.dll`) or the file `F5.IsHandler.dll`. This table shows each of the Data Gathering Agent files and the IIS versions that support each file.

Data Gathering Agent	IIS version 5.0	IIS version 6.0	IIS version 7.0
<code>f5isapi.dll</code> (32-bit) <code>f5isapi64.dll</code> (64-bit)	X	X	N/A
<code>F5.IsHandler.dll</code> (32-bit, 64-bit, and .NET)	N/A	X	X

Installing the Data Gathering Agent `f5isapi.dll` or `f5isapi64.dll` on an IIS 5.0 server

You can install the file `f5isapi.dll` or `f5isapi64.dll` on IIS versions 5.0 or 6.0.

Important: Do not install either of these files on IIS version 7.0 or 7.5. For IIS servers running version 7.0 or 7.5, install the file `F5.IsHandler.dll` instead.

Perform this task to install the Data Gathering Agent `f5isapi.dll` or `f5isapi64.dll` on an IIS 5.0 server.

1. Download the **Data Gathering Agent** (`f5isapi.dll` or `f5isapi64.dll`) from the BIG-IP® system to the Windows platform. You can find this plug-in in either the `/var/windlls` or the `/usr/local/www/docs/agents` directory on the BIG-IP system.
2. Copy `f5isapi.dll` or `f5isapi64.dll` to the directory `C:\Inetpub\scripts`.
3. Open the Internet Services Manager.
4. In the left pane of the Internet Services Manager, open the folder `machine_name\Default Web Site\Script`, where `machine_name` is the name of the server you are configuring. The contents of Scripts folder opens in the right pane.
5. In the right pane, right-click **`f5isapi.dll`** or **`f5isapi64.dll`**, and select **Properties**. The Properties dialog box for `f5isapi.dll` or `f5isapi64.dll` opens.
6. Clear **Logvisits**. (Logging of each visit to the agent quickly fills up the log files.)
7. Click the File Security tab.
The File Security options appears.
8. In the **Anonymous access and authentication control group** box, click **Edit**.
The Authentication Methods dialog box opens.
9. In the dialog box, clear all check boxes, then select **Basic Authentication**.
10. In the Authentication Methods dialog box, click **OK** to accept the changes.
11. In the Properties dialog box, click **Apply**.
The WMI Data Gathering Agent is now ready to be used.

Once you have installed the plug-in, you must configure a WMI monitor, associate the configured monitor with the pool member, and set the load balancing method to Dynamic Ratio.

Installing the Data Gathering Agent f5isapi.dll or f5isapi64.dll on an IIS 6.0 server

You can install the file f5isapi.dll or f5isapi64.dll on IIS versions 5.0 or 6.0.

Important: Do not install either of these files on IIS version 7.0 or 7.5. For IIS servers running version 7.0 or 7.5, install the file F5.IsHandler.dll instead.

Perform this task to install the Data Gathering Agent f5isapi.dll or f5isapi64.dll on an IIS 6.0 server.

1. Create a **scripts** directory under the web site document root (C:\InetPub\wwwroot for *Default Website*).
2. Set the properties of the **scripts** directory to **scripts and executables**.
3. Copy the file f5isapi.dll or f5isapi64.dll to the created scripts directory.
4. Start IIS manager (**inetmgr**) and navigate to the **scripts** directory.
5. On the right pane, select the file name f5isapi.dll or f5isapi64.dll.
6. Select **Properties->File Security->Authentication and Access Control** and ensure that the settings **anonymous user** and **Basic Authentication** are checked.
7. If you want to allow all unknown extensions, then in IIS Manager, navigate to **Web Server Extensions -> All Unknown ISAPI extensions** and allow all unknown extensions. Otherwise, proceed to step 8.
8. If you want to allow the file f5isapi.dll or f5isapi64.dll only, navigate to **Web Server Extensions -> Tasks: Add a New Webserver Extension**. Then:
 - a) In the Name field, select **F5 ISAPI** and click **Add** for the required files. This requests a path to the file.
 - b) Browse to the file f5isapi.dll or f5isapi64.dll, using the path C:\InetPub\wwwroot\scripts\f5isapi.dll for Default Website, and click **OK**.
 - c) Check the **Set Extension Status to Allowed** box, and click **OK**. The value **F5 ISAPI** should now appear in the extensions list as **Allowed**.

Once you have installed the plug-in, you must configure a WMI monitor, associate the configured monitor with the pool member, and set the load balancing method to Dynamic Ratio.

Installing the Data Gathering Agent F5.IsHandler.dll on an IIS 6.0 server

You can install the file F5.IsHandler.dll on IIS versions 6.0 and 7.0.

Perform this task to install the Data Gathering Agent F5.IsHandler.dll on an IIS 6.0 server.

1. Create a **scripts** directory under the directory C:\Inetpub. (C:\Inetpub\scripts).
2. Create a \bin directory under the **scripts** directory (C:\Inetpub\scripts\bin).
3. Set the properties of the **scripts** directory to **scripts and executables**.
4. Copy the file F5.IsHandler.dll to the directory C:\Inetpub\scripts\bin.

In the C:\Inetpub\scripts directory, create the file **web.config**. This example shows a web.config file on an IIS server running version 6.0. In the example, the path value f5isapi.dll, although appearing

to be incorrect, is actually correct. It is the type value, `F5.IsHandler`, that directs the server to the correct file.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.web>
    <httpHandlers>
      <clear />
      <add verb="*" path="f5isapi.dll" type="F5.IsHandler" />
    </httpHandlers>
  </system.web>
</configuration>
```

5. From the Start menu, choose Control Panel and double-click **Administration Tools**.
6. Double-click **Internet Information Services**.
This opens the IIS Management Console.
7. Expand the name of the local computer.
8. Allow the file `ASP.NET v2.0build_number`:
 - a) Select **Web Server Extensions**.
 - b) Select **ASP.NET v2.0build_number**.
 - c) Click **Allow**.
9. Create a new virtual directory named `scripts`:
 - a) Expand **Websites** and *Default Web Site*.
 - b) Right-click *Default Web Site*, choose **New**, and choose **Virtual Directory**.
 - c) Click **Next**.
 - d) Type `scripts` for the alias and click **Next**.
 - e) Type the directory you created in step 1 (`C:\Inetpub\scripts\`) and click **Next**.
 - f) Click **Next** again.
 - g) Click **Finished**.
10. Create an application pool for the file `F5.IsHandler.dll`:
 - a) Right-click **Application Pools**, choose **New**, and choose **Application Pool**.
 - b) Type `F5 Application Pool` in the **Application Pool ID** box and click **OK**.
11. Right click `scripts` and select **properties**.
12. Set up the application pool:
 - a) Click the **Virtual Directory** tab.
 - b) From the **Application Pool** list, select **F5 Application Pool**.
13. Set up the mappings:
 - a) Click the **Configuration** button.
 - b) On the Mappings tab of the Application Configuration screen, click **Add**.
 - c) For the executable, type the file name `C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll`.
 - d) For the file name extension, type `.dll`.
 - e) Clear the box for **Check that file exists** and click **OK**.
 - f) On the Application Configuration screen, click **OK**.
14. Set up directory security:

- a) Click the **Directory Security** tab.
- b) Click the **Edit** button.
- c) Disable authentication by clicking the **Anonymous Access and Integrated Windows** box.
- d) Check the **Basic Authentication** box and click **OK**.

***Note:** If you are not authenticating locally, you might need to set the default domain or realm.*

15. Set up the ASP.NET program:

- a) Click the **ASP.NET** tab.
- b) From the ASP.NET version list, select **2.0.buildnumber** (for example 2.0.50727).

16. On the scripts Properties page, click **OK**.

17. Set up access to the IIS metabase:

- a) Run the command `aspnet_regiis -ga <ASP.NETUsername>`.
- b) See the web site <http://support.microsoft.com/?kbid=267904>.

Once you have installed the plug-in, you must configure a WMI monitor, associate the configured monitor with the pool member, and set the load balancing method to Dynamic Ratio.

Installing the Data Gathering Agent F5.IsHandler.dll on an IIS 7.0 server

***Important:** Do not install the files `f5isapi.dll` or `f5isapi64.dll` on IIS version 7.0.*

Perform this task to install the Data Gathering Agent F5.IsHandler.dll on an IIS 7.0 server

1. Create a scripts directory under the directory C:\Inetpub. (C:\Inetpub\scripts).
2. Create a \bin directory under the scripts directory (C:\Inetpub\scripts\bin).
3. Copy the file F5.IsHandler.dll to the directory C:\Inetpub\scripts\bin.
4. In the C:\Inetpub\scripts directory, create the file web.config. This figure shows an example of web.config file on an IIS server running version 7.0.

```
<?xml version="1.0" encoding="UTF-8"?>
  <configuration>
    <system.webServer>
      <handlers>
        <clear />
        <add name="F5IsHandler" path="f5isapi.dll"
verb="*" type="F5.IsHandler" modules="ManagedPipelineHandler"
scriptProcessor="" resourceType="Unspecified" requireAccess="Script"
preCondition="" />
      </handlers>
      <security>
        <authentication>
          <anonymousAuthentication
enabled="false" />
        </authentication>
      </security>
    </system.webServer>
  </configuration>
```

Important: In this example, the path value `f5isapi.dll`, although appearing to be incorrect, is actually correct. It is the type value, `F5.IsHandler`, that directs the server to the correct file.

5. Allow anonymous authentication to be overridden by using the `appcmd` command to set the override mode in the machine-level `applicationHost.config` file.

```
appcmd set config "Default Web Site/scripts"
/section:anonymousAuthentication /overrideMode:Allow

/commit:APPHOST
```

6. Set up a new application pool for the file `F5.IsHandler.dll`:
 - a) From the Start menu, choose Control Panel.
 - b) Choose Administrative Tools
 - c) Choose Internet Information Services (IIS) Manager.
 - d) From **Connections**, expand `MachineName` (`MachineName\UserName`).
 - e) Right click the **Application Pools** menu and choose **Add Application Pool**.
 - f) In the **Name** box, type `F5 Application Pool`.
 - g) Click **OK**.
7. Create a new application named `scripts`:
 - a) Expand **Web Sites and MachineName**.
 - b) Right-click `MachineName` and choose **Add Application**.
 - c) In the **Alias** box, type `scripts`.
 - d) To change the application pool, click **Select**.
 - e) For the physical path, type the directory you created in step 1 (`C:\Inetpub\scripts\`).
 - f) Click **OK**.
8. Change the **Authentication** setting to **Basic Authentication**:
 - a) Select `scripts`.
 - b) In the center pane, double click **Authentication**.
 - c) Verify that the status of all items under **Authentication** is **Disabled**, except for the **Basic Authentication** item. To enable or disable an authentication item, right click the name and choose **Enable** or **Disable**.
9. Ensure that the `bin` directory and/or the `webconfig` files are removed from the hidden segments list.
10. To view a list of hidden items, click the `scripts` application (see step 7), double-click **Request Filtering**, and click the **Hidden Segments** section.

Once you have installed the plug-in, you must configure a WMI monitor, associate the configured monitor with the pool member, and set the load balancing method to Dynamic Ratio.

Installing the Data Gathering Agent `F5.IsHandler.dll` on an IIS 7.5 server

Important: Do not install the files `f5isapi.dll` or `f5isapi64.dll` on IIS version 7.5. For IIS servers running version 7.5, always install the file `F5.IsHandler.dll`.

Perform this task to install the Data Gathering Agent `F5.IsHandler.dll` on an IIS 7.5 server

1. Create a scripts directory under the directory C:\Inetpub (C:\Inetpub\scripts).
2. Create a \bin directory under the **scripts** directory (C:\Inetpub\scripts\bin).
3. Copy the file F5.IsHandler.dll to the directory C:\Inetpub\scripts\bin.
4. In the C:\Inetpub\scripts directory, create the file web.config.

```
<?xml version="1.0" encoding="UTF-8"?>
  <configuration>
    <system.webServer>
      <handlers>
        <clear />
        <add name="F5IsHandler" path="f5isapi.dll"
verb="*" type="F5.IsHandler"
modules="ManagedPipelineHandler"
scriptProcessor="" resourceType="Unspecified"
requireAccess="Script" preCondition=""
/>
      </handlers>
      <security>
        <authentication>
          <anonymousAuthentication enabled="false"
/>
        </authentication>
      </security>
    </system.webServer>
  </configuration>
```

Important: In the above example, the **path** value `f5isapi.dll`, although appearing to be incorrect, is actually correct. It is the **type** value, `F5.IsHandler`, that directs the server to the correct file.

5. Allow anonymous authentication to be overridden by using the `appcmd` command to set the override mode in the machine-level `applicationHost.config` file.

```
appcmd set config "Default Web Site/scripts"
/section:anonymousAuthentication
/overrideMode:Allow /commit:APPHOST
```

Note: `appcmd` is located in `\windows\system32\ntesrv.`

6. Set up a new application pool for the file `F5.IsHandler.dll`:
 - a) From the **Start** menu, choose **Control Panel**.
 - b) Choose **Administrative Tools**.
 - c) Choose **Internet Information Services (IIS) Manager**.
 - d) From **Connections**, expand `MachineName (MachineName\UserName)`.
 - e) Right click the **Application Pools** menu and choose **Add Application Pool**.
 - f) In the **Name** box, type `F5 Application Pool`.
 - g) Click **OK**.
 - h) From the **Application Pools** list, right click **F5 Application Pool** and choose **Advanced Settings**.
 - i) Under the **Process Model** List, click **Identity**, and then click the button to the right of **ApplicationPoolIdentity**.
 - j) Click the drop down for Built-in account, and choose **NetworkService**.
 - k) Click **OK**.
 - l) Click **OK**.
7. Create a new application named `scripts`:

- a) Expand **Web Sites and *MachineName***.
- b) Right click *MachineName* and choose **Add Application**.
- c) In the **Alias** box, type *scripts*.
- d) Change the application pool, click **Select**, select **F5 Application Pool** from the **Application Pool** drop-down, and click **OK**.
- e) For the physical path, type the directory you created in step 1 (C:\Inetpub\scripts\).
- f) Click **OK**.

8. Change the **Authentication setting to **Basic Authentication**:**

- a) Select **scripts**.
- b) In the center pane, double click **Authentication**.
- c) Verify that the status of all items under **Authentication** is **Disabled**, except for the **Basic Authentication** item. To enable or disable an authentication item, right click the name and choose **Enable** or **Disable**.

Once you have installed the plug-in, you must configure a WMI monitor, associate the configured monitor with the pool member, and set the load balancing method to Dynamic Ratio.

Index

A

- Accept-Encoding header
 - about 77
- Action on Service Down setting 51
- administrative partitions
 - and iRules 159
- algorithms
 - for load balancing 53
- all-match strategy
 - about 37
- Application Acceleration Manager
 - enabling 78, 82
- authentication
 - with PAM 123
- authentication modules
 - types of 123
- automatic node creation 46
- availability
 - of pool members 55

B

- best-match strategy
 - about 37
- Browsers
 - workarounds for compression 77

C

- cache server pools 157
- certificate-based authorization
 - and SSL LDAP 125
- certificate revocation
 - with CRLDP 128
 - with SSL OCSP 126
- certificates
 - and LDAP database 125
- chunked encoding 65
- chunking actions 66
- classification
 - of traffic 153
- class match command 162
- client access
 - controlling through LDAP 125–126
- client credentials
 - and HTTP 124
- client requests
 - distributing 49
- client traffic
 - redirecting to virtual servers 111
- clone pools
 - described 32
- clustered multiprocessing 35
- CMP 35
- compression
 - configuring for symmetric optimization 89
- connection management 19

- connection overload 51
- connection pooling
 - and XForwarded For header 68, 70
 - with OneConnect 129
- connection rate limitvirtual serverspool membersnodes
 - about 31
 - about connection rate limits 31
- connection reaping 19
- Connection reaping 20
- connections
 - and NATs 143, 145
 - and SNATs 148–149
 - resuming 140
- connection timeouts 20–21
- content adaptation
 - for requests/responses 86
- content-based routing 92
- Content-Type header 101
- content types
 - for HTML content modification 101
- cookie decryption 68
- cookie encryption 68
- cookie insertion
 - into HTTP headers 107–109
- cookie persistence
 - defined 107
- cookie persistence methods 107
- cookies
 - from servers 108
- cookie values
 - mapping to nodes 108
 - overwriting 107
- counters
 - and Statistics profiles 132
- CRLDP authentication
 - defined 128
- CRLs
 - and CRLDP 128
 - and SSL OCSP 126
- custom profiles
 - 60
 - as parent profiles 61

D

- data gathering agents
 - installing 167–168, 170–171
- data groups
 - defined 162
 - storage of 163
- data streams
 - replacing strings in 133
- default profiles
 - about 59
 - as parent profiles 61
- destination address persistence
 - defined 110

- destinations
 - and virtual servers [27](#)
- Diameter messages
 - sending [86](#)
- Diameter profile
 - purpose of [86](#)
- disabling and enabling pool members [51](#)
- DNS profile type
 - defined [133](#)
- DNSSEC
 - enabling [85](#)
- DNS traffic
 - managing [85](#)
- domain translation
 - and Set-Cookie header [92](#)
- dropped connections [31](#)
- dynamic load balancing
 - and plug-ins [141](#)

E

- electronic trading
 - about configuring FIX profile [96](#)
 - about logging FIX messages [98](#)
 - about steering traffic [96](#)
 - about tag substitution [96](#)
 - about using SSL encryption [98](#)
 - about validating FIX messages [96](#)
- error codes
 - from HTTP server responses [65](#)
- event declarations
 - defined [156](#)
- excess client headers [71](#)
- excess server headers [72](#)
- explicit proxy settings
 - [72](#)
 - bad request message [74](#)
 - bad response message [74](#)
 - connection failed message [73](#)
 - default connect handling [73](#)
 - DNS lookup failed message [73](#)
 - dns resolver [72](#)
 - host names [73](#)
 - route domain [72](#)
 - tunnel name [73](#)

F

- F5.IsHandler.dll file
 - installing [168](#), [170–171](#)
- f5lsapi.dll file
 - installing [167–168](#)
- f5lsapi64.dll file
 - installing [167–168](#)
- failure
 - and pool members [52](#)
- fallback error codes [65](#)
- fallback hosts [64](#)
- Fast HTTP profiles
 - purpose and benefits [116](#)
- Fast L4 profiles
 - purpose of [115](#)

- Fast L4 profile settings
 - partial listing of [116](#)
- first-match strategy
 - about [37](#)
- FIX profile
 - about configuring for electronic trading [96](#)
 - about full parsing validation [96](#)
 - about logging FIX messages [98](#)
 - about quick parsing validation [96](#)
 - about steering traffic [96](#)
 - about tag substitution [96](#)
 - about using SSL encryption [98](#)
 - about validating FIX messages [96](#)
- FIX protocol
 - supported versions [96](#)
- FTP commands
 - translating for IPv6 [84](#)
- FTP traffic
 - managing [84](#)

H

- hardware acceleration
 - for Layer 4 traffic [115](#)
- hash method
 - for cookie persistence [108](#)
- hash persistence
 - creating [110](#)
 - defined [110](#)
- header contents
 - erasing [65](#)
- header insertion
 - for pool members [107](#)
- headers
 - intercepting [107](#)
- header size
 - of HTTP requests [71](#)
- health monitors
 - about [45](#)
 - and pools [50](#)
 - specifying minimum [50](#)
- HTML content
 - manipulating [102](#)
 - modifying [101](#)
- HTML rules
 - types of [101](#)
- HTML tag attributes
 - modifying [101](#)
- HTTP::redirect command [158](#)
- HTTP/1.1 pipelining [72](#)
- HTTP allow truncated redirect [71](#)
- HTTP basic auth realm [64](#)
- HTTP compression
 - and buffering size [76](#)
 - and HTTP/1.0 [77](#)
 - and server response length [76](#)
 - browser workarounds for [77](#)
 - managing Content-Type responses with [75](#)
 - managing URI responses with [75](#)
- HTTP compression methods [76](#)
- HTTP Compression profile
 - about [74](#), [81](#)

- HTTP Compression profile (*continued*)
 - options 75, 82
- HTTP content adaptation 86
- HTTP cookie persistence
 - defined 107
- HTTP error codes
 - and fallback 65
- HTTP excess client headers 71
- HTTP excess server headers 72
- HTTP header insertion 65
- HTTP Location header 67
- HTTP maximum header count 71
- HTTP oversize client headers 71
- HTTP oversize server headers 71
- HTTP parent profile 64
- HTTP profile introduction 63
- HTTP profiles
 - and proxy mode 63
 - purpose of 63
 - Via Header settings 69
- HTTP proxy mode 63
- HTTP redirection 64
- HTTP redirections
 - rewriting 67
- HTTP request header size 71
- HTTP request latency
 - minimizing 93
- HTTP requests
 - initiating multiple 72
- HTTP response headers 65
- HTTP unknown methods 72

I

- ICAP profiles
 - for content adaptation 86
- ICAP servers
 - for content adaptation 86
- idle connections 19–20
- idle server-side connections 20
- idle timeout settings 21
- IDS systems
 - copying traffic to 32
- iFile commands 164
- iFile files
 - defined 163
- IIS
 - installing agents on 167–168, 170–171
- IIS version support 167
- inbound connections
 - and NATs 143
 - and SNATs 148
- IP address encoding
 - for cookie persistence 108
- IPv4 address format
 - converting to IPv4 85
- IPv6 address format
 - and FTP traffic 84
- iRule commands
 - and hash persistence 110
 - types of 156, 160
- iRule context 159

- iRule evaluation 159
- iRule event types 159
- iRules
 - and counters 132
 - and profiles 161
 - and virtual servers 160
 - basic elements of 155
 - described 155
 - for HTML content replacement 101
 - for persistence 113
 - ordering of 160
- iSession profiles
 - about 89
 - modifying compression 89

K

- Kerberos authentication
 - defined 128

L

- LAN traffic optimization
 - and TCP protocol 118
- latency
 - minimizing 93
- Layer 4 processing
 - offloading to hardware 115
- LDAP
 - and record matching 125
- LDAP authentication
 - defined 124
 - for access control 124
- LDAP authorization
 - types of 126
- LDAP credentials
 - types of 125
- LDAP database
 - searching 125
- linear white space 68
- load balancing methods 53
- load balancing pools
 - about 49
- Local Traffic Manager module
 - summary of 19
- local traffic policy matching
 - about 37
 - about actions 41
 - about all-match strategy 37
 - about best-match strategy 37
 - about conditions 38
 - about first-match strategy 37
 - about matching strategies 37
 - about rules 38
 - actions operands settings 41
 - controls settings 38
 - operands settings 39
 - requires settings 38
- logging
 - for DNS traffic 133
 - for HTTP traffic 133

M

Manual Resume feature [140](#)
 mean opinion score, *See* MOS
 message-oriented applications
 and SCTP profiles [120](#)
 methods, unknown [72](#)
 Microsoft RDP persistence
 defined [111](#)
 mirroring
 of connections [31](#)
 monitor associations [142](#)
 monitor destinations [138](#)
 monitoring
 types of [135](#)
 monitor instances [142](#)
 monitor plug-in files [165](#)
 monitor plug-ins
 and dynamic load balancing [141](#)
 monitors
 and pools [50](#)
 custom [137](#)
 pre-configured [136](#)
 removing [46](#)
 specifying minimum [50](#)
 types of [136](#)
 monitor settings
 about [136](#)
 MOS
 and Video Quality of Experience [99](#)
 MPTCP
 and mobile traffic optimization [119](#)
 MSRDP persistence
 defined [111](#)

N

named counters
 and Statistics profiles [132](#)
 NATs
 and pools [51](#)
 defined [143](#)
 for inbound connections [143](#)
 for outbound connections [145](#)
 network map
 about [21](#)
 network map display [22](#)
 node addresses
 and route domain IDs [45](#)
 node address setting
 about [45](#)
 node availability [46](#)
 node command [157](#)
 nodes
 and connection rates [47](#)
 assigning ratio weights to [47](#)
 associating monitors with [142](#)
 creating automatically [46](#)
 creating explicitly [46](#)
 defined [45](#)
 node states [47](#)
 node status [47](#)

NTLM

and OneConnect [131](#)

NTLM profile type
 defined [131](#)

O

object filtering [21](#)
 object relationships [22](#)
 object summary display [22](#)
 OCSP authentication
 defined [126](#)
 OneConnect
 and NTLM [131](#)
 OneConnect connection pooling [66](#)
 OneConnect feature
 and SNATs [148](#)
 OneConnect profiles
 purpose of [105](#)
 OneConnect profile type
 defined [129](#)
 operators
 defined [156](#)
 outbound connections
 and NATs [145](#)
 and SNATs [149](#)
 oversize server headers [71](#)

P

Packet Velocity ASIC
 for Layer 4 traffic [115](#)
 PAM technology
 defined [123](#)
 parent profiles [60](#)
 partitions
 and iRules [159](#)
 passwords
 and NTLM profiles [131](#)
 path translation
 and Set-Cookie header [92](#)
 persistence
 about [103](#)
 and destination IP addresses [110](#)
 and pools [106](#)
 and source IP addresses [112](#)
 and virtual addresses [105](#), [107](#)
 and virtual servers [106](#)
 for Microsoft RDP environments [111](#)
 for SIP traffic [112](#)
 for SSL sessions [112](#)
 using iRules [113](#)
 persistence criteria
 specifying [105](#)
 persistence mirroring
 about [31](#)
 persistence profiles
 types of [103](#)
 plug-ins
 and dynamic load balancing [141](#)
 pool command [157](#)
 pool features [49](#)

- pool member availability 55
- pool members
 - and passive failure 52
- pool member services 51
- pool member settings 56
- pool member states 58
- pool member status 58
- pools
 - about 49
 - and cookie persistence 107
 - and health monitors 50
 - and QoS levels 52
 - and session persistence 106
 - and ToS levels 51
 - purpose of 49
- pools and pool members
 - associating monitors with 142
- pool status 58
- port 443
 - and rewriting redirections 67
- port 4443
 - and rewriting redirections 67
- port encoding
 - for cookie persistence 109
- ports
 - and cookie persistence 109
- priority group activation 55
- profile command 161
- profiles
 - about HTTP Compression 74, 81
 - about iSession 89
 - about TCP 117
 - about Web Acceleration 78, 82
 - and iRules 161
 - and virtual servers 61
 - default 59
 - defined 81
 - described 59
 - Web Acceleration settings 78, 82
- profile settings
 - inheriting 61
 - overriding with iRules 162
- profile types
 - 59
 - miscellaneous 129
- Protocol profiles 115
- Proxy Via header 68
- PVA acceleration
 - for Layer 4 traffic 115

Q

- QoE, See video Quality of Experience
- QoS levels
 - and pools 52
- Quality of Service levels
 - and pools 52

R

- RADIUS authentication
 - defined 124

- RADIUS messages
 - sending 87
- RADIUS profiles
 - purpose of 87
- RealNetworks servers
 - and dynamic load balancing 141
- RealServer monitors 165
- RealSystem monitor plug-in
 - installing and compiling 166
- reaping 20
- record matching
 - and SSL LDAP 125
- remote authentication
 - and CRLDP 128
 - with Kerberos delegation 128
 - with LDAP 124
 - with RADIUS 124
 - with SSL LDAP 124
 - with SSL OCSP 126
 - with TACACS+ 124
- remote authentication modules
 - types of 123
- Request Adapt profiles
 - for content adaptation 86
- request latency
 - minimizing 93
- Request Logging profile type
 - defined 133
- requests
 - queuing 52
- resource availability
 - designating 140
- Response Adapt profiles
 - for content adaptation 86
- response chunking 65
- Reverse mode 139–140
- reverse proxy servers 91
- Rewrite profiles
 - about 90
 - rules for URI matching 91
- route domain IDs
 - for virtual server addresses 29
- RTSP protocol
 - defined 85
 - over UDP 85
- RTSP proxy configuration
 - described 85

S

- SCTP profiles
 - defined 120
- server arrays
 - optimizing 110, 112
- server availability
 - designating 140
- server connections
 - pooling of 129
- server information
 - as cookies 108
- server load balancing 53

- server names
 - and cookie persistence *108*
 - as cookies *107*
- servers
 - and cookie generation *108*
- service availability *51*
- service interruptions *31*
- session broker *111*
- session broker tokens *111*
- session data
 - ignoring *104–105*
- session directory *111*
- session persistence
 - about *103*
 - and OneConnect *104*
 - and pools *106*
 - and virtual addresses *105, 107*
 - and virtual servers *106*
- session persistence profiles
 - types of *103*
- Set-Cookie header
 - translation of *92*
- Set-Cookie translation
 - about *90*
- simple persistence
 - defined *112*
- SIP persistence
 - defined *112*
- SIP protocol
 - defined *87, 112*
- Slow Ramp Time setting *51*
- snat and snatpool commands *158*
- SNAT automap feature *150–151*
- SNAT pools *150–151*
- SNATs
 - and pools *51*
 - and VLANs *152*
 - defined *30, 147*
 - for inbound connections *148*
 - for outbound connections *149*
- SNAT translation *150–152*
- SNAT types *151*
- SNMP agents
 - and dynamic load balancing *141*
- SOCKS profile
 - described *95*
- source address persistence
 - defined *112*
- source IP addresses
 - and OneConnect *129*
- SPDY protocol
 - purpose of *93*
- SSL certificates
 - and LDAP *125*
- SSL LDAP
 - and record matching *125*
- SSL LDAP authentication
 - defined *124*
- SSL OCSP authentication
 - defined *126*
- SSSL persistence
 - defined *112*

- Statistics profile type
 - defined *132*
- sticky persistence
 - defined *110*
- storage
 - of data groups *163*
- streaming-media servers
 - controlling *85*
- Stream profile type
 - defined *133*
- string replacement
 - with Stream profiles *133*

T

- TACACS+ authentication
 - defined *124*
- tag replacement
 - in HTML content *102*
- Tcl
 - and iRules *155*
- Tcl expressions
 - for header insertion *65*
- TCP express
 - optimizing mobile traffic *118–119*
- TCP Express *115*
- TCP profiles
 - about *117*
 - and mobile traffic optimization *118–119*
 - optimized for LANs *118*
 - optimized for WANs *118*
- TCP request queuing *52*
- timeouts
 - for server-side connections *20*
 - for session persistence *20*
- timeout settings *19*
- Time Until Up feature *141*
- Tools Command Language
 - and iRules *155*
- ToS levels
 - and pools *51*
- traffic classes
 - defined *153*
- traffic control
 - through profiles *59*
- traffic filters
 - default *59*
- Transparent mode *139–140*
- Type of Service levels
 - and pools *51*

U

- UDP profiles *120*
- UIE persistence
 - and iRules *113*
- Universal Inspection Engine *113*
- unknown methods *72*
- URI rules
 - requirements for specifying *92*
- URI translation
 - and Set-Cookie header *92*

- URI translation (*continued*)
 - example of 90–91
- URI translation rules 91
- user credentials
 - and NTLM profiles 131
- user groups
 - and LDAP 125–126
- user roles
 - and LDAP 125–126

V

- Vary header
 - about 76
- Via header
 - about identifying intermediate routers 69
 - about identifying protocols for intermediate routers 69
 - about using in requests and responses 68
 - overview 68
- video quality of experience
 - about 98
- video Quality of Experience
 - and mean opinion score 99
 - and MOS 99
- virtual addresses
 - about 33
 - and session persistence 105
- virtual address status 35
- virtual server destinations 27

- virtual servers
 - and iRules 160
 - and route domains 29
 - and session persistence 106
 - defined 25
- virtual server settings 25
- virtual server status 35
- virtual server types 25
- VLAN groups
 - and Packet Velocity ASIC 115
- VLANs
 - and SNATs 152

W

- WAN traffic optimization
 - and TCP protocol 118
- Web Acceleration profile
 - about 78, 82
 - settings 78, 82
- Web Acceleration Profile
 - tmsh statistics description 83
- Windows servers
 - and dynamic load balancing 141
- Windows session directory 111

X

- XForwarded For header 68, 70
- XML content-based routing 92

