Pulse Secure Virtual Traffic Manager: TrafficScript Guide

Supporting Pulse Secure Virtual Traffic Manager 19.2

Product Release	**19.2**
Published	**15 July, 2019**
Document Version	**1.0**

Pulse Secure, LLC
2700 Zanker Road,
Suite 200 San Jose
CA 95134

www.pulsesecure.net

© 2019 by Pulse Secure, LLC. All rights reserved.

Pulse Secure and the Pulse Secure logo are trademarks of Pulse Secure, LLC in the United States. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Pulse Secure, LLC assumes no responsibility for any inaccuracies in this document. Pulse Secure, LLC reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Pulse Secure Virtual Traffic Manager: TrafficScript Guide

The information in this document is current as of the date on the title page.

END USER LICENSE AGREEMENT

The Pulse Secure product that is the subject of this technical documentation consists of (or is intended for use with) Pulse Secure software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.pulsesecure.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Contents

PREFACE 1
Document conventions 1
Text formatting conventions 1
Command syntax conventions 1
Notes and Warnings 2
Requesting Technical Support 2
Self-Help Online Tools and Resources 2
Opening a Case with PSGSC 3

INTRODUCTION 5
About This Guide 5
The TrafficScript Language 5
TrafficScript Examples 6
Application of Rules 7
Using a TrafficScript Rule 8

TRAFFICSCRIPT SYNTAX 13
Statements 13
Constants 13
Variables 14
Expressions 15
Operators 15
Type Casts in TrafficScript 18
Conditionals 18
The “If” Statement 19
The "switch" Statement 19
Loops 21
“for” Loops 21
‘while’ Loops 21
“do” Loops 21
“foreach” Loops 22
Other Flow Control 22
Complex Data Types 22
Arrays 22
Hashes 23
The Global Associative Array 24
The Process-Local Associative Array 25

© 2019 Pulse Secure, LLC.

The Connection-Local Array 25
Libraries 25
Functions 27
Escaping Regular Expressions 27
Creating New Subroutines in TrafficScript 28
Syntax 28
Request and Response Rules 29
Processing Multiple Requests and Responses 30
Specialized Protocol Handling Functions 30
Processing Other Protocols 31
The State Machine in Detail 31
Controlling the State Machine 31

SAMPLE TRAFFICSCRIPT RULES 33
Routing by Content Type 33
Restricting Access Based on the Time of Day 33
Customer Prioritization 34
Routing Based on XML Traffic 35
Example: Google Search Request 35
Authenticating User Access 36
Synchronizing Requests and Responses 38
Streaming HTTP Responses 39
Managing FTP Connections 40

TROUBLESHOOTING 43
Checking Syntax 43
Debugging Rules 43
Request and Response Rules 44
A Special Note About pool.use and pool.select 45

FUNCTION REFERENCE 47
TrafficScript Core Functions 47
array.append(array1, array2) 47
array.contains(array, value) 48
array.copy(array) 48
array.create(size, [default]) 48
array.filter(array, pattern, [flags]) 48
array.join(array, [separator]) 49
array.length(array) 49
array.pop(array) 49
array.push(array, value) 50

© 2019 Pulse Secure, LLC.

ARRAY.RESIZE(ARRAY, SIZE, [DEFAULT]) 50
ARRAY.REVERSE(ARRAY) 50
ARRAY.SHIFT(ARRAY) 50
ARRAY.SORT(ARRAY, [REVERSE]) 51
ARRAY.SORTNUMERICAL(ARRAY, [REVERSE]) 51
ARRAY.SPLICE(ARRAY, OFFSET, LENGTH, [VALUES]) 52
ARRAY.UNSHIFT(ARRAY, VALUE) 52
HASH.CONTAINS(HASH, KEY) 52
HASH.COUNT(HASH) 53
HASH.DELETE(HASH, KEY) 53
HASH.EMPTY(HASH) 53
HASH.KEYS(HASH) 53
HASH.VALUES(HASH) 54
JSON.DESERIALIZE(JSON_STRING) 54
JSON.SERIALIZE(OBJECT) 54
LANG.ASSERT(CONDITION, MESSAGE) 55
LANG.CHR(NUMBER) 55
LANG.DUMP(VARIABLE) 55
LANG.ISARRAY(DATA) 55
LANG.ISHASH(DATA) 56
LANG.MAX(PARAM1, PARAM2) 56
LANG.MIN(PARAM1, PARAM2) 56
LANG.ORD(STRING) 57
LANG.TOARRAY(VALUES) 57
LANG.TODOUBLE(VALUE) 57
LANG.TOHASH(VALUES) 57
LANG.TOINT(VALUE) 57
LANG.TOSTRING(VALUE) 58
LANG.TOCHAR(NUMBER) 58
LANG.WARN(MESSAGE) 58
MATH.ACOS(X) 58
MATH.ASIN(X) 59
MATH.ATAN(ANGLE) 59
MATH.CEIL(VALUE) 59
MATH.COS(ANGLE) 59
MATH.EXP(POWER) 59
MATH.FABS(VALUE) 60
MATH.FLOOR(VALUE) 60
MATH.LN(VALUE) 60
MATH.LOG(VALUE) 60
MATH.POW(NUM, POWER) 61

© 2019 Pulse Secure, LLC.

MATH.RANDOM(RANGE) 61
MATH.RINT(VALUE) 61
MATH.SIN(ANGLE) 61
MATH.SQRT(NUM) 61
MATH.TAN(ANGLE) 62
STRING.BERTOINT(STRING) 62
STRING.IREPLACE(STRING, SEARCH, REPLACEMENT) - DEPRECATED 62
STRING.IREPLACEALL(STRING, SEARCH, REPLACEMENT) - DEPRECATED 62
STRING.APPEND(STR1, STR2, ...) 62
STRING.BASE64DECODE(STRING) 63
STRING.BASE64ENCODE(STRING) 63
STRING.BYTESTODOTTED(STRING) 63
STRING.BYTESTOINT(STRING) 64
STRING.CMP(STR1, STR2) 64
STRING.CONTAINS(HAYSTACK, NEEDLE) 64
STRING.CONTAINSI(HAYSTACK, NEEDLE) 65
STRING.COUNT(HAYSTACK, NEEDLE, [START]) 65
STRING.DECRYPT(STRING, PASSPHRASE) 65
STRING.DOTTEDTOBYTES(IP ADDRESS) 66
STRING.DROP(STRING, COUNT) 66
STRING.ENCRYPT(STRING, PASSPHRASE) 66
STRING.ENDSWITH(STRING, SUFFIX) 67
STRING.ENDSWITHI(STRING, SUFFIX) 67
STRING.ESCAPE(STRING) 67
STRING.EXTRACTHOST(STRING) 68
STRING.EXTRACTPORT(STRING) 68
STRING.FIND(HAYSTACK, NEEDLE, [START]) 68
STRING.FINDI(HAYSTACK, NEEDLE, [START]) 69
STRING.FINDR(HAYSTACK, NEEDLE, [DISTANCEFROMENDTOSTART]) 69
STRING.HASH(STRING) 69
STRING.HASHMD5(STRING) 70
STRING.HASHSHA1(STRING) 70
STRING.HASHSHA256(STRING) 70
STRING.HASHSHA384(STRING) 70
STRING.HASHSHA512(STRING) 71
STRING.HEXTOINT(STRING) 71
STRING.HEXDECODE(ENCODED STRING) 71
STRING.HEXENCODE(STRING) 71
STRING.HTMLDECODE(ENCODEDSTRING) 72
STRING.HTMLENCODE(STRING) 72
STRING.ICMP(STR1, STR2) 72

© 2019 Pulse Secure, LLC.

STRING.INSERTBYTES(STRING, INSERTION, OFFSET) 72
STRING.INTTOBER(NUMBER) 73
STRING.INTTOBYTES(NUMBER, [WIDTH]) 73
STRING.INTTOHEX(STRING) 73
STRING.IPMASKMATCH(IP ADDRESS, CIDR IP SUBNET) 74
STRING.LEFT(STRING, COUNT) 74
STRING.LEN(STRING) 74
STRING.LENGTH(STRING) 74
STRING.LOWERCASE(STRING) 75
STRING.NORMALIZEIPADDRESS(STRING) 75
STRING.RANDOMBYTES(LENGTH) 75
STRING.REGEXESCAPE(STRING) 75
STRING.REGEXMATCH(STRING, REGEX, [FLAGS])) 76
STRING.REGEXSUB(STRING, REGEX, REPLACEMENT, [FLAGS]) 76
STRING.REPLACE(STRING, SEARCH, REPLACEMENT) 77
STRING.REPLACEALL(STRING, SEARCH, REPLACEMENT) 77
STRING.REPLACEALLI(STRING, SEARCH, REPLACEMENT) 78
STRING.REPLACEBYTES(STRING, REPLACEMENT, OFFSET) 78
STRING.REPLACEI(STRING, SEARCH, REPLACEMENT) 78
STRING.REVERSE(STRING) 79
STRING.RIGHT(STRING, COUNT) 79
STRING.SKIP(STRING, COUNT) 79
STRING.SPLIT(STRING, [SEPARATOR]) 79
STRING.SPRINTF(FORMAT STRING, ARGUMENTS) 80
STRING.STARTSWITH(STRING, PREFIX) 80
STRING.STARTSWITHI(STRING, PREFIX) 80
STRING.SUBSTRING(STRING, BASE, END) 80
STRING.TRIM(STRING) 81
STRING.UNESCAPE(ESCAPED STRING) 81
STRING.UPPERCASE(STRING) 81
STRING.URLENCODE(STRING) 81
STRING.URLENCODEEXCEPTRESERVED(STRING) 82
STRING.VALIDIPADDRESS(STRING) 82
STRING.WILDMATCH(STRING, PATTERN) 82
STRING.GMTIME.PARSE(STR) 83
SYS.DOMAINNAME() 83
SYS.GETENV(VARIABLE) 83
SYS.GETNETWORKINTERFACES(HASH_OF_OPTIONS) 83
SYS.GETPID() 84
SYS.HOSTNAME() 84
SYS.TIME() 84

© 2019 Pulse Secure, LLC.

SYS.TIMETOSTRING(UNIXTIME) 85
SYS.GMTIME.FORMAT(FORMAT, UNIXTIME) 85
SYS.LOCALTIME.FORMAT(FORMAT, UNIXTIME) 86
SYS.TIME.HIGHRES() 87
SYS.TIME.HOUR(UNIXTIME) 87
SYS.TIME.MINUTES(UNIXTIME) 87
SYS.TIME.MONTH() 88
SYS.TIME.MONTHDAY(UNIXTIME) 88
SYS.TIME.SECONDS(UNIXTIME) 88
SYS.TIME.WEEKDAY(UNIXTIME) 89
SYS.TIME.YEAR(UNIXTIME) 89
SYS.TIME.YEARDAY(UNIXTIME) 89
TRAFFIC MANAGER FUNCTIONS 90
AUTH.QUERY(AUTHENTICATOR, USER, [PASSWORD]) 91
CONNECTION.BYPASSPROTOCOLHANDLING() 92
CONNECTION.CHECKLIMITS([POOLNAME]) 92
CONNECTION.CLOSE(DATA, [READ]) 93
CONNECTION.DISCARD() 93
CONNECTION.GETBANDWIDTHCLASS() - DEPRECATED 94
CONNECTION.GETCOMPLETIONREASONCODE() 94
CONNECTION.GETCOMPLETIONREASONINFO() 95
CONNECTION.GETDATA(COUNT) - DEPRECATED 96
CONNECTION.GETDATALEN() - DEPRECATED 96
CONNECTION.GETLINE(OFFSET) - DEPRECATED 96
CONNECTION.GETLOCALIP() - DEPRECATED 96
CONNECTION.GETLOCALPORT() - DEPRECATED 96
CONNECTION.GETMEMORYUSAGE() 97
CONNECTION.GETNODE() 97
CONNECTION.GETPERSISTENCE() 97
CONNECTION.GETPOOL() 97
CONNECTION.GETPROXYCLIENTIP() 97
CONNECTION.GETPROXYCLIENTPORT() 98
CONNECTION.GETPROXYHEADERTYPE() 98
CONNECTION.GETPROXYSERVERIP() 99
CONNECTION.GETPROXYSERVERPORT() 99
CONNECTION.GETPROXYTLVDATA(TLV_TYPE) 100
CONNECTION.GETPROXYVERSION() 100
CONNECTION.GETREMOTEIP() - DEPRECATED 100
CONNECTION.GETREMOTEPORT() - DEPRECATED 101
CONNECTION.GETSERVICELEVELCLASS() 101
CONNECTION.GETVIRTUALSERVER() 101

© 2019 Pulse Secure, LLC.

CONNECTION.SETBANDWIDTHCLASS(NAME) - DEPRECATED 101
CONNECTION.SETDATA(REQUEST DATA) - DEPRECATED 101
CONNECTION.SETIDEMPOTENT(RESEND) - DEPRECATED 102
CONNECTION.SETPERSISTENCE(NAME) 102
CONNECTION.SETPERSISTENCEKEY(VALUE) 102
CONNECTION.SETPERSISTENCENODE(VALUE) 103
CONNECTION.SETSERVICELEVELCLASS(LEVEL) 103
CONNECTION.SLEEP(MILLISECONDS) 104
CONNECTION.DATA.GET(KEY) 104
CONNECTION.DATA.SET(KEY, VALUE) 104
COUNTER.INCREMENT(COUNTER, [AMOUNT]) 105
COUNTER64.INCREMENT(COUNTER, [AMOUNT]) 105
DATA.GET(KEY) 106
DATA.GETMEMORYFREE() 106
DATA.GETMEMORYUSAGE() 106
DATA.REMOVE(KEY) 106
DATA.RESET([PREFIX]) 107
DATA.SET(KEY, VALUE) 107
DATA.LOCAL.GET(KEY) 108
DATA.LOCAL.GETMEMORYFREE() 108
DATA.LOCAL.GETMEMORYUSAGE() 108
DATA.LOCAL.REMOVE(KEY) 109
DATA.LOCAL.RESET([PREFIX]) 109
DATA.LOCAL.SET(KEY, VALUE) 109
EVENT.EMIT(CUSTOM EVENT NAME, MESSAGE) 110
GEO.GETCITY(IP) 110
GEO.GETCOUNTRY(IP) 110
GEO.GETCOUNTRYCODE(IP) 110
GEO.GETDISTANCEKM(LAT1, LON1, LAT2, LON2) 111
GEO.GETDISTANCEMILES(LAT1, LON1, LAT2, LON2) 111
GEO.GETIPDISTANCEKM(IP1, IP2) 111
GEO.GETIPDISTANCEMILES(IP1, IP2) 111
GEO.GETLATITUDE(IP) 111
GEO.GETLOCATION() 112
GEO.GETLOCATIONLONLAT() 112
GEO.GETLONGITUDE(IP) 112
GEO.GETREGION(IP) 113
GEO.GETREGIONCODE(IP) 113
GLB.GETDOMAIN() 113
GLB.GETIPS() 113
GLB.SERVICE.AVOIDLOCATION(LOCATION) 114

GLB.SERVICE.GETLOCATIONWEIGHT(LOCATION, [SERVICE]) 114
GLB.SERVICE.GETNAME() 115
GLB.SERVICE.GETNEAREST LOCATION() 115
GLB.SERVICE.IGNORELOCATION(LOCATION) 115
GLB.SERVICE.ISLOCATIONLIVE(LOCATION, [SERVICE]) 116
GLB.SERVICE.LISTDOMAINS([SERVICE]) 116
GLB.SERVICE.LISTLOCATIONS([SERVICE]) 117
GLB.SERVICE.PREFERLOCATION(LOCATION) 117
GLB.SERVICE.SKIP() 118
GLB.SERVICE.USELOCATION(LOCATION) 118
HTTP.ADDHEADER(NAME, VALUE) 119
HTTP.ADDRESPONSEHEADER(NAME, VALUE) 119
HTTP.CHANGESITE(NAME) 119
HTTP.COOKIE(NAME) - DEPRECATED 120
HTTP.DISABLECLIENTKEEPALIVE() 120
HTTP.DOESFORMPARAMEXIST(PARAMETER) 120
HTTP.GETBODY([COUNT]) 121
HTTP.GETBODYLINES([COUNT]) 121
HTTP.GETCLIENTVERSION() 122
HTTP.GETCOOKIE(NAME) 122
HTTP.GETCOOKIES() 123
HTTP.GETFORMPARAM(PARAMETER, [SEPARATOR]) 123
HTTP.GETFORMPARAMNAMES(SEPARATOR) - DEPRECATED 123
HTTP.GETFORMPARAMS() 124
HTTP.GETHEADER(NAME) 124
HTTP.GETHEADERNAMES() - DEPRECATED 124
HTTP.GETHEADERS() 124
HTTP.GETHOSTHEADER() 125
HTTP.GETMETHOD() 125
HTTP.GETMULTIPARTATTACHMENT(PART) 125
HTTP.GETPATH() 126
HTTP.GETQUERYSTRING() 126
HTTP.GETRAWQUERYSTRING() 126
HTTP.GETRAWURL() 126
HTTP.GETREQUEST() 127
HTTP.GETRESPONSE() 127
HTTP.GETRESPONSEBODY([COUNT]) 128
HTTP.GETRESPONSEBODYLINES([COUNT]) 128
HTTP.GETRESPONSECODE() 129
HTTP.GETRESPONSECOOKIE(NAME) 129
HTTP.GETRESPONSECOOKIES() 129

© 2019 Pulse Secure, LLC.

HTTP.GETRESPONSEHEADER(NAME) 129
HTTP.GETRESPONSEHEADERNAMES() - DEPRECATED 130
HTTP.GETRESPONSEHEADERS() 130
HTTP.GETRESPONSEVERSION() 130
HTTP.GETSCHEME() 130
HTTP.GETVERSION() 131
HTTP.HEADEREXISTS(NAME) 131
HTTP.LISTFORMPARAMNAMES() 131
HTTP.LISTHEADERNAMES() 132
HTTP.LISTRESPONSEHEADERNAMES() 132
HTTP.NORMALIZEPATH(URL) 132
HTTP.REDIRECT(ADDRESS) 133
HTTP.REMOVECOOKIE(NAME) 133
HTTP.REMOVEHEADER(NAME) 133
HTTP.REMOVERESPONSECOOKIE(NAME) 134
HTTP.REMOVERESPONSEHEADER(NAME) 134
HTTP.RESPONSEHEADEREXISTS(NAME) 134
HTTP.SCRUBREQUESTHEADERS(HEADER1, HEADER2, ...) 135
HTTP.SCRUBRESPONSEHEADERS(HEADER1, HEADER2, ...) 135
HTTP.SENDRESPONSE(CODE, TYPE, BODY, HEADERS) 136
HTTP.SETBODY(BODY) 136
HTTP.SETCOOKIE(NAME, VALUE) 136
HTTP.SETESCAPEDPATH(URL) 137
HTTP.SETHEADER(NAME, VALUE) 137
HTTP.SETIDEMPOTENT(RESEND) 137
HTTP.SETMETHOD(METHOD) 138
HTTP.SETPATH(URL) 138
HTTP.SETQUERYSTRING(QUERYSTRING) 139
HTTP.SETRAWPATH(URL) 139
HTTP.SETRAWQUERYSTRING(QUERYSTRING) 139
HTTP.SETRESPONSEBODY(BODY, [TRANSFER-ENCODING]) 139
HTTP.SETRESPONSECODE(CODE, [MESSAGE]) 140
HTTP.SETRESPONSECOOKIE(NAME, VALUE, [OPTIONS]) 140
HTTP.SETRESPONSEHEADER(NAME, VALUE) 140
HTTP.APTIMIZER.BYPASS() 141
HTTP.APTIMIZER.USE(SCOPE, PROFILE) 141
HTTP.AUTH.GETAUTHENTICATEDUSER() 142
HTTP.CACHE.DISABLE() 142
HTTP.CACHE.ENABLE() 142
HTTP.CACHE.EXISTS([POOLNAME]) 143
HTTP.CACHE.RESPONDIFCACHED([POOLNAME]) 143

HTTP.CACHE.SETKEY(KEY) 144
HTTP.COMPRESS.DISABLE() 144
HTTP.COMPRESS.ENABLE() 145
HTTP.CONNECTION.SENDORIGIN(ORIGIN1, ORIGIN2, ...) 145
HTTP.KERBEROS.GETCLIENTPRINCIPALNAME() 146
HTTP.KERBEROS.GETDELEGATEDTICKET() 146
HTTP.KERBEROS.GETPRINCIPAL() 146
HTTP.KERBEROS.GETTARGETPRINCIPALNAME() 147
HTTP.KERBEROS.ISENABLED() 147
HTTP.KERBEROS.SETCLIENTPRINCIPALNAME(PRINCIPALNAME) 148
HTTP.KERBEROS.SETENABLED(ENABLED) 148
HTTP.KERBEROS.SETPRINCIPAL(PRINCIPALCATALOGITEM) 148
HTTP.KERBEROS.SETTARGETPRINCIPALNAME(PRINCIPALNAME) 149
HTTP.REQUEST.DELETE(URL, [HEADERS], [TIMEOUT]) 149
HTTP.REQUEST.GET(URL, [HEADERS], [TIMEOUT]) 150
HTTP.REQUEST.HEAD(URL, [HEADERS], [TIMEOUT]) 151
HTTP.REQUEST.POST(URL, POST DATA, [HEADERS], [TIMEOUT]) 151
HTTP.REQUEST.PUT(URL, PUT DATA, [HEADERS], [TIMEOUT]) 152
HTTP.REQUEST.SSL.GET(URL, CA_CERTIFICATES, [HEADERS], [TIMEOUT], [VERIFY-HOSTNAME]) 153
HTTP.STREAM.CONTINUEFROMBACKEND([DATA]) 154
HTTP.STREAM.FINISHRESPONSE([DATA]) 155
HTTP.STREAM.READBULKRESPONSE(COUNT, [DELIMITER]) 155
HTTP.STREAM.READRESPONSE(COUNT, [DELIMITER]) 156
HTTP.STREAM.STARTRESPONSE(RESP_CODE, CONTENT_TYPE, [CONTENT_LENGTH, HEADERS]) 157
HTTP.STREAM.WRITERESPONSE(DATA) 158
JAVA.RUN(JAVA EXTENSION CLASS NAME, [OPTIONS]) 158
LOG.ERROR(MESSAGE) 158
LOG.INFO(MESSAGE) 159
LOG.WARN(MESSAGE) 159
NET.DNS.RESOLVEHOST(HOSTNAME) 159
NET.DNS.RESOLVEHOST6(HOSTNAME) 159
NET.DNS.RESOLVEIP(IP ADDRESS) 160
POOL.ACTIVENODES(POOL) 160
POOL.CHECKNODE(POOL, HOST, PORT) 160
POOL.DISABLEDNODES(POOL) 161
POOL.DRAININGNODES(POOL) 161
POOL.FAILEDNODES(POOL) 161
POOL.LISTACTIVENODES(POOL) 162
POOL.LISTALLNODES(POOL) 162

© 2019 Pulse Secure, LLC.

POOL.LISTDISABLEDNODES(POOL) 162
POOL.LISTDRAININGNODES(POOL) 163
POOL.LISTFAILEDNODES(POOL) 163
POOL.SELECT(POOL, [HOST, PORT]) 163
POOL.USE(POOL, [HOST, PORT]) 164
RATE.GETBACKLOG(CLASS_NAME, [CONTEXT]) 165
RATE.USE(CLASS_NAME, [CONTEXT]) 165
RATE.USE.NOQUEUE(CLASS_NAME, [CONTEXT]) 166
RECENTCONNS.EXCLUDE() 166
RECENTCONNS.INCLUDE() 167
RECENTCONNS.MARKEDFORINCLUSION() 167
REQUEST.AVOIDNODE(NODE) 167
REQUEST.ENDSAT(OFFSET) 168
REQUEST.ENDSWITH(REGEX) 168
REQUEST.GET([COUNT]) 169
REQUEST.GETBANDWIDTHCLASS() 169
REQUEST.GETDSCP() 169
REQUEST.GETDESTIP() 170
REQUEST.GETDESTPORT() 170
REQUEST.GETLENGTH() 170
REQUEST.GETLINE([REGEX], [OFFSET]) 170
REQUEST.GETLOCALIP() 171
REQUEST.GETLOCALPORT() 171
REQUEST.GETLOGENABLED() - DEPRECATED 171
REQUEST.GETMAXTRANSACTIONDURATION() 172
REQUEST.GETREMOTEIP() 172
REQUEST.GETREMOTEPORT() 172
REQUEST.GETRETRIES() 172
REQUEST.GETTOS() - DEPRECATED 173
REQUEST.ISRESENDABLE() 173
REQUEST.RETRY() 173
REQUEST.SENDRESPONSE(DATA) 174
REQUEST.SET(REQUEST DATA) 175
REQUEST.SETBANDWIDTHCLASS(NAME) 175
REQUEST.SETDSCP(6-BIT DSCP FIELD) 175
REQUEST.SETIDEMPOTENT(RESEND) 176
REQUEST.SETLOGENABLED(ENABLED) - DEPRECATED 176
REQUEST.SETMAXCONNECTIONATTEMPTS(INT) 176
REQUEST.SETMAXREPLYTIME(SECONDS) 177
REQUEST.SETMAXTIMEDOUTCONNECTIONATTEMPTS(INT) 177
REQUEST.SETMAXTRANSACTIONDURATION(INT) 177

REQUEST.SETREMOTEIP(IPADDR) 178
REQUEST.SETTOS(TYPE OF SERVICE) - DEPRECATED 178
REQUEST.SETVIRTUALSERVERTIMEOUT(SECONDS) 178
REQUEST.SKIP(COUNT) 179
REQUESTLOG.EXCLUDE() 179
REQUESTLOG.INCLUDE() 180
REQUESTLOG.MARKEDFORINCLUSION() 180
RESOURCE.EXISTS(FILENAME) 180
RESOURCE.GET(FILENAME) 181
RESOURCE.GETLINES(FILENAME) 181
RESOURCE.GETMD5(FILENAME) 181
RESOURCE.GETMTIME(FILENAME) 182
RESPONSE.APPEND(RESPONSE DATA) 182
RESPONSE.CLOSE() 182
RESPONSE.FLUSH(COUNT) 183
RESPONSE.GET([COUNT]) 183
RESPONSE.GETBANDWIDTHCLASS() 184
RESPONSE.GETDSCP() 184
RESPONSE.GETLENGTH() 184
RESPONSE.GETLINE([REGEX], [OFFSET]) 184
RESPONSE.GETLOCALIP() 185
RESPONSE.GETLOCALPORT() 185
RESPONSE.GETREMOTEIP() 185
RESPONSE.GETREMOTEPORT() 186
RESPONSE.GETTOS() - DEPRECATED 186
RESPONSE.SET(RESPONSE DATA) 186
RESPONSE.SETBANDWIDTHCLASS(NAME) 186
RESPONSE.SETDSCP(6-BIT DSCP FIELD) 187
RESPONSE.SETTOS(TYPE OF SERVICE) - DEPRECATED 187
RTSP.ADDREQUESTHEADER(NAME, VALUE) 187
RTSP.ADDRESPONSEHEADER(NAME, VALUE) 188
RTSP.GETMETHOD() 188
RTSP.GETPATH() 188
RTSP.GETRAWURL() 188
RTSP.GETREQUEST() 189
RTSP.GETREQUESTBODY([COUNT]) 189
RTSP.GETREQUESTBODYLINES(COUNT) 189
RTSP.GETREQUESTHEADER(NAME) 190
RTSP.GETREQUESTHEADERNAMES() - DEPRECATED 190
RTSP.GETREQUESTHEADERS() 190
RTSP.GETRESPONSE() 190

© 2019 Pulse Secure, LLC.

RTSP.GETRESPONSEBODY([COUNT]) 191
RTSP.GETRESPONSEBODYLINES(COUNT) 191
RTSP.GETRESPONSECODE() 191
RTSP.GETRESPONSEHEADER(NAME) 192
RTSP.GETRESPONSEHEADERNAMES() - DEPRECATED 192
RTSP.GETRESPONSEHEADERS() 192
RTSP.GETVERSION() 192
RTSP.LISTREQUESTHEADERNAMES() 193
RTSP.LISTRESPONSEHEADERNAMES() 193
RTSP.REDIRECT(PATH) 193
RTSP.REMOVEREQUESTHEADER(NAME) 194
RTSP.REMOVERESPONSEHEADER(NAME) 194
RTSP.REQUESTHEADEREXISTS(NAMES) 194
RTSP.RESPONSEHEADEREXISTS(NAME) 194
RTSP.SENDRESPONSE(CODE, BODY, HEADERS) 195
RTSP.SETMETHOD(METHOD) 195
RTSP.SETPATH(URL) 195
RTSP.SETREQUESTBODY(BODY) 196
RTSP.SETREQUESTHEADER(NAME, VALUE) 196
RTSP.SETRESPONSEBODY(BODY) 196
RTSP.SETRESPONSECODE(CODE, [MESSAGE]) 197
RTSP.SETRESPONSEHEADER(NAME, VALUE) 197
RULE.GETNAME() 197
RULE.GETSTATE() 198
SIP.ADDREQUESTHEADER(NAME, VALUE, AT_TOP) 198
SIP.ADDRESPONSEHEADER(NAME, VALUE, AT_TOP) 198
SIP.GETMETHOD() 199
SIP.GETREQUEST() 199
SIP.GETREQUESTBODY() 199
SIP.GETREQUESTBODYLINES() 200
SIP.GETREQUESTHEADER(NAME) 200
SIP.GETREQUESTHEADERNAMES() - DEPRECATED 200
SIP.GETREQUESTHEADERS() 200
SIP.GETREQUESTURI() 201
SIP.GETRESPONSE() 201
SIP.GETRESPONSEBODY() 201
SIP.GETRESPONSEBODYLINES() 202
SIP.GETRESPONSECODE() 202
SIP.GETRESPONSEHEADER(NAME) 202
SIP.GETRESPONSEHEADERNAMES() - DEPRECATED 203
SIP.GETRESPONSEHEADERS() 203

SIP.GETVERSION() 203
SIP.LISTREQUESTHEADERNAMES() 203
SIP.LISTRESPONSEHEADERNAMES() 204
SIP.REDIRECT(CONTACT) 204
SIP.REMOVEREQUESTHEADER(NAME) 204
SIP.REMOVERESPONSEHEADER(NAME) 205
SIP.REQUESTHEADEREXISTS(NAME) 205
SIP.RESPONSEHEADEREXISTS(NAME) 205
SIP.SENDRESPONSE(CODE, REASON, [HEADERS], [BODY]) 206
SIP.SETMETHOD(METHOD) 206
SIP.SETREQUESTBODY(BODY) 206
SIP.SETREQUESTHEADER(NAME, VALUE) 207
SIP.SETREQUESTURI(URI) 207
SIP.SETRESPONSEBODY(BODY) 208
SIP.SETRESPONSECODE(CODE, [MESSAGE]) 208
SIP.SETRESPONSEHEADER(NAME, VALUE) 208
SLM.CONFORMING([CLASS NAME]) 209
SLM.ISOK([CLASS_NAME]) 209
SLM.THRESHOLD([CLASS_NAME]) 210
SSL.CLIENTCERT() 210
SSL.CLIENTCERTALGORITHM() 211
SSL.CLIENTCERTCHAIN() 211
SSL.CLIENTCERTCOMMONNAME() 211
SSL.CLIENTCERTENDDATE() 212
SSL.CLIENTCERTHASH() 212
SSL.CLIENTCERTISSUER() 212
SSL.CLIENTCERTPUBLICKEY() 213
SSL.CLIENTCERTSHA1() 213
SSL.CLIENTCERTSERIAL() 213
SSL.CLIENTCERTSERIALDEC() 214
SSL.CLIENTCERTSTARTDATE() 214
SSL.CLIENTCERTSTATUS() 214
SSL.CLIENTCERTSUBJECT() 215
SSL.CLIENTCERTVERSION() 215
SSL.CLIENTCIPHER() 215
SSL.CLIENTSUPPORTSSECURERENEGOTIATION() 216
SSL.GETCLIENTCLOSEALERT() 216
SSL.GETSERVERCLOSEALERT() 216
SSL.GETTLSSERVERNAME() 217
SSL.ISSSL() 217
SSL.REQUESTCERT() 218

© 2019 Pulse Secure, LLC.

SSL.REQUIRECERT() 218
SSL.SERVERCERT() 219
SSL.SERVERCERTALGORITHM() 219
SSL.SERVERCERTCOMMONNAME() 220
SSL.SERVERCERTENDDATE() 220
SSL.SERVERCERTHASH() 220
SSL.SERVERCERTISSUER() 221
SSL.SERVERCERTNAME() 221
SSL.SERVERCERTPUBLICKEY() 221
SSL.SERVERCERTSHA1() 222
SSL.SERVERCERTSERIAL() 222
SSL.SERVERCERTSTARTDATE() 222
SSL.SERVERCERTSUBJECT() 222
SSL.SERVERCERTVERSION() 223
SSL.SERVERSITENAME() 223
SSL.SETCLIENTCLOSEALERT(ALERTFLAG) 223
SSL.SETSERVERCLOSEALERT(ALERTFLAG) 224
SSL.SETTLSSERVERNAME(SERVERNAME) 224
SSL.SSLKEYLOGLINE() 225
SSL.SSLSESSIONID() 225
STATS.GETQUEUETIME() 225
STATS.GETSERVERRESPONSETIME() 226
STATS.GETTRANSACTIONDURATION() 226
SYS.GETRESTAPIPORT() 226
SYS.ISFIPS() 227
TCP.CLOSE(SOCK) 227
TCP.CONNECT(IP, PORT, [TIMEOUT]) 227
TCP.READ(SOCKET, MAXIMUM, [TIMEOUT]) 228
TCP.WRITE(SOCKET, DATA, [TIMEOUT]) 228
UDP.SENDTO(HOST, PORT, DATA) 229
XML.VALIDATE(DOCUMENT, DTD) - DEPRECATED 229
XML.VALIDATE.DTD(DOCUMENT, DTD) 229
XML.VALIDATE.XSD(DOCUMENT, SCHEMA) 229
XML.XPATH.MATCHNODECOUNT(DOC, NSPACEMAP, QUERY) 230
XML.XPATH.MATCHNODESET(DOC, NSPACEMAP, QUERY) 231
XML.XSLT.TRANSFORM(DOCUMENT, STYLESHEET) 231

© 2019 Pulse Secure, LLC.

© 2019 Pulse Secure, LLC.

Preface

- Document conventions 1
- Requesting Technical Support 2

Document conventions

The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in Pulse Secure technical documentation.

Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used in the flow of the text to highlight specific words or phrases.

Format	Description
bold text	Identifies command names
	Identifies keywords and operands
	Identifies the names of user-manipulated GUI elements
	Identifies text to enter at the GUI
italic text	Identifies emphasis
	Identifies variables
	Identifies document titles
`Courier Font`	Identifies command output
	Identifies command syntax examples

Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
bold text	Identifies command names, keywords, and command options.
italic text	Identifies a variable.
[ ]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.

© 2019 Pulse Secure, LLC.

Convention	Description
{ **x** | **y** | **z** }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x | **y**	A vertical bar separates mutually exclusive elements.
< >	Non-printing characters, for example, passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, member[member...].
\	Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Notes and Warnings

Note, Attention, and Caution statements might be used in this document.

Note: A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION

An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

CAUTION

A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

Requesting Technical Support

Technical product support is available through the Pulse Secure Global Support Center (PSGSC). If you have a support contract, file a ticket with PSGSC.

- Product warranties—For product warranty information, visit https://support.pulsesecure.net/product-service-policies/

Self-Help Online Tools and Resources

For quick and easy problem resolution, Pulse Secure provides an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: https://support.pulsesecure.net
- Search for known bugs: https://support.pulsesecure.net
- Find product documentation: https://www.pulsesecure.net/techpubs
- Download the latest versions of software and review release notes: https://support.pulsesecure.net
- Open a case online in the CSC Case Management tool: https://support.pulsesecure.net

© 2019 Pulse Secure, LLC.

- To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: https://support.pulsesecure.net

For important product notices, technical articles, and to ask advice:

- Search the Pulse Secure Knowledge Center for technical bulletins and security advisories: https://kb.pulsesecure.net
- Ask questions and find solutions at the Pulse Community online forum: https://community.pulsesecure.net

Opening a Case with PSGSC

You can open a case with PSGSC on the Web or by telephone.

- • Use the Case Management tool in the PSGSC at https://support.pulsesecure.net.
- • Call 1-844 751 7629 (Toll Free, US).

For international or direct-dial options in countries without toll-free numbers, see https://support.pulsesecure.net/support/support-contacts/

© 2019 Pulse Secure, LLC.

© 2019 Pulse Secure, LLC.

Introduction

This chapter introduces the TrafficScript rules language, and provides a brief overview of it's usage. This chapter includes the following sections:

- About This Guide 5
- The TrafficScript Language. 5
- Application of Rules. 7
- Using a TrafficScript Rule 8

About This Guide

The *Pulse Secure Virtual Traffic Manager: TrafficScript Guide* describes the Pulse Secure Virtual Traffic Manager (Traffic Manager) TrafficScript language.

This guide introduces you to the syntax and commands used in TrafficScript, and is intended as a complete reference to all TrafficScript functions available in the Traffic Manager.

The TrafficScript Language

By using the TrafficScript language, you can write tailored traffic management rules to inspect, manage and route requests and responses within your Traffic Manager deployment.

TrafficScript rules can manage connections in any TCP-based or UDP-based protocol. The Traffic Manager executes designated rules whenever a new connection or network request is received, whenever it receives a response from a back-end server node, or at the completion of a transaction. TrafficScript rules can inspect the incoming and outgoing data in the connection, and other aspects such as the remote client address. They can also connect to external TCP or HTTP services on demand.

The TrafficScript rules can then modify the request or response (for example, rewriting the URL or headers in an HTTP request), set session persistence parameters, or decide how to route the request to the most appropriate pool. This makes it possible to control precisely how traffic is managed, using rules designed to meet to your specific hosting requirements.

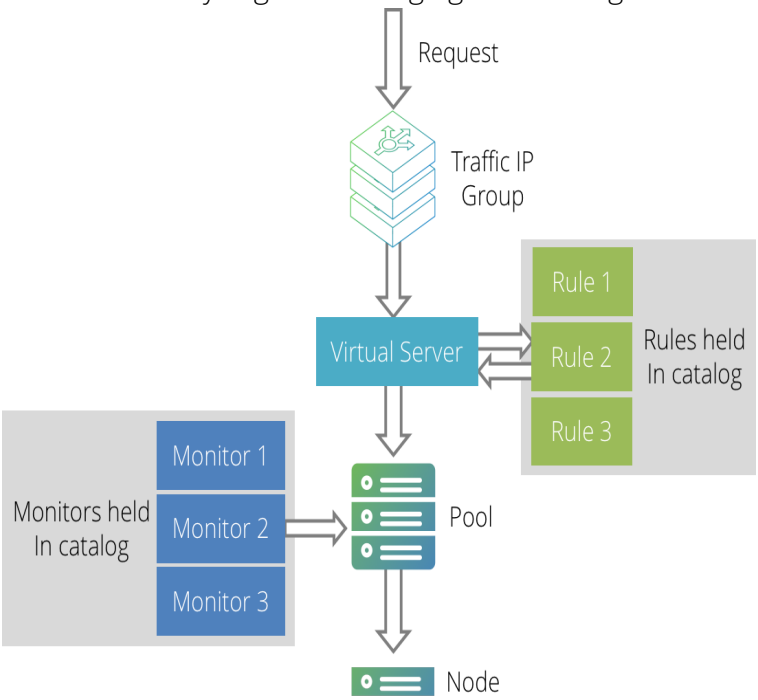
This manual introduces the TrafficScript language, provides details of syntax and usage, and then includes a full reference guide to the available function set for version 19.2.

TrafficScript rules are stored in the Rules Catalog and applied to your Virtual Servers.

Note: TrafficScript rules can use regular expressions. Before using regular expressions, read and understand the security considerations in the "Administration System Security" chapter of the *Pulse Secure Virtual Traffic Manager: User's Guide*.

© 2019 Pulse Secure, LLC.

FIGURE 1 Analyzing and managing traffic using TrafficScript rules



Use TrafficScript to perform any of these traffic management tasks:

- Inspect incoming or outgoing traffic and rewrite it fully or in part as desired.
- Restrict a Web site to a certain range of IP addresses.
- Apply selective management to elements such as Web spiders.
- Enable or disable functions for a given request or response (such as compression)
- Retry requests that generate errors a maximum number of times.
- Future-proof your services against any change in the back-end components of the system.
- Work around broken links and content on your Web site.

TrafficScript Examples

The following example TrafficScript rule can be used with HTTP requests. It handles the request as follows:

- Requests for “www.example.co.uk” are rewritten to “www.example.com”.
- Requests for .jsp pages are routed to a set of application servers (the pool named “JSPServers”).
- Requests for URLs beginning “/secure” are only allowed during office hours.

```
# Rewrite host header if necessary
if( http.getHeader( "Host" ) == "www.example.co.uk" ) {
  http.setHeader( "Host", "www.example.com" );
}

$path = http.getPath();

# Give .jsp requests to the 'JSPServers' pool
```


© 2019 Pulse Secure, LLC.

```
if( string.endsWith( $path, ".jsp" ) ) {
  pool.use( "JSPServers" );
}

# Deny access to /secure outside office hours
if( string.startsWith( $path, "/secure" ) ) {
  if( sys.time.hour() < 9 || sys.time.hour() >= 18 ){
    connection.discard();
  }
}
```

The next example rule can be used with HTTP responses. It processes the response as follows:

- If the HTTP status code is 404 or 5xx, retry the request a maximum of 3 times.
- If the response contains references to www.example.co.uk, rewrite it by changing these references to www.example.com.

```
$code = http.getResponseCode();
if( $code == 404 || $code >= 500 ) {
  if( request.getRetries() < 3 ) {
    # Avoid the current node when we retry,
    # if possible
    request.avoidNode( connection.getNode() );
    request.retry();
  } else {
    http.sendResponse( "312 Redirect", "text/plain",
      "", "Location: /" );
  }
}

# We're only going to process text/html responses, so
# break out of the rule if the response is of a
# different type...
if( http.getResponseHeader(
      "Content-Type" ) != "text/html" ) break;

$response = http.getResponseBody();

if( string.contains( $response,"www.example.co.uk" ) ) {
  $response = string.regexsub( $response,
         "www.example.co.uk", "www.example.com", "g" );
  http.setResponseBody( $reponse );
}
```

Application of Rules

Rules are typically used by a virtual server to choose a pool to handle a request. The rule can inspect any part of the request, possibly modify it, and decide which pool should handle the request.

Further uses of TrafficScript rules include:

- You can use a rule to dictate session persistence information to a pool. After inspecting the request the rule can use the `connection.setPersistenceKey()` function to provide a string to persist on. This string is used by the "Universal" session persistence method to identify the session the request belongs to.
- Rules can be used to check the response from the server and modify it, or even retry the request (if possible) if a transient error was detected.
- If you use various back-end systems with different presentation styles or even different protocols, rules can be used to integrate them into a single, coherent and consistent service. Incoming requests can be rewritten into the format suitable for the required service, and responses can be rewritten into a single, consistent form.
- For example, HTTP requests that involve a database lookup can be rewritten into SOAP request for a Web Service; the XML response can then be transformed into a suitable HTML document to return using HTTP.
- Rules can override the classes assigned to a connection by the Virtual Server or the Pool. This way, they can specify custom behaviour for each connection. For example, connections to a slow resource can be given a longer response time tolerance. Classes you can assign in this way include "Service Level Monitoring", "Session Persistence" and "Bandwidth Management".
- Service Protection classes can use rules. If you have associated a Service Protection class with your Virtual Server, it inspects the incoming packets. The class might use a rule to check the packet for a match with known Web worms or viruses. This rule is executed before the main processing of the Virtual Server is carried out.

Using a TrafficScript Rule

TrafficScript rules are stored in the Rules catalog. You can use the Rules catalog to create rules, upload them from an external source, modify or duplicate them, and delete unused rules as required.

You can configure a Virtual Server to execute one or more rules from the catalog each time it receives a new request or response, or at completion of a transaction. This way, several different Virtual Servers can use the same rule, and modifications to the rule take effect on all Virtual Servers.

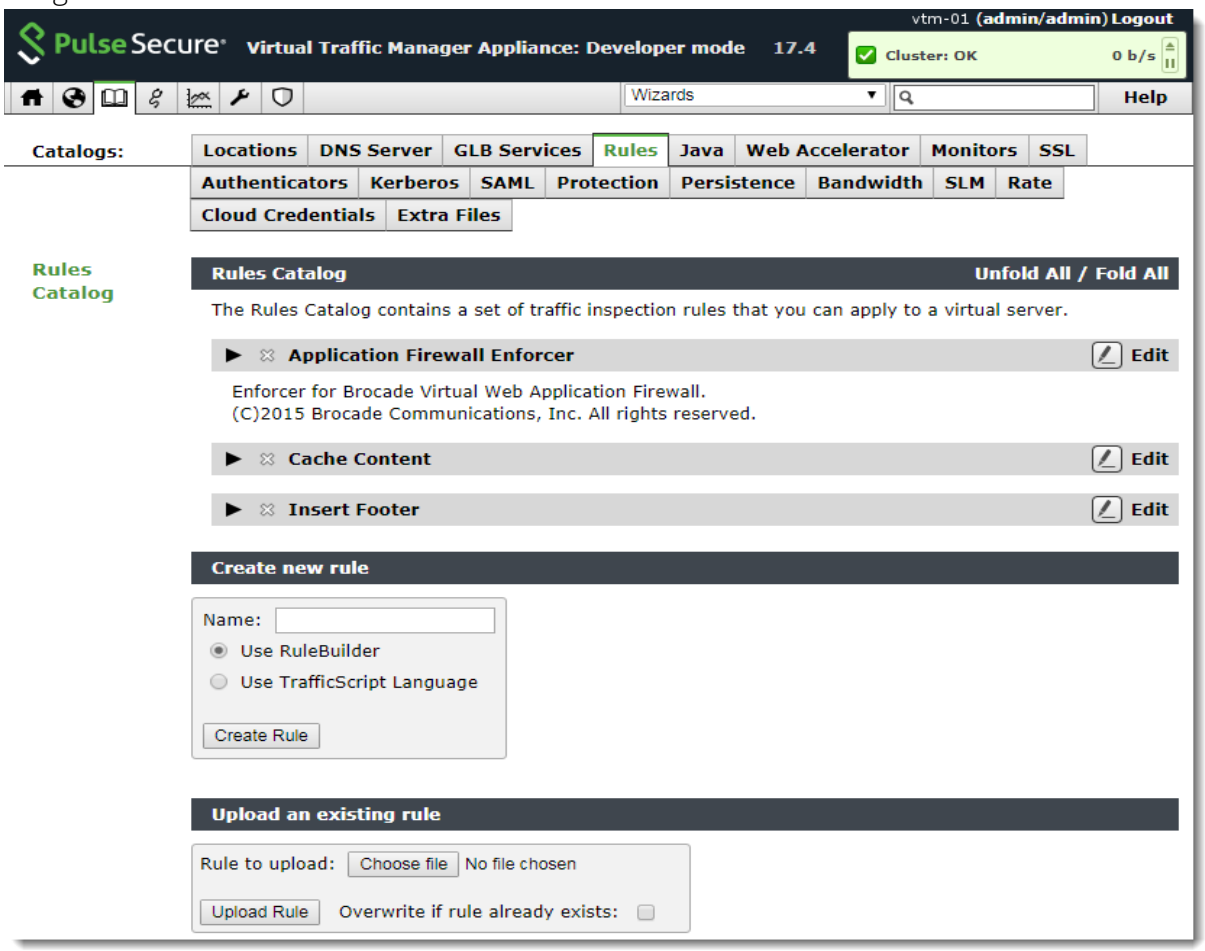
To use TrafficScript rules, first create a new rule or upload a previously created rule using the Rules catalog. Then configure your Virtual Server to reference and use this rule.

To Create a Rule in the Catalog

© 2019 Pulse Secure, LLC.

1. Click **Configure > Catalogs > Rules Catalog**.

FIGURE 2 Creating a new rule



2. In the “Create New Rule” section, type a name for your rule.
3. Click **Use** TrafficScript Language.
4. To create the rule, click Create Rule.
5. On the TrafficScript editing page, type your rule content into the **Rule** text box. For longer rules, you might want to consider writing the rule in a separate text editor before pasting it in. To view the online function reference, click TrafficScript Help. To syntax check your rule, click Check Syntax.
6. Optionally add a free text description of the rule into the **Notes** text box.
7. To save your changes, click Update.

Note: Some TrafficScript functions are protocol specific (such as 'http.getHeader()') and can only be used when handling associated connections. Some functions are only appropriate in a request rule or a response rule. For example, a function that modifies a parameter of a response has no effect if used in a request rule (as the response has not yet been received). For more details, refer to the documentation for each function in this guide, or in the Online Help.

To Upload a Rule to the Catalog

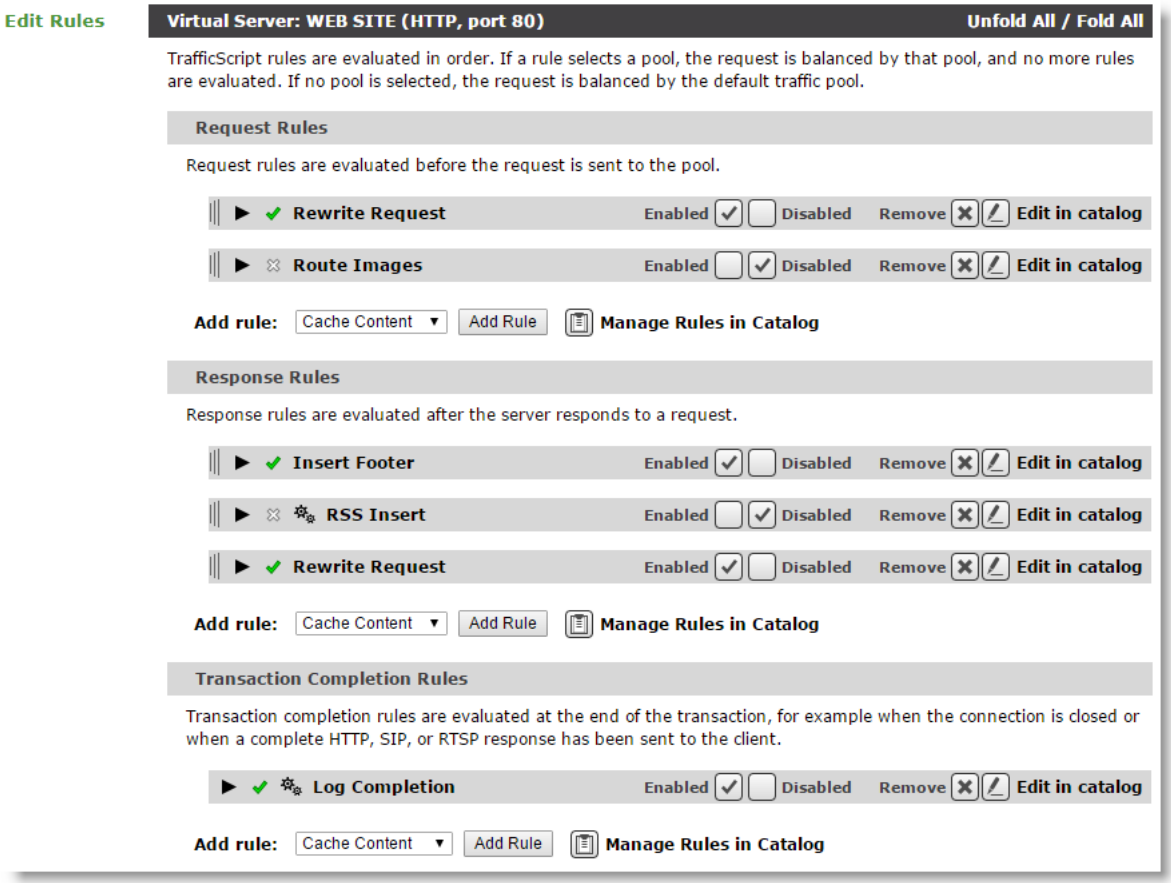
Note: Your rules should be in plain text, and the Traffic Manager uses the full filename (including extension) as the rule name.The Traffic Manager performs a syntax check on the uploaded rule file. Warnings and errors are displayed on the Rule catalog page, the Diagnose page, and in the Event Log.

1. Click **Catalogs > Rules Catalog**.
2. In the "Upload an existing rule" section, click **Choose file** to select a rule file from your local filesystem.
3. Choose whether to overwrite existing rules that have the same name as your filename.
4. Click **Upload Rule**.

To Configure a Virtual Server to Use a Rule

1. Click **Configure > Virtual Servers** and edit your selected Virtual Server.
2. To view and modify the request, response, and transaction completion rules associated with this Virtual Server, click Rules.

FIGURE 3 Applying a rule to a virtual server



You can choose new rules to add to each rule type from the drop-down selection at the bottom of each list.

© 2019 Pulse Secure, LLC.

3. Rules are executed in a specified order. If the first rule does not make a final decision about a request or response, the second rule is tested, and so on. To reorder the list, use the mouse pointer to drag the rule bars up and down as desired.
4. To enable or disable individual rules in the list, use the controls provided. To remove a rule completely from the list, click **Remove**. Note that removed rules are not deleted from the catalog, they are just disassociated with your Virtual Server.
5. For non-http protocols, you can specify whether the rule should be executed just once (against the first request or response), or against every request and response in the protocol dialogue.

Note: This option is not necessary for HTTP virtual servers because HTTP is a single request-response protocol, and requests within a keepalive connection are processed independently.

You can test the effect of a new rule by enabling and disabling it for your test Virtual Server.

© 2019 Pulse Secure, LLC.

© 2019 Pulse Secure, LLC.

TrafficScript Syntax

TrafficScript is the scripting language provided by the Traffic Manager. An administrator can create TrafficScript rules to process requests, implementing suitable logic to ensure that requests are handled in the most appropriate way. TrafficScript is similar to many other programming or scripting languages, such as C or Perl. This chapter describes the syntax of the language.

This chapter contains the following sections:

- Statements 13
- Constants 13
- Variables 14
- Expressions 15
- Conditionals 18
- Loops 21
- Other Flow Control 22
- Complex Data Types 22
- Functions 27
- Escaping Regular Expressions 27
- Creating New Subroutines in TrafficScript 28
- Request and Response Rules 29
- The State Machine in Detail 31

Statements

A command in TrafficScript is called a statement. Each statement ends with a semicolon (;), and a TrafficScript rule typically contains several statements.

```
http.setHeader( "Host", "secure.mysite.com" );
pool.use( "mypool" );
```

Any text between a hash (#) and the end of the line is called a comment and is ignored.

```
# Write a message to the event log file
log.info( "Starting to run rule now!" );
$body = http.getBody( ); # get the POST data
```

Constants

TrafficScript allows you to specify integer, boolean, floating point, and string values:

- Integers: sequences of digits, such as “23”; hex format (0xff) and octal format (\377) are also supported.
- Boolean: TrafficScript provides two boolean keyword constants, `true` and `false`, for use in comparison expressions. These constants are synonymous with 1 and 0 respectively, and can be used where an explicit truth test or return value is required.

© 2019 Pulse Secure, LLC.

- Floating point: decimal point (3.14) and scientific (5.6e-2) notations are both supported.
- Strings: strings are character sequences, enclosed by double (") quotes or single (') quotes.

  In double-quoted strings, special characters can be escaped using the standard backslash "\" notation (for example, "\n" is a newline character), using octal notation ("\012" is also a newline) or using hexadecimal notation ("\x0A" is a newline).

  In single-quoted strings, no escaping is performed. For example, "\n" is not converted to the newline character.

  In both types of string, for improved readability, strings can be broken across lines using a single "\" followed by a newline:

```
# The following two lines both create the string 'Hello world'

$single = 'Hello \
world';

$double = "Hello \
world";
```

  Although, in the case of a single quoted string, not specifying the "\" at the end of the line still causes TrafficScript to insert a newline character.

Be aware of escaping rules when writing regular expressions in TrafficScript. For example, in a double-quoted TrafficScript string, the character "\" automatically escapes the next character, so if you want a literal "\" in your string, double-escape it using "\\". For this reason, regular expressions are often written using single-quoted strings.

Variables

Variables are used in TrafficScript to store values while the rule is executed.

A simple variable can store an integer, boolean, floating point, or string value. More complex data types, such as arrays and hashes, can also be stored in a variable. The value is interpreted as the correct type depending on its context.

Simple data type variables are immutable when passed into a function. In other words, it is a copy of the variable argument that is modified. Complex types, however, are treated differently. They are passed *by reference* into a TrafficScript function or user-defined subroutine (for more details, see "Creating New Subroutines in TrafficScript" on page 28), and as such operations on the stored value have a rule-wide effect.

Variables exist for the duration of the execution of the rule. Values stored in variables are discarded when the rule completes.

Variable names always start with the dollar ($) character, and the value of a variable is set by the assignment operator "=".

```
$path = http.GetPath();
$bytes = connection.getDataLen();
```


© 2019 Pulse Secure, LLC.

```
$pi = 3.14157;
```

Expressions

Expressions in TrafficScript are created from combinations of literal values, variables, evaluated functions, and operators. Expressions are evaluated when the rule is executed.

You can use expressions to:

- Construct complex strings from several different values.
- Create complex tests for “if” conditions or “while” loops.
- Perform mathematical calculations.

You can use an expression anywhere a literal value (or variable) is suitable. You normally see expressions:

- In assignment statements, assigning a value to a variable.
- In conditions – “if/else” statements and “while” loops.
- As function arguments.

For example:

```
$message = "The URL path is " . http.GetPath();
$four = 2 + 2;

# Sets $ratio to "75%" (for example)
$ratio = ( $a / ($a + $b) * 100 ) . "%";

$contentLength = http.getHeader( "Content-Length" );
if( $contentLength > 1024 * 1024 ) {
  log.warn( "Large request body: ".$contentLength );
}
```

Operators

Expressions are constructed from operands (variables, literal values, and so on) and operators. Operators perform calculations or tests on their operands.

Operands in an expression are automatically promoted to the appropriate type: string, integer, or float in accordance with the typecasting rules described in “Type Casts in TrafficScript” on page 18.

The following sections provide details for the operator types you can use.

Mathematical

The operators “+”, “-”, “*” and “/” treat their operands as integers or doubles and add, subtract, multiply, or divide them.

The prefix operator “-” promotes its operand to an integer or double and returns its negation.

```
4 + 7.5 * $a
```

```
-$b / $c - 1
```

The modulus operator "%" takes integer operands, and calculates the remainder after division.

```
7 % 3    # Returns 1
3 % 7    # Returns 3
```

String Concatenation

The operator "." promotes its operands to strings and concatenates them.

```
"The message is " . $bytes . " bytes long."
( $a / ($a + $b ) * 100 ) . " percent"
```

The ".=" operator appends its second operand to the first:

```
$message = "The name is ";
$message .= "Bond, James Bond";
```

Comparison

The operators "==", "!=", ">", "<", ">=" and "<=" compare their operands and return `true` (1) or `false` (0). If both arguments are strings, a string comparison is performed; otherwise the operands are promoted to integers or floats and compared. For further information on typecasting promotion rules, see "Type Casts in TrafficScript" on page 18.

```
1 > 3.14      # false
"99" > 100    # false (performs integer comparison)
"99" > "100"  # true (performs string comparison)
$a == $b      # are the values of $a and $b the same?
```

Boolean

The operators "&&" and "||" perform boolean "and" and "or" tests. The prefix operator "!" performs a Boolean "not" test.

These operators treat their operands as either *true* or *false*, and return `true` (1) or `false` (0) accordingly:

- A non-zero number operand or non-empty string operand is *true*.
- A zero number operand or empty string operand is *false*.

```
"foo" && !0              # true
( 1 < 2 ) && ( 3 < 4 )   # true
$a || $b                 # true if $a or $b is true
```

Increment and Decrement

The following table shows the effect of increment and decrement operations:

Operation	Description
$foo++	Increment $foo after it has been referenced
$foo--	Decrement $foo after it has been referenced
++$foo	Increment $foo before it has been referenced

© 2019 Pulse Secure, LLC.

Operation	Description
--$foo	Decrement $foo before it has been referenced
$foo += 5	Add 5 to the value of $foo ($foo = $foo + 5)
$foo -= 5	Subtract 5 from the value of $foo ($foo = $foo - 5)

Bitwise Operators

The operators "&" (bitwise-AND), "|" (bitwise-OR), and "^" (bitwise-XOR) perform bitwise operations on their integer arguments. Strings and floats are typecast to integers using TrafficScript typecasting rules (see "Type Casts in TrafficScript" on page 18).

```
0x1234 & 255          # 0x34
1 | 2 | 4             # 7
1 ^ 3                 # 2
```

The prefix operator "~" (bitwise-NOT) performs a bitwise NOT on its integer argument.

```
~1 & 0xffff           # 65534
```

The bitwise left-shift and right-shift operators ("<<" and ">>") perform left and right bit-shifts on their integer argument.

```
1 << 2                # 4
2 >> 1                # 1
```

Assignment Operators

In addition to +=, -=, increment, and decrement operators, TrafficScript supports the following mathematical and bitwise assignment operators:

```
$foo *= 5              # Product equals ($foo = $foo * 5)
$foo /= 2              # Quotient equals ($foo = $foo / 5)
$foo %= 2              # Modulo equals ($foo = $foo % 5)
$foo <<= 2             # Bit-shift left equals ($foo = $foo << 2)
$foo >>= 2             # Bit-shift right equals ($foo = $foo >> 2)
$foo &= 2              # Bitwise AND equals ($foo = $foo & 2)
$foo |= 2              # Bitwise OR equals ($foo = $foo | 2)
$foo ^= 2              # Bitwise XOR equals ($foo = $foo ^ 2)
```

Precedence

Complex expressions follow the standard rules of precedence. Parentheses "(" and ")" can be used to group sub-expressions.

© 2019 Pulse Secure, LLC.

Type Casts in TrafficScript

Variables in TrafficScript can contain data of various types: integer, floating point (double), string, boolean, array, and hash. TrafficScript automatically casts (converts) values and variables into the correct type when you evaluate an expression or call a function. See the following table for details:

Value	Cast to *integer*	Cast to *double*	Cast to *string*	Cast to *boolean*	Cast to *array*	Cast to *hash*
14 (integer)	14	14.0	"14"	true	[14]	[ ]
3.25 (double)	3	3.25	"3.25"	true	[3.25]	[ ]
3.75 (double)	4	3.75	"3.75"	true	[3.75]	[ ]
"abcde"(string)	0	0.0	"abcde"	true	["abcde"]	[ ]
"3.25" (string)	3	3.25	"3.25"	true	["3.25"]	[ ]
"3.75" (string)	4	3.75	"3.75"	true	["3.75"]	[ ]
"14str" (string)	14	14.0	"14str"	true	["14str"]	[ ]
"3.2.7" (string)	3	3.2	"3.2.7"	true	["3.2.7"]	[ ]
0 (integer)	0	0.0	"0"	false	[0]	[ ]
"0" (string)	0	0.0	"0"	false	["0"]	[ ]
"" (string)	0	0.0	""	false	[""]	[ ]
[ ] (array)	Err	Err	Err	false	[ ]	[ ]
["0"] (array)	Err	Err	Err	true	["0"]	[ ]
["A" => 1] (hash)	Err	Err	Err	true	[["A" => 1]]	["A" => 1]

Strings and doubles are rounded up or down to the nearest integer value when they are cast to integers.

```
$int = 10;
$double = 2.71828;

string.len( $int );     # casts to string, returns 2
string.len( $double ); # casts to string, returns 7

# Set $string to "10, 2.71828"
$string = $int . ", " . $double;

# Convert $string to a number, and add 4:
$r = $string + 4;  # $r is 14
```

Conditionals

TrafficScript provides two primary conditional statement types, “If” and “Switch”.

© 2019 Pulse Secure, LLC.

The “If” Statement

TrafficScript provides “if” and “if/else” statements for conditional execution. The condition is an expression, which is evaluated.

The return value of the expression determines whether the condition is “true” or “false”:

- A non-zero number or non-empty string is true.
- A zero number or empty string is false.

```
if( <condition> ) {
  <statement list>
}
```

or

```
if( <condition> ) {
  <statement list>
} else {
  <statement list>
}
```

TrafficScript evaluates the given condition. If it is true (the value is non-zero, or is a non-empty string), TrafficScript executes the statement list that follows. In the case of an “If/Else” statement, and the condition evaluates to false, TrafficScript instead executes the second statement list following “else”.

For example:

```
$path = http.getPath();
if( string.startsWith( $path, "/secure" ) ) {
  pool.use( "secure pool" );
} else {
  pool.use( "non-secure pool" );
}
```

The "switch" Statement

TrafficScript provides a "switch" statement for cases where one if-else expression can evaluate to a number of different values, each requiring the Traffic Manager to take a different action. For example, a Traffic Manager might serve several different domains, each with its own back-end. By checking the host value of the HTTP header, you can use the switch statement to select the correct back-end using a number of cases.

The syntax for the switch statement is:

```
switch( <expression>[, <function name>] ) {
   case <expression list 1>:
      <statement list 1>;
   case <expression list 2>:
      <statement list 2>;
   case <expression list 3>:
      <statement list 3>;
   ...
   [default:
       <statement list>;]
}
```

© 2019 Pulse Secure, LLC.

By default, the Traffic Manager executes the first case containing an expression that evaluates to be equal ("==") to the switch `<expression>`.

You can specify an alternative comparison function using the optional `<function name>` argument. This argument accepts the name of a user defined or built in function, declared without any brackets or arguments. It must accept two arguments, with the switch expression being the first argument and the current case expression being the second. If this function returns a value that evaluates to true, the Traffic Manager executes the corresponding case statement list. For example, to perform matches on regular expressions, set `<function name>` to the TrafficScript function "string.regexmatch".

For each case statement, you can specify either a single expression or a comma-separated list of multiple expressions to be used in the comparison. A case is executed if the evaluation function returns true for any of the case's expressions. In this context, the commas in the expression list act as the "or" logical operator.

The optional `default` statement must only be used as the last case and is always executed if reached.

A typical usage example:

```
$host = http.getHostHeader();
 switch( $host ) {
    case "secure.example.com", "ssl.example.com":
       pool.use( "secure pool" );
    case "www.example.com", "www.example.org", "www.example.net":
       pool.use( "non-secure pool" );
    default:
       pool.use( "discard" );
 }
```

An example using a custom evaluation function:

```
$host = http.getHostHeader();
 switch( $host, string.startsWith ) {
    case "secure", "ssl":
       pool.use( "secure pool" );
    case "www":
       pool.use( "non-secure pool );
    default:
       pool.use( "discard" );
 }
```

A switch statement is additionally affected by the following:

- You can use the `break` statement to break out of a switch statement entirely and continue execution after the switch. This has no effect on surrounding loops.
- TrafficScript does not provide an implicit fall-through. In other words, a break at the end of a case body is redundant.
- You can use any variable type within a switch.
- A switch has no return value.
- The Traffic Manager evaluates `<expression>` only once and stores the result for case comparison. If `<expression>` is a function, it is therefore executed only once.

© 2019 Pulse Secure, LLC.

Loops

A body of code can be executed a number of times using a loop. TrafficScript supports “for”, “while”, “do”, and “foreach” loops.

“for” Loops

A “for” loop contains an initialization step, a test and an increment step, bracketing the body of code to be executed on each iteration:

```
for( <initialization> ; <condition> ; <increment> ) {
   <statement list>
}
```

For example:

```
for( $count = 0; $count < 10; $count++ ) {
   log.info( "In loop, count = " . $count );
}
```

This loop will print the message 10 times, with $count running from 0 to 9.

‘while’ Loops

The “while” loop evaluates a condition, and while the condition is true, it executes the enclosed block of code:

```
while( <condition> ) {
   <statement list>
}
```

For example:

```
$count = 0;
while( $count < 10 ) {
   log.info( "In loop, count = " . $count );
   $count = $count + 1;
}
```

This loop will print the message 10 times, with $count running from 0 to 9.

“do” Loops

The “do” loop executes the enclosed block of code, then checks the condition. It repeats the code while the condition evaluates to true:

```
do {
   <statement list>
} while( <condition> );
```

For example:

```
$count = 0;
do {
   log.info( "In loop, count = " . $count );
   $count = $count + 1;
} while( $count < 10 );
```

This loop will print the message 10 times, with $count running from 0 to 9.

© 2019 Pulse Secure, LLC.

TrafficScript contains a number of helper functions to manipulate complex data, such as HTTP requests or long strings. Consequently, it is rarely necessary to use loops in TrafficScript.

"foreach" Loops

Typically used with arrays and hashes, a "foreach" loop executes the bracketed body of code with each element of the supplied array in turn. The variable to the left of the "in" operator is initialized with the next array element prior to executing the code block.

```
foreach ( <element> in <array> ) {
   <statement list>
}
```

For example:

```
$array = [ "Alex", "Matt", "Oliver", "Laurence" ];
$i = 0;

foreach ( $element in $array ) {
   log.info( "Element #" . $i . " " . $element );
   $i++;
}
```

For an example of a foreach loop with a hash, see "Hashes" on page 23.

Other Flow Control

The "break" and "continue" statements can be used to restart or exit loop or rules processing.

Inside a while loop, "break" causes execution of the loop to stop. Execution then continues to the statement after the loop. "continue" causes the loop code to restart.

Inside a rule (outside any containing loops), "break" causes execution of the rule to stop. Execution then proceeds to the next TrafficScript rule. "continue" causes the rule to be restarted from the beginning.

For example:

```
# We're only interested in processing HTTP text/html
# responses...

$mime = http.getResponseHeader( "Content-Type" );
if( !string.startsWith( $mime, "text/html" )) break;

# proceed with processing for text/html...
```

Complex Data Types

Arrays

An array in TrafficScript is a structured variable that stores a list of values. For example, you can define a list of names using the following code:

© 2019 Pulse Secure, LLC.

```
$array = [ "Alex", "Matt", "Oliver", "Laurence" ];
```

The values in this array can then be looked up:

```
$someone = $array[0];
log.info($someone);
```

This instructs the Traffic Manager to print the string "Alex" to the event log. TrafficScript has functions that make it easy to work with array structures. For example, to print all of the names stored in $array using a "for" loop, first determine the number of elements that $array stores. TrafficScript provides the function array.length() for this purpose. For example:

```
$array = [ "Alex", "Matt", "Oliver", "Laurence" ];

$arraylen = array.length($array);
log.info("My array has " . $arraylen . " elements.\n");

for ( $i = 0; $i < $arraylen; $i++ ){
   log.info ( "Element #" . $i . " " . $array[$i]);
}
```

When applied as a rule, this code causes the following output in the event log.

FIGURE 4 Example event log messages

✓	26/May/2010:03:39:10 +0100	INFO	**Rule** *arrays*, **Virtual Server** *Foo*: Element #3 Laurence
✓	26/May/2010:03:39:10 +0100	INFO	**Rule** *arrays*, **Virtual Server** *Foo*: Element #2 Oliver
✓	26/May/2010:03:39:10 +0100	INFO	**Rule** *arrays*, **Virtual Server** *Foo*: Element #1 Matt
✓	26/May/2010:03:39:10 +0100	INFO	**Rule** *arrays*, **Virtual Server** *Foo*: Element #0 Alex
✓	26/May/2010:03:39:10 +0100	INFO	**Rule** *arrays*, **Virtual Server** *Foo*: My array has 4 elements.

Alternatively, you can bypass the requirement to know the array length by using a "foreach" loop. The code below produces output similar to the previous example:

```
$array = [ "Alex", "Matt", "Oliver", "Laurence" ];
$i = 0;

log.info ("My array has " . array.length($array) . " elements.\n");

foreach ( $element in $array ) {
   log.info( "Element #" . $i . " " . $element );
   $i++;
}
```

For further information on array-specific functions, refer to the TrafficScript Reference in the Traffic Manager Online Help.

Hashes

A hash in TrafficScript is similar to an array, but instead of storing a list of values it stores a list of key/value pairs. Hashes are sometimes referred to as associative arrays. You can define a hash using the following code:

```
$hash = [ "orange" => "fruit",
          "apple" => "fruit",
          "cabbage" => "vegetable",
          "pear" => "fruit" ];
```

To print all of the keys and values stored in the hash, first get a list of keys in the form of an array - using the function `hash.keys()` - that you can then use in a foreach loop to print all of the values. For example:

```
foreach ( $key in hash.keys($hash)){
   log.info("Key: " . $key . "; Value: " . $hash[$key] . ";");
}
```

Combining the above two samples of code results in the following output in the event log:

FIGURE 5 Example event log messages

```
26/May/2010:04:47:21 +0100 INFO Rule arrays, Virtual Server Foo: Key: orange; Value: fruit;
26/May/2010:04:47:21 +0100 INFO Rule arrays, Virtual Server Foo: Key: apple; Value: fruit;
26/May/2010:04:47:21 +0100 INFO Rule arrays, Virtual Server Foo: Key: cabbage; Value: vegetable;
26/May/2010:04:47:21 +0100 INFO Rule arrays, Virtual Server Foo: Key: pear; Value: fruit;
```

The Global Associative Array

This is the first of three persistent associative arrays that TrafficScript can access.

Access the global array using the `data.set()` and `data.get()` functions. Data set in this array is persistent, and can be read from a later script. This array is of fixed size (the size is defined by the Traffic Manager global setting **trafficscript!data_size**). When it fills up, you cannot add further entries without first removing existing entries.

Maintaining the Array

As mentioned, use `data.set()` and `data.get()` to add and lookup array items. To delete individual elements, use `data.remove()`.

To determine the amount of memory in use by the global array, use the function `data.getMemoryUsage()`. To delete all entries in the array, use `data.reset()`. You can delete a subset of the entries using `data.reset( "prefix" )`.

If you want to store several different types of data globally, use a consistent prefix to start the name of each key. Then, if you need to free memory, you can identify and delete all of the data that is stored for one particular purpose (for example, a global cache that grows continually, but can safely be deleted and reconstructed if necessary).

Example: An Indexed Array

You can use the global associative array to create an indexed array named “myarray” as follows:

```
# Declare a subroutine to calculate factorials
sub factorial( $n ) {
   if( $n == 0 ) return 1;
   return $n*factorial( $n-1 );
}

# Put entries into the array
$c = 0;
while( $c <= 10 ) {
   $msg = "Did you know that ". $c ."! is ". factorial( $c ) ."?" ;
```


© 2019 Pulse Secure, LLC.

```
   data.set( "myarray".$c, $msg );
   $c++;
}

# Look up several entries.  Note: the 1000th entry is empty
$msg = "";
$msg .= "Index 5:    ".data.get( "myarray5" )."\n";
$msg .= "Index 10:   ".data.get( "myarray10" )."\n";
$msg .= "Index 1000: ".data.get( "myarray1000" )."\n";

# delete the entire array (but no other data stored by data.set)
data.reset( "myarray" );

http.sendResponse( "200 OK", "text/plain", $msg, "" );
```

The Process-Local Associative Array

This is the second of three persistent associative arrays that TrafficScript can access.

When data has to be persistent to the whole system, but not globally unique, the process-local associative array provides the best balance between flexibility and performance. Entries in the global array (using the TrafficScript commands `data.set()` and `data.get()`), are unique across the whole system. This means that on a Symmetric Multiprocessing (SMP) system, when a process inserts a new array entry, it has to prevent all the other processes from accessing that same entry at the same time. The Traffic Manager achieves this with locks that hold the other processes back until the first process has finished. As such, if entry insertions and updates are happening very frequently, the system can spend a lot of time waiting to access the global array.

To alleviate this, the Traffic Manager provides the process-local array; accessed using `data.local.*`. Entries in the process-local array can be read from any TrafficScript context that takes place in the same process as the one that added them, across requests and connections. Essentially, the process-local array trades memory for performance: access and manipulation is instant, no locks are involved, but data is present once per process instead of just once across the system. In most cases this trade-off is acceptable, and in some cases, especially where the application logic requires the use of the `data.reset()` function, the performance benefits of using the process-local array are substantial.

The Connection-Local Array

This is the third of three persistent associative arrays that TrafficScript can access.

When processing a connection, it is sometimes useful to store information calculated in one rule for retrieval in a later rule. You can do this using a connection-local associative array, using the `connection.data.set()` and `connection.data.get()` functions.

Information stored in this way can only be retrieved by a TrafficScript rule that is processing the same connection, and all information is destroyed (and the memory freed) when the connection completes.

Libraries

TrafficScript rules that contain subroutines can be used as libraries. Take the following rule for example:

```
sub headbug(){
   # Prints each header to the event log.

   $headers = http.listHeaderNames();

   foreach ($header in $headers){
      log.info( $header . ": " . http.getheader($header));
   }
}
```

The `http.listHeaderNames()` function returns an array of header names that were sent in the HTTP request.

To make this routine available as a library, save it as a separate rule and use the `import` statement to use it in your other rules. For example, by saving the above routine as a rule named "foo", you can then import and use the `headbug()` subroutine using the following code:

```
import foo;

foo.headbug();
```

You can alternatively employ a locally declared alias for the imported library, such as:

```
import foo as mylib;

mylib.headbug();
```

This provides a means to reference an imported library with a more suitable, or perhaps shorter, local name. Functionally, there is no difference between this mechanism and the directly named example.

Applying this rule to a Virtual Server causes the names and values for each header of each request that the Virtual Server processes to be logged to the event log. You can modify the first rule (or library) so that it creates a hash of each of the headers using the following example:

```
sub headbug(){
   # Prints each header to the event log.

   $headers = http.listHeaderNames();

   foreach ($header in $headers){
      $headhash = [ $header => http.getheader($header) ];
      log.info( $header . ": " . $headhash[ $header ] );
   }
}
```

Although this library performs the same function, it does so slightly differently in that it creates a data structure that can be used later. To be able to use this structure later, however, you must extract it from the subroutine. To achieve this, alter the `headbug()` subroutine to return the $headhash data structure, then modify the calling code:

```
# foo (library rule)

sub headbug(){
   # Prints each header to the event log.

   $headers = http.listHeaderNames();
```


© 2019 Pulse Secure, LLC.

```
   $headhash = [];

   foreach ($header in $headers){
      $headhash[ $header ] = http.getheader($header);
      #log.info( $header . ": " . $headhash[ $header ] );
   }

   return($headhash);
}

# bar (calling rule)

import foo;

$headhash = foo.headbug();

foreach ($header in hash.keys($headhash)){
   log.info( $header . ": " . $headhash[ $header ]);
}
```

Functions

A function performs an action, and returns a value. Functions are used to provide useful capabilities to TrafficScript.

TrafficScript contains a large number of functions to manipulate data or manage the current request. For ease of use, TrafficScript functions are grouped into families. For example, functions that operate on HTTP requests all begin with "`http.`", such as `http.getHeader()` or `http.setBody()`.

To call a function, use its name followed by a pair of parentheses "()". Many functions take one or more values as parameters, and these are listed inside the parentheses.

```
connection.discard();
http.setHeader( "Cookie: type=chocolate" );
$hello = string.append( "Hello", " ", "world", "!" );
```

Function names are not case sensitive. So, the function `lang.todouble()` can be invoked using `lang.toDouble()`, `Lang.ToDouble()` or `lang.todouble()`.

For a description of all TrafficScript functions applicable to this product version, see "Function Reference" on page 47

Escaping Regular Expressions

Several TrafficScript functions take regular expressions as arguments. TrafficScript uses the PCRE regular expression library.

Regular expressions might contain a number of special characters:

- . matches any single character.

- ? indicates that the previous expression is optional.
- * matches any number of the previous expression.
- ^ matches the beginning of a string.
- $ matches the end of a string.

If you want to match a literal period (.), or other special character in your regular expression, escape it with a backslash (\) character.

Like many other scripting languages, double-quoted TrafficScript strings use "\" as an escape character, so to place a regular expression like "^192\.168\." into a TrafficScript double-quoted string, you must double-escape the "\" character:

```
# Sets the regex string as ^192\.168\.
# The two examples below have the same effect
$regex = "^192\\.168\\.";
$regex = '^192\.168\.';
if ( string.regexMatch( $ip, $regex ) ) {
    # IP is on 192.168.* network
}
```

Note that it's not often necessary to use regular expressions in TrafficScript for the following reasons:

- To search strings, you can use `string.IPMaskMatch()`, `string.Contains()`, `string.startsWith()` and `string.endsWith()`.
- To search and replace within strings, you can use `string.replace()` and `string.replaceAll()`.

Creating New Subroutines in TrafficScript

You can create new TrafficScript subroutines to improve the efficiency of your software. Once created, these routines can be used in rules or procedures just like the predefined functions.

Syntax

The syntax to create a new subroutine is:

```
sub function_name ($var1, $var2)
{
      <code>
      return "return value"
}
```

Subroutines can be called just as you would call any normal function:

```
$ret = function_name( $foo, $var );
```

This example would return the value given by the subroutine into the variable $ret.

Subroutine Position and Name Restrictions

User subroutine names cannot be the same as built-in TrafficScript functions.

Subroutines can be declared above or below where they are used. For example, the following code is correct:

© 2019 Pulse Secure, LLC.

```
test();
sub test()
{
     log.info("Attention! Test running");
}
```

Local Variables

Variables within subroutines are local to that section and do not affect the result when used outside of the subroutine.

For example, the following lines of code print out an empty string, as $var is not available (passed in as an argument) within the subroutine:

```
$var = "abc";
sub function()
{
      log.info($var);
}
function();
```

$1 to $9 Variables

TrafficScript uses $1 to $9 as global variables, so they can be used to return extra data from subroutines in addition to the (optional) return statement that returns a single value.

For example, the following code prints "Hello World":

```
sub greetings()
{
     $1="Hello ";
     return "World"
}
$ret = greetings();log.info($1 . $ret)
```

Request and Response Rules

TrafficScript rules are assigned to a Virtual Server. The Traffic Manager allows a rule to be used as a “request rule”, a “response rule”, or a “transaction completion rule”:

- Request rules are executed when the first request data is received from a client. A request rule can read further data if the client's request is not complete. When the request rules assigned to a virtual Server have finished executing, a connection to a back-end server is made and the request data is streamed to the server.
- Response rules are executed when the first data in the response is read from the back-end server. A response rule can then read further data from the server, and can flush the current data to the client. When the response rules assigned to a Virtual Server have finished executing, the remaining data is streamed to the client.

© 2019 Pulse Secure, LLC.

- Transaction completion rules are applied at the end of the transaction when, for example, the client or server closes the connection, or when the connection times out. In the case of HTTP, RTSP and SIP Virtual Servers, transactions are also considered complete when a full response has been sent to the client, even if the connection is kept alive to handle another request.

Processing Multiple Requests and Responses

If the client sends another request down the same network connection, the Traffic Manager can be configured to run the request and response rules again against this new request. To configure this behavior, use the "run once/run every time" setting for each rule in the Virtual Server configuration. Use `request.endsAt()` and `request.endsWith()` to parse persistent connections that contain multiple requests and responses.

Note: The run once/run every time option is not relevant for HTTP connections. Even though a client might send several HTTP requests down the same persistent connection, they are automatically separated into single connections internally.

Note: If you are parsing persistent connections on which the client and server can send data at any point in time in any order, use the "Generic Streaming" Virtual Server type along with `request.get()` and `response.get()`.

Specialized Protocol Handling Functions

Processing HTTP

The Traffic Manager is fully conversant in the HTTP protocol. On a new HTTP connection, request rules are not started until all the HTTP headers have been received. Specialized TrafficScript functions (such as `http.getPath()` and `http.setHeader()`) are available to parse and manipulate the request.

When an HTTP response is received from a back-end server, it does not execute the response rules until it has received all the HTTP headers in the response. Again, specialized TrafficScript functions (such as `http.setResponseHeader()` and `http.setResponseCookie()`) are available to manipulate the response.

Note: You cannot use the lower-level connection, request, and response functions such as `request.getLine()` or `response.set()` to modify an HTTP request. Use the equivalent specialized HTTP functions instead.

Although multiple HTTP requests can be submitted down a single TCP connection (using keepalives and pipelining), the Traffic Manager transparently separates these requests and handles each individually. For this reason, the option to run rules once or every time is not relevant to HTTP requests.

HTTPS traffic decrypted by the Traffic Manager is handled in exactly the same way. The HTTP payload requests can be inspected and manipulated just as if they were sent in plain text.

© 2019 Pulse Secure, LLC.

Other Specialized Protocols

TrafficScript offers high-level functions for several other protocols, including SIP and RTSP. Always use the high-level functions to read and write request data where possible, rather than lower level functions such as `request.get()` or `response.set()`.

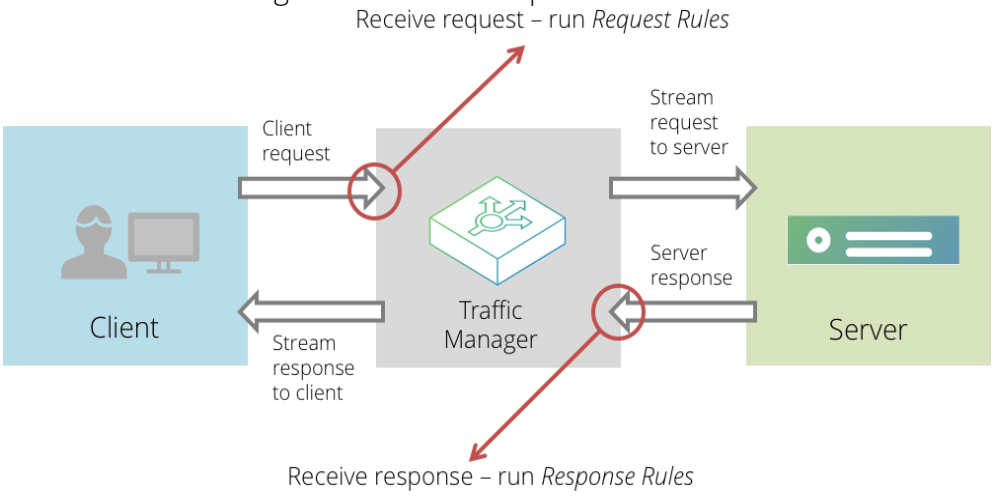
Processing Other Protocols

The Traffic Manager can handle other protocols (TCP and UDP based) using `request.get()`, `request.set()`, `response.get()`, `response.set()`, and related functions to read requests.

Note: UDP is a connectionless protocol. To set a UDP response, use the TrafficScript function `connection.close( $data, 0 );`.

The State Machine in Detail

FIGURE 6 Processing rules in TrafficScript



Controlling the State Machine

The following TrafficScript functions can be used to control the state machine when processing requests and responses:

Function	Notes
`pool.use()`	When used in a request rule, aborts all rules processing and specify the pool to give the request to.
`request.sendResponse()` `http.sendResponse()`	When used in a request rule, specifies the response to send to the client. The current request is discarded and no data is sent to any back-end servers. When the request rules finish, the response rules are run on the provided response.

Function	Notes
`response.close()`	In a response rule, close the connection to the server. When the response rules complete, pending response data is sent to the client and the Traffic Manager then waits for a new request from the client.
`request.endsAt()` `request.endsWith()`	In a request rule, use these functions to extract individual requests from the incoming data stream and process them synchronously in a request-response manner. For some protocols, a client might send several requests in one go. These functions can be used to process the requests one-by-one.
`request.retry()`	In a response rule, retry the request if possible. All request data that was read before and during a request rule is cached. A request can be retried if all the request data was read and cached; it can't be retried if data was subsequently streamed between the client and the server before a response was received.
`response.flush()`	In a response rule, flush the current response data. Use `response.get()` or `response.getLine()` to read further response data. This function can be used to manually stream data from the server to the client, ensuring that the response rule does not complete until the response has completed.
`connection.close()` `connection.discard()`	These functions can be used to abort a connection, optionally providing a response. The abort is immediate - no further rules are run at either the request or response stage.

© 2019 Pulse Secure, LLC.

Sample TrafficScript Rules

This chapter contains the following sections:

- Routing by Content Type 33
- Restricting Access Based on the Time of Day 33
- Customer Prioritization 34
- Routing Based on XML Traffic 35
- Authenticating User Access 36
- Synchronizing Requests and Responses 38
- Streaming HTTP Responses 39
- Managing FTP Connections 40

Routing by Content Type

This example inspects the URL in an HTTP request. It decides which pool to send the request to, based on the value of the URL.

Suppose your Web site uses a number of different technologies, including Microsoft ASP pages and Oracle JSP as well as static HTML content. Microsoft Windows based devices serve the ASP pages, Oracle servers handle JSP, and a set of commodity Linux servers serve the static content. You want to direct your traffic to different servers depending on the specific content.

Set up a pool for each of these groups called “windows”, “oracle” and “linux” respectively. The following rule directs network traffic according to the type of content.

```
$path = http.getPath();
if( string.endsWith( $path, ".jsp" )) {
     pool.use( "oracle" );
}    else if( string.endsWith( $path, ".asp" )) {
         pool.use( "windows" );
} else {
     pool.use( "linux" );
}
```

Restricting Access Based on the Time of Day

This example only allows access to a particular service during designated office hours (between 9am and 6pm, Monday to Friday). It discards all connections that occur outside these times.

```
$dayofweek = sys.time.weekDay();
$hourofday = sys.time.hour();

# $dayofweek: Sunday is 1, Saturday is 7
# $hourofday: office hours are between 9am and 5:59pm
if( $dayofweek == 1 ||
```

```
    $dayofweek == 7 ||
    $hourofday < 9  ||
    $hourofday >= 18 ) {
  log.warn( "Warning: access out of hours!" );
  connection.discard();
}
```

In practice, it may be more appropriate to direct restricted traffic to a separate *error pool* of servers rather than just dropping the connection without warning. The servers in the error pool can be configured to return an appropriate error message before closing the connection. The procedure for doing this depends on the protocol being balanced.

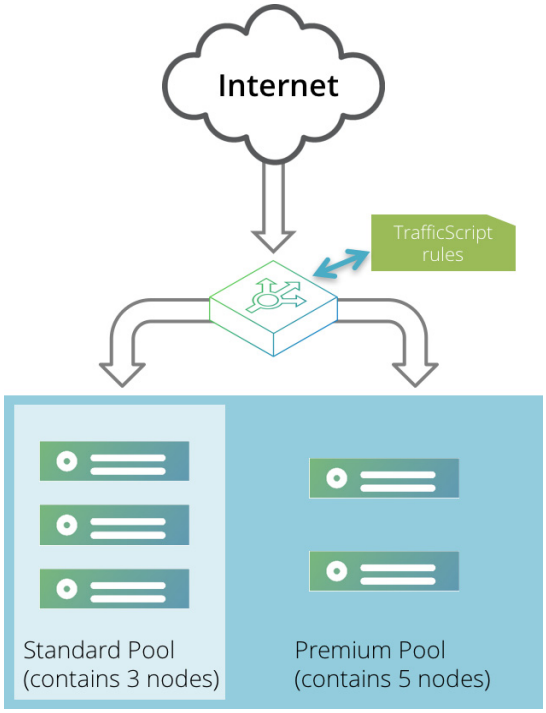
Customer Prioritization

This example inspects the cookie in an HTTP request. It uses the value of the cookie to determine which pool to direct the request to. One pool is faster than the other because it contains machines that are reserved for premium users.

A company has a customer base divided into gold and silver membership. It wishes to give priority to the gold customers and has five servers: yellow, green, blue, black, and purple.

Two pools are created: standard, for silver customers, containing machines yellow, green and blue; and premium, for gold customers, which includes all five of the servers. Thus black and purple are only available to gold customers.

FIGURE 7 Customer prioritization scheme inserting cookies using TrafficScript



The site uses a cookie login system, with the customer type encoded in the cookie. Different membership levels can be detected, and sent to the correct pool:

```
$cookie = http.getHeader( "cookie" );
```


© 2019 Pulse Secure, LLC.

```
if( string.contains( $cookie, "gold" )) {
   pool.use( "premium" );
} else {
   pool.use( "standard" );
}
```

Routing Based on XML Traffic

TrafficScript includes support for parsing XML documents using XPath, an industry-standard language used to query XML documents.

XML documents are used by SOAP-based protocols such as Web Services, and enable complex data to be exchanged and understood automatically without user intervention.

An XML document is organized into a tree structure of *nodes*[1]. Each node might contain a piece of data, or other nodes. XPath can navigate through these nodes to extract specific data from the XML document. This data can then be used to make routing decisions on the traffic.

The XPath 1.0 specification is available at http://www.w3.org/TR/xpath.

Example: Google Search Request

The Google™ search engine has a Web Services interface that accepts SOAP requests for search queries. A request for a search for Pulse Secure consists of an HTTP POST containing the following XML body data:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
      <namesp1:doGoogleSearch xmlns:namesp1="urn:GoogleSearch">
         <key xsi:type="xsd:string">googleUniqueID</key>
         <q xsi:type="xsd:string">Pulse Secure</q>
         <start xsi:type="xsd:int">0</start>
         <maxResults xsi:type="xsd:int">10</maxResults>
         <filter xsi:type="xsd:boolean">false</filter>
         <restrict xsi:type="xsd:string"/>
         <safeSearch xsi:type="xsd:boolean">false</safeSearch>
         <lr xsi:type="xsd:string"/>
         <ie xsi:type="xsd:string">latin1</ie>
         <oe xsi:type="xsd:string">latin1</oe>
      </namesp1:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that the SOAP body contains a “doGoogleSearch” node. This contains the parameters of the search request.

1. Note that these are unrelated to Traffic Manager back-end nodes.

An Internet service might implement or proxy doGoogleSearch requests and the Traffic Manager might be used to manage the traffic to this service.

For example, it might be necessary to split doGoogleSearch requests according to the specified maximum number of results. If “maxResults” is greater than 100, the request is to be sent to pool “googleLarge”, otherwise it should be sent to pool “google”.

A TrafficScript rule can use the functions `xml.XPath.MatchNodeSet()` and `xml.XPath.MatchNodeCount()` to query the SOAP request body and test XML nodes:

```
# Read the entire body of the SOAP/HTTP request
$body = http.getBody( 0 );

# XML parameters lie in the "urn:GoogleSearch" XML
# namespace:
$googlens = "xmlns:googlens=\"urn:GoogleSearch\"";

# Test for the presence of a "doGoogleSearch" node.
# If present, get the value of the "maxResults"
# parameter and choose the pool

if( xml.XPath.MatchNodeCount( $body, $googlens,
    "//googlens:doGoogleSearch" ) ) {

  $maxResults = xml.XPath.MatchNodeSet( $body, $googlens,
    "//googlens:doGoogleSearch/maxResults/text()" );

  if( $maxResults >= 100 ) {
    pool.use( "googleLarge" );
  } else {
    pool.use( "google" );
  }
}
```

Authenticating User Access

The following rule uses HTTP Basic authentication to ask the remote client for a user name and password. It checks the client’s user name, password, and IP address using a local Web server running an authentication application.

The rule looks for a header called “Authorization”. This should contain a string containing the word `Basic`, followed by a base-64 encoded string. When decoded, this string is of the form `username:password`.

If the string is malformed or the user name or password is absent, the rule returns a “401 Authorization Required” message to the client. This causes the user’s Web browser to prompt them for a user name and password.

If a user name and password have been supplied, the rule uses these details, together with the DNS name of the client, to query the local Web server using `http.request.get()`. The rule expects a “200 OK” response from the Web server to indicate success. Any response code other than 200 results in the rule sending the user the response “403 Forbidden”.

© 2019 Pulse Secure, LLC.

Note that if the user is successfully authenticated, the rule performs no positive action with the request. It is passed on to any other rules the Virtual Server is using, or its default pool.

```
# Determine the username and password, and send a
# '401 Authorization Required' header if there isn't
# one.

$authheader = http.getHeader( "Authorization" );

# Decode the Authorization header; it starts with
# 'Basic', followed by a base64 encoded
# username:password string

if( string.startsWith( $authheader, "Basic " ) ) {
  $encuserpasswd = string.skip( $authheader, 6 );
  $userpasswd = string.base64decode($encuserpasswd);

  $i = string.find( $userpasswd, ":" );
  $user = string.substring( $userpasswd, 0, $i-1 );
  $password = string.skip( $userpasswd, $i+1 );
}

# If the client did not provide an Authorization
# header, indicate that authorization is required
if( $user == "" ) {
   http.sendResponse( "401 Authorization required",
      "text/html", "Please login\n",
      "WWW-Authenticate: Basic realm=\"secure
        server\"" );
}

# Check the supplied username and password by
# querying a local access server with the username,
# password and remote host.

$rhost = net.dns.resolveIP(connection.getRemoteIP());

$querystring = "user=" . string.escape( $user ) .
   "&passwd=" . string.escape( $password ) .
   "&rhost=" . string.escape( $rhost );

http.request.get(
   "http://server/auth.cgi?" . $querystring );

if( $1 != 200 ) {
   # access was denied
   http.sendResponse( "403 Forbidden",
      "text/html", "Access denied\n", "" );
}

# The user is allowed access
```

This rule could be optimized to cache responses from the local Web server, using `data.set()` and `data.get()`.

Synchronizing Requests and Responses

This example assumes that we are managing a simple line-based protocol. Each request is one line long, delimited by a new-line character (\n). Each response might be very long, but is finished by a single line containing a dot (".\n").

This protocol is similar to many simple line-based protocols like POP, IMAP or SMTP, although they each have more sophisticated ways if indicating when a response has finished. The protocol is persistent - there might be many requests and responses within a single connection.

You can use a simple client-first or server-first Virtual Server to manage this protocol.

You might want to intercept a particular request and return a response directly from the Traffic Manager without forwarding the request to the back-end server, but it might be desirable to keep the connection open and forward other requests to the server. This example permits all requests other than "`HACKME\n`", for which "`403 Go Away!\n.\n`" is returned

A first attempt might be as follows:

```
# get the request
$req = request.getLine();

if( $req == "HACKME\n" ) {
   request.sendResponse( "403 Go away!\n.\n" );
}
```

This simple filter has great potential for being subverted. A determined hacker could:

- Try sending the requests byte-by-byte to fool any parsing code (this would not work in this case).
- Try sending two requests in one packet - the first is valid, the second is “`HACKME`”. The above filtering code lets the entire packet through.
- Send one request and a partial "`HACK`" in one packet, then send "`ME\n`" in a second packet when the attacker receives the response from the first request. Our sample filter does not recognize the fragmented "`HACKME`" request.

To avoid these potential errors, it is necessary to synchronize individual requests, processing each request in turn and buffering up any additional data for the next request:

```
# get the request
$req = request.endsWith( "\n" );

if( $req == "HACKME\n" ) {
   request.sendResponse( "403 Go away!\n.\n" );
}
```

Using `request.endsWith()` causes data to be read up to, and including, the “\n”. This data alone is then processed, going through the request and response stage of the state machine. When a response has been received from the server, the next request starts processing, which might contain data that was postponed from the previous request.

This simply and effectively guards against attempts to subvert the parser by sending two requests in one, or by sending partial requests.

© 2019 Pulse Secure, LLC.

This solution has an unfortunate side effect. If a user sends two requests in quick succession, and the first request causes the server to send a large response, with the second request being the "`HACKME`" message that causes the Traffic Manager to send a response itself.

If the "`HACKME`" request is received while the server is still responding to the previous request, the Traffic Manager's direct response might be interleaved with the server's response, resulting in a corrupt data stream back to the client.

The solution is to synchronize responses in the data stream, so that processing of the next request is delayed until the previous request has completed. This can be achieved with a simple response rule:

```
$res = response.getLine();
while( $res != ".\n" ) {
   response.flush( string.len( $res ) );
   $res = response.getLine();
}
```

The response rule does not complete until a line containing ".\n" is received. When it completes, all remaining response data is flushed to the client and the next request is processed.

Streaming HTTP Responses

The low-level function `response.flush()` should not be used to manage HTTP responses. Pulse Secure recommends using the `http.stream.*` functions instead.

If you wish to generate or modify an HTTP response, the simplest way to do so is to read the response in its entirety (using `http.getResponseBody()`), manipulate it, and then write it using `http.setResponseBody()`. However, this can be inefficient when managing large HTTP responses for the following reasons:

- Data is not written to the client until the response rule completes. If it takes time to read the entire response and manipulate it, this can potentially cause the client to timeout and close the connection.
- The entire response needs to be managed in memory, which can be very memory-inefficient. The memory is not discarded until the rule has completed and the data is written to the client.

As an alternative, you can stream HTTP responses, reading and writing them in smaller quantities. Response data is written as soon as it is available, ensuring lower memory use and better performance.

The following functions manage HTTP streaming:

Function	Notes
`http.stream.startResponse()`	This function should be called before any data is written to the client. The headers for the response as assembled and written to the client. No response header manipulation can be performed after this point.
`http.stream.readResponse()`	Reads and returns a portion of the HTTP response body, limited by length or a delimiter character.

Function	Notes
`http.stream.writeResponse()`	Writes the supplied body data to the client, but holds the connection open for additional body data if required.
`http.stream.finishResponse()`	Indicates that response streaming has finished. Rules processing is halted and any remaining response data is sent to the client.

The following TrafficScript rule processes HTML responses line by line, making a simple substitution:

```
$status = http.getResponseCode();
if( $status != 200 ) break;

$type = http.getResponseHeader("Content-Type");
if( !string.startsWith( $type, "text/") ) break;

http.stream.startResponse( $rcode, $type );

while( $line = http.stream.readResponse( 2048, "\n" ) ) {
   $line = string.replaceAll( $line, "TrafficScript", "TRAFFICSCRIPT" );
   http.stream.writeResponse( $line );
}

http.stream.finishResponse();
```

Note: The `http.stream.*` series of TrafficScript functions cannot be used in conjunction with the Traffic Manager's Web content optimization technology - Aptimizer. Executing any of these functions bypasses Aptimizer completely and return the response directly to the client.

Managing FTP Connections

This sample TrafficScript rule manages incoming FTP connections. It implements a proxy for the login stage, waiting until the remote user has provided a suitable user name and password.

When using this rule, configure it as a "run every time" request rule:

```
$req = string.trim( request.endswith( "\n" ) );

if( string.regexmatch( $req, "USER (.*)" ) ) {
   connection.data.set( "user", $1 );
   $msg = "331 Password required for ".$1."!!\r\n";
   request.sendresponse( $msg );
   break;
}

if( !string.regexmatch( $req, "PASS (.*)" ) ) {
   # Are we connected?
   if( connection.getNode() ) { break; }
   request.sendresponse( "530 Please log in!!\r\n" );
   break;
```


© 2019 Pulse Secure, LLC.

```
}

$user = connection.data.get( "user" );
$pass = $1;

# In this case, we'll permit any password that is
# the uppercase version of the username
# Do your own authentication here; for example,
# call a remote server with http.request.get

if( string.uppercase( $user ) != $pass ) {
   request.sendresponse(
      "530 Incorrect user or password!!\r\n" );
   break;
}

# now, replay the correct request against a new
# server instance, disconnecting from any FTP server
# we are already connected to
response.close();

connection.data.set( "state", "connecting" );
request.set(
   "USER anonymous\r\nPASS ".$user."\r\n" );

# select the pool we want...
if( $user == "ftp@example.com" ) {
   pool.select( "Traffic Manager FTP pool" );
}
if( $user == "ftp@customer.com" ) {
   pool.select( "Customer FTP pool" );
}
# the default pool is 'discard', so other users
# are dropped
```

To use this rule, configure it as a “run every time” response rule:

```
if( connection.data.get("state") == "connecting" ) {
   # We've just connected.  Slurp the first line
   # (the serverfirst banner), the second line (the
   # 331 need password) and then replace the
   # serverfirst banner

   $first = response.getLine();
   $second = response.getLine( "\n", $1 );
   $remainder = string.skip( response.get(), $1 );
   response.set( $first.$remainder );
   connection.data.set( "state", "" );
}
```

Remember to configure a server-first banner for the FTP Virtual Server.

© 2019 Pulse Secure, LLC.

Troubleshooting

Writing and debugging complex TrafficScript rules is an involved process. This chapter lists some useful techniques.

This chapter contains the following sections:

- Checking Syntax. 43
- Debugging Rules 43
- Request and Response Rules. 44
- A Special Note About pool.use and pool.select 45

Checking Syntax

A syntax error occurs if you make an error in your TrafficScript rule that prevents the Traffic Manager from fully understanding the rule.

When editing a rule in the Admin UI, you can check the syntax at any time using the Check Syntax button on the **Catalogs > Rules > Edit** page.

You can also check the syntax of a TrafficScript rule from the command line using the zeus.zxtm binary as follows:

```
$ export ZEUSHOME=/usr/local/zeus
$ $ZEUSHOME/zxtm/bin/zeus.zxtm --rulesyntaxcheck rulefile
Compilation failed, with 1 error
Error: line 20: Wrong number of arguments to function pool.use

       pool.use( );
                 ^
```

TrafficScript is not case-sensitive. Function names and variables can be referred to using any combination of case.

Debugging Rules

The TrafficScript function `log.info()` can be used to log a message to the error log (`ZEUSHOME/log/errors`).

Consider the following code:

```
if( string.contains( http.getPath(), "root.exe" ) ) {
  log.info( "Discarding potential nimda attack" );
  connection.discard();
}
```

This appends the following message to your error log file:

```
$ export ZEUSHOME=/usr/local/zeus
```

```
$ tail $ZEUSHOME/log/errors
[20/Dec/2014:10:24:46 +0000] INFO   rules/RuleName  rulelogmsginfo  vservers/VSname
Discarding potential nimda attack
```

You can also inspect your error log file by clicking **Event Log** in the Traffic Manager Admin UI.

When you are debugging a rule, use `log.info()` to print out progress messages as the rule executes. `log.info()` requires a string parameter; you can construct complex strings by appending variables and literals together using the "`.`" operator:

```
$msg = "Received ".connection.getDataLen()." bytes.";
log.info( $msg );
```

The functions `log.warn()` and `log.error()` are similar to `log.info()`. They prefix error log messages with a higher priority - either "WARN" or "ERROR". Use the Event Log viewer to filter out low-priority messages.

Be careful when printing out connection data verbatim as the connection data might contain control characters or other non-printable characters. You can encode data using either "`string.hexEncode()`" or "`string.escape()`".

You should use "`string.hexEncode()`" if the data is binary, and "`string.escape()`" if the data contains readable text with a small amount of non-printable characters.

Request and Response Rules

In some circumstances, a poorly written request or response rule can stall a connection. If you call a function such as `request.getLine()`, the connection can block until it receives another line of input. If you've mis-parsed the connection and the client or server are not going to send any more input, the connection can stall indefinitely (until it is timed out).

Take great care to correctly parse a protocol if you write a detailed handler for it. Your implementation of the handler determines precisely how the protocol is handled. When developing such a handler, several techniques are useful:

- Use `log.info()` to emit verbose debugging information, for example, when each rule starts and when it ends.
- Use a system call tracer such as `strace` or `truss` to monitor exactly what data is being read and written to the network.

The script `ZEUSHOME/zxtm/bin/trace` can assist with this. Type `trace --help` for more information.

- Use `tcpdump` or `wireshark` to monitor what the client and server is sending, but be aware that if a rule is blocked reading from the client, it does not notice that the server has sent any data.

Use the **Activity > Connections** page in the Admin UI to monitor the connections and individual requests being handled by the Traffic Manager. This data can be further enhanced with the Traffic Manager's Request Tracing capability. With this feature enabled, use the **Connections** page to provide detailed information about the time-line for each connection. This can be particularly useful by showing if a connection is being stalled when running a TrafficScript rule. Refer to the *Pulse Secure Virtual Traffic Manager: User's Guide* for more information on this feature.

© 2019 Pulse Secure, LLC.

A Special Note About pool.use and pool.select

The `pool.use()` and `pool.select()` TrafficScript functions are used to decide which Pool should be used to send the connection to a back-end server. They each take one argument, the name of the Pool to use.

By default, `pool.use()` and `pool.select()` only accept a string literal as the Pool name:

```
pool.use( "my webserver pool" );
```

This allows the Traffic Manager to inspect a rule and determine precisely which pools are referenced by any Virtual Servers that use the rule. This information is used as follows:

- To *type check* the usage of a Pool, to ensure that only configuration settings relevant to the protocol the Pool is managing are displayed, and to ensure that the same Pool is not used by Virtual Servers that manage different traffic protocols.
- To determine which Pools are in use, so it can monitor the behavior of the nodes.
- To display the Pools referenced by a Virtual Server, in the configuration summary, front page and other parts of the UI.
- To determine when a Pool is no longer used, so that error information can be removed.

However, in some limited circumstances, it is efficient to use a variable to specify the desired Pool.

```
if( $poolname == "pool1" ) {
   pool.use( "pool1" );
} else if( $poolname == "pool2" ) {
   pool.use( "pool2" );
} else if( $poolname == "pool3" ) {
   pool.use( "pool3" );
} else if( $poolname == "pool4" ) {
   pool.use( "pool4" );
} # etc.
```

It would be much more preferable to write the following:

```
pool.use( $poolname );
```

To enable this behavior, enable the trafficscript!variable_pool_use setting on the **System > Global Settings > Other Settings** page in the Admin UI.

Be aware that if you enable the above behavior, the Traffic Manager is not able to determine accurately which Pools are referenced by a rule, and this limits the internal consistency checks and error monitoring that it performs for those Pools.

© 2019 Pulse Secure, LLC.

© 2019 Pulse Secure, LLC.

Function Reference

This chapter contains the following sections:

- TrafficScript Core Functions 47
- Traffic Manager Functions 90

TrafficScript Core Functions

TrafficScript core functions provide the basic function set for the TrafficScript language, including support for the core TrafficScript types, mathematical functions, and date and time manipulation.

The core functions are grouped into several families:

- array.: These functions facilitate the use and manipulation of arrays within TrafficScript.
- hash.: These functions facilitate the use and manipulation of hashes (or key/value pairs) within TrafficScript.
- json.: These functions provide the ability to convert between TrafficScript arrays/hash variables and JavaScript Object Notation (JSON) strings.
- lang.: These functions deal with language-specific tasks like forced type conversions.
- math.: These functions provide mathematical operations.
- string.: These functions operate on strings and other sequences of data.
- string.gmtime.: These functions are used to format time values.
- sys.: These functions return operating-system related parameters, such as the hostname.
- sys.(gmtime./.localtime./.time.): These functions are used to format time values.

TrafficScript function names are not case sensitive. Therefore, the function `lang.todouble()` can be invoked using `lang.toDouble()`, `Lang.ToDouble()`, and `lang.todouble()`.

Many functions take parameters and return values with specific types. TrafficScript casts variables and values to the appropriate type as described in "Type Casts in TrafficScript" on page 18.

array.append(array1, array2)

Appends each element of array2 to the end of array1. Note that this behaviour is different to that of array.push(), which would add one new element to the end of array1 containing array2. The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
# Append 4, 5 and 6 to the array
$numbers = [ 1, 2, 3 ];
array.append( $numbers, [ 4, 5, 6 ] );
```

© 2019 Pulse Secure, LLC.

```
# $numbers is now [ 1, 2, 3, 4, 5, 6 ]
```

See also: “array.push(array, value)”

array.contains(array, value)

Returns whether or not the supplied array contains the specified value as an element. Note that it will not match elements inside sub-arrays.

Sample Usage

```
$array = [ "one", "two", [ "three", "four" ] ];

# true
array.contains( $array, "one" );

# false
array.contains( $array, "three" );
```

See also: “array.filter(array, pattern, [flags])”

array.copy(array)

Returns a copy of the supplied array. This is semantically equivalent to `$arraycopy = $array`, however it will warn if the variable passed to it is not an array.

Sample Usage

```
# Loop around a sorted version of an array without
# permanently sorting it
$arr = [ "London", "Berlin", "Lisbon", "Paris" ];
foreach( $val in array.sort( array.copy( $arr ) )) {
   log.info( $val );
}

# $arr will be unsorted here
```

See also: “array.create(size, [default])”

array.create(size, [default])

Creates a new array of the specified size and optionally fills the array with the default data.

Sample Usage

```
# Create an array of length 20 with all the elements
# set to zero.
$zeroarray = array.create( 20, 0 );
```

See also: “array.resize(array, size, [default])”, “array.copy(array)”

array.filter(array, pattern, [flags])

Removes elements from the supplied array that do not match the pattern. The optional 'flags' parameter contains a string of single-letter options. The following options are supported:

© 2019 Pulse Secure, LLC.

- 'i', meaning 'case insensitive' - letters in the pattern match both upper and lower case letters.
- '!', meaning negate - elements of the array that match the regular expression are removed.

The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
# Get the extension headers for this request
$headers = http.listHeaderNames();
$extensions = array.filter( $headers, "^X-", "i" );
```

array.join(array, [separator])

Concatenates all the elements of the supplied array into a string separated by the separator. The elements will be separated by a space if no separator is supplied. Note that a warning will be printed should the supplied array itself contain any arrays or hashes.

Sample Usage

```
# Print a comma-separated list of names
$names = [ "alice", "bob", "mallory" ];
log.info( array.join( $names, ", " ) );
```

See also: “string.split(string, [separator])”

array.length(array)

Returns the length of the supplied array

Sample Usage

```
# See how many HTTP headers there are
$headers = http.listHeaderNames();
log.info( "There are "
          . array.length( $headers )
          . " headers in this request" );
```

array.pop(array)

Removes the last element of the supplied array and returns it as the result of the function.

Sample Usage

```
$stack = [];
array.push( $stack, 3 );
array.push( $stack, 2 );
array.push( $stack, 1 );
# $stack is now [ 3, 2, 1 ];

# Prints 1 2 3
while( array.length( $stack ) > 0 ) {
   log.info( array.pop( $stack ) );
```

© 2019 Pulse Secure, LLC.

```
}
```

See also: "array.unshift(array, value)", "array.push(array, value)", "array.shift(array)"

array.push(array, value)

Adds the supplied value to the end of the supplied array. The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
# Append 4 to the array
$numbers = [ 1, 2, 3 ];
array.push( $numbers, 4 );
```

See also: "array.shift(array)", "array.unshift(array, value)", "array.pop(array)"

array.resize(array, size, [default])

Resizes the supplied array to the specified size. If the size of the array is being increased and the default parameter is specified then the new elements added to the array will be set to the default parameter. If the new size is smaller than the original size then the appropriate number of elements will be removed from the end of the array. The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
# We're only interested in the first
# 10 lines of body data - but fill in the
# rest with blank lines if there are fewer
# than 10 lines.
$body = array.resize( http.getBodyLines(), 10, "" );
```

See also: "array.create(size, [default])"

array.reverse(array)

Reverses the elements of the supplied array. The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
$array = [ 1, 2, 3, [ 4, 5 ] ];

# array will be [ [ 4, 5 ], 3, 2, 1 ]
array.reverse( $array );
```

See also: "array.sort(array, [reverse])"

array.shift(array)

Removes the first element of the supplied array and returns it as the result of the function.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$array = [ 1, 2, 3, 4 ];
# Empty an array from the front while printing
# its contents
while( array.length( $array ) > 0 ) {
   log.info( array.shift( $array ) );
}
```

See also: "array.unshift(array, value)", "array.push(array, value)", "array.pop(array)"

array.sort(array, [reverse])

Sorts the supplied array alphanumerically. If the optional reverse parameter is true, then the array will be sorted in reverse. The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
# Sort the keys of a hash before iterating over them
foreach( $key in array.sort( hash.keys( $hash ) ) ) {
   log.info( $key . " maps to " . $hash[$keys] );
}

# Sort these numbers alphanumerically
$sorted = array.sort( [ 1, 2, 3, 4, 10, 11 ] );

# The result will be [ 1, 10, 11, 2, 3, 4 ]

# Reverse sort these numbers alphanumerically
$rev = array.sort( [ 1, 2, 3, 4, 10, 11 ], true );

# The result will be [ 4, 3, 2, 11, 10, 1 ]
```

See also: "array.sortNumerical(array, [reverse])"

array.sortNumerical(array, [reverse])

Sort the supplied array numerically in ascending order. If the optional reverse parameter is true, then the array will be sorted in descending order. The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
# Sort an array of numbers
$numbers = [ 2, 10, "3", "4", 24, "11" ];
array.sortNumerical( $numbers );

# $numbers is now [ 2, "3", "4", 10, "11", 24 ]

# Reverse sort an array of numbers
array.sortNumerical( $numbers, true );

# $numbers is now [ 24, "11", 10, "4", "3", 2 ]
```

See also: "array.sort(array, [reverse])"

© 2019 Pulse Secure, LLC.

array.splice(array, offset, length, [values])

Replace elements in the supplied array. Any elements between offset and length will be removed from the array and any extra values specified after the length parameter will be inserted into the array at that location. The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
$numbers = [ 0, 1, 2, 3, 4, 5 ];

# Starting from element 2, remove one element
# and insert "a" and "b" into the array
array.splice( $numbers, 2, 1, "a", "b" );

# $numbers is now [ 0, 1, "a", "b", 3, 4, 5 ]

# Starting at element 0, remove 3 elements
array.splice( $numbers, 0, 3 );

# $numbers is now [ "b", 3, 4, 5 ]
```

See also: “array.filter(array, pattern, [flags])”

array.unshift(array, value)

Adds the supplied value to the front of the supplied array. The return value of this function can be used as the argument to another function to perform multiple operations on the same array.

Sample Usage

```
# Prepend 1 to the array
$numbers = [ 2, 3, 4 ];
array.unshift( $numbers, 1 );
```

See also: “array.shift(array)”, “array.push(array, value)”, “array.pop(array)”

hash.contains(hash, key)

Returns whether the supplied hash contains a particular key.

Sample Usage

```
# See if bob is in the hash of users
#  and if so, whether his password matches
if( hash.contains( $users, "bob" ) ) {
   if( $users["bob"] == $password ) {
      # Access granted
   } else {
      # Access denied
   }
} else {
    # User does not exist
}
```

See also: “hash.keys(hash)”

© 2019 Pulse Secure, LLC.

hash.count(hash)

Returns the number of items in the hash.

Sample Usage

```
# See how many HTTP headers there are
$headers = http.getHeaders();
log.info( "There are "
          . hash.count( $headers )
          . " headers in this request" );
```

hash.delete(hash, key)

Deletes the specified key from the hash. The return value of this function can be used as the argument to another function to perform multiple operations on the same hash.

Sample Usage

```
$hash = [ "orange" => "fruit",
          "apple" => "fruit",
          "cabbage" => "vegetable",
          "pear" => "fruit" ];

hash.delete( $hash, "cabbage" );

# keys will be orange, apple and pear
$keys = hash.keys( $hash );
```

See also: "hash.keys(hash)"

hash.empty(hash)

Removes all the values from the hash. The return value of this function can be used as the argument to another function to perform multiple operations on the same hash.

Sample Usage

```
# Empty the contents of a hypothetical set object
sub set.empty( $set ) {
   # You cannot do '$set = [];' here because that
   # would be reassigning the input variable
   hash.empty( $set );
}

set.empty( $my_set );
```

See also: "hash.keys(hash)"

hash.keys(hash)

Returns an array containing the keys that map to values in the supplied hash data structure.

Sample Usage

```
# Get the set of IPs that have connected to the site
$ips = data.get( "connected_ips" );
$ips[request.getRemoteIP()] = 1;
data.set( "connected_ips", $ips );
$connected_set = hash.keys( $ips );
log.info( array.length( $connected_set ) .
          " unique IPs have connected to the site" );
```

See also: "hash.values(hash)"

hash.values(hash)

Returns an array containing the values that have been mapped to in the hash.

Sample Usage

```
# Add up the values of a hash
$values = hash.values( $hash );
$total = 0;
foreach( $val in $values ) {
   $total += $val;
}
log.info( "The total is " . $total );
```

See also: "hash.keys(hash)"

json.deserialize(json_string)

Converts the supplied string in JavaScript Object Notation (JSON) into a TrafficScript array or hash variable. If the supplied string is not in the correct format then a warning will be printed and the result will be empty.

Sample Usage

```
# Deserialising a JSON array
$json_array = '[ "element 1", "element 2" ]';
$array = json.deserialize( $json_array );

# Deserialising a JSON hash
$json_object = '{ "key 1":"value 1", \
                  "key 2":[ "array element" ] }';
$hash = json.deserialize( $json_object );
```

See also: "json.serialize(object)"

json.serialize(object)

Converts the supplied array or hash variable into JavaScript Object Notation (JSON). This format is commonly used to exchange data between online applications.

Sample Usage

```
# Serialising an array
$array = [ "element 1", "element 2" ];
$json_array = json.serialize( $array );
```


© 2019 Pulse Secure, LLC.

```
# Serialising a hash
$hash = [ "key 1" => "value 1",
          "key 2" => [ "array element" ] ];
$json_object = json.serialize( $hash );
```

See also: "json.deserialize(json_string)"

lang.assert(condition, message)

If the condition is false, prints a warning to the log with the current line number and terminates the rule. If the condition is true then no messages will be printed and the rule will continue as normal.

Sample Usage

```
# Make sure that this rule is only run with
# Virtual Servers that are using SSL Decryption.
lang.assert( ssl.isSSL(),
             "This rule should only be used \
              with decrypted SSL connections" );
```

See also: "lang.warn(message)"

lang.chr(number)

Converts a number to the corresponding ASCII character. chr() may be used as an alias for lang.chr().

Sample Usage

```
# Pick a random letter from A - Z
$char = lang.chr( math.random( 26 ) + 65 );
```

Alternative name: chr

See also: "lang.ord(string)"

lang.dump(variable)

Converts the supplied variable into a human-readable string. This function is useful for printing the contents of arrays and hashes when debugging TrafficScript rules.

Sample Usage

```
# Print the headers of the current HTTP request
log.info( lang.dump( http.getHeaders() ) );
```

lang.isarray(data)

Returns whether or not the supplied data is an array.

Sample Usage

```
sub passMeAnArray( $array ) {
   # Test whether the supplied data is an array
   if( !lang.isArray( $array ) ) {
```

```
      return false;
   }
   # ...
}
```

See also: "lang.ishash(data)"

lang.ishash(data)

Returns whether or not the supplied data is a hash.

Sample Usage

```
sub passMeAHash( $hash ) {
   # Test whether the supplied data is a hash
   if( !lang.isHash( $hash ) ) {
      return false;
   }
   # ...
}
```

See also: "lang.isarray(data)"

lang.max(param1, param2)

Returns the maximum value of the two parameters provided. If both parameters are strings, it uses a string comparison; otherwise, the parameters are promoted to integers or doubles and compared. max() may be used as a shorthand for lang.max().

Sample Usage

```
$r = lang.max( 9, 10 ); # returns 10
$s = lang.max( "9", "10" ); # returns "9"
$r = lang.max( 2, "4.8" ); # returns 4.8
```

Alternative name: max

See also: "lang.min(param1, param2)"

lang.min(param1, param2)

Returns the minimum value of the two parameters provided. If both parameters are strings, it uses a string comparison; otherwise, the parameters are promoted to integers or doubles and compared. min() may be used as a shorthand for lang.min().

Sample Usage

```
$r = lang.min( 9, 10 ); # returns 9
$s = lang.min( "9", "10" ); # returns "10"
$r = lang.min( 2, "4.8" ); # returns 2
```

Alternative name: min

See also: "lang.max(param1, param2)"

© 2019 Pulse Secure, LLC.

lang.ord(string)

Converts an ascii character to an integer. ord() may be used as a shorthand alias for lang.ord().

Sample Usage

```
# Get the integer value of a character.
$val = lang.ord( "A" );
```

Alternative name: ord

See also: "lang.toString(value)", "lang.chr(number)"

lang.toArray(values)

Returns an array of the supplied values.

Sample Usage

```
$arr = lang.toArray( "10", "hello" );
```

See also: "lang.toDouble(value)", "lang.toString(value)", "lang.toInt(value)"

lang.toDouble(value)

Returns the double (floating point) value of its parameter, using the TrafficScript type-casting rules.

Sample Usage

```
$r = lang.toDouble( "3.14157" ); # returns 3.14157
$r = lang.toDouble( 10 );        # returns 10
$r = lang.toDouble( 3.14175 );   # returns 3.14175
$r = lang.toDouble( "..." );     # returns 0.0
```

Alternative name: toDouble

See also: "lang.toInt(value)", "lang.toString(value)"

lang.toHash(values)

Returns an hash of the supplied key value pairs.

Sample Usage

```
$hash = lang.toHash( "ten", 10, "eleven", 11 );
```

See also: "lang.toDouble(value)", "lang.toString(value)", "lang.toInt(value)", "lang.toArray(values)"

lang.toInt(value)

Returns the integer value of its parameter, using the TrafficScript type-casting rules.

Sample Usage

```
$r = lang.toInt( "10xxx" ); # returns 10
$r = lang.toInt( 3.14157 ); # returns 3
$r = lang.toInt( 10 );      # returns 10
```

© 2019 Pulse Secure, LLC.

```
$r = lang.toInt( "..." );   # returns 0
```
Alternative name: toInt

See also: "lang.toDouble(value)", "lang.toString(value)"

lang.toString(value)

Returns the string value of its parameter, using the TrafficScript type-casting rules.

Sample Usage

```
$s = lang.toString( 10 ); # returns "10"
$s = lang.toString( 3.14157 ); # returns "3.14157"
$s = lang.toString( "10" ); # returns "10"
```
Alternative name: toString

See also: "lang.toInt(value)", "lang.toDouble(value)", "lang.chr(number)", "lang.ord(string)"

lang.tochar(number)

This function is an alias for lang.chr.

Sample Usage

```
# Pick a random letter from A - Z
$char = lang.tochar( math.random( 26 ) + 65 );
```
Alternative name: tochar

See also: "lang.chr(number)"

lang.warn(message)

Prints a warning to the log with the line number that this function call appears on. If strict error checking is enabled then the rule will abort.

Sample Usage

```
$decoded = string.decrypt( $password, "passphrase" );
if( !$decoded ) {
   lang.warn( "Failed to decrypt string" );
} else {
   # String decrypt succeeded...
}
```
See also: "lang.assert(condition, message)"

math.acos(x)

Calculates the arc cosine of x and returns an angle in radians in the range 0 to pi.

Sample Usage

```
$acos = math.acos( 0 ); # returns pi/2 (approx.)
```


© 2019 Pulse Secure, LLC.

Alternative name: acos

See also: "math.cos(angle)"

math.asin(x)

Calculates the arc sine of x and returns an angle in radians in the range -pi/2 to pi/2.

Sample Usage

```
$asin = math.asin( 0 ); # returns 0 (approx.)
```

Alternative name: asin

See also: "math.sin(angle)"

math.atan(angle)

Calculates the arc tangent of x and returns an angle in radians in the range -pi/2 to pi/2.

Sample Usage

```
$atan = math.atan( 0 ); # returns 0 (approx.)
```

Alternative name: atan

See also: "math.tan(angle)"

math.ceil(value)

Returns the smallest integer greater than or equal to its parameter.

Sample Usage

```
$r = math.ceil( 6.28 ); # returns 7
$r = math.ceil( "4" ); # returns 4
```

Alternative name: ceil

See also: "math.floor(value)", "math.rint(value)", "math.fabs(value)"

math.cos(angle)

Interprets its parameter as an angle in radians and returns its cosine.

Sample Usage

```
$cos = math.cos( 6.28314 ); # returns 1 (approx.)
```

Alternative name: cos

See also: "math.sin(angle)", "math.tan(angle)"

math.exp(power)

Calculates e raised to the power of its parameter and returns the result.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$r = math.exp( math.ln( 10 ) ); # returns 10
```
Alternative name: exp

See also: "math.ln(value)", "math.pow(num, power)"

math.fabs(value)

Interprets its parameter as a floating point number and returns its absolute value.

Sample Usage

```
$r = math.fabs( -6.28 ); # returns 6.28
```
Alternative name: fabs

See also: "math.floor(value)", "math.ceil(value)", "math.rint(value)"

math.floor(value)

Returns the largest integer not greater than its parameter.

Sample Usage

```
$r = math.floor( 4.001 ); # returns 4
$r = math.floor( 17 ); # returns 17
```
Alternative name: floor

See also: "math.ceil(value)", "math.rint(value)", "math.fabs(value)"

math.ln(value)

Returns the natural logarithm of its parameter.

Sample Usage

```
$ln = math.ln( 2.71828 ); # returns 1 (approximately)
```
Alternative name: ln

See also: "math.log(value)", "math.exp(power)"

math.log(value)

Returns the base10 logarithm of its parameter.

Sample Usage

```
$log = math.log( 100 ); # returns 2
```
Alternative name: log

See also: "math.ln(value)", "math.pow(num, power)"

© 2019 Pulse Secure, LLC.

math.pow(num, power)

Raises its first parameter to the power of its second parameter and returns the result.

Sample Usage

```
$r = math.pow( 2, 3 ); # returns 8
```

Alternative name: pow

See also: "math.ln(value)", "math.exp(power)", "math.sqrt(num)"

math.random(range)

Returns a pseudorandom integer greater than or equal to zero, and less than its parameter. This function uses a quick pseudo-random number generator meaning that the output of this function is not suitable for cryptographic purposes.

Sample Usage

```
# returns a value in the range 0 to 99
$rand = math.random( 100 );
```

Alternative name: random

See also: "string.randomBytes(length)"

math.rint(value)

Rounds its parameter by returning the integer closest to its value.

Sample Usage

```
$r = math.rint( 4.25 ); # returns 4
$r = math.rint( 4.75 ); # returns 5
```

Alternative name: rint

See also: "math.floor(value)", "math.ceil(value)", "math.fabs(value)"

math.sin(angle)

Interprets its parameter as an angle in radians and returns its sine.

Sample Usage

```
$sin = math.sin( 6.28314 ); # returns 0 (approx.)
```

Alternative name: sin

See also: "math.cos(angle)", "math.tan(angle)"

math.sqrt(num)

Returns the square root of its parameter.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$root = math.sqrt( 2 ); # returns 1.414 (approx.)

if( lang.toString( math.sqrt( $num ) ) == "nan" ) {
   # $num is negative or NaN ...
}
```

Alternative name: sqrt

See also: "math.pow(num, power)"

math.tan(angle)

Interprets its parameter as an angle in radians and returns its tangent.

Sample Usage

```
$tan = math.tan( 6.28314 ); # returns 0 (approx.)
```

Alternative name: tan

See also: "math.sin(angle)", "math.cos(angle)"

string.BERToInt(string)

Converts a BER compressed integer into an integer.

Sample Usage

```
# $r = 200
$r = string.BERToInt( "\201\110" );
```

Alternative name: BERToInt

See also: "string.intToBER(number)", "string.bytesToInt(string)"

string.Ireplace(string, search, replacement) - deprecated

This function has been deprecated. Use instead "string.replaceI(string, search, replacement)".

Returns a new string, copied from the original string, with the first occurrence of the search string in the supplied string replaced by the replacement string. This function is case-insensitive.

string.IreplaceAll(string, search, replacement) - deprecated

This function has been deprecated. Use instead "string.replaceAlll(string, search, replacement)".

Returns a new string, copied from the original string, with all occurrences of the search string in the supplied string replaced by the replacement string. This function is case-insensitive.

string.append(str1, str2, ...)

Returns the result of concatenating all of its inputs together as strings.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Returns "The answer is 42"
$s = string.append( "The ", "answer ", "is " , 42 );
```

Alternative name: append

string.base64decode(string)

Decodes a base64-encoded string and returns the result.

Base64 encoding is used for MIME-encoded messages, and in the HTTP Basic Authorization header.

Sample Usage

```
# Decodes a username and password from HTTP
# BASIC authentication
$h = http.getHeader( "Authorization" );
if( string.startsWith( $h, "Basic " ) ) {
   $enc = string.skip( $h, 6 );
   $userpasswd = string.base64decode( $enc );
   log.info( "User used: ".$userpasswd );
}
```

Alternative name: base64decode

See also: “string.base64encode(string)”, “string.hexdecode(encoded string)”, “string.unescape(escaped string)”

string.base64encode(string)

Returns the base64-encoded version of the provided string. This converts each group of three characters into a 4-character string containing just [A-Za-z0-9+/], and '=' for padding.

Base64 encoding is used for MIME-encoded messages, and in the HTTP Basic Authorization header.

Sample Usage

```
# Encodes a username and password for HTTP
# BASIC authentication
$enc = string.base64encode( "user:passwd" );
$h = "Basic ".$enc;
http.setHeader( "Authorization", $h );
```

Alternative name: base64encode

See also: “string.base64decode(string)”, “string.hexencode(string)”, “string.escape(string)”

string.bytesToDotted(string)

Converts a network ordered byte string into an IP address.

Sample Usage

```
# The first 4 bytes are the IP address
$ipstr = string.substring( $msg, 0, 3 );
```

```
log.info( "IP is ".string.bytesToDotted( $ipstr ) );
```
Alternative name: bytesToDotted

See also: “string.dottedToBytes(IP address)”, “string.bytesToInt(string)”

string.bytesToInt(string)

Converts a byte string in network order to an integer. The byte string should be either 1, 2 or 4 bytes long.

Sample Usage

```
# $msg starts with a 2-bytes length
$lenstr = string.substring( $msg, 0, 1 );
$len = string.bytesToInt( $lenstr );
$msg = string.substring( $msg, 2, 2+$len-1 );
```
Alternative name: bytesToInt

See also: “string.intToBytes(number, [width])”, “string.bytesToDotted(string)”, “string.BERToInt(string)”

string.cmp(str1, str2)

Compares its two parameters as strings in a case-sensitive manner. It returns a negative value if str1 is less than str2; zero if they are equal, and a positive value if str1 is greater than str2.

Sample Usage

```
if( string.cmp( $a, "HTTP/1.0" ) == 0 ) {
   # $a is "HTTP/1.0"
}

if( string.cmp( $a, $b ) < 0 ) {
   # $a is less than $b
}
```
Alternative name: cmp

See also: “string.icmp(str1, str2)”

string.contains(haystack, needle)

Searches for the provided string (the needle) in the given source (the haystack).

It returns 1 if the 'needle' was found, or 0 otherwise.

Sample Usage

```
if( string.contains( $cookie, "chocolate" ) ) {
   # The cookie contains chocolate ...
}
```
Alternative name: contains

© 2019 Pulse Secure, LLC.

See also: “string.containsI(haystack, needle)”, “string.find(haystack, needle, [start])”, “string.findr(haystack, needle, [distanceFromEndToStart])”, “string.startsWith(string, prefix)”, “string.endsWith(string, suffix)”

string.containsI(haystack, needle)

Searches for the provided string (the needle) in the given source (the haystack). It is case-insensitive.

It returns 1 if the 'needle' was found, or 0 otherwise.

Sample Usage

```
if( string.containsI( $path, "danger" ) ) {
   # The path contains danger ...
}
```

Alternative name: containsI

See also: “string.contains(haystack, needle)”, “string.findI(haystack, needle, [start])”, “string.startsWithI(string, prefix)”, “string.endsWithI(string, suffix)”

string.count(haystack, needle, [start])

Searches from the start of a string, counting the number of times that the provided search string (the needle) is found inside the given string (the haystack). An optional parameter can specify the start position for the search.

It returns the number of times that the string is found.

Sample Usage

```
# Returns 2
$r = string.count( "This is it!", "is" );

# Returns 1, no overlaps allowed
$s = string.count( "ooo", "oo" );
```

Alternative name: count

See also: “string.find(haystack, needle, [start])”, “string.contains(haystack, needle)”

string.decrypt(string, passphrase)

Returns the decrypted version of a string that has previously been encrypted using string.encrypt(). The passphrase supplied must match that given to string.encrypt(), otherwise the decoding will fail.

An empty string is returned if the decrypt or the integrity check fails.

Sample Usage

```
# Decrypt the 'kart' cookie
$cookie = http.getcookie( "kart" );
if( $cookie != "" ) {
```

```
   $cookie = string.decrypt( $cookie, $passphrase );
   if( $cookie == "" ) {
      log.warn( "User modified kart cookie" );
      http.removecookie( "kart" );
   } else {
      http.setcookie( "kart", $cookie );
   }
}
```

Alternative name: decrypt

See also: "string.encrypt(string, passphrase)"

string.dottedToBytes(IP address)

Converts an IP address to a network order byte string.

Sample Usage

```
# Prepend the client IP onto $msg
$ip = request.getRemoteIP();
$msg = string.dottedToBytes( $ip ).$msg;
```

Alternative name: dottedToBytes

See also: "string.bytesToDotted(string)", "string.intToBytes(number, [width])"

string.drop(string, count)

Returns all but the last 'count' characters from the end of the provided string. An empty string will be returned if 'count' is greater than the length of the original string.

Sample Usage

```
# returns "www.example"
$s = string.drop( "www.example.com", 4 );
```

Alternative name: drop

See also: "string.skip(string, count)", "string.trim(string)"

string.encrypt(string, passphrase)

Encrypts a string using the provided pass phrase. The returned string is encrypted using the AES block cipher, using an expanded form of the passphrase as the cipher key. A MAC is also added to ensure the integrity of the string.

This is open to replay attacks, and as such, should not be used to encrypt sensitive data, such as credit card details.

Sample Usage

```
# Encrypt the 'kart' cookie
$cookie = http.getresponsecookie( "kart" );
if( $cookie != "" ) {
```


© 2019 Pulse Secure, LLC.

```
   $cookie = string.encrypt( $cookie, $passphrase );
   http.setresponsecookie( "kart", $cookie );
}
```

Alternative name: encrypt

See also: "string.decrypt(string, passphrase)"

string.endsWith(string, suffix)

Returns 1 if the provided string ends with the given suffix, and 0 otherwise.

Sample Usage

```
if( string.endsWith( $url, ".cgi") ) {
   # Request is for a CGI script ...
}
```

Alternative name: endsWith

See also: "string.endsWithI(string, suffix)", "string.startsWith(string, prefix)", "string.contains(haystack, needle)"

string.endsWithI(string, suffix)

Returns 1 if the provided string ends with the given suffix, and 0 otherwise. It is case-insensitive.

Sample Usage

```
if( string.endsWithI( $path, "victory") ) {
   # The path ends with victory
}
```

Alternative name: endsWithI

See also: "string.endsWith(string, suffix)", "string.startsWithI(string, prefix)", "string.containsI(haystack, needle)"

string.escape(string)

Returns a percent-encoded version of its parameter.

Control characters and spaces (character value <= 32) and '%' characters are each replaced by a '%' symbol, followed by their 2-digit hex value.

Sample Usage

```
# returns "Hello%20World!%0D%0A"
$s = string.escape( "Hello World!\r\n" );
```

Alternative name: escape

See also: "string.unescape(escaped string)", "string.hexencode(string)", "string.regexescape(string)", "string.urlencode(string)"

string.extractHost(string)

Returns the host part of the supplied address if it is a valid IP or hostname. Otherwise the empty string is returned.

Sample Usage

```
sub lookup_proxy( $path ) {
   # Code to map a path to a node
}
# Send certain requests to an external server
$forward = lookup_proxy( http.getPath() );
if( $forward
    && $host = string.extractHost( $forward ) ) {
   if( !$port = string.extractPort( $forward ) ) {
      $port = 80;
   }
   pool.use( "External", $host, $port );
}
```

Alternative name: extractHost

See also: "string.extractPort(string)"

string.extractPort(string)

Returns the port part of the supplied address if both the host and port of the supplied address are valid. Otherwise the empty string is returned.

Sample Usage

```
# Get the SIP request's next destination
$next = sip.getRequestHeader( "Route" );
string.regexmatch( $next, "^<sip:(.*?)>" );
$next = $1;
# Get the port number of the destination
if( $port = string.extractPort( $next ) ) {
   if( $port < 1024 ) {
      # Don't forward the message
      sip.sendResponse( "403", "Forbidden" );
   }
}
```

Alternative name: extractPort

See also: "string.extractHost(string)"

string.find(haystack, needle, [start])

Reports whether the provided search string (the needle) is contained inside the given string (the haystack). An optional parameter can specify the start position for the search.

It returns the location of the first instance of the search string; note that character positions start at 0.

If it could not find the search string, it returns -1.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$r = string.find( "This is it!", "is" ); # Returns 2
```

Alternative name: find

See also: "string.findI(haystack, needle, [start])", "string.findr(haystack, needle, [distanceFromEndToStart])", "string.contains(haystack, needle)"

string.findI(haystack, needle, [start])

Reports whether the provided search string (the needle) is contained inside the given string (the haystack). An optional parameter can specify the start position for the search. It is case-insensitive.

It returns the location of the first instance of the search string; note that character positions start at 0.

If it could not find the search string, it returns -1.

Sample Usage

```
# Returns 0
$r = string.findI( "The way of the warrior", "the" );
```

Alternative name: findI

See also: "string.find(haystack, needle, [start])", "string.containsI(haystack, needle)"

string.findr(haystack, needle, [distanceFromEndToStart])

Searches backwards from the end of a string, determining whether the provided search string (the needle) is contained inside the given string (the haystack). An optional parameter can specify the start position for the search, measured from the end of the string (so setting it to 1 skips the last character in the string).

It returns the location of the last instance of the search string; note that character positions start at 0.

If it could not find the search string, it returns -1.

Sample Usage

```
# Returns 8 (the 'i' in it)
$r = string.findr( "This is it!", "i" );

# Returns 5 (the 'i' in is)
$r = string.findr( "This is it!", "i", 3 );
```

Alternative name: findr

See also: "string.find(haystack, needle, [start])", "string.contains(haystack, needle)"

string.hash(string)

Returns a number representing a hash of the provided string. When using string.hash(), negative numbers can be returned for the hash value. The returned value should not be relied on to be consistent across different releases of the software.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$hash = string.hash( http.getRawURL() );
connection.setPersistenceKey( $hash % 1000 );
```

Alternative name: hash

string.hashMD5(string)

Returns the MD5 hash of the provided string. The returned string will be 16 bytes long, and may contain non-printable characters.

Sample Usage

```
$hash = string.hashMD5( $data );
```

Alternative name: hashMD5

See also: "string.hexencode(string)", "string.hashSHA1(string)", "string.hashSHA256(string)", "string.hashSHA384(string)", "string.hashSHA512(string)"

string.hashSHA1(string)

Returns the SHA1 hash of the provided string. The returned string will be 20 bytes long, and may contain non-printable characters.

Sample Usage

```
$hash = string.hashSHA1( $data );
```

Alternative name: hashSHA1

See also: "string.hexencode(string)", "string.hashMD5(string)", "string.hashSHA256(string)", "string.hashSHA384(string)", "string.hashSHA512(string)"

string.hashSHA256(string)

Returns the SHA-256 hash of the provided string. The returned string will be 32 bytes long, and may contain non-printable characters.

Sample Usage

```
$hash = string.hashSHA256( $data );
```

Alternative name: hashSHA256

See also: "string.hexencode(string)", "string.hashMD5(string)", "string.hashSHA1(string)", "string.hashSHA384(string)", "string.hashSHA512(string)"

string.hashSHA384(string)

Returns the SHA-384 hash of the provided string. The returned string will be 48 bytes long, and may contain non-printable characters.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$hash = string.hashSHA384( $data );
```
Alternative name: hashSHA384

See also: “string.hexencode(string)”, “string.hashMD5(string)”, “string.hashSHA1(string)”, “string.hashSHA256(string)”, “string.hashSHA512(string)”

string.hashSHA512(string)

Returns the SHA-512 hash of the provided string. The returned string will be 64 bytes long, and may contain non-printable characters.

Sample Usage

```
$hash = string.hashSHA512( $data );
```
Alternative name: hashSHA512

See also: “string.hexencode(string)”, “string.hashMD5(string)”, “string.hashSHA1(string)”, “string.hashSHA256(string)”, “string.hashSHA384(string)”

string.hexToInt(string)

Converts a hexadecimal number to an integer. Returns the first valid hexadecimal value found in the string, or 0. A prefix of "0x" is accepted, but not required. Negative numbers are also valid.

Sample Usage

```
# Returns 10.
$int = string.hexToInt( "0000a" );
```
See also: “string.intToHex(string)”

string.hexdecode(encoded string)

Returns the hex-decoded version of the provided string. This interprets each character pair as a 2-digit hex value, replacing it with the corresponding 8-bit character. It does not verify that the original string was correctly encoded.

Sample Usage

```
# returns "hello"
$s = string.hexdecode("68656C6C6F");
```
Alternative name: hexdecode

See also: “string.hexencode(string)”, “string.base64decode(string)”, “string.unescape(escaped string)”

string.hexencode(string)

Returns the hex-encoded version of the provided string . This converts each character into a two-character hex representation, doubling the length of the string.

Sample Usage

```
# Returns "68656C6C6F"
$s = string.hexencode( "hello" );
```

Alternative name: hexencode

See also: “string.hexdecode(encoded string)”, “string.base64encode(string)”, “string.escape(string)”

string.htmldecode(encodedstring)

Returned the HTML-decoded version of a string, converting symbols such as < and "

Sample Usage

```
$body = string.htmldecode( http.getBody( 0 ) );
```

Alternative name: htmldecode

See also: “string.htmlencode(string)”

string.htmlencode(string)

Returns the encoded version of the supplied string to make it safe for including in HTML. It converts '<' to <, '>' to >, '" to " and '&' to &. Control characters are hex-encoded.

Sample Usage

```
$html = string.htmlencode( $parameter );
```

Alternative name: htmlencode

See also: “string.htmldecode(encodedstring)”, “string.urlencode(string)”

string.icmp(str1, str2)

Compares its two parameters as strings in a case-insensitive manner. It returns a negative value if str1 is less than str2; zero if they are equal, and a positive value if str1 is greater than str2.

Sample Usage

```
if( string.icmp( $a, "Content-Length" ) == 0 ) {
   # $a is "Content-Length"
}
```

Alternative name: icmp

See also: “string.cmp(str1, str2)”

string.insertBytes(string, insertion, offset)

Returns a new string with the supplied string inserted into the original string at the offset specified. If offset < 0, or offset > length(string), the original string is returned unchanged. If offset == 0 the insertion string is prepended; if offset == length(string) the insertion string is appended.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# $r = "he was Othello"
$r = string.insertbytes( "hello", " was Othe", 2 );

# $r = "hello world"
$r = string.insertbytes( "hello", " world", 5 );

# $r = "I say hello"
$r = string.insertbytes( "hello", "I say ", 0 );
```

Alternative name: insertBytes

See also: "string.replaceBytes(string, replacement, offset)"

string.intToBER(number)

Converts an integer into a BER compressed integer (which is a variable-length binary string encoding).

Sample Usage

```
# $r = "\201\110"
$r = string.intToBER( 200 );
```

Alternative name: intToBER

See also: "string.BERToInt(string)", "string.intToBytes(number, [width])"

string.intToBytes(number, [width])

Converts an integer to a network order byte string of the specified width. Only widths of 1, 2 and 4 are permitted, and the width defaults to 4 if it is not supplied.

Sample Usage

```
# Prepend $msg with its length, as a 4-byte int
$msg = string.insertBytes( $msg,
   string.intToBytes( string.len( $msg ) ),
   0 );
```

Alternative name: intToBytes

See also: "string.bytesToInt(string)", "string.bytesToDotted(string)", "string.intToBER(number)"

string.intToHex(string)

Converts an integer into a hexadecimal string.

Sample Usage

```
# Returns "0000cafe"
$hex = string.intToHex( 51966 );
```

See also: "string.hexToInt(string)"

string.ipmaskmatch(IP Address, CIDR IP Subnet)

Returns 1 if the provided IP address is contained in the CIDR IP Subnet, and 0 otherwise.

It interprets its first parameter as a string containing an IP address, and its second parameter as an CIDR IP subnet. CIDR IP subnets can be of the form "10.0.1.0/24", "10.0.1.0/255.255.255.0", "10.0.1." or "10.0.1.1".

For IPv6, the standard notation of "2001:200:0:8002::/64" is supported.

Sample Usage

```
if( string.ipmaskmatch( $ip, "10.0.0.0/8" ) ) {
   # IP is in the 10.*.*.* subnet ...
}
```

Alternative name: ipmaskmatch

See also: “string.validIPAddress(string)”

string.left(string, count)

Returns the first 'count' characters of the provided string. An empty string will be returned if 'count' is less than or equal to zero.

Sample Usage

```
# returns "#!"
$s = string.left( "#!/bin/sh", 2 );
```

Alternative name: left

See also: “string.skip(string, count)”

string.len(string)

Interprets its parameter as a string and returns its length (in bytes).

Sample Usage

```
$len = string.len( "Hello world!" ); # returns 12
```

Alternative name: len

string.length(string)

This function is an alias for string.len.

Sample Usage

```
$len = string.length( "Hello world!" ); # returns 12
```

Alternative name: length

See also: “string.len(string)”

© 2019 Pulse Secure, LLC.

string.lowercase(string)

Returns a new string containing all characters in the provided string converted to lowercase. The provided string is not modified.

Sample Usage

```
$s = string.lowercase("AbCdEfG"); # Returns "abcdefg"
```

Alternative name: lowercase

See also: “string.uppercase(string)”

string.normalizeIPAddress(string)

Returns a unique string representation of an IP address: all leading zeros are removed, and for IPv6 addresses the longest occurrence of a block of more than one consecutive zero is replaced by "::". If there is more than one such block then the first one is replaced. This normal form can be used to compare IP addresses without ambiguity and is also the form used by TrafficScript functions returning IP addresses. If the string is not a valid IP address, the empty string is returned.

Sample Usage

```
$remote_ip = request.getremoteip();
$client = string.normalizeipaddress(
            "2001::0157:0:0:0:0:006a" );
# $client is now "2001:0:157::6a"
if( $remote_ip == $client ){
   # do something specific for this client
}
```

Alternative name: normalizeIPAddress

See also: “string.validIPAddress(string)”

string.randomBytes(length)

Returns a string of the supplied length filled with random bytes. The random number generator used by this function generates output that is suitable for use in cryptographic operations.

Sample Usage

```
# Get a string 16 bytes long filled with random data
$str = string.randomBytes( 16 );
```

Alternative name: randomBytes

See also: “math.random(range)”

string.regexescape(string)

Returns a version of its parameter suitable for using inside a regex match as a string literal.

All characters in the string that aren't a-z, A-Z, 0-9 or '_' are escaped using a backslash.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$str = "10.100.230.5";
$escaped = string.regexescape( $str );
if( string.regexmatch( $line, $escaped ) ) {
   log.info( "Line matched: " . $str );
}
```

Alternative name: regexescape

See also: "string.escape(string)"

string.regexmatch(string, regex, [flags]))

Performs a regular expression match on the supplied string. If the regular expression 'regex' contains bracketed sub-expressions, then the variables $1 ... $9 will be set to the matching substrings.

Note that the '\' character is an escape character in TrafficScript strings enclosed with double quotation marks. If you need to put a literal '\' in a regular expression, you must escape it as '\\' or enclose the string in single quotation marks. To match a literal string inside a regular expression use the string.regexEscape function.

The optional 'flags' parameter contains a string of single-letter options. The following options are supported:

- 'i', meaning 'case insensitive' - letters in the pattern match both upper and lower case letters.

It returns 1 if matched, and 0 otherwise.

If the regular expression is constructed from client supplied data, see the security considerations in the "Administration System Security" chapter of the User's Guide.

Sample Usage

```
$id = "user=Joe, password=Secret";
if( string.regexmatch(
   $id, "^user=(.*), password=(.*)$" ) ) {
   log.info( "Got UserID: ".$1.", Password: ".$2 );
}

$name = "Name( Bob )";
string.regexmatch( $name, 'Name\( (\S+) \)' );
log.info( $1 ); # prints Bob
```

Alternative name: regexmatch

See also: "string.wildmatch(string, pattern)", "string.regexsub(string, regex, replacement, [flags])", "string.regexescape(string)"

string.regexsub(string, regex, replacement, [flags])

Returns a new string containing the results of a regular expression match on the supplied string, replacing each matching substring with the supplied replacement. The replacement string may contain $1 .. $9 substitutions, which reference bracketed sub-expressions in the regular expression.

© 2019 Pulse Secure, LLC.

Note that the '\' character is an escape character in TrafficScript strings enclosed with double quotation marks. If you need to put a literal '\' in a regular expression, you must escape it as '\\' or enclose the string in single quotation marks. To match a literal string inside a regular expression use the string.regexEscape function.

The optional 'flags' parameter contains a string of single-letter options. The following options are supported:

- 'g', meaning 'global replace' - apply the regex pattern as many times as possible.
- 'i', meaning 'case insensitive' - letters in the pattern match both upper and lower case letters.

string.regexsub() returns the string with the replacements.

If the regular expression is constructed from client supplied data, see the security considerations in the "Administration System Security" chapter of the User's Guide.

Sample Usage

```
# Rewrite incoming URL
$url = string.regexsub($url, "^/(.*)/secure",
                       "/$1/private");


# Remove cookieless session from URL
# e.g. "/(sessionid)/app/index" => "/app/index"
$url = string.regexsub( $url, '/\(\S+\)/', "/" );
```

Alternative name: regexsub

See also: “string.regexmatch(string, regex, [flags]))”

string.replace(string, search, replacement)

Returns a new string, copied from the original string, with the first occurrence of the search string in the supplied string replaced by the replacement string. This function is case-sensitive.

Sample Usage

```
# Rewrite incoming URL
$url = string.replace($url, "/secure", "/private");
```

Alternative name: replace

See also: “string.replaceI(string, search, replacement)”, “string.replaceAll(string, search, replacement)”, “string.replaceAlll(string, search, replacement)”

string.replaceAll(string, search, replacement)

Returns a new string, copied from the original string, with all occurrences of the search string in the supplied string replaced by the replacement string. This function is case-sensitive.

Sample Usage

```
# Rewrite incoming URL
$url = string.replaceAll($url, "/in", "/out");
```

Alternative name: replaceAll

See also: “string.replaceAllI(string, search, replacement)”, “string.replace(string, search, replacement)”, “string.replaceI(string, search, replacement)”

string.replaceAllI(string, search, replacement)

Returns a new string, copied from the original string, with all occurrences of the search string in the supplied string replaced by the replacement string. This function is case-insensitive.

Sample Usage

```
# Rewrite incoming URL
$url = string.replaceAllI($url, "/in", "/out");
```

Alternative name: replaceAllI

See also: “string.replaceAll(string, search, replacement)”, “string.replaceI(string, search, replacement)”, “string.replace(string, search, replacement)”

string.replaceBytes(string, replacement, offset)

Returns a new string, copied from the original string, with the appropriate bytes replaced by the replacement string at the supplied offset. The new string is the same length as the original string.

If offset < 0, or offset >= length(string), or length(replacement) == 0, string.replaceBytes returns a copy of the original string.

Sample Usage

```
# $r = "hi lo"
$r = string.replaceBytes( "hello", "i ", 1 );

# $r = "helwo"
$r = string.replaceBytes( "hello", "world", 3 );
```

Alternative name: replaceBytes

See also: “string.insertBytes(string, insertion, offset)”

string.replaceI(string, search, replacement)

Returns a new string, copied from the original string, with the first occurrence of the search string in the supplied string replaced by the replacement string. This function is case-insensitive.

Sample Usage

```
# Rewrite incoming URL
$url = string.replaceI($url, "/secure", "/private");
```

Alternative name: replaceI

See also: “string.replace(string, search, replacement)”, “string.replaceAllI(string, search, replacement)”, “string.replaceAll(string, search, replacement)”

© 2019 Pulse Secure, LLC.

string.reverse(string)

Returns the characters of a string in reverse order.

Sample Usage

```
$c = string.reverse( "esrever" ); # Returns "reverse"
```

Alternative name: reverse

string.right(string, count)

Returns the last 'count' characters of the provided string. An empty string will be returned if 'count' is less than or equal to zero.

Sample Usage

```
# returns "sh"
$s = string.right( "#!/bin/sh", 2 );
```

Alternative name: right

See also: “string.drop(string, count)”

string.skip(string, count)

Returns all but the first 'count' characters from the input string. An empty string will be returned if 'count' is greater than the length of the original string.

Sample Usage

```
# returns "/bin/sh"
$s = string.skip( "#!/bin/sh", 2 );
```

Alternative name: skip

See also: “string.drop(string, count)”, “string.trim(string)”

string.split(string, [separator])

Returns an array containing the substrings of the supplied string that are delimited by the separator. The separator defaults to a space character, so applying this method to a string without supplying the separator character will return an array containing all the individual words in the string. Using the empty string as the separator will return an array with each character as a separate element.

Sample Usage

```
$words = string.split( http.getResponseBody() );
log.info( "There were " . array.length( $words )
          . " words in the response" );
```

See also: “array.join(array, [separator])”

string.sprintf(format string, arguments)

Behaves like the C library sprintf() function. Only %s, %c, %d and %f are supported. The function returns the generated string.

Sample Usage

```
$text = string.sprintf(
   "Apples: %3d Oranges: %3d\n",
   $apples, $oranges );
```

Alternative name: sprintf

See also: "string.append(str1, str2, ...)"

string.startsWith(string, prefix)

Returns 1 if the provided string starts with the given prefix, and 0 otherwise.

Sample Usage

```
if( string.startsWith( $url, "http://" ) ) {
   # URL is of the form http://host/uri ...
}
```

Alternative name: startsWith

See also: "string.startsWithI(string, prefix)", "string.endsWith(string, suffix)", "string.contains(haystack, needle)"

string.startsWithI(string, prefix)

Returns 1 if the provided string starts with the given prefix, and 0 otherwise. It is case-insensitive.

Sample Usage

```
if( string.startsWithI( $path, "/tea" ) ) {
   # The path starts with tea ...
}
```

Alternative name: startsWithI

See also: "string.startsWith(string, prefix)", "string.endsWithI(string, suffix)", "string.containsI(haystack, needle)"

string.substring(string, base, end)

Returns the substring starting at character position 'base' and ending at position 'end'.

Note that character positions start at 0, and the end position is inclusive.

Sample Usage

```
# Set $s to "is is a"
$s = string.substring( "This is a string", 2, 8 );
```

Alternative name: substring

© 2019 Pulse Secure, LLC.

string.trim(string)

Returns the result of removing leading and trailing white space (and control characters) from its input

Sample Usage

```
$s = string.trim( " 1234 " ); # returns "1234"
```

Alternative name: trim

See also: "string.skip(string, count)", "string.drop(string, count)"

string.unescape(escaped string)

Returns the escape-decoded version of its parameter.

%-encoded characters are replaced with their decoded versions. %u-encoded characters are replaced with their UTF-8 representation. If there are illegal digits which cannot safely be converted, the variable $1 is set to 0 and the result contains '_' in place of the '%'. Such malicious %-escaped URLs are a common way of attacking unassuming servers or applications, and by handling them in this way, the attack is thwarted, but some information about a suspicious request is retained.

Sample Usage

```
# returns "\r\n100%"
$s = string.unescape("%0D%0A100%25");
# returns "something_BGelse"
$s = string.unescape("something%BGelse");
# returns "file.ida"
$s = string.unescape("file.%u0069%u0064%u0061");
```

Alternative name: unescape

See also: "string.escape(string)", "string.hexdecode(encoded string)"

string.uppercase(string)

Returns a new string containing all characters in the provided string converted to uppercase. The provided string is not modified.

Sample Usage

```
$s = string.uppercase("aBcDeFg"); # Returns "ABCDEFG"
```

Alternative name: uppercase

See also: "string.lowercase(string)"

string.urlencode(string)

Returns the URL-encoded version of the supplied string to make it safe for including in URLs. It converts anything other than A-Z a-z 0-9 . - _ to percent+hex form. It encodes reserved characters (RFC 3986) and % as well. string.urlencodeexceptreserved() should be used instead to exclude those.

© 2019 Pulse Secure, LLC.

Sample Usage

```
#$url will be
# %2Fcheck%20price%3Famount%3D12%26currency%3D%24
$url = string.urlencode(
             "/check price?amount=12&currency=$");
```

Alternative name: urlencode

See also: "string.unescape(escaped string)", "string.htmlencode(string)"

string.urlencodeexceptreserved(string)

Returns the URL-encoded version of the supplied string to make it safe for including in URLs. It converts characters except - unreserved characters A-Z a-z 0-9 . - _ - reserved characters / ? # [ ] @ ! $ & ' ( ) - % according to RFC 3986 to percent+hex form.

Sample Usage

```
# $url will be
# "/check%20price?amount=12&currency=$"
# where space ' ' gets encoded to %20
# and leaving /, ?, = and $ as they were.
$url = string.urlencodeexceptreserved(
           "/check price?amount=12&currency=$");
```

Alternative name: urlencodeexceptreserved

See also: "string.unescape(escaped string)", "string.htmlencode(string)", "string.urlencode(string)"

string.validIPAddress(string)

Returns 4 if the string provided is an IPv4 address, 6 if it is an IPv6 address and 0 if it is not a valid IP address.

Sample Usage

```
$ipversion = string.validIPAddress( $x );
if( 4 == $ipversion ) {
   # do something specific to IPv4
} else if( 6 == $ipversion ) {
   # do something specific to IPv6
} else {
   # error: not a valid IP address
}
```

Alternative name: validIPAddress

See also: "string.ipmaskmatch(IP Address, CIDR IP Subnet)"

string.wildmatch(string, pattern)

Performs a shell-like wild match on the supplied string. The pattern may contain the wildcard characters '?' (which matches a single character) and '*' (which matches any substring).

It returns 1 if matched, and 0 otherwise.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$url = http.getPath();
if( string.wildmatch( $url, "*.cgi" ) ) {
   # Is a request for a CGI script ...
}
```

Alternative name: wildmatch

See also: "string.regexmatch(string, regex, [flags]))"

string.gmtime.parse(str)

Parses the supplied string as a time stamp and returns the time in seconds since the epoch (1st Jan 1970). Dates before the epoch or after 2038 will produce unexpected results.

Note: Timezone information contained inside the time string is ignored. The time is always assumed to be in GMT.

Sample Usage

```
# Parse a string into unix time format
$str = "Tue, 21 Oct 2008 13:44:26 GMT";
$time = string.gmtime.parse( $str );
```

See also: "sys.gmtime.format(format, unixtime)"

sys.domainname()

Returns the domain name of the host machine. For example, if the machine is named "server1.example.com", sys.domainname() will return "example.com".

Sample Usage

```
$domainname = sys.domainname();
```

Alternative name: domainname

See also: "sys.hostname()"

sys.getenv(variable)

Returns the named environment variable, or the empty string if the environment variable does not exists.

Sample Usage

```
$zeushome = sys.getenv( "ZEUSHOME" );
```

Alternative name: getenv

sys.getnetworkinterfaces(hash_of_options)

Returns a list of network interfaces on the system, with a hash of the following for each one: Bcast, pppdest, IfName, netmask, MacAddr, IP, up

The following options can be supplied in a hash in order to alter the output:

"include_lo" => [0|1] This option controls whether the list returned should include loopback addresses. (Default: 1)

"up" => [0|1] If 1, only return interfaces marked as UP. If 0, ignore the UP status. (Default: 0)

Sample Usage

```
$info = sys.getnetworkinterfaces( ["include_lo" => 0,
                                   "up" => 1 ] );
if($info)
{
   foreach( $ipinfo in $info )  {
      $ip = $ipinfo["IP"];
      $name = $ipinfo["IfName"];
      $mac =  $ipinfo["MacAddr"];
      log.info("---------------------"
             . "\n interface = " . $name
             . "\n IP = " . $ip
             . "\n MAC = " . $mac
            );
   }
}else {
   log.info("No network interfaces returned");
}
```

sys.getpid()

Returns the process id of the current process.

Sample Usage

```
$mypid = sys.getpid();
```

Alternative name: getpid

sys.hostname()

Returns the hostname of the host machine. For example, if the machine is named "server1.example.com", sys.hostname() will return "server1".

Sample Usage

```
$hostname = sys.hostname();
```

Alternative name: hostname

See also: “sys.domainname()”

sys.time()

Returns the current system time as the number of seconds since midnight, 1/1/1970.

Sample Usage

```
$unixtime = sys.time();
```


© 2019 Pulse Secure, LLC.

Alternative name: time

See also: "sys.timeToString(unixtime)", "sys.localtime.format(format, unixtime)", "sys.gmtime.format(format, unixtime)", "sys.time.highres()"

sys.timeToString(unixtime)

Takes the time in seconds since midnight, 1/1/1970 and if the optional unixtime parameter is provided, returns a formatted string representing that time. If the unixtime parameter is not given, it returns the current time as a formatted string.

Sample Usage

```
# Returns "[01/Feb/2004:12:24:51 +2000]"
$tm = sys.timeToString( sys.time() );
```

Alternative name: timeToString

See also: "sys.time()"

sys.gmtime.format(format, unixtime)

Converts the time into a string format. This function converts using GM time - see sys.localtime.format() to convert using localtime.

Format	Meaning	Format	Meaning
%a	Mon Tue Wed ...	%A	Monday Tuesday ...
%b	Jan Feb Mar ...	%B	January February ...
%d	Day of month "01"-"31"	%D	%m/%d/%y
%e	Day of month " 1"-"31"	%H	Hour of day "00-23"
%h	Equivalent to %b	%I	Hour of day "01" - "12"
%j	Julian day of the year "001" to "366"	%m	Month of year "01" - "12"
%M	Minute "00" - "59"	%n	Newline character
%p	AM/PM	%r	Time in %I:%M:%S [AM|PM]
%R	%H:%M	%S	Seconds, output as a number between "00" and "61"
%t	Tab character	%T	%H:%M:%S
%u	Day of week (1 = Monday, 7 = Sunday)	%w	Day of week (0 = Sunday, 6 = Saturday)
%y	Year (without century) "00" to "99"	%Y	Year "0000" to "9999"
%Z	Time zone ("GMT")	%%	"%"

© 2019 Pulse Secure, LLC.

If supplied, the optional 'unixtime' parameter specifies the number of seconds since midnight 1/1/1970, and the function returns a formatted string representing that time. If the 'unixtime' value is not provided, the current time will be returned.

Sample Usage

```
# Return a time suitable for an HTTP header
# e.g. Mon, 14 Aug 2006 12:39:01 GMT
$str = sys.gmtime.format( "%a, %d %b %Y %T GMT" );
```

Alternative name: gmtime.format

See also: “sys.time()”, “sys.time.seconds(unixtime)”, “sys.time.minutes(unixtime)”, “sys.time.hour(unixtime)”, “sys.time.weekday(unixtime)”, “sys.time.monthday(unixtime)”, “sys.time.month()”, “sys.time.year(unixtime)”, “sys.time.yearday(unixtime)”, “sys.localtime.format(format, unixtime)”, “string.gmtime.parse(str)”

sys.localtime.format(format, unixtime)

Converts the time into a string format. This function converts using localtime - see sys.gmtime.format() to convert using GMT.

Format	Meaning	Format	Meaning
%a	Mon Tue Wed ...	%A	Monday Tuesday ...
%b	Jan Feb Mar ...	%B	January February ...
%d	Day of month "01"-"31"	%D	%m/%d/%y
%e	Day of month " 1"-"31"	%H	Hour of day "00-23"
%h	Equivalent to %b	%I	Hour of day "01" - "12"
%j	Julian day of the year "001" to "366"	%m	Month of year "01" - "12"
%M	Minute "00" - "59"	%n	Newline character
%p	AM/PM	%r	Time in %I:%M:%S [AM|PM]
%R	%H:%M	%S	Seconds, output as a number between "00" and "61"
%t	Tab character	%T	%H:%M:%S
%u	Day of week (1 = Monday, 7 = Sunday)	%w	Day of week (0 = Sunday, 6 = Saturday)
%y	Year (without century) "00" to "99"	%Y	Year "0000" to "9999"
%Z	Time zone (from $TZ)	%%	"%"

© 2019 Pulse Secure, LLC.

If supplied, the optional 'unixtime' parameter specifies the number of seconds since midnight 1/1/1970, and the function returns a formatted string representing that time. If the 'unixtime' value is not provided, the current time will be returned.

Sample Usage

```
# Return a formatted string
# e.g. Mon, 14 Aug 2006 12:39:01 EST
$str = sys.localtime.format( "%a, %d %b %Y %T EST" );
```

Alternative name: localtime.format

See also: “sys.time()”, “sys.time.seconds(unixtime)”, “sys.time.minutes(unixtime)”, “sys.time.hour(unixtime)”, “sys.time.weekday(unixtime)”, “sys.time.monthday(unixtime)”, “sys.time.month()”, “sys.time.year(unixtime)”, “sys.time.yearday(unixtime)”, “sys.gmtime.format(format, unixtime)”

sys.time.highres()

Returns the current system time as the number of seconds and microseconds since midnight, 1/1/1970. The value is returned as a double, e.g. 1138417190.823265

Sample Usage

```
$time = sys.time.highres();
```

Alternative name: time.highres

See also: “sys.timeToString(unixtime)”, “sys.localtime.format(format, unixtime)”, “sys.gmtime.format(format, unixtime)”, “sys.time()”

sys.time.hour(unixtime)

Returns the hour of the day in local time (0-23).

If optional parameter 'unixtime' is supplied, then this specifies the time since midnight 1/1/1970 otherwise the current time will be used.

Sample Usage

```
$hour = sys.time.hour();
```

Alternative name: time.hour

See also: “sys.time()”, “sys.time.seconds(unixtime)”, “sys.time.minutes(unixtime)”, “sys.time.weekday(unixtime)”, “sys.time.monthday(unixtime)”, “sys.time.yearday(unixtime)”, “sys.time.month()”, “sys.time.year(unixtime)”, “sys.localtime.format(format, unixtime)”

sys.time.minutes(unixtime)

Returns the minutes after the hour in local time (0-59).

If optional parameter 'unixtime' is supplied, then this specifies the time since midnight 1/1/1970 otherwise the current time will be used.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$mins = sys.time.minutes();
```
Alternative name: time.minutes

See also: “sys.time()”, “sys.time.seconds(unixtime)”, “sys.time.hour(unixtime)”, “sys.time.weekday(unixtime)”, “sys.time.monthday(unixtime)”, “sys.time.yearday(unixtime)”, “sys.time.month()”, “sys.time.year(unixtime)”, “sys.localtime.format(format, unixtime)”

sys.time.month()

Returns the month of the year in local time (1-12).

If optional parameter 'unixtime' is supplied, then this specifies the time since midnight 1/1/1970 otherwise the current time will be used.

Sample Usage

```
# Find out what the month is tomorrow.
$month = sys.time.month( sys.time() + 86400);
```
Alternative name: time.month

See also: “sys.time()”, “sys.time.seconds(unixtime)”, “sys.time.minutes(unixtime)”, “sys.time.hour(unixtime)”, “sys.time.weekday(unixtime)”, “sys.time.monthday(unixtime)”, “sys.time.yearday(unixtime)”, “sys.time.year(unixtime)”, “sys.localtime.format(format, unixtime)”

sys.time.monthday(unixtime)

Returns the day of the month in local time (1-31).

If optional parameter 'unixtime' is supplied, then this specifies the time since midnight 1/1/1970 otherwise the current time will be used.

Sample Usage

```
$dayofmonth = sys.time.monthday();
```
Alternative name: time.monthday

See also: “sys.time()”, “sys.time.seconds(unixtime)”, “sys.time.minutes(unixtime)”, “sys.time.hour(unixtime)”, “sys.time.weekday(unixtime)”, “sys.time.yearday(unixtime)”, “sys.time.month()”, “sys.time.year(unixtime)”, “sys.localtime.format(format, unixtime)”

sys.time.seconds(unixtime)

Returns the seconds after the minute in local time. Normally, it returns a number in the range of (0-59), but can be up to 61 to allow for leap seconds.

If optional parameter 'unixtime' is supplied, then this specifies the time since midnight 1/1/1970 otherwise the current time will be used.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$secs = sys.time.seconds();
```
Alternative name: time.seconds

See also: “sys.time()”, “sys.time.minutes(unixtime)”, “sys.time.hour(unixtime)”, “sys.time.weekday(unixtime)”, “sys.time.monthday(unixtime)”, “sys.time.yearday(unixtime)”, “sys.time.month()”, “sys.time.year(unixtime)”, “sys.localtime.format(format, unixtime)”

sys.time.weekday(unixtime)

Returns the day of the week in local time (1-7). Sunday has the value 1; Saturday has the value 7.

If optional parameter 'unixtime' is supplied, then this specifies the time since midnight 1/1/1970 otherwise the current time will be used.

Sample Usage

```
$dayofweek = sys.time.weekday();
```
Alternative name: time.weekday

See also: “sys.time()”, “sys.time.seconds(unixtime)”, “sys.time.minutes(unixtime)”, “sys.time.hour(unixtime)”, “sys.time.monthday(unixtime)”, “sys.time.yearday(unixtime)”, “sys.time.month()”, “sys.time.year(unixtime)”, “sys.localtime.format(format, unixtime)”

sys.time.year(unixtime)

Returns the year in local time (1970-2038).

If optional parameter 'unixtime' is supplied, then this specifies the time since midnight 1/1/1970 otherwise the current time will be used.

Sample Usage

```
$year = sys.time.year();
```
Alternative name: time.year

See also: “sys.time()”, “sys.time.seconds(unixtime)”, “sys.time.minutes(unixtime)”, “sys.time.hour(unixtime)”, “sys.time.weekday(unixtime)”, “sys.time.monthday(unixtime)”, “sys.time.yearday(unixtime)”, “sys.time.month()”, “sys.localtime.format(format, unixtime)”

sys.time.yearday(unixtime)

Returns the day of the year in local time (1-366).

If optional parameter 'unixtime' is supplied, then this specifies the time since midnight 1/1/1970 otherwise the current time will be used.

Sample Usage

```
$dayofyear = sys.time.yearday();
```

© 2019 Pulse Secure, LLC.

Alternative name: time.yearday

See also: "sys.time()", "sys.time.seconds(unixtime)", "sys.time.minutes(unixtime)", "sys.time.hour(unixtime)", "sys.time.weekday(unixtime)", "sys.time.monthday(unixtime)", "sys.time.month()", "sys.time.year(unixtime)", "sys.localtime.format(format, unixtime)"

Traffic Manager Functions

Traffic Manager TrafficScript functions provide the means to inspect, manipulate and direct traffic within the Traffic Manager.

The TrafficScript functions are grouped into several families:

- auth.: Functions to query traffic manager administration user authenticators;
- connection.: These low-level functions allow you to query and manipulate the connection directly;
- connection.data.: These functions allow you to set and query connection-local data;
- counter./counter64.: Provides a simple counter increment mechanism;
- data.: These functions allow you to store and retrieve persistent variables;
- event.: Provides the ability to trigger a custom event;
- geo.: These functions provide various geo-location abilities to the Traffic Manager Multi-Site Cluster Management functionality;
- glb.: Functions to control and fine-tune GSLB (Global Server Load Balancing) decisions;
- http.: These helper functions allow you to query and manipulate HTTP connections easily, without having to parse and interpret the connection directly;
- http.aptimizer.: Provides functions to enable (or bypass) web content optimization on your services;
- http.cache.: These functions give you control over the Web Cache used for caching HTTP content;
- http.compress.: These functions allow you to control whether or not compressible responses are compressed;
- http.request.: These functions can be used to issue HTTP GET or POST requests, and return the result;
- http.stream.: These functions can be used to manipulate streaming data over HTTP;
- java.: Runs a named Java Extension;
- log.: These functions can be used to append messages to the error log file;
- net.dns.: These functions can be used to perform DNS lookups in a TrafficScript rule;
- pool.: These functions are used to select the pool to balance the connection with;
- rate.: Use these functions to select a rate shaping class;
- request.: These low-level functions allow you to query and manipulate the client-side of the connection directly;
- resource.: These functions query external resources;

© 2019 Pulse Secure, LLC.

- response.: These low-level functions allow you to query and manipulate the serverside of the connection directly;
- rtsp.: These functions provide control over Real Time Streaming Protocol (RTSP) traffic;
- rule.: These functions can examine the current rule processing situation;
- sip.: These functions provide control over Session Initiation Protocol (SIP) traffic;
- slm.: These functions are used to query and assign service level monitoring properties to a connection.
- ssl.: These functions are used to query the SSL parameters of encrypted connections;
- tcp.: These functions are used to read from, and write data to, a TCP socket;
- xml.: These functions are used to query, manipulate and validate XML data.

auth.query(authenticator, user, [password])

Queries the named authenticator for information about "user" and, if supplied, checks that "password" matches the password on record for that user. Authenticators can be configured from the "Catalog > Authenticators" page of the Administration Server.

After looking up the user, this function will return a hash with the following values set:

- "OK" - Set to true if the query was successful. Set to false if the user could not be found, if the supplied password was incorrect, or if the authenticator encountered a problem.
- "Error" - Set to an error message if the authenticator encountered a problem, for example if it was unable to contact the authentication server.

Authenticators can be configured to provide information about the supplied user, for example the groups to which they belong. This information will also be contained in the hash returned by the auth.query() function.

Sample Usage

```
# Verify the user's password using an LDAP
# authenticator called 'ldap'
$auth = auth.query( "ldap", $user, $pass );
if( $auth['Error'] ) {
   log.error(
      "Error with authenticator 'ldap': " .
      $auth['Error']
   );
   connection.discard();
} else if( !$auth['OK'] ) {
   # Unauthorised
   http.sendResponse( "403 Permission Denied",
      "text/html", "Incorrect username or password",
      ""
   );
}

# Allow through members of the 'admin' group using
# the 'group' attribute returned by the authenticator
if( $auth['group'] != "admin" ) {
```

```
   http.sendResponse( "403 Permission Denied",
      "text/html",
      "You do not have permission to view this page",
      ""
   );
}
```

Restrictions

Cannot be used in completion rules.

connection.bypassProtocolHandling()

This function instructs the connection to cease any protocol-specific processing, and just forward data between the client and server. If no server has been selected at this point, a load-balancing decision will still be made as per the pool configuration.

After calling this function the rule will terminate and no further rules will be processed on this connection. This feature is not supported with HTTP/2. If this function is used on an HTTP/2 stream, it will instead close the stream with an error indicating that the client should retry using HTTP/1.1.

Sample Usage

```
$upgrade_header = string.lowercase(
   http.getResponseHeader( "Upgrade" ) );
$conn_header = string.lowercase(
   http.getResponseHeader( "Connection" ) );

if( $upgrade_header == "websocket" &&
    $conn_header == "upgrade" ) {
   request.setMaxReplyTime( 300 );
   connection.bypassProtocolHandling();
}
```

See also: “connection.close(Data, [Read])”, “connection.discard()”

Restrictions

Cannot be used in completion rules.

connection.checkLimits([poolname])

This function checks to see if this connection will be queued due to backend connection or transaction limits. The function returns 1 if the connection is within configured maximum limits for the named pool (See the max_connections_pernode and the max_transactions_per_node settings in Pool > Protocol Settings). The function returns 0 if the connection will exceed the configured maximum limits and would be queued.

If the named pool does not exist, your traffic manager will log a warning message and a value of -1 will be returned.

Note that even if this function returns 1, another child process may use the last available slot before this connection is forwarded to a node.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# A connection is queued if it exceeds
# max_connections_pernode or
# max_transactions_per_node.
# Send the client to a low bandwidth overflow pool.
if( !connection.checkLimits( "pool" ) ) {
   pool.use( "hiload_lowbw" );
}
```

Restrictions

Can only be used with TCP-based protocols.

connection.close(Data, [Read])

Writes the provided data directly back to the client. After the data has been sent, the connection is closed.

Use of connection.close is deprecated for HTTP virtual servers. Use http.sendResponse with http.disableClientKeepalive instead.

The optional second argument specifies whether data should continue to be read in from the client after sending this response, and wait for it to close the connection. If set to 0, the connection will close immediately. If non-zero, the traffic manager will wait and read any remaining data from the connection before closing it.

The default behaviour is to wait, because some client software will not read a response until it has sent its entire request.

Sample Usage

```
# Send an instant response and close the connection
connection.close( "Server closing connection\n" );
```

See also: “connection.discard()”, “connection.bypassProtocolHandling()”, “http.sendResponse(code, type, body, headers)”

Restrictions

Cannot be used in completion rules.

connection.discard()

For HTTP/2, stops processing the current request, and leaves the TCP connection established. For other protocols, immediately closes the current connection and stops processing rules. This is equivalent to the function call 'pool.use("discard")'.

Sample Usage

```
# Drop this connection NOW!
connection.discard();
```

See also: “connection.close(Data, [Read])”, “connection.bypassProtocolHandling()”, “http.sendResponse(code, type, body, headers)”

Restrictions

Cannot be used in completion rules.

connection.getBandwidthClass() - deprecated

This function has been deprecated. Use instead "response.getBandwidthClass()".

Returns the current bandwidth class for the connection to the client, or an empty string if no class is set.

connection.getCompletionReasonCode()

Returns a short string code describing the reason a particular transaction completed. A list of the existing codes is given below.

Code	Description
COMPLETE	The transaction completed.
DISCARDED_BY_RULE	The transaction was ended by a TrafficScript rule.
CONFIG_CHANGED	The transaction was purged due to a configuration change.
PARENT_CONNECTION_ENDED	The transaction was purged as its parent connection ended.
CONNECT_FAILURE_SERVER	Could not establish a connection to the server.
CONNECTION_ERROR	There was a problem with the connection.
CONNECTION_CLOSED_SERVER	The connection was closed by the server.
CONNECTION_CLOSED_CLIENT	The connection was closed by the client.
TIMEOUT_NODE_QUEUE	No node was available to handle the connection.
TIMEOUT_PROCESSING_RULE	The connection timed out while processing a TrafficScript rule.
TIMEOUT_PROCESSING_WAX	The connection timed out while optimizing a resource.
TIMEOUT	The connection timed out during client/server communication.
TIMEOUT_CONNECT_CLIENT	The connection timed out before receiving any data from the client.
TIMEOUT_CONNECT_SERVER	The connection timed out before receiving any data from the server.
TIMEOUT_READ_SERVER	The connection timed out waiting for the server to send a response.
TIMEOUT_MAXDURATION	The total transaction duration was exceeded.
TIMEOUT_SIP_MAXDURATION	The total SIP transaction duration was exceeded.
TIMEOUT_SIP_REQUEST	Timed out waiting for a SIP response.
TIMEOUT_UDP_NORESPONSE	Timed out waiting for a response from the back-end node.
TIMEOUT_UDP_INACTIVE	UDP connection was inactive for too long.

© 2019 Pulse Secure, LLC.

Code	Description
SESSION_PERSISTENCE_FAILURE	Failed to find an appropriate node for session persistence.
NODE_LOST	The connection's back-end node was lost.
SOCKET_LOST	The connection's socket was lost.
READ_FAILURE	Failed reading from connection socket.
WRITE_FAILURE	Failed writing to connection socket.
BAD_REQUEST	Bad or malformed request.
BAD_RESPONSE	Bad or malformed response.
SIP_REQUEST_TOO_LARGE	The SIP request entity was too large.
INTERNAL_ERROR	An unexpected failure occurred.

Sample Usage

```
$reasonCode = connection.getCompletionReasonCode();
if ( $reasonCode != "COMPLETE" ) {
   # Erroneous/interrupted transaction.
   recentconns.include();
}
```

See also: "connection.getCompletionReasonInfo()"

connection.getCompletionReasonInfo()

Obtain more detailed information about the reason a particular connection completed. The information is returned in a hash, with the following keys:

Hash key	Details
code	The completion reason code (see connection.getCompletionReasonCode).
message	A detailed message describing the reason code.
iserror	Contains 1 if the transaction ended due to an error.
isservererror	Contains 1 if the transaction ended due to a server error.
isclienterror	Contains 1 if the transaction ended due to a client error.
istimeout	Contains 1 if the transaction ended because it exceeding any timeout duration.
isprocessingtimeout	Contains 1 if the transaction ended because it exceeded rule or optimization processing timeouts.

Sample Usage

```
$info = connection.getCompletionReasonInfo();
if( $info['iserror'] ) {
```

```
   log.info( "Transaction error detected. Code: " .
      $info['code'] . " Message: " .
      $info['message']
   );
}
```

See also: "connection.getCompletionReasonCode()"

connection.getData(count) - deprecated

This function has been deprecated. Use instead "request.get([count])".

Returns the first 'count' bytes of data provided by the client.

Warning: you can stall a connection by asking it to read more data than the remote client will provide. Combine this with connection.getDataLen() to reliably read data from a connection.

connection.getDataLen() - deprecated

This function has been deprecated. Use instead "request.getLength()".

Returns the number of bytes of data already received from the client. This can be combined with connection.getData() to reliably read data from a connection without stalling if no data is available.

connection.getLine(offset) - deprecated

This function has been deprecated. Use instead "request.getLine([regex], [offset])".

Returns a line of input data provided by the client. The line separator is '\n', and this is stripped off before returning the line. connection.getline() takes a single byte-count argument which indicates where to start scanning for a line; a value of '0' begins at the start, so returns the first line.

When connection.getline() returns, the variable $1 is updated to point to the start of the next line in the datastream.

You can iterate through the lines of input data by using $1 as the iterator variable.

Restrictions

Not available when the virtual server's internal protocol is 'generic streaming'.

connection.getLocalIP() - deprecated

This function has been deprecated. Use instead "request.getLocalIP()".

Returns the IP address that the client connected to, i.e. the address local to this machine.

connection.getLocalPort() - deprecated

This function has been deprecated. Use instead "request.getLocalPort()".

© 2019 Pulse Secure, LLC.

Returns the network port number that the client connected to. (e.g. port 80 is normal for a web server)

connection.getMemoryUsage()

Returns an estimate of the amount of memory currently in use for this connection, in bytes. Memory is primarily used for buffering data, and the memory usage can be tuned using the various buffer size settings.

Sample Usage

```
# How much memory are we using?
$memoryusage = connection.getMemoryUsage();
```

connection.getNode()

Returns the name of the back-end node that this request is connected to. If a back-end node has not been chosen, which is normally the case in request rules, it returns the empty string.

Sample Usage

```
# Which node is used for this connection
$nodename = connection.getNode();
```

See also: "connection.getPool()", "connection.getVirtualServer()", "request.avoidNode(node)"

connection.getPersistence()

In a Response rule this function returns the name of the current Session Persistence class used for this connection, or whatever class has been set by a previous use of connection.setPersistence().

Sample Usage

```
$class = connection.getPersistence();
```

See also: "connection.setPersistence(name)"

connection.getPool()

Returns the name of the pool that this request is connected to. If a pool has not been chosen, it returns the empty string.

Sample Usage

```
# Where are we connected to?
$poolname = connection.getPool();
```

See also: "connection.getNode()", "connection.getVirtualServer()"

connection.getProxyClientIP()

Returns the client IP address if one is contained in a connection PROXY header sent by the client. A client IP address is contained only in headers of a TCP or UDP type.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# If the request is from a local address
# then log it
$c_ip = connection.getProxyClientIP();
$subnet = "192.168.0.0/24";
if ( string.ipMaskMatch( $c_ip, $subnet ) ) {
   $full_addr = $c_ip . ":" .
                connection.getProxyClientPort();
   log.info( "Request from " . $full_addr );
}
```

See also: “connection.getProxyHeaderType()”, “connection.getProxyVersion()”, “connection.getProxyClientPort()”, “connection.getProxyServerIP()”, “connection.getProxyServerPort()”, “connection.getProxyTLVData(tlv_type)”

connection.getProxyClientPort()

Returns the client port if one is contained in a connection PROXY header sent by the client. A client port is contained only in headers of a TCP or UDP type.

Sample Usage

```
# If the request is from a local address
# then log it
$c_ip = connection.getProxyClientIP();
$subnet = "192.168.0.0/24";
if ( string.ipMaskMatch( $c_ip, $subnet ) ) {
   $full_addr = $c_ip . ":" .
                connection.getProxyClientPort();
   log.info( "Request from " . $full_addr );
}
```

See also: “connection.getProxyHeaderType()”, “connection.getProxyVersion()”, “connection.getProxyClientIP()”, “connection.getProxyServerIP()”, “connection.getProxyServerPort()”, “connection.getProxyTLVData(tlv_type)”

connection.getProxyHeaderType()

Returns the protocol or type of the connection's PROXY header, if available. This is not necessarily the same as the protocol in use on the connection itself. If PROXY protocol support is enabled, the value returned will be one of "TCP4", "TCP6", "UDP4", "UDP6", "UNIX STREAM", "UNIX DATAGRAM", "LOCAL" or "UNKNOWN". If PROXY protocol support is disabled, the empty string will be returned.

Sample Usage

```
# Pick a pool based on the IP version used
# by the client
$proxytype = connection.getProxyHeaderType();
if ( $proxytype == "TCP6" ) {
   pool.use( "IPv6-pool" );
} else if ( $proxytype == "TCP4" ) {
   pool.use( "IPv4-pool" );
} else if ( $proxytype == "LOCAL" ) {
```


© 2019 Pulse Secure, LLC.

```
   # A health check - end the connection in an
   # orderly manner, as if an empty page has
   # been successfully retrieved
   http.disableClientKeepalive();
   http.sendResponse( "200 OK",
                      "text/plain", "", "" );
} else {
   # Disallow all other client connection types
   connection.discard();
}
```

See also: “connection.getProxyVersion()”, “connection.getProxyClientIP()”, “connection.getProxyClientPort()”, “connection.getProxyServerIP()”, “connection.getProxyServerPort()”, “connection.getProxyTLVData(tlv_type)”

connection.getProxyServerIP()

Returns the server IP address if one is contained in a connection PROXY header sent by the client. A server IP address is contained only in headers of a TCP or UDP type.

Sample Usage

```
# Requests sent to a specific address are
# directed to one particular node
if ( connection.getProxyServerIP() ==
                              "10.18.73.234" &&
     connection.getProxyServerPort() == 80 ) {
   pool.use( "mypool", "10.18.73.1", 8080 );
}
```

See also: “connection.getProxyHeaderType()”, “connection.getProxyVersion()”, “connection.getProxyClientIP()”, “connection.getProxyClientPort()”, “connection.getProxyServerPort()”, “connection.getProxyTLVData(tlv_type)”

connection.getProxyServerPort()

Returns the server port if one is contained in a connection PROXY header sent by the client. A server port is contained only in headers of a TCP or UDP type.

Sample Usage

```
# Requests sent to a specific address are
# directed to one particular node
if ( connection.getProxyServerIP() ==
                              "10.18.73.234" &&
     connection.getProxyServerPort() == 80 ) {
   pool.use( "mypool", "10.18.73.1", 8080 );
}
```

See also: “connection.getProxyHeaderType()”, “connection.getProxyVersion()”, “connection.getProxyClientIP()”, “connection.getProxyClientPort()”, “connection.getProxyServerIP()”, “connection.getProxyTLVData(tlv_type)”

© 2019 Pulse Secure, LLC.

connection.getProxyTLVData(tlv_type)

Returns an array of strings that contain the data from each TLV of the given type that is included in the PROXY protocol version 2 data sent on the connection, if any. This requires that PROXY protocol support is enabled and that a valid PROXY protocol version 2 header was received on the connection. If PROXY protocol support is disabled, no PROXY protocol header was received on the connection, PROXY protocol version 1 was used for the connection, or no TLV of the given type was included in the PROXY protocol version 2 header, then an empty list is returned.

Sample Usage

```
# Reject connections with no "AUTHORITY" TLV,
# i.e. type 2
$authority = connection.getProxyTLVData( 2 );
if ( array.length( $authority ) == 0 ) {
   # No authority TLV means the connection
   # is not authorised
   connection.discard();
} else if ( array.contains( $authority,
                      "badhost.example.com" ) ) {
   # Also disallow the bad host as the authority
   connection.discard();
}
```

See also: “connection.getProxyHeaderType()”, “connection.getProxyVersion()”, “connection.getProxyClientIP()”, “connection.getProxyClientPort()”, “connection.getProxyServerIP()”, “connection.getProxyServerPort()”

connection.getProxyVersion()

Returns the version number of the PROXY protocol in use on the connection, if available. If PROXY protocol support is enabled and a PROXY protocol header was sent by the client, the value returned will be either 1 or 2. If PROXY protocol support is disabled then 0 will be returned.

Sample Usage

```
# Only connections that use PROXY protocol
# version 2 are permitted
$proxyversion = connection.getProxyVersion();
if ( $proxyversion != 2 ) {
   # Either version 1, or no PROXY protocol
   # header was sent
   connection.discard();
}
```

See also: “connection.getProxyHeaderType()”, “connection.getProxyClientIP()”, “connection.getProxyClientPort()”, “connection.getProxyServerIP()”, “connection.getProxyServerPort()”, “connection.getProxyTLVData(tlv_type)”

connection.getRemoteIP() - deprecated

This function has been deprecated. Use instead “request.getRemoteIP()”.

© 2019 Pulse Secure, LLC.

Returns the remote IP address of the client.

connection.getRemotePort() - deprecated

This function has been deprecated. Use instead "request.getRemotePort()".

Returns the remote port of the client's connection.

connection.getServiceLevelClass()

Returns the current service level class for the connection, or an empty string if no class is set.

Sample Usage

```
$class = connection.getServiceLevelClass();
```

See also: "connection.setServiceLevelClass(level)"

connection.getVirtualServer()

Returns the name of the Virtual Server that the rule is running under.

Sample Usage

```
# Are we on the secure site?
if( connection.getVirtualServer() == "secure" ) {
   pool.use( "secure" );
}
```

See also: "connection.getNode()", "connection.getPool()"

connection.setBandwidthClass(name) - deprecated

This function has been deprecated. Use instead "response.setBandwidthClass(name)".

Sets the bandwidth class for the current connection to the client. Providing an empty class name removes the bandwidth class from the connection. It returns zero if an error occurs (for example, if the bandwidth class does not exist), and 1 otherwise.

Restrictions

Cannot be used in completion rules.

connection.setData(request data) - deprecated

This function has been deprecated. Use instead "request.set(request data)".

Replaces the input data read from the client with the provided string.

This is a low-level routine that should be used with care. For protocols with their own higher-level protocol managers (e.g. HTTP), use the higher level routines to alter the input data (e.g. http.setHeader() and http.setBody()).

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

connection.setIdempotent(resend) - deprecated

This function has been deprecated. Use instead "request.setIdempotent(resend)".

Marks a request as resendable or non-resendable.

An idempotent request has no detrimental side effects, so it can safely be attempted multiple times. A non-idempotent request has a side effect - for example, it may update a database, or initiate a purchase.

By default, all non-HTTP requests are marked as idempotent. If a back-end node fails to generate a correct response when a request is initially forwarded to it, your traffic manager will attempt to resend the request to another node.

connection.setIdempotent() can override this behaviour. If 'resend' is zero, this indicates that the request should only be attempted against one back-end node.

If 'resend' has a non-zero value, this indicates that if a request is made to a back-end node and a correct response is not received, your traffic manager should retry the request against another back-end node.

Note that a request cannot be resent once it has begun streaming data between the client and the node. Additionally, UDP connections cannot be marked as resendable (the UDP client application should handle failed UDP responses).

Restrictions

Cannot be used in completion rules.

connection.setPersistence(name)

Sets the Session Persistence class that will be used for the connection. This is used to override the default Session Persistence class that will be used once a Pool is selected.

If no parameter is given then the current Session Persistence class will be removed and the Pool's default session persistence class will be used for this connection.

Sample Usage

```
connection.setPersistence( "sales" );
```

See also: "connection.getPersistence()", "connection.setPersistenceKey(value)"

Restrictions

Cannot be used in completion rules.

connection.setPersistenceKey(value)

Sets the value of the Session Persistence key that is used by a Universal Session Persistence type class.

© 2019 Pulse Secure, LLC.

Setting the value to the empty string will remove any persistence key from the connection.

A Session Persistence class that uses Universal Session Persistence attempts to ensure that every connection that provides the same key is directed to the same back-end node.

This function has no effect if a different type of session persistence class is ultimately used.

Sample Usage

```
$value = http.getHeader( "User-Agent" ) .
         request.getRemoteIP();
connection.setPersistenceKey( $value );
connection.setPersistence( "my persistence class" );
```

See also: "connection.setPersistence(name)"

Restrictions

Cannot be used in completion rules.

connection.setPersistenceNode(value)

Sets the back-end node to be used by a NamedNode Persistence class.

A Session Persistence class that uses NamedNode Persistence will then ensure that this node will be used for the request. The node must be valid and exist in the Pool being used. If no port number is given, or if the port number is not valid, then if there is a node with a matching name, it will be used. For example, if the node 'web:80' is specified, but there is only a 'web:443', then that node will be used instead. This is to help share session persistence between different services on the same machine.

This function has no effect if a different type of session persistence class is ultimately used.

Sample Usage

```
# Use the node 'web:80' for this request
connection.setPersistenceNode( "web:80" );
```

See also: "connection.setPersistence(name)"

Restrictions

Cannot be used in completion rules.

connection.setServiceLevelClass(level)

Sets the service level class for the current connection. It returns zero if an error occurs (for example, if the service level class does not exist), and 1 otherwise

Sample Usage

```
connection.setServiceLevelClass( "gold" );
```

See also: "connection.getServiceLevelClass()"

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

connection.sleep(milliseconds)

Pauses processing of the current connection for the specified number of milliseconds. This can be used to rate-limit particular clients; for example, those asking for particular files, or from particular locations, or using particular user-agents.

Sample Usage

```
# Pause this connection for 2 seconds
connection.sleep( 2000 );
```

Restrictions

Cannot be used in completion rules.

connection.data.get(key)

Returns the value that was previously stored with the given key using connection.data.set() in the current connection, or returns the empty string if no data was stored.

Sample Usage

```
# Track the state of this connection so that we
# can process the response correctly
$req = request.get( 5 );
if( string.startsWith( $req, "LOGIN" ) ) {
   connection.data.set( "state", "login" );
} else {
   connection.data.set( "state", "" );
}

# ...
# in the response, ...
$state = connection.data.get( "state" );
```

See also: “connection.data.set(key, value)”, “data.get(key)”

connection.data.set(key, value)

Stores a value for this connection, associating it with the provided key. The value can be retrieved later when processing the same connection, using connection.data.get(). Once the connection finishes, the value cannot be retrieved.

Sample Usage

```
# Track the state of this connection so that we
# can process the response correctly
$req = request.get( 5 );
if( string.startsWith( $req, "LOGIN" ) ) {
```


© 2019 Pulse Secure, LLC.

```
   connection.data.set( "state", "login" );
} else {
   connection.data.set( "state", "" );
}

# ...
# in the response, ...
$state = connection.data.get( "state" );
```

See also: “connection.data.get(key)”, “data.set(key, value)”

Restrictions

Cannot be used in completion rules.

counter.increment(counter, [amount])

Increments the numbered counter. These counters are readable via SNMP, and can be graphed on the Current Activity page on the Administration Server.

By default, the counter is incremented by one, but you can also supply an amount to increment the counter by. If a negative amount is supplied the counter is decremented, but please keep in mind that a decrement is indiscernible from a wrap-around.

Sample Usage

```
# Increment the first user counter by 100
counter.increment( 1, 100 );
# Increment the first user counter by 1
counter.increment( 1 );
```

See also: “counter64.increment(counter, [amount])”

counter64.increment(counter, [amount])

Increments the numbered 64 bit counter. These counters are readable via SNMP, and can be graphed on the Current Activity page on the Administration Server. 64 bit counters exist in a separate name- and storage space from the 32 bit counters, i.e. they don't overlap in any way.

By default, the counter is incremented by one, but you can also supply an amount to increment the counter by. If a negative amount is supplied the counter is decremented, but please keep in mind that a decrement is indiscernible from a wrap-around. Also please note that the supplied increment amount is limited to signed 32 bit range even though the counter is 64 bit.

Sample Usage

```
# Increment the first 64 bit user counter by 100
counter64.increment( 1, 100 );
# Increment the first 64 bit user counter by 1
counter64.increment( 1 );
```

See also: “counter.increment(counter, [amount])”

© 2019 Pulse Secure, LLC.

data.get(key)

Returns the value that was previously stored with the given key using data.set(), or returns the empty string if no data was stored.

Values stored in this way are persistent; a value stored in one rule can later be retrieved by a different rule handling a different connection. Thus, a rule can maintain persistent state across connections.

Sample Usage

```
data.set( "count", 7 );

# In another rule or connection...
$value = data.get( "count" ); # Returns 7
```

See also: “data.set(key, value)”, “data.remove(key)”, “data.getMemoryUsage()”

data.getMemoryFree()

Returns the amount of free space, in bytes, available for storing information with data.set().

If memory space is low, then data.reset() can be used to clear some or all of the entries from the storage. Alternatively, the upper limit on memory can be configured using trafficscript!data_size on the Global Settings page.

Sample Usage

```
# If running low on storage, clear some temporary
# data that other rules have stored.
$bytes = data.getMemoryFree();
if( $bytes < 1024 ) {
   data.reset( "temp-" );
}
```

See also: “data.set(key, value)”, “data.reset([prefix])”, “data.getMemoryUsage()”

data.getMemoryUsage()

Returns an estimate of the amount of memory, in bytes, used by entries that have been stored by data.set().

This can be used to verify if a rule is storing excessive amounts of data, starving the host machine of memory.

Sample Usage

```
$bytes = data.getMemoryUsage();
```

See also: “data.set(key, value)”, “data.reset([prefix])”, “data.getMemoryFree()”

data.remove(key)

Removes the value that was previously associated with the given key using data.set().

data.remove() returns 1 if the item did exist, or 0 if it was not found.

© 2019 Pulse Secure, LLC.

Sample Usage

```
data.set( "cache-$url", $obj );

# In another rule or connection...
data.remove( "cache-$url" );
```

See also: “data.set(key, value)”, “data.get(key)”, “data.reset([prefix])”

data.reset([prefix])

Removes some or all of the mappings created by data.set(). With no arguments, it removes all keys. With a single argument, it removes all keys that begin with the supplied string.

NOTE: Should you intend to delete a specific individual key, data.remove() is more suited to this task. It offers greater performance and also ensures there is no risk of accidentally deleting other keys that start with the same string.

Sample Usage

```
# Free some memory if we've used too much
if( data.getMemoryUsage() > 102400 ) {
   data.reset( "mappings-" );
}
```

See also: “data.set(key, value)”, “data.remove(key)”, “data.getMemoryUsage()”, “data.getMemoryFree()”

data.set(key, value)

Stores a value, associating it with the provided key. The value can be retrieved later using data.get(), even from a different rule or connection.

The value will be stored as a string, implicit conversion of floating point numbers to strings can cause some precision loss. You can convert a floating point number into a string with no precision loss using 'string.sprintf("%f", $val)'. Arrays and hashes are serialised before storing and will be deserialised into their original form when retrieved with data.get().

To prevent memory problems, there is an upper limit on the amount of data that can be stored in the TrafficScript data storage. This means that the data.set() may fail. The upper limit can be configured using trafficscript!data_size on the Global Settings page.

data.set() returns true if the entry was stored, or false if there was no room.

Sample Usage

```
# Associate $value with $key for future retrieval
data.set( $key, $value );

# Run this code only once:
if( !data.get( "runonce" ) ) {
   # Do initialization
   # ...
   data.set( "runonce", 1 );
}
```

See also: "data.get(key)", "data.remove(key)", "data.getMemoryUsage()"

data.local.get(key)

Returns the value that was previously stored with the given key using data.local.set(), or returns the empty string if no data was stored.

Values stored in this way are persistent; a value stored in one rule can later be retrieved by a different rule handling a different connection as long as it is handled by the same child process. Thus, a rule can maintain persistent state across connections. The value can be different or non-existent on another child process.

data.local.get() and data.local.set() are useful for caching data where coherence is not required.

Sample Usage

```
data.local.set( "count", 7 );

# In another rule or connection handled
# by the same child process ...
$value = data.local.get( "count" ); # Returns 7
```

See also: "data.local.set(key, value)", "data.local.remove(key)", "data.local.getMemoryUsage()"

data.local.getMemoryFree()

Returns the amount of free space, in bytes, available for storing information with data.local.set().

If memory space is low, then data.local.reset() can be used to clear some or all of the entries from the storage. Alternatively, the upper limit on memory can be configured using trafficscript!data_local_size on the Global Settings page.

Sample Usage

```
# If running low on storage, clear some temporary
# data that other rules have stored.
$bytes = data.local.getMemoryFree();
if( $bytes < 1024 ) {
   data.local.reset( "temp-" );
}
```

See also: "data.local.set(key, value)", "data.local.reset([prefix])", "data.local.getMemoryUsage()"

data.local.getMemoryUsage()

Returns an estimate of the amount of memory, in bytes, used by entries that have been stored by data.local.set().

This can be used to verify if a rule is storing excessive amounts of data, starving the host machine of memory.

Sample Usage

```
$bytes = data.local.getMemoryUsage();
```

See also: "data.local.set(key, value)", "data.local.reset([prefix])", "data.local.getMemoryFree()"

© 2019 Pulse Secure, LLC.

data.local.remove(key)

Removes the value that was previously associated with the given key using data.local.set().

data.local.remove() returns 1 if the item did exist, or 0 if it was not found.

Sample Usage

```
data.local.set( "cache-$url", $obj );

# In another rule or connection...
data.local.remove( "cache-$url" );
```

See also: "data.local.set(key, value)", "data.local.get(key)", "data.local.reset([prefix])"

data.local.reset([prefix])

Removes some or all of the mappings created by data.local.set(). With no arguments, it removes all keys. With a single argument, it removes all keys that begin with the supplied string.

NOTE: Should you intend to delete a specific individual key, data.local.remove() is more suited to this task. It offers greater performance and also ensures there is no risk of accidentally deleting other keys that start with the same string.

Sample Usage

```
# Free some memory if we've used too much
if( data.local.getMemoryUsage() > 102400 ) {
   data.local.reset( "mappings-" );
}
```

See also: "data.local.set(key, value)", "data.local.remove(key)", "data.local.getMemoryUsage()", "data.local.getMemoryFree()"

data.local.set(key, value)

Stores a value, associating it with the provided key. The value can be retrieved later using data.local.get(), even from a different rule or connection processed by the same child process.

The value will be stored as a string, implicit conversion of floating point numbers to strings can cause some precision loss. You can convert a floating point number into a string with no precision loss using 'string.sprintf("%f", $val)'. Arrays and hashes are serialised before storing and will be deserialised into their original form when retrieved with data.local.get().

To prevent memory problems, there is an upper limit on the amount of data that can be stored in the local TrafficScript data storage. This means that the data.local.set() may fail. The upper limit can be configured using trafficscript!data_local_size on the Global Settings page.

data.local.set() returns true if the entry was stored, or false if there was no room.

Sample Usage

```
# Associate $value with $key for future retrieval
```

```
data.local.set( $key, $value );

# Run this code only once:
if( !data.local.get( "runonce" ) ) {
   # Do initialization
   # ...
   data.local.set( "runonce", 1 );
}
```

See also: “data.local.get(key)”, “data.local.remove(key)”, “data.local.getMemoryUsage()”

event.emit(custom event name, message)

Triggers a Custom Event identified by the 'custom event name'. Actions can be associated with the Custom Event by configuring an Event Type to contain a Custom Event with the specified 'custom event name', and then associating that Event Type with an Action.

The 'custom event name' cannot contain '/' or control codes.

In addition to custom actions, a log message will be produced containing the eventid and the message.

Sample Usage

```
# Emit a debug statement.
event.emit( "debug1", "Some debug information" );
```

See also: “log.info(message)”, “log.warn(message)”, “log.error(message)”

geo.getCity(ip)

Returns the city of the supplied IPv4 address, or the empty string.

Sample Usage

```
# Get this IP's city, such as Santa Clara
$city = geo.getCity( "216.250.81.96" );
```

geo.getCountry(ip)

Returns the country name of the supplied IPv4 address, or the empty string.

Sample Usage

```
# Get this IP's country, such as United States
$country = geo.getCountry( "216.250.81.96" );
```

geo.getCountryCode(ip)

Returns the two-character country code of the supplied IPv4 address, or the empty string.

Sample Usage

```
# Get this IP's country code, such as US
$countryCode = geo.getCountryCode( "216.250.81.96" );
```


© 2019 Pulse Secure, LLC.

geo.getDistanceKM(lat1, lon1, lat2, lon2)

Returns the distance in kilometres between two points on the earth's surface (identified by latitude and longitude), or -1 on error.

Sample Usage

```
# Get the distance between two lat-long points in km
$d = geo.getDistanceKM( "52.2338", "0.1529",
                        "37.4062", "-121.9765" );
```

geo.getDistanceMiles(lat1, lon1, lat2, lon2)

Returns the distance in miles between two points on the earth's surface (identified by latitude and longitude), or -1 on error.

Sample Usage

```
# Get the miles between two lat-long points
$d = geo.getDistanceMiles( "52.2338", "0.1529",
                           "37.4062", "-121.9765" );
```

geo.getIPDistanceKM(ip1, ip2)

Returns the distance in kilometres between the locations of two IPv4 addresses. It will return -1 unless both locations can be found.

Sample Usage

```
# Get the distance between two IPs in kilometres
$d = geo.getIPDistanceKM( "11.12.13.14",
                          "25.26.27.28" );
```

geo.getIPDistanceMiles(ip1, ip2)

Returns the distance in miles between the locations of two IPv4 addresses. It will return -1 unless both locations can be found.

Sample Usage

```
# Get the distance between two IPs in miles
$d = geo.getIPDistanceMiles( "1.2.3.4", "5.6.7.8" );
```

geo.getLatitude(ip)

Returns the decimal latitude of the supplied IPv4 address, or the empty string if the location is unknown. This may be accurate to city, region or only country level: to find out, check whether geo.getCity() and geo.getRegion() return empty strings.

Sample Usage

```
# Get this IP's decimal latitude (positive = north),
```

```
# such as 37.3961
$lat = geo.getLatitude( "216.250.81.96" );
```

geo.getLocation()

Returns the name of the location in which this traffic manager is based.

Note that traffic managers can be assigned to locations only when vTM Multi-Site Cluster Management is enabled.

Sample Usage

```
# Run this part of the rule if the request was
# sent to a traffic manager located in Cambridge
if( geo.getLocation() == "Cambridge" ) {
   # ...
}
```

See also: "geo.getLocationLonLat()"

geo.getLocationLonLat()

Returns a hash containing the longitude and latitude of the location to which this request was sent.

Note that traffic managers can be assigned to locations only when vTM Multi-Site Cluster Management is enabled and locations can be assigned a latitude and longitude only when using the Global Load Balancing feature.

Sample Usage

```
# How for away from the traffic manager did this
# request originate from?
$loc = geo.getLocationLonLat();
$ip = request.getRemoteIP();
$reqlon = geo.getLongitude( $ip );
$reqlat = geo.getLatitude( $ip );
$reqloc = geo.getCountry( $ip );
log.info( "Request was sent from " .
          $reqloc . ", " .
          geo.getDistanceMiles( $loc["latitude"],
                                $loc["longitude"],
                                $reqlat, $reqlon ) .
          " miles away from here" );
```

See also: "geo.getLocation()"

geo.getLongitude(ip)

Returns the decimal longitude of the supplied IPv4 address, or the empty string if the location is unknown. This may be accurate to city, region or only country level: to find out, check whether geo.getCity() and geo.getRegion() return empty strings.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Get this IP's decimal longitude (positive = east),
# such as -121.962
$long = geo.getLongitude( "216.250.81.96" );
```

geo.getRegion(ip)

Returns the region (e.g. US state) of the supplied IPv4 address, or the empty string if the location is unknown or doesn't have a region.

Sample Usage

```
# Get this IP's region, such as California
$state = geo.getRegion( "216.250.81.96" );
```

geo.getRegionCode(ip)

Returns the region code (e.g. US state abbreviation) of the supplied IPv4 address, or the empty string. The code for a given region may differ between software versions.

Sample Usage

```
# Get this IP's region code, such as CA
$stateCode = geo.getRegionCode( "64.69.78.223" );
```

glb.getDomain()

Get the domain currently being processed.

This function can only be used in GLB service rules.

Sample Usage

```
# Disable balancing of the example.com domain
# if the client is in Boston, USA
$domain = glb.getDomain();
$ip = request.getRemoteIP();
if( $domain == "example.com"
      && geo.getCity( $ip ) == "Boston"
      && geo.getCountryCode( $ip ) == "US" ) {
   glb.service.skip();
}
```

glb.getIPs()

Get an array of the IPs associated with the domain currently being processed.

This function can only be used in GLB service rules.

Sample Usage

```
# Log the IPs that were returned by the DNS server
```

```
# for the domain test.example.com
if( glb.getDomain() == "test.example.com" ) {
   $msg = "IPs returned to " . request.getRemoteIP()
        . " for domain test.example.com were: "
        . array.join( glb.getIPs(), ", " );
   log.info( $msg );
}
```

glb.service.avoidLocation(location)

Avoid the specified location when re-writing the DNS response for the current domain.

The effect of this is to narrow down the list of available locations to just those that are not avoided.

If a location is both preferred and avoided being preferred will take precedence. If all locations are avoided (and none preferred) the outcome of the balancing algorithm will be as if no preferences have been specified.

If the location is unknown, unconfigured, or dead this preference will have no effect on the balancing decision. (In these cases a warning event will be raised and logged.)

This function can only be used in GLB service rules.

Sample Usage

```
# Do not use the UK between 01:00 and 03:00 GMT if
# possible to use another location
if( sys.gmtime.format( '%H' ) >= 1
      && sys.gmtime.format( '%H' ) < 3 ) {
   glb.service.avoidLocation( 'United Kingdom' );
}
```

Restrictions

Cannot be used in completion rules.

glb.service.getLocationWeight(location, [service])

Get the weight of the specified location, as configured for the Weighted Random load balancing algorithm for the specified service, or for the current service if no service is specified and the function is used in a GLB service rule.

If the service is not using the Weighted Random load balancing algorithm then the weight for all locations will be reported as 0.

If the location does not exist, or is not configured for the specified service, the function will return -1 and will print a warning to the event log.

A service must be specified if this function is used outside a GLB service rule.

Sample Usage

```
# If debugging, never use the lowest weighted node
#  - instead, keep it free for testing purposes
```


© 2019 Pulse Secure, LLC.

```
if( $debugging ) {
   # Build a list of weight -> location mappings
   $locs = glb.service.listLocations();
   $weights = [];
   foreach( $loc in $locs ) {
      $w = glb.service.getLocationWeight( $loc );
      $weights[$w] = $loc;
   }
   # Ignore the lowest weighted location
   $lowest = array.sortNumerical(
       hash.keys( $weights )
   )[0];
   glb.service.ignoreLocation( $weights[$lowest] );
}
```

glb.service.getName()

Get the name of the current GLB service configuration.

This is the service that has been picked to handle the current domain. It is this service that will have specified that the rule be executed.

This function can only be used in GLB service rules.

Sample Usage

```
# Disable the service named MyService on Sundays
$name = glb.service.getName();
if( $name == "MyService"
      && sys.time.weekday() == 1 ) {
   glb.service.skip();
}
```

glb.service.getNearestLocation()

Get the name of the traffic manager location calculated to be the closest to geographic location the client IP. Note that only live traffic manager locations are considered in this calculation.

If no nearest traffic manager location could be calculated the empty string is returned.

This function can only be used in GLB service rules.

Sample Usage

```
# Log the traffic manager location that is closest
# to the client
$nearest = glb.service.getNearestLocation();
log.info( "Location nearest to " .
   request.getRemoteIP() . " is " . $nearest );
```

glb.service.ignoreLocation(location)

Do not consider the specified location when re-writing the DNS response for the current domain.

This removes the specified location from the list of available locations.

Multiple locations can be ignored, if all available locations are ignored it will be as if all locations are dead. Ignoring locations takes precedence over preferring locations.

If the location is unknown, unconfigured, or dead this preference will have no effect on the balancing decision. (In these cases a warning event will be raised and logged.)

This function can only be used in GLB service rules.

Sample Usage

```
# Never use the UK between 01:00 and 03:00 GMT
if( sys.gmtime.format( '%H' ) >= 1
      && sys.gmtime.format( '%H' ) < 3 ) {
   glb.service.ignoreLocation( 'United Kingdom' );
}
```

Restrictions

Cannot be used in completion rules.

glb.service.isLocationLive(location, [service])

Check if a location is live. A non-live location includes those for which monitors have failed, or those that are in the "draining" list for the specified service, or for the current service if no service is specified and the function is used in a GLB service rule.

The function returns 1 if the location is live, 0 if it isn't, and -1 if the location does not exist or is not configured for the current service.

A service must be specified if this function is used outside a GLB service rule.

Sample Usage

```
# Log warning if first location is dead
$locs = glb.service.listLocations();
$status = glb.service.isLocationLive($locs[0]);
if( $status == 0 ) {
   log.warn( "Location " . $locs[0] . " is dead." );
}
```

glb.service.listDomains([service])

Get an array of the domains the specified GLB service is configured to handle, or the domains that the current GLB service is configured to handle if no service is specified.

This function can only be used in GLB service rules unless the service parameter is specified.

Sample Usage

```
# Log the domains the current GLB service is
# configured to handle.
```


© 2019 Pulse Secure, LLC.

```
$domains = glb.service.listDomains();
log.info( "Configured domains: " .
   array.join( $domains, ", " ) );
```

glb.service.listLocations([service])

Get a priority-ordered array of the locations configured for the specified GLB service, or the current GLB service if none is specified. This is the value of the "location_order" config key.

This function can only be used in GLB service rules unless the service parameter is specified.

Sample Usage

```
# Apply a custom global load-balancing algorithm
# without accessing a back-end DNS server, but
# using stored configuration on the traffic manager.
# Rule is used on a standard DNS virtual server.
$locations = glb.service.listLocations( "Gallery" );
foreach( $loc in $locations ) {
   if( glb.service.isLocationLive(
                              $loc, "Gallery" ) ) {
      # Return DNS response with IP of location
   }
}
# Return DNS response to fallback IP - all locations
# are down
```

glb.service.preferLocation(location)

Prefer the specified location when re-writing the DNS response for the current domain. More than one location may be preferred by calling this function multiple times.

The effect of this is to narrow down the list of available locations to just those that have been selected as preferred. However, if no preferred locations are available balancing will fall back to other available locations.

If a location is both preferred and avoided being preferred will take precedence. If all locations are preferred the outcome of the balancing algorithm will be as if no preferences have been specified.

If the location is unknown, unconfigured, or dead this preference will have no effect on the balancing decision. (In these cases a warning event will be raised and logged.)

This function can only be used in GLB service rules.

Sample Usage

```
# Prefer to use Australia between 01:00 and 03:00 GMT
# But if it isn't available fall back to balancing
if( sys.gmtime.format( '%H' ) >= 1
      && sys.gmtime.format( '%H' ) < 3 ) {
   glb.service.preferLocation('Australia');
}
```

Restrictions

Cannot be used in completion rules.

glb.service.skip()

Skip the current service configuration. This means that no futher service rules will be executed and no balancing will occur for the domain currently being processed. The DNS records for the domain will be sent to the client unchanged. (Further A-record domains in the current DNS response may still be processed.)

The glb.service.skip() call does not return.

This function can only be used in GLB service rules.

Sample Usage

```
# Do not balance requests from the local network
$ip = request.getRemoteIP();
if( string.IPMaskMatch( $ip, "10.0.0.0/16" ) ) {
   glb.service.skip();
}
```

Restrictions

Cannot be used in completion rules.

glb.service.useLocation(location)

Force use of the specified location when re-writing the DNS response for the current domain.

If the location has IPs configured for the current service that match IPs returned in the DNS response for the current domain the response will be re-written to use only the IPs configured for the location.

If the location is unknown, unconfigured, or dead this preference will have no effect on the balancing decision. (In these cases a warning event will be raised and logged.)

This function can only be used in GLB service rules.

Sample Usage

```
# Force use of Australia between 01:00 and 03:00 GMT
if( sys.gmtime.format( '%H' ) >= 1
      && sys.gmtime.format( '%H' ) < 3 ) {
   glb.service.useLocation( 'Australia' );
}
```

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

http.addHeader(name, value)

Modifies the current HTTP request, adding an HTTP header with the supplied value. A case-insensitive lookup is first performed in order to find any existing headers. If a match is found, a duplicate header will be added to the message along with the new value.

If an invalid header name is specified, the function will print a warning and return without modifying the HTTP request.

Sample Usage

```
# Add a host header if it is missing
if( !http.headerExists( "Host" ) ) {
   http.addHeader( "Host", "unknown" );
}
```

See also: “http.setHeader(name, value)”, “http.getHeader(name)”, “http.removeHeader(name)”, “http.headerExists(name)”, “http.addResponseHeader(name, value)”

http.addResponseHeader(name, value)

Adds an HTTP header to the HTTP response that will be sent back to the client. If the header already exists in the response, then this value will be appended to the existing value. The header lookup is case-insensitive.

If an invalid header name is specified, the function will print a warning and return without modifying the HTTP response.

Sample Usage

```
# Set a cookie to remember this user
http.addResponseHeader( "Set-Cookie",
   "id=12345678; path=/" );
```

See also: “http.setResponseHeader(name, value)”, “http.getResponseHeader(name)”, “http.removeResponseHeader(name)”, “http.addHeader(name, value)”

http.changeSite(name)

Redirect users to a new website. It is a more sophisticated version of http.redirect() that will preserve the original path that the request asked for. Note that it sends back a HTTP 301 redirect rather than the 302 response returned by http.redirect(). For instance, if the original request was for "http://www.example.com/image/image.jpg", then http.changeSite("example.co.uk") would redirect the user to "http://example.co.uk/image/image.jpg"

The redirection will preserve the original path (and any query string) of the request, together with the port number and protocol. If you wish to force any of these details, then you can specify them as part of the supplied host name. e.g. http.changeSite("https://www.example.com") will always send people to a SSL-encrypted site.

You can also add on a prefix to the path of the URL, e.g. http.changeSite("www.example.com/oldsite") would redirect a request for "http://www.example.com/missing/page.html" to "http://www.example.com/oldsite/missing/page.html".

© 2019 Pulse Secure, LLC.

If the original request matches the supplied redirection, then http.changeSite() will take no action and let the request continue. This ensures that no 'redirection loops' occur.

Sample Usage

```
# Send users to our new site
if( http.getHostHeader() == "www.zeus.com" ) {
   http.changeSite( "www.example.com" );
}
```

See also: “http.sendResponse(code, type, body, headers)”, “http.redirect(address)”

Restrictions

Cannot be used in completion rules.

http.cookie(name) - deprecated

This function has been deprecated. Use instead “http.getCookie(name)”.

Returns the value of the named cookie.

http.disableClientKeepalive()

Overrides the 'keepalive' setting on the virtual server for the current request by adding a 'Connection: close' header to the response.

Sample Usage

```
# Refuse permission and ensure that the connection
# will not be kept alive.
http.disableClientKeepalive();
http.sendResponse( "403 Permission Denied",
                   "text/html", "Go away",
                   "" );
```

See also: “http.getHeader(name)”

Restrictions

Cannot be used in completion rules.

http.doesFormParamExist(Parameter)

Checks whether a form parameter is present, either in the URL query string, or if not found and the request is a POST, from the POST body data. It returns 1 if the parameter is present, and 0 if not.

This is useful when there are form parameters with no value, e.g a query string like 'foo=bar&stuff&x=y' - the 'stuff' parameter has no value, but is present.

Sample Usage

```
# Did the user ask for fries with that?
if( http.doesFormParamExist( "fries" ) ) {
```


© 2019 Pulse Secure, LLC.

```
   # Do something
}
```

See also: “http.getFormParam(Parameter, [Separator])”, “http.listFormParamNames()”, “http.getFormParams()”

http.getBody([count])

Returns the body data of the request. HTTP clients use the body data for sending file uploads or for HTML form parameters.

If the optional 'count' parameter is supplied, http.getBody() will only read and return this number of bytes. If count is 0, http.getBody() returns the entire request.

If the request has no body, then this returns an empty string.

http.getBody() will work in a response rule only if the entire request was read before the request rules completed (for example, if another call to http.getBody() was made in a request rule). Otherwise, request data will have been streamed to the server and not cached. If the request data is not available, this function will return the empty string.

This function is not usable in response rules, as the body data of the request will no longer be accessible.

To read HTML form parameters, it is easier to use http.getFormParam() as that will work for GET and POST requests.

Sample Usage

```
# Read the entire request body
$body = http.getBody();
```

See also: “http.getBodyLines([count])”, “http.setBody(body)”, “http.getResponseBody([count])”, “http.getFormParams()”, “http.listFormParamNames()”

Restrictions

Cannot be used in completion rules.

http.getBodyLines([count])

Splits the body data of the HTTP request into individual lines and returns an array of the data.

If the optional 'count' parameter is supplied, http.getBodyLines() will only read and return this number of bytes. If count is 0, http.getBodyLines() returns the entire request.

If the request has no body, then this returns an empty array. This function is not usable in response rules, as the body data of the request will no longer be accessible.

To read HTML form parameters, it is easier to use http.getFormParams() as that will work for GET and POST requests.

Sample Usage

```
# Read the entire request body
$body = http.getBodyLines();

# If the body data is some special format we can
#  understand then we can process it line-by-line
if( array.length( $body ) > 0 ) {
   if( array.shift( $body ) == "Special data" ) {
      foreach( $line in $body ) {
         # handle special body data lines
      }
   }
}
```

See also: "http.getBody([count])", "http.getResponseBodyLines([count])"

Restrictions

Cannot be used in completion rules.

http.getClientVersion()

Returns the HTTP version in use by the client.

This function differs from http.getVersion() in that it returns the version of the HTTP request sent by the client, rather than that of the HTTP request the traffic manager will send to the back-end server.

As an example, if the client is using HTTP/2 but the traffic manager has converted the request to HTTP/1.1 before processing it, http.getClientVersion() will return "HTTP/2" whereas http.getVersion() will return "HTTP/1.1".

Sample Usage

```
if ( http.getClientVersion() == "HTTP/2" ) {
   log.info( "HTTP/2 request detected." );
   recentconns.include();
}
```

See also: "http.getVersion()", "http.getResponseVersion()"

http.getCookie(name)

Returns the named cookie in the incoming HTTP request.

http.getCookie is a helper method that makes it easier to parse the HTTP Cookie header and extract the values of that particular cookie, rather than using http.getHeader() directly.

An HTTP 'Cookie' header can have multiple values, such as

Cookie: user-id=Joe; user-type=gold

http.getCookie(...) will retrieve the 'Cookie' header line and parse it, returning the value of the cookie. If the cookie does not exist, http.getCookie() will return the empty string.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Get the PHP session cookie
$cookie = http.getCookie( "PHPSESSIONID" );
```

See also: “http.getCookies()”, “http.setCookie(name, value)”, “http.removeCookie(name)”, “http.getResponseCookie(name)”

http.getCookies()

Returns a hash containing the names of all the cookies in this request mapped to their values.

An HTTP 'Cookie' header can have multiple values, such as

Cookie: user-id=Joe; user-type=gold

http.getCookies() will retrieve the 'Cookie' header line and parse it, returning a hash mapping all the cookie names to their values.

Sample Usage

```
# List all of the cookie names and their values
$cookies = http.getCookies();
foreach( $cookie in hash.keys( $cookies ) ) {
   log.info( $cookie . ": " . $cookies[$cookie] );
}
```

See also: “http.getCookie(name)”, “http.setCookie(name, value)”, “http.removeCookie(name)”, “http.getResponseCookies()”

http.getFormParam(Parameter, [Separator])

Returns the %-decoded form parameter from the URL query string, or if not found and the request is a POST, from the POST body data.

If the parameter is provided twice, only the first will be returned, unless the optional separator is provided, in which case all matches will be returned, separated with this string.

Sample Usage

```
# See what drink the user wanted
$drink = http.getFormParam( "drink" );
```

See also: “http.setQueryString(querystring)”, “http.getRawURL()”, “http.getBody([count])”, “http.listFormParamNames()”, “http.getFormParams()”

http.getFormParamNames(Separator) - deprecated

This function has been deprecated. Use instead “http.listFormParamNames()”.

Returns a list containing the names of all the form parameters present in the URL query string and, if the request is a POST, in the POST body data. The names are returned as a single string, separated by the string supplied to this function. If the same parameter appears multiple times in the request, it will only appear once in the list returned by this function.

© 2019 Pulse Secure, LLC.

http.getFormParams()

Returns a hash mapping the names of all the form parameters present in the URL query string and, if the request is a POST, in the POST body data to their values. If the same parameter appears multiple times in the request then it will be mapped to an array of values in the hash that is returned by this function.

Sample Usage

```
# Log all of the form parameters and values
$params = http.getFormParams();
foreach( $param in hash.keys( $params ) ) {
   $values = $params[$param];
   if( lang.isArray( $values ) ) {
      log.info($param . "=" .
               array.join( $values, ", " )
              );
   } else {
      log.info($param . "=" . $values);
   }
}
```

See also: “http.getFormParam(Parameter, [Separator])”, “http.doesFormParamExist(Parameter)”, “http.listFormParamNames()”

http.getHeader(name)

Returns the value of a named HTTP header in the HTTP request, or the empty string if the header does not exist or has an empty value. The lookup is case-insensitive.

Sample Usage

```
# Get the browser name and version
$browser = http.getHeader( "User-Agent" );

# this returns the same value
$browser = http.getHeader( "user-agent" );
```

See also: “http.getHeaders()”, “http.setHeader(name, value)”, “http.addHeader(name, value)”, “http.removeHeader(name)”, “http.headerExists(name)”, “http.getResponseHeader(name)”, “http.getHostHeader()”, “http.getHeaders()”

http.getHeaderNames() - deprecated

This function has been deprecated. Use instead “http.listHeaderNames()”.

Returns a list of all the headers that are present in the request.

The headers are returned as a single string, separated by spaces.

http.getHeaders()

Returns a hash containing all the header names in the request mapped to their values.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Show all the headers in the request
$headers = http.getHeaders();

foreach( $header in hash.keys( $headers ) ) {
   log.info( $header . "=" . $headers[$header] );
}
```

See also: “http.getHeader(name)”, “http.setHeader(name, value)”, “http.addHeader(name, value)”, “http.removeHeader(name)”, “http.headerExists(name)”, “http.getResponseHeaders()”, “http.getHostHeader()”

http.getHostHeader()

Returns the HTTP Host header. This value is lowercased and has the port removed. Any trailing full stop is also removed. For example if the Host header is 'www.Example.com:80' then http.getHostHeader() returns 'www.example.com'.

Sample Usage

```
# Get the unambiguous Host header
$hostheader = http.getHostHeader();
```

See also: “http.getHeader(name)”

http.getMethod()

Returns the HTTP method that was used to make the request, such as GET or POST.

Sample Usage

```
# Was this an HTTP POST?
if( http.getMethod() == "POST" ) {
   # handle POST request ...
}
```

See also: “http.setMethod(method)”

http.getMultipartAttachment(part)

Returns the specified data out of a multipart encoded HTTP request. The data contained in the part is returned on success, or "" if it doesn't exist for that part. $1 contains the Content Type of the part, and $2 contains the complete headers for that part.

Sample Usage

```
$count = 0;
while( $data =
       http.getMultipartAttachment( $count ) ) {
  log.info( "Data was " . $data );
  log.info( "Content-Type was " . $1 );
  log.info( "Headers were " . $2 );
  $count = $count + 1;
}
```

http.getPath()

Returns the %-decoded path in the HTTP request URL, stripping the query string if one was provided. If there is a leading scheme and authority prefix, this is removed as well, so the URL "http://www.example.com/content?page=44" will be returned as "/content". If a %-encoded value is found, which isn't valid hexadecimal (for example, %G), this isn't converted or removed, but instead the '%' character is replaced by a '_' character, since malformed %-encoded values could be used as part of a malicious attack to fool security checks against particular URLs.

Sample Usage

```
# Retrieve the path
$path = http.getPath();
```

See also: “http.setPath(url)”, “http.getRawURL()”, “http.getQueryString()”, “http.getRawQueryString()”, “http.normalizePath(url)”

http.getQueryString()

Returns the %-decoded query string in the URL, or the empty string if no query string was provided.

Sample Usage

```
# Was there a query string?
$qs = http.getQueryString();
if( $qs ) {
   # Handle query string ...
}
```

See also: “http.setQueryString(querystring)”, “http.getRawQueryString()”, “http.setRawQueryString(querystring)”, “http.getRawURL()”

http.getRawQueryString()

Returns the raw (non %-decoded) query string in the URL, or the empty string if no query string was provided.

Sample Usage

```
# Was there a query string?
$qs = http.getRawQueryString();
if( $qs ) {
   # Handle query string ...
}
```

See also: “http.setRawQueryString(querystring)”, “http.getQueryString()”, “http.setQueryString(querystring)”, “http.getRawURL()”

http.getRawURL()

Returns the raw (non-decoded) URL data provided by the client in the first line of the HTTP request, after the method and before the HTTP version specifier.

© 2019 Pulse Secure, LLC.

The raw URL data includes both the path and query string if supplied and is not decoded. It may also contain the protocol and hostname if the client sent them. It could contain %-escaped characters that can be used to disguise the contents of the URL. Use http.getPath() or http.getQueryString() to return the %-decoded version of the path or query string.

This function could return raw urls of various forms including:

- http://www.example.com/file.html
- /file.html
- /path/../file.html
- /file.html?querystring
- /file.html?qs%encoded

You can use http.normalizePath() on the path component to remove any '/../' or '/./' references.

In general it is better to use http.getPath() and http.getQueryString() to avoid the need to process the raw url.

Sample Usage

```
$rawurl = http.getRawURL();
if( string.contains( $rawurl, "%00" ) ) {
   # Something suspicious here ...
   connection.discard();
}
```

See also: “http.getPath()”, “http.getQueryString()”, “http.getRawQueryString()”, “string.unescape(escaped string)”, “http.normalizePath(url)”

http.getRequest()

Returns the full HTTP request and headers, but does not include any body data.

Sample Usage

```
# Check that the request is not too big
# for our servers
$request = http.getRequest();
if( string.len( $request ) > 2048 ) {
   http.sendResponse( "413 Request too large",
                      "text/plain",
                      "Request too large", "" );
}
```

See also: “http.listHeaderNames()”, “http.getHeader(name)”, “http.getBody([count])”

http.getResponse()

Returns the beginning of the HTTP response (the status line and the headers), up to but not including the empty line that separates the headers from the body. The returned string does not include any body data.

Sample Usage

```
# See if the response included any
# headers that start with 'Foo'
$response = http.getResponse();
if( string.regexMatch( $response, "^Foo" )) {
   log.info( "A 'Foo' header was found" );
}
```

See also: "http.listResponseHeaderNames()", "http.getResponseHeader(name)", "http.getResponseBody([count])"

http.getResponseBody([count])

Returns the body of the HTTP response.

If the response has chunked transfer encoding this function will return the de-chunked body. Similarly if the response is gzip or deflate compressed the body will be returned uncompressed.

If the optional 'count' parameter is provided, http.getResponseBody() will read and return the first 'count' bytes of the response. If count is 0, http.getResponseBody() will return the entire response.

Sample Usage

```
# Read the entire response body
$body = http.getResponseBody();
```

See also: "http.setResponseBody(body, [transfer-encoding])", "http.getBody([count])"

Restrictions

Cannot be used in completion rules.

http.getResponseBodyLines([count])

Splits the body data of the HTTP request into individual lines and returns an array of the data.

If the response has chunked transfer encoding this function will return the de-chunked body. Similarly if the response is gzip or deflate compressed the body will be returned uncompressed.

If the optional 'count' parameter is provided, http.getResponseBodyLines() will read and return the first 'count' bytes of the response. If count is 0, http.getResponseBodyLines() will return the entire response.

Sample Usage

```
# Read the entire response body
$body = http.getResponseBodyLines();
```

See also: "http.setResponseBody(body, [transfer-encoding])", "http.getBodyLines([count])"

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

http.getResponseCode()

Returns the status code from the first line of the HTTP response.

Sample Usage

```
# Log 404 responses
if( http.getResponseCode() == 404 ) {
   log.info( "404 page for " . http.getPath() );
}
```

See also: "http.setResponseCode(code, [message])"

http.getResponseCookie(name)

Returns the value of the named cookie in the HTTP response.

http.getResponseCookie() is a helper method to make it easier to parse the HTTP Set-Cookie header and extract the values of that particular cookie, rather than using http.getResponseHeader() directly.

If the cookie does not exist, http.getResponseCookie() will return the empty string.

This function should be called in a response rule; it has no effect in a request rule.

Sample Usage

```
# Get the PHP session cookie
$cookie = http.getResponseCookie( "PHPSESSIONID" );
```

See also: "http.getResponseCookies()", "http.setResponseCookie(name, value, [options])", "http.removeResponseCookie(name)", "http.getCookie(name)"

http.getResponseCookies()

Returns a hash containing the names of all the cookies being set in this response, mapped to their values.

This function should be called in a response rule; it has no effect in a request rule.

Sample Usage

```
# List all of the cookies that are being set in
# the response and their values
$cookies = http.getResponseCookies();
foreach( $cookie in hash.keys( $cookies ) ) {
   log.info( $cookie . ": " . $cookies[$cookie] );
}
```

See also: "http.getResponseCookie(name)", "http.setResponseCookie(name, value, [options])", "http.removeResponseCookie(name)", "http.getCookies()"

http.getResponseHeader(name)

Returns the value of a named HTTP header in the HTTP response, or the empty string if the header does not exist or has an empty value. The header lookup is case-insensitive.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Get the mime type of the response
$mime = http.getResponseHeader( "Content-Type" );

# note that the MIME type may look like
# 'text/html; charset=ISO-8859-1'
```

See also: "http.setResponseHeader(name, value)", "http.addResponseHeader(name, value)", "http.removeResponseHeader(name)", "http.responseHeaderExists(name)", "http.getHeader(name)"

http.getResponseHeaderNames() - deprecated

This function has been deprecated. Use instead "http.listResponseHeaderNames()".

Returns a list of all the headers that are present in the response.

The headers are returned as a single string, separated by spaces.

http.getResponseHeaders()

Returns a hash containing all the header names in the response mapped to their values.

Sample Usage

```
# Show all the headers in the response
$headers = http.getResponseHeaders();

foreach( $header in hash.keys( $headers ) ) {
   log.info( $header . "=" . $headers[$header] );
}
```

See also: "http.getResponseHeader(name)", "http.setResponseHeader(name, value)", "http.addResponseHeader(name, value)", "http.removeResponseHeader(name)", "http.responseHeaderExists(name)", "http.getHeaders()"

http.getResponseVersion()

Returns the version of the HTTP protocol being used. It returns the version string in the first line of the HTTP response, such as 'HTTP/1.1'. It will return the empty string in the case of HTTP/0.9 response.

Sample Usage

```
# Get the HTTP response version
$version = http.getResponseVersion();
```

See also: "http.getVersion()"

http.getScheme()

Returns the Scheme portion of the HTTP request URL. If no scheme was included in the request, this function will return the empty string. For HTTP/2 requests, the scheme is received from the client as the ":scheme" pseudo-header. For HTTP/1.x requests, the scheme is normally omitted unless the traffic manager is being used as an explicit HTTP proxy.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Retrieve the scheme
$scheme = http.getScheme();
```

See also: "http.getPath()", "http.getRawURL()", "http.getQueryString()", "http.getRawQueryString()"

http.getVersion()

Returns the version string from the HTTP request being sent to the back-end server. For HTTP/0.9 requests, the returned value will be an empty string as there is no version specifier in HTTP/0.9. The returned value will not necessarily match the HTTP version in use by the client as the traffic manager might convert the request before processing it. To obtain the HTTP version in use by the client connection, use http.getClientVersion().

Sample Usage

```
if ( http.getVersion() == "HTTP/1.1" ) {
   log.info( "Sending HTTP/1.1 request to server." );
}
```

See also: "http.getClientVersion()", "http.getResponseVersion()"

http.headerExists(name)

Reports if a named header exists or not. It is similar to http.getHeader(), but makes it possible to distinguish between a header not being present and a header having no value.

The lookup is case-insensitive.

It returns 1 if the header exists, and 0 if it does not.

Sample Usage

```
# Add a host header if it is missing
if( !http.headerExists( "Host" ) ) {
   http.addheader( "Host", "unknown" );
}
```

See also: "http.getHeader(name)"

http.listFormParamNames()

Returns an array containing the names of all the form parameters present in the URL query string and, if the request is a POST, in the POST body data. If the same parameter appears multiple times in the request, it will only appear once in the list returned by this function.

Sample Usage

```
# Log all of the form parameters and values
$params = http.listFormParamNames();
foreach( $param in $params ) {
   log.info($param . "=" .
            http.getFormParam($param, ","));
}
```

See also: "http.getFormParam(Parameter, [Separator])", "http.doesFormParamExist(Parameter)"

© 2019 Pulse Secure, LLC.

http.listHeaderNames()

Returns a list of all the headers that are present in the request.

The headers are returned as an array.

Sample Usage

```
# Log all of the header names and values
$headers = http.listHeaderNames();
foreach( $header in $headers ) {
   log.info($header . "=" . http.getHeader($header));
}
```

See also: "http.getHeader(name)", "http.removeHeader(name)", "http.getRequest()", "http.listResponseHeaderNames()"

http.listResponseHeaderNames()

Returns a list of all the headers that are present in the response.

The headers are returned as an array.

Sample Usage

```
# Log all of the header names and values
$headers = http.listResponseHeaderNames();
foreach( $header in $headers ) {
   log.info($header . "=" . http.getHeader($header));
}
```

See also: "http.getResponseHeader(name)", "http.removeResponseHeader(name)", "http.getResponse()", "http.listHeaderNames()"

http.normalizePath(url)

Flattens a decoded URL path, converting '//' to '/', '/./' to '/', and flattening '/a/../' to '/'. It returns the flattened path string.

If the file system path is invalid, this function returns the empty string. Invalid paths include those that contain disallowed characters like '\0', invalid hex-escapes, or that use '../' sequences to reference a location outside the local root.

This function should be used on the retrieved URL before attempting path matching for access control. Remember to pass in a %-decoded URL path, to prevent disguised paths and control codes from being 'hidden' in the path.

Sample Usage

```
# Check for access to /secure
$path = http.normalizePath( http.getPath() );
if( !$path ) {
   # bad path ...
} else if( string.startsWith( $path, "/secure" ) ) {
```


© 2019 Pulse Secure, LLC.

```
   # request to restricted area: check credentials
}
```

See also: “http.getPath()”

http.redirect(address)

Sends back an HTTP 302 temporary redirect response, which will instruct a web browser to fetch the specified URL.

The URL specified will be used as the value of the 'Location' header in the response sent to the client. As described in RFC 7231, section 7.1.2, the value of the Location header must be an ASCII-encoded URI-reference, with any illegal characters properly percent-encoded.

Care must be taken when constructing the URL for this function from an untrusted source. As a security measure, if any characters of ASCII value 32 and below are found in the supplied URL, http.redirect will percent-encode them, however this action will not mitigate all potential security problems.

This function, excepting the additional percent-encoding step, is the equivalent of http.sendResponse("302 Moved Temporarily", "text/html", "", "Location: " . $url);

Sample Usage

```
# Redirect all 404 error pages to our front page
if( http.getResponseCode() == 404 ) {
   http.redirect( "http://www.example.com/" );
}
```

See also: “http.sendResponse(code, type, body, headers)”, “http.changeSite(name)”

Restrictions

Cannot be used in completion rules.

http.removeCookie(name)

Removes the named cookie from the incoming HTTP request.

http.removeCookie is a helper method that makes it easier to parse the HTTP Cookie header and remove a particular cookie, rather than using http.getHeader() and http.setHeader() directly.

Sample Usage

```
# Remove the 'Priority' cookie
http.removeCookie( "Priority" );
```

See also: “http.getCookie(name)”, “http.setCookie(name, value)”, “http.removeResponseCookie(name)”

http.removeHeader(name)

Removes a named header if it exists in the request.

Sample Usage

```
# Remove the 'Accept-Language' header if it exists
http.removeHeader( "Accept-Language" );
```

See also: "http.getHeader(name)", "http.addHeader(name, value)", "http.setHeader(name, value)", "http.removeResponseHeader(name)"

http.removeResponseCookie(name)

Removes a cookie from the HTTP response.

This function should be called in a response rule; it has no effect in a request rule.

Sample Usage

```
# Remove the 'Priority' cookie
http.removeResponseCookie( "Priority" );
```

See also: "http.getResponseCookie(name)", "http.setResponseCookie(name, value, [options])", "http.removeCookie(name)"

http.removeResponseHeader(name)

Removes the named HTTP header from the HTTP response. The header lookup is case-insensitive.

Sample Usage

```
# Remove the 'Location' response header
http.removeResponseHeader( "Location" );
```

See also: "http.setResponseHeader(name, value)", "http.getResponseHeader(name)", "http.addResponseHeader(name, value)", "http.scrubResponseHeaders(header1, header2, ...)", "http.removeHeader(name)"

http.responseHeaderExists(name)

Reports if a named header exists in the HTTP response. It is similar to http.getResponseHeader(), but makes it possible to distinguish between a header not being present and a header having no value.

Header lookups are case-insensitive.

It returns 1 if the header exists, and 0 if it does not.

Sample Usage

```
if( http.responseHeaderExists( "Location" ) ) {
   # Web server is redirecting user to
   # another location
}
```

See also: "http.getResponseHeader(name)"

© 2019 Pulse Secure, LLC.

http.scrubRequestHeaders(header1, header2, ...)

Limits the allowed HTTP request headers to a known set. The allowed headers can either be passed in as a list or space separated in a single argument.

Care should be taken when using this function to ensure that the headers that are required for connection handling are let through. At the very least, the following should be allowed: Connection, Content-Length, Transfer-Encoding, Content-Type, Host For a complete list of HTTP headers, refer to RFC2616. Protocols that extend HTTP, such as WebDAV, use other headers.

Sample Usage

```
# Remove all headers, except the Connection,
# Content-Type, Transfer-Encoding, Content-Length
# and Host headers.

# These 2 examples are identical

http.scrubRequestHeaders( "Host",
   "Connection", "Content-Type",
   "Transfer-Encoding", "Content-Length" );

http.scrubRequestHeaders(
   "host connection content-type transfer-encoding".
   " content-length" );
```

See also: “http.removeHeader(name)”, “http.scrubResponseHeaders(header1, header2, ...)”

http.scrubResponseHeaders(header1, header2, ...)

Limits the allowed HTTP response headers to a known set. The allowed headers can either be passed in as a list or space separated in a single argument.

Care should be taken when using this function to ensure that the headers that are required for connection handling are let through. At the very least, the following should be allowed: Connection, Content-Length, Transfer-Encoding, Location For a complete list of HTTP headers, refer to RFC2616. Protocols that extend HTTP, such as WebDAV, use other headers.

Sample Usage

```
# Remove all headers, except the Date, Connection,
# Content-Type, Transfer-Encoding, Content-Length
# and Location headers.

# These 2 examples are identical

http.scrubResponseHeaders( "Date",
   "Connection", "Content-Type",
   "Transfer-Encoding", "Content-Length",
   "Location" );

http.scrubResponseHeaders(
   "date connection content-type transfer-encoding".
```

```
   " content-length location" );
```

See also: “http.removeResponseHeader(name)”, “http.scrubRequestHeaders(header1, header2, ...)”

http.sendResponse(code, type, body, headers)

Hands back an HTTP response to the client instead of balancing the request via a pool onto a node. It generates a correct HTTP response from the response code, content type, body data and headers supplied. Multiple headers should be separated with \r\n (note, however, that your traffic manager may override some of these headers, e.g. the 'Connection' header).

For more sophisticated behaviour (for example, compressing responses using http.compress.enable) the http.stream.* methods should be used to respond to a client.

Sample Usage

```
# Deny access and close connection
http.sendResponse( "403 Permission Denied",
         "text/html", "Go away",
         "Set-Cookie: denied=Yes\r\nX-Foo: Bar");
```

See also: “connection.close(Data, [Read])”, “connection.discard()”

Restrictions

Cannot be used in completion rules.

http.setBody(body)

Sets the request body for this HTTP request to the supplied string, replacing any request body already present.

This also updates the 'Content-Length' header in the request to the length of the new body data.

Sample Usage

```
# Change the order!
$body = http.getBody();
$body = string.regexsub( $body, "Buy", "Sell", "g" );
http.setBody( $body );
```

See also: “http.getBody([count])”, “http.setResponseBody(body, [transfer-encoding])”

Restrictions

Cannot be used in completion rules.

http.setCookie(name, value)

Sets the value of the named cookie in the incoming HTTP request.

http.setCookie is a helper method that makes it easier to parse the HTTP Cookie header and set the value of a particular cookie, rather than using http.getHeader() and http.setHeader() directly.

An HTTP 'Cookie' header can have multiple values, such as

© 2019 Pulse Secure, LLC.

Cookie: user-id=Joe; user-type=gold

Sample Usage

```
# Set the priority cookie
http.setCookie( "Priority", "Gold" );
```

See also: "http.getCookie(name)", "http.removeCookie(name)", "http.setResponseCookie(name, value, [options])"

http.setEscapedPath(url)

Replaces the path portion of the request URL with the supplied value. If the replacement value contains a '?', this function will also replace the query string; otherwise, any query string is preserved. Characters whose ASCII value is less than or equal to 32 and '%' will be %-encoded in the replacement value.

Sample Usage

```
# Make customer buy widget:
http.setEscapedPath( "/purchase?product=widget" );
```

See also: "http.getPath()", "http.setPath(url)", "http.setRawPath(url)"

http.setHeader(name, value)

Sets the value of the named HTTP header.

A case-insensitive lookup is first performed in order to identify any existing headers, replacing the value where a match is found.

Note that the "Connection" header is manipulated by your traffic manager and that changing its value in TrafficScript may not give the expected results.

If an invalid header name is specified, the function will print a warning and return without modifying the HTTP request.

Sample Usage

```
# Add (or replace) an X-Forwarded-By header
http.setHeader( "X-Forwarded-By",
   "vTM " . sys.hostname() );
```

See also: "http.addHeader(name, value)", "http.getHeader(name)", "http.headerExists(name)", "http.removeHeader(name)", "http.setResponseHeader(name, value)"

http.setIdempotent(resend)

Marks a request as resendable or non-resendable.

An idempotent request has no detrimental side effects, so it can safely be submitted multiple times. A simple page retrieval is generally idempotent. A non-idempotent request has a side effect - for example, it may update a database, or initiate a purchase. The HTTP/1.1 specification regards all GET, HEAD, PUT, DELETE, OPTIONS and TRACE requests as idempotent.

© 2019 Pulse Secure, LLC.

Your traffic manager tags these requests as 'resendable'; if the request is submitted to a back-end node and a correct response is not received, your traffic manager will resubmit the request to another back-end node. All other requests, such as POST requests are not resent if a back-end node fails to generate a correct response.

http.setIdempotent() can override this behaviour. If 'resend' has a non-zero value, this indicates that if the request is submitted to a back-end node and a correct response is not received, your traffic manager should resubmit the request to another back-end node.

If 'resend' is zero, this indicates that the request should only be attempted against one back-end node.

Note that a request cannot be resent if it has begun streaming data from the client to the node before it detects the failure. To avoid this, you can read the entire request within the TrafficScript rule, so that it is buffered in its entirety internally in the TrafficManager.

Sample Usage

```
# This request can be resent
if( http.getMethod() == "POST" ) {
   # Force the {{product_name}}
   # to read the entire HTTP body
   http.getBody();
   # Mark this request as resendable
   http.setIdempotent( 1 );
}
```

See also: “request.setIdempotent(resend)”

Restrictions

Cannot be used in completion rules.

http.setMethod(method)

Sets the HTTP method, changing the original request.

Sample Usage

```
# Force HTTP POSTs to GETs
if( http.getMethod() == "POST" ) {
   http.setBody( "" );
   http.setMethod( "GET" );
}
```

See also: “http.getMethod()”

http.setPath(url)

Replaces the path portion of the request URL with the supplied value. If the replacement value contains a '?', this function will also replace the query string; otherwise, any query string is preserved. This function will %-encode any character excluding following set as defined in RFC3986 - Unreserved: A-Z, a-z, 0-9, ., _, - - Reserved: !, *, (,), ;, :, @, &, =, +,' $, ',', ?, /, #, [,] - '%'

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Make customer buy widget:
http.setPath( "/purchase?product=widget" );
```

See also: “http.getPath()”, “http.setRawPath(url)”, “http.setEscapedPath(url)”, “string.urlencode(string)”

http.setQueryString(querystring)

Replaces the query-string portion of the request URL with the supplied replacement. Any control characters in the replacement are %-encoded.

Sample Usage

```
# Rewrite the query string
http.setQueryString( $newqs );
```

See also: “http.getQueryString()”, “http.setRawQueryString(querystring)”, “http.getRawQueryString()”

http.setRawPath(url)

Replaces the path portion of the request URL with the supplied value. If the replacement value contains a '?', this function will also replace the query string; otherwise, any query string is preserved. URL will not be %-encoded.

Sample Usage

```
# Make customer buy widget:
http.setRawPath( "/purchase?product=widget" );
```

See also: “http.getPath()”, “http.setPath(url)”, “http.setEscapedPath(url)”

http.setRawQueryString(querystring)

Replaces the query-string portion of the request URL with the supplied replacement. Unlike http.setQueryString, control characters are not encoded.

Sample Usage

```
# Rewrite the query string
http.setRawQueryString( "foo=%20bar" );
```

See also: “http.getRawQueryString()”, “http.setQueryString(querystring)”, “http.getQueryString()”

http.setResponseBody(body, [transfer-encoding])

Sets the response body for this HTTP response to the supplied string, replacing any response body already present.

This also updates the 'Content-Length' header in the response to the length of the new body data. In addition the 'Content-Encoding' header is removed as we only ever set body data which is not encoded or compressed. If the server is still sending the original response body when this function is called, the connection to the server will be harmlessly dropped.

The optional 'transfer-encoding' parameter indicates the encoding of the body data (for example, 'chunked').

© 2019 Pulse Secure, LLC.

Sample Usage

```
$body = http.getResponseBody( 0 );
$body = string.regexsub( $body, "Buy", "Sell", "g" );
http.setResponseBody( $body );
```

See also: "http.getResponseBody([count])", "http.setBody(body)"

Restrictions

Cannot be used in completion rules.

http.setResponseCode(code, [message])

Sets the status code and message in the first line of the HTTP response.

Sample Usage

```
# Not enough credit!
http.setResponseCode( "402", "Payment required" );
```

See also: "http.getResponseCode()"

http.setResponseCookie(name, value, [options])

Sets a cookie in the HTTP response. If the named cookie already exists, this function replaces its value.

The options are a semi-colon separated list of cookie options, such as "domain", "path", "expires" and "secure".

If the named cookie exists and no 'options' are provided, the current options for the named cookie are preserved.

This function may be called from a request rule or a response rule.

Sample Usage

```
# Set the priority cookie
http.setResponseCookie( "Priority", "Gold" );

# Set a username cookie, with various options
http.setResponseCookie( "Username", "mork",
   "domain=example.com; path=/cgi-bin" );
```

See also: "http.getResponseCookie(name)", "http.removeResponseCookie(name)", "http.setCookie(name, value)"

http.setResponseHeader(name, value)

Sets a HTTP header in the HTTP response that will be sent back to the client. If the header already exists in the response, then it will be replaced with this new value. The header lookup is case-insensitive.

Note that this function should not be used with the Connection header, i.e. setResponseHeader("Connection", value) since it may not give the expected results.

© 2019 Pulse Secure, LLC.

If an invalid header name is specified, the function will print a warning and return without modifying the HTTP request.

Sample Usage

```
# Change the server string
http.setResponseHeader( "Server",
   "vTM" );
```

See also: “http.addResponseHeader(name, value)”, “http.getResponseHeader(name)”, “http.removeResponseHeader(name)”, “http.setHeader(name, value)”

http.aptimizer.bypass()

Do not use Web Accelerator to optimize the result of this request. Note that http.aptimizer.bypass() should not be used to turn off content optimization for requests that have been previously optimized as this might result in some requests not being served. Instead, create a new Web Accelerator profile with the mode set to 'Idle' and assign this profile to the request using http.aptimizer.use().

Sample Usage

```
# Bypass content optimization if the query string
# contains a 'bypass_aptimizer' parameter. This could
# be used to quickly compare the difference between
# optimized and unoptimized versions of a web page.

if( http.doesFormParamExist( "bypass_aptimizer" ) ) {
   http.aptimizer.bypass();
}
```

See also: “http.aptimizer.use(scope, profile)”

Restrictions

Cannot be used in completion rules.

http.aptimizer.use(scope, profile)

Use the specified Web Accelerator scope and profile to optimize web content served to the client that sent this request. If the named Web Accelerator scope or profile does not exist then a warning message will be logged and the traffic manager will choose a scope matching the request based on the Web Accelerator settings configured for the virtual server, and use the associated profile. If no scope matches the request then no optimization will occur. If the virtual server does not have Web Accelerator enabled, then a warning message will be logged and no optimization will occur.

Sample Usage

```
# Optimize web content for clients based in Australia
$ip = request.getRemoteIP();
if( geo.getCountry( $ip ) == "Australia" ) {
   http.aptimizer.use(
      "Any hostname or path",
      "Remote Users"
```

```
   );
}
```

See also: "http.aptimizer.bypass()"

Restrictions

Cannot be used in completion rules.

http.auth.getAuthenticatedUser()

If the HTTP request has been authenticated via Authentication configured on the relevant Virtual Server, the authenticated user name can be fetched with this function. Basic-Auth or non-authenticated requests will return the empty string.

Sample Usage

```
$u = http.auth.getAuthenticatedUser();
log.info( "Authenticated User: " . $u );
# backend checks this header:
http.setHeader( "X-Frontend-Auth-User", $u );
```

http.cache.disable()

Prevents this response from being cached. In a request rule, this additionally prevents a cache lookup for this request.

Sample Usage

```
# don't cache static content...
if( $staticcontent ) { http.cache.disable(); }
```

See also: "http.cache.enable()", "http.cache.setkey(key)"

Restrictions

Cannot be used in completion rules.

http.cache.enable()

Performs the opposite function to http.cache.disable(), and re-enables the default caching behaviour when the dynamic caching option in the virtual server is enabled.

Sample Usage

```
# only cache what we explicitly enable
http.cache.disable();  # turn off everything
if( $agent == "googlebot" || $is_app1 ) {
   http.cache.enable();
}
```

See also: "http.cache.disable()", "http.cache.setkey(key)"

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

http.cache.exists([poolname])

Returns 1 if the current request can (currently) be responded to from the cache, otherwise 0.

Note that even if the request is in the cache at the time of this call, it may be removed from the cache by the time that TrafficScript processing has finished and the traffic manager can send it. If a cached response must be guaranteed, http.cache.respondIfCached() should be used.

A pool name can be provided as optional argument in order to make the lookup use the specified pool. Without this argument, the lookup will use the pool previously selected with pool.select or the virtual server's default pool.

This function always returns 0 if called in a response rule.

Sample Usage

```
# Use a rate class only if the page is going to be
# served from the backend.
if( !http.cache.exists() ) {
   # The page cannot be served from the cache.
   # The traffic manager will have to get the page
   # from a back-end server, so rate-limit the
   # connection:
   rate.use( "rate" );
}
```

See also: “http.cache.enable()”, “http.cache.disable()”, “http.cache.respondIfCached([poolname])”, “pool.select(Pool, [Host, Port])”

http.cache.respondIfCached([poolname])

Sends a cached response to the client without any further rule processing and without connecting to a back-end server. If no match is found in the cache or if the request does not allow cached responses, rule processing continues normally. If the response can be served from the cache, no statements after this function call will be processed and the client will get the cached page.

A pool name can be provided as optional argument in order to make the lookup use the specified pool. Without this argument, the lookup will use the pool previously selected with pool.select or the virtual server's default pool.

This function does nothing if called in a response rule.

Sample Usage

```
# Use a rate class only if the page is going to be
# served from the backend.
http.cache.respondIfCached();
#
```

```
# If we get here, the page could not be served
# from the cache.  The traffic manager will have
# to get the page from a back-end server, so
# rate-limit the connection:
rate.use( "rate" );
```

See also: "http.cache.enable()", "http.cache.disable()", "http.cache.exists([poolname])", "pool.select(Pool, [Host, Port])"

Restrictions

Cannot be used in completion rules.

http.cache.setkey(key)

Allows multiple variants of the same URL to be considered distinct objects, even if the standard 'Vary' RFC semantics would consider the pages identical. Cached objects will be stored with this key, and subsequent requests for the same URL will only match if the same key is provided. An example use is to provided different cached content based on a portion of the User-Agent field of the request.

Note that successive uses of this function will overwrite the previous use rather than append the new key to it.

Sample Usage

```
# internal/external users see different pages
http.cache.setkey( $am_internal_user );
```

See also: "http.cache.enable()", "http.cache.disable()"

Restrictions

Cannot be used in completion rules.

http.compress.disable()

Stops this HTTP response from being compressed. This function overrides the Virtual Server Content Compression settings, so this is useful for stopping particular MIME types for certain browsers from being compressed.

Sample Usage

```
# Don't compress text/css pages
if( $contenttype == "text/css" ) {
    http.compress.disable();
}
```

See also: "http.compress.enable()"

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

http.compress.enable()

Allows this individual HTTP response to be compressed. If this function is called for an HTTP response, then the Virtual Server settings for Content Compression are ignored, and the response will be compressed, assuming that the client supports compression. This function return 0 if it successfully enables compression

Sample Usage

```
# Compress all text pages for Gecko
if( string.startswith( $contenttype, "text/" ) &&
    string.contains( $useragent, "Gecko" ) ) {

    http.compress.enable();
}
```

See also: "http.compress.disable()"

Restrictions

Cannot be used in completion rules.

http.connection.sendOrigin(origin1, origin2, ...)

When a request is received via an HTTP/2 stream, calling this function will result in the traffic manager sending an ORIGIN (RFC 8336) frame to the client on the control stream. The ORIGIN frame will have the supplied URLs in the array provided as its entries.

The ORIGIN frame informs the browser that the specified URL(s) are provided authoritatively by this same virtual server, and requests for these websites can be coalesced into the same HTTP/2 connection. The URLs must start with http:// or https://, contain an optional port number, and must not provide a trailing path.

The http.connection.sendOrigin() function has no effect if the request is not a valid HTTP/2 request. If the function is called for additional requests sent over the same HTTP/2 connection, subsequent ORIGIN frames will be sent only if the URLs supplied differ from the previous invocation of http.connection.sendOrigin() for that connection. Browser behavior in response to receiving an ORIGIN frame may vary.

Sample Usage

```
http.connection.sendorigin(
                    [ "https://www.example.com",
                      "https://api.example.com" ] );
log.info( "Sent ORIGIN frame" );
```

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

http.kerberos.getClientPrincipalName()

Return the principal name to be used for this client in Kerberos Protocol Transition. If http.kerberos.setClientPrincipalName was called, returns the value set by it, otherwise if a principal name can be deduced from any client certificate, returns that. The empty string will be returned in the case of no principal name, or failure.

Sample Usage

```
$principal = http.kerberos.getClientPrincipalName();
if( ! $principal ) {
   if( ssl.clientCertStatus() != "OK" &&
      ssl.clientSupportsSecureRenegotiation() ) {
      ssl.requestCert();
      $principal =
         http.kerberos.getClientPrincipalName();
   }
   if( ! $principal ) {
      # Fall back to forms based authentication
      http.redirect( "/login.cgi" );
   }
}
```

See also: "http.kerberos.setClientPrincipalName(PrincipalName)"

http.kerberos.getDelegatedTicket()

Attempts to get a Kerberos ticket delegated to this client using Protocol Transition, encoded for use in an Authorization header. Uses the value returned by http.kerberos.getClientPrincipalName to identify the client principal to be delegated to. If a pool has been set (with pool.use, or in a response rule), uses the Kerberos Principal Configuration set for that pool (if any), otherwise uses the one set for the virtual server.

Sample Usage

```
$ticket = http.kerberos.getDelegatedTicket();
if ($ticket) {
   http.setHeader('Authorization',
                  'Negotiate ' . $ticket);
   request.retry();
}
```

See also: "http.kerberos.setClientPrincipalName(PrincipalName)"

Restrictions

Cannot be used in completion rules.

http.kerberos.getPrincipal()

Returns the name of the Kerberos principal catalog item that is currently configured for use by the traffic manager when performing Kerberos Protocol Transition. If http.kerberos.setPrincipal was called, returns the value set by it, otherwise the value from the configuration will be returned if present. The empty string will be returned in the case of no configured principal name, or failure.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$principal = http.kerberos.getPrincipal();
if( ! string.len( $principal ) ) {
   # Provide a user friendly error page
   http.redirect( "/error.html" );
}
```

See also: "http.kerberos.setPrincipal(PrincipalCatalogItem)"

http.kerberos.getTargetPrincipalName()

Return the principal name to be used for the target service in Kerberos Protocol Transition. If http.kerberos.setTargetPrincipalName was called, returns the value set by it, otherwise the value from the configuration will be returned if present. The empty string will be returned in the case of no configured principal name, or failure.

Sample Usage

```
$principal = http.kerberos.getTargetPrincipalName();
if( ! $principal ) {
   # Provide a user friendly error page
   http.redirect( "/error.html" );
}
```

See also: "http.kerberos.setTargetPrincipalName(PrincipalName)"

http.kerberos.isEnabled()

Returns true if Kerberos protocol transition is currently enabled on the connection, otherwise false.

Sample Usage

```
$user = "";
$basic = http.getHeader( "Authorization" );
$failed_auth = ! string.startsWith( $basic,
                                    "Basic " );
if( ! $failed_auth ) {
   $basic = string.skip( $basic, 6 );
   $dec = string.base64decode( $basic );
   $up = string.split( $dec, ":" );
   $user = $up[0]; $pass = $up[1];
   $auth = auth.query( "LDAP-Authenticator",
                       $user, $pass );
   $failed_auth = ! $auth['OK'];
}

if( $failed_auth ) {
   http.sendResponse( "401 Unauthenticated",
                      "text/plain",
                      "Authentication Required",
                      "WWW-Authenticate: ".
                      "Basic realm=\"YourRealm\"");
} else {
   # If Kerberos Protocol Transition is enabled for
```

```
   # the connection, ensure the user name is used.
   if( http.kerberos.isEnabled() ) {
      http.kerberos.setClientPrincipalName( $user );
   }
}
```

See also: "http.kerberos.setEnabled(Enabled)"

http.kerberos.setClientPrincipalName(PrincipalName)

Sets the principal name to be used for this client in Kerberos Protocol Transition.

Sample Usage

```
java.run( "com.example.GetClientPrincipal" );
$principal =
   connection.data.get('KerberosClientPrincipal');
http.kerberos.setClientPrincipalName( $principal );
```

See also: "http.kerberos.getClientPrincipalName()"

Restrictions

Cannot be used in completion rules.

http.kerberos.setEnabled(Enabled)

Explicitly specify whether Kerberos protocol transition should be used on the connection. The value set will override any configuration on the virtual server.

Sample Usage

```
$client = http.kerberos.getClientPrincipalName();

if( string.len( $client ) > 0 ) {
   http.kerberos.setEnabled( true );
}
```

See also: "http.kerberos.isEnabled()"

Restrictions

Cannot be used in completion rules.

http.kerberos.setPrincipal(PrincipalCatalogItem)

Sets the name of the Kerberos principal catalog item that is to be used for performing Kerberos Protocol Transition.

Sample Usage

```
if( geo.getLocation() == "Cambridge" ) {
   http.kerberos.setPrincipal( "Cambridge Realm" );
}
```

See also: "http.kerberos.getPrincipal()"

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

http.kerberos.setTargetPrincipalName(PrincipalName)

Sets the principal name to be used for the target service for Kerberos Protocol Transition.

Sample Usage

```
if( http.getResponseCode() == 401 ) {
   $node = connection.getNode();
   if( string.len( $node ) > 0 ) {
      http.kerberos.setTargetPrincipalName(
            "HTTP/" . $node . "@EXAMPLE.COM" );
      $ticket = http.kerberos.getDelegatedTicket();
      if ($ticket) {
         http.setHeader( 'Authorization',
                         'Negotiate ' . $ticket );
         request.retry();
      }
   }
}
```

See also: "http.kerberos.getTargetPrincipalName()"

Restrictions

Cannot be used in completion rules.

http.request.delete(url, [headers], [timeout])

Issues an HTTP DELETE request for a remote web page. $1 is set to the HTTP response code (e.g. 204 for OK), or is 0 if there was an error. $2 is set to the Content-Type of the response (or an error message if there was an error). $3 is set to the content-headers of the response, in the format: [<header-line>\r\n]*. $4 is set to the first line of the HTTP response. The entire original response may be reconstructed as $4."\r\n".$3."\r\n".$body, where $body is the result of http.request.get.

HTTPS pages can be requested by using the https:// prefix for the url.

A timeout parameter can be given (in seconds). If the request does not complete in this time, then an error will be given instead.

Requests made with http.request.delete() will use keepalive connections to the destination server. Provided the request was completed successfully, these keepalive connections will be re-used by any invocation of http.request.delete(), http.request.head() or http.request.get() to the same destination host:port, from any rule, in any virtual server. If the request is to a node in a pool, the connection will be shared with other requests your traffic manager makes to the same node. If the request made with http.request.delete() was unsuccessful, your traffic manager closes the connection.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Delete a resource from the server
$book_id = http.getHeader( "X-Book-ID" );
http.request.delete( "http://www.example.com/books/"
                     . $book_id );
if( $1 != 204 ) {
   http.sendResponse( $1, "text/plain", "",
      "Could not delete book " . $book_id );
}
```

See also: "http.request.get(url, [headers], [timeout])", "http.request.post(url, POST data, [headers], [timeout])", "http.request.put(url, PUT data, [headers], [timeout])", "http.request.head(url, [headers], [timeout])"

Restrictions

Cannot be used in completion rules.

http.request.get(url, [headers], [timeout])

Issues an HTTP request for a remote web page and returns the body of the page requested. $1 is set to the HTTP response code (e.g. 200 for OK), or is 0 if there was an error. $2 is set to the Content-Type of the response (or an error message if there was an error). $3 is set to the content-headers of the response, in the format: [<header-line>\r\n]*. $4 is set to the first line of the HTTP response. The entire original response may be reconstructed as $4."\r\n".$3."\r\n".$body, where $body is the result of http.request.get.

HTTPS pages can be requested by using the https:// prefix for the url.

A timeout parameter can be given (in seconds). If the request does not complete in this time, then an error will be given instead.

Requests made with http.request.get() will use keepalive connections to the destination server. Provided the request was completed successfully, these keepalive connections will be re-used by any invocation of http.request.get() or http.request.head() to the same destination host:port, from any rule, in any virtual server. If the request is to a node in a pool, the connection will be shared with other requests your traffic manager makes to the same node. If the request made with http.request.get() was unsuccessful, your traffic manager closes the connection.

Sample Usage

```
$body = http.request.get( "https://www.example.com/",
   "Cookie: foo=bar" );
if( $1 ) {
   http.sendResponse( $1, $2, $body, "" );
} else {
   log.info( "An error occurred: " . $2 );
}
```

See also: "http.request.ssl.get(url, ca_certificates, [headers], [timeout], [verify-hostname])", "http.request.head(url, [headers], [timeout])", "http.request.post(url, POST data, [headers], [timeout])", "http.request.put(url, PUT data, [headers], [timeout])", "http.request.delete(url, [headers], [timeout])"

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

http.request.head(url, [headers], [timeout])

Issues an HTTP HEAD request for a remote web page. $1 is set to the HTTP response code (e.g. 200 for OK), or is 0 if there was an error. $2 is set to the Content-Type of the response (or an error message if there was an error). $3 is set to the content-headers of the response, in the format: [<header-line>\r\n]*. $4 is set to the first line of the HTTP response. The entire original response may be reconstructed as $4."\r\n".$3."\r\n".$body, where $body is the result of http.request.get.

HTTPS pages can be requested by using the https:// prefix for the url.

A timeout parameter can be given (in seconds). If the request does not complete in this time, then an error will be given instead.

Requests made with http.request.head() will use keepalive connections to the destination server. Provided the request was completed successfully, these keepalive connections will be re-used by any invocation of http.request.head() or http.request.get() to the same destination host:port, from any rule, in any virtual server. If the request is to a node in a pool, the connection will be shared with other requests your traffic manager makes to the same node. If the request made with http.request.head() was unsuccessful, your traffic manager closes the connection.

Sample Usage

```
# Check that this site is working
http.request.head( "http://www.example.com/" );
if( $1 != 200 ) {
   pool.use( "Backup site" );
}
```

See also: “http.request.get(url, [headers], [timeout])”, “http.request.post(url, POST data, [headers], [timeout])”, “http.request.put(url, PUT data, [headers], [timeout])”, “http.request.delete(url, [headers], [timeout])”

Restrictions

Cannot be used in completion rules.

http.request.post(url, POST data, [headers], [timeout])

Issues an HTTP POST request for a remote web page, and returns the body of the page requested. $1 is set to the HTTP response code (e.g. 200 for OK), or is 0 if there was an error. $2 is set to the Content-Type of the response (or an error message if there was an error). $3 is set to the content-headers of the response, in the format: [<header-line>\r\n]*. $4 is set to the first line of the HTTP response. The entire original response may be reconstructed as $4."\r\n".$3."\r\n".$body, where $body is the result of http.request.post.

HTTPS pages can be requested by using the https:// prefix for the url.

A timeout parameter can be given (in seconds). If the request does not complete in this time, then an error will be given instead.

Requests made with http.request.post() will always create a new connection to the destination server and will not use an existing connection. After a request made with http.request.post() has finished successfully, its connection can be re-used by any invocation of http.request.get() or http.request.head() to the same destination host:port, from any rule, in any virtual server. If the request was to a node in a pool, the connection will also be shared with other requests your traffic manager makes to the same node. If the request made with http.request.post() was unsuccessful, your traffic manager closes the connection.

Sample Usage

```
$body = http.request.post("http://www.example.com/",
   "data",
   "Cookie: foo=bar\nContent-Type: text/plain" );
if( $1 ) {
   http.sendResponse( $1, $2, $body, "" );
}
```

See also: “http.request.get(url, [headers], [timeout])”, “http.request.head(url, [headers], [timeout])”, “http.request.put(url, PUT data, [headers], [timeout])”, “http.request.delete(url, [headers], [timeout])”

Restrictions

Cannot be used in completion rules.

http.request.put(url, PUT data, [headers], [timeout])

Issues an HTTP PUT request for a remote web page, and returns the body of the page requested. $1 is set to the HTTP response code (e.g. 200 for OK), or is 0 if there was an error. $2 is set to the Content-Type of the response (or an error message if there was an error). $3 is set to the content-headers of the response, in the format: [<header-line>\r\n]*. $4 is set to the first line of the HTTP response. The entire original response may be reconstructed as $4."\r\n".$3."\r\n".$body, where $body is the result of http.request.put.

HTTPS pages can be requested by using the https:// prefix for the url.

A timeout parameter can be given (in seconds). If the request does not complete in this time, then an error will be given instead.

Requests made with http.request.put() will always create a new connection to the destination server and will not use an existing connection. After a request made with http.request.put() has finished successfully, its connection can be re-used by any invocation of http.request.get() or http.request.head() to the same destination host:port, from any rule, in any virtual server. If the request was to a node in a pool, the connection will also be shared with other requests your traffic manager makes to the same node. If the request made with http.request.put() was unsuccessful, your traffic manager closes the connection.

Sample Usage

```
$body = http.request.put("http://www.example.com/",
   "data",
```


© 2019 Pulse Secure, LLC.

```
   "Cookie: foo=bar\nContent-Type: text/plain" );
if( $1 ) {
   http.sendResponse( $1, $2, $body, "" );
}
```

See also: "http.request.get(url, [headers], [timeout])", "http.request.head(url, [headers], [timeout])", "http.request.post(url, POST data, [headers], [timeout])", "http.request.delete(url, [headers], [timeout])"

Restrictions

Cannot be used in completion rules.

http.request.ssl.get(url, ca_certificates, [headers], [timeout], [verify-hostname])

Issues an HTTPS request for a remote web page and returns the body of the page requested. Its behavior is identical to that of http.request.get when used with an HTTPS URL except for the following differences:

- The URL must start with 'https://', e.g https://www.iana.org/, i.e. this function can only be used for secure communication.
- http.request.ssl.get will verify the server's certificate during the SSL Handshake. It will do so using the CA certificates provided in the second argument. This argument has to be an array of certificates. Certificates must be provided in PEM format. If one of the provided strings is not a valid certificate in PEM format, or if the array is empty, the function will fail. Therefore, verification of the peer's certificate is mandatory.
- If the fifth argument has a non-zero value http.request.ssl.get will also check whether the certificate's common name(s) match the hostname specified in the URL.
- Kept-alive connections are not re-used.

Sample Usage

```
$issuer = '-----BEGIN CERTIFICATE-----
MIIDxTCCAq2gAwIBAgIQAqxcJmoLQJuPC3nyrkYldzANBgkq
hkiG9w0BAQUFADBsMQswCQYDVQQGEwJVUzEVMBMGA1UEChMM
RGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3d3cuZGlnaWNlcnQu
Y29tMSswKQYDVQQDEyJEaWdpQ2VydCBIaWdoIEFzc3VyYW5j
ZSBFViBSb290IENBMB4XDTA2MTExMDAwMDAwMFoXDTMxMTEx
MDAwMDAwMFowbDELMAkGA1UEBhMCVVMxFTATBgNVBAoTDERp
Z2lDZXJ0IEluYzEZMBcGA1UECxMQd3d3LmRpZ2ljZXJ0LmNv
bTErMCkGA1UEAxMiRGlnaUNlcnQgSGlnaCBBc3N1cmFuY2Ug
RVYgUm9vdCBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBAMbM5XPm+9S75S0tMqbf5YE/yc0lSbZxKsPVlDRn
ogocsF9ppkCxxLeyj9CYpKlBWTrT3JTWPNt0OKRKzE0lgvdK
pVMSOO7zSW1xkX5jtqumX8OkhPhPYlG++MXs2ziS4wblCJEM
xChBVfvLWokVfnHoNb9Ncgk9vjo4UFt3MRuNs8ckRZqnrG0A
FFoEt7oT61EKmEFBIk5lYYeBQVCmeVyJ3hlKV9Uu5l0cUyx+
mM0aBhakaHPQNAQTXKFx01p8VdteZOE3hzBWBOURtCmAEvF5
OYiiAhF8J2a3iLd48soKqDirCmTCv2ZdlYTBoSUeh10aUAsg
EsxBu24LUTi4S8sCAwEAAaNjMGEwDgYDVR0PAQH/BAQDAgGG
MA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFLE+w2kD+L9H
AdSYJhoIAu9jZCvDMB8GA1UdIwQYMBaAFLE+w2kD+L9HAdSY
```

```
JhoIAu9jZCvDMA0GCSqGSIb3DQEBBQUAA4IBAQAcGgaX3Nec
nzyIZgYIVyHbIUf4KmeqvxgydkAQV8GK83rZEWWONfqe/EW1
ntlMMUu4kehDLI6zeM7b41N5cdblIZQB2lWHmiRk9opmzN6c
N82oNLFpmyPInngiK3BD41VHMWEZ71jFhS9OMPagMRYjyOfi
ZRYzy78aG6A9+MpeizGLYAiJLQwGXFK3xPkKmNEVX58Svnw2
Yzi9RKR/5CYrCsSXaQ3pjOLAEFe4yHYSkVXySGnYvCoCWw9E
1CAx2/S6cCZdkGCevEsXCS+0yx5DaMkHJ8HSXPfqIbloEpw8
nL+e/IBcm2PN7EeqJSdnoDfzAIJ9VNep+OkuE6N36B9K
-----END CERTIFICATE-----';


$body = http.request.ssl.get(
                            "https://www.iana.org/",
                            [ $issuer ],
                            "User-Agent: MyRule/0.1",
                            60,
                            true );
if( $1 ) {
   http.sendResponse( $1, $2, $body, "" );
} else {
   log.info( "An error occurred: " . $2 );
}
```

See also: “http.request.get(url, [headers], [timeout])”

Restrictions

Cannot be used in completion rules.

http.stream.continueFromBackend([data])

Stops streaming any data from the current rule and lets your traffic manager send remaining data from backend. The 'data' parameter can be used to send the last block to be streamed. Rule processing will finish and no further statements in this or subsequent rules will be executed. Unlike http.stream.finishResponse(), any data coming from the backend server will continue to be sent from the backend to the client normally.

Note that this function will behave exactly like http.stream.finishResponse() if run from a request rule.

Sample Usage

```
http.stream.startResponse(
  "200", "text/html", "", "Server: vTM");

while( 1 ) {
  # read full lines but at most 4k of data:
  $data = http.stream.readBulkResponse( 4096, "\n" );

  # No more data to read; break out of this loop
  if( 0 == string.length( $data ) ) {
    break;
  }

  if( string.find( $data, "foo" ) >= 0 ) {
     $data = string.replace( $data,"foo","bar" );
```


© 2019 Pulse Secure, LLC.

```
    # Job done, exit now. Any remaining body data
    # will continue to be sent from backend.
    http.stream.continueFromBackend( $data );
  }

  # stream data to client if not found.
  http.stream.writeResponse( $data );
}

http.stream.finishResponse();
```

See also: "http.stream.startResponse(resp_code, content_type, [content_length, headers])", "http.stream.readBulkResponse(count, [delimiter])", "http.stream.readResponse(count, [delimiter])", "http.stream.writeResponse(data)", "http.stream.finishResponse([data])"

Restrictions

Cannot be used in completion rules.

http.stream.finishResponse([data])

Indicates that 'data' is the last block to be streamed for the current transaction. Rule processing will stop after http.stream.finishResponse() has been called, i.e. the remaining statements of the present rule will not be evaluated and no subsequent rules will be run.

Sample Usage

```
http.stream.startResponse(
  "200", "text/html", "", "Server: vTM");

while( 1 ) {
  # read full lines but at most 4k of data:
  $data = http.stream.readBulkResponse( 4096, "\n" );
  if( $data == "" )
    break; # server has finished the response
  # invert server's logic:
  $data = string.replaceAllI( $data, "yes", "no" );
  http.stream.writeResponse( $data );
}

http.stream.finishResponse();
```

See also: "http.stream.startResponse(resp_code, content_type, [content_length, headers])", "http.stream.readBulkResponse(count, [delimiter])", "http.stream.readResponse(count, [delimiter])", "http.stream.writeResponse(data)", "http.sendResponse(code, type, body, headers)"

Restrictions

Cannot be used in completion rules.

http.stream.readBulkResponse(count, [delimiter])

Reads (and consumes) data from the server, so that TrafficScript can manipulate the data and send a modified version to the client.

Reads the number of bytes specified by 'count' from the body of the HTTP response supplied by the server. Data from the response is only returned up to and including the "last" occurrence of 'delimiter' if a non-empty delimiter has been specified. Unlike http.getResponseBody(), it also removes the data returned from the server's response. When the end of the response from the server has been reached, an empty string is returned.

If the delimiter partially matches at the end of the specified number of bytes, the data returned will include the full delimiter (thus returning slightly more data than specified). If the delimiter is not found in the specified number of bytes, then the specified number of bytes of data will be returned.

Sample Usage

```
# Stream a HTTP response back, changing the content
# as it is read in.
http.stream.startResponse(
  "200", "text/html", "", "Server: vTM" );

while( 1 ) {
  # read several full lines but at most 4k of data:
  $data = http.stream.readBulkResponse( 4096, "\n" );
  if( $data == "" )
    break; # server has finished the response
  # invert server's logic:
  $data = string.replaceAllI( $data, "yes", "no" );
  http.stream.writeResponse( $data );
}

http.stream.finishResponse();
```

See also: “http.stream.readResponse(count, [delimiter])”, “http.getResponseBody([count])”, “http.stream.writeResponse(data)”, “http.stream.finishResponse([data])”, “http.stream.startResponse(resp_code, content_type, [content_length, headers])”

Restrictions

Cannot be used in completion rules.

http.stream.readResponse(count, [delimiter])

Similar to http.stream.readBulkResponse(), but only returns response body data up to and including the "first" occurrence of delimiter. The two functions behave identically if no delimiter is provided.

Sample Usage

```
http.stream.startResponse(
  "200", "text/html", "", "Server: vTM" );

while( 1 ) {
  # read full line but at most 4k of data:
  $data = http.stream.readResponse( 4096, "\n" );
  if( $data == "" )
    break; # server has finished the response
  # invert server's logic:
```


© 2019 Pulse Secure, LLC.

```
  $data = string.replaceAllI( $data, "yes", "no" );
  http.stream.writeResponse( $data );
}

http.stream.finishResponse();
```

See also: "http.stream.readBulkResponse(count, [delimiter])", "http.getResponseBody([count])", "http.stream.writeResponse(data)", "http.stream.finishResponse([data])", "http.stream.startResponse(resp_code, content_type, [content_length, headers])"

Restrictions

Cannot be used in completion rules.

http.stream.startResponse(resp_code, content_type, [content_length, headers])

Sets up an HTTP response from which data can be streamed later by calling http.stream.writeResponse(). http.stream.startResponse() can only be called once per HTTP transaction. Only 'resp_code' and 'content_type' are mandatory arguments. However, it is recommended to specify the 'content_length' if possible. If it is provided and a valid integer, your traffic manager will not stream more than that number of bytes. A set of headers (separated by "\r\n") can be provided in the optional fourth argument. In a response rule, if no fourth argument is given, the response headers from the back-end will be sent on to the client (note, however, that your traffic manager may override some of these headers, e.g. the 'Connection' header)

Sample Usage

```
http.stream.startResponse(
  "200", "text/html", "",
  "Server: vTM\r\nX-Hello: World");

while( 1 ) {
  # read full lines but at most 4k of data:
  $data = http.stream.readBulkResponse( 4096, "\n" );
  if( $data == "" )
    break; # server has finished the response
  # invert server's logic:
  $data = string.replaceAllI( $data, "yes", "no" );
  http.stream.writeResponse( $data );
}

http.stream.finishResponse();
```

See also: "http.stream.writeResponse(data)", "http.stream.finishResponse([data])", "http.stream.readBulkResponse(count, [delimiter])", "http.stream.readResponse(count, [delimiter])", "http.sendResponse(code, type, body, headers)"

Restrictions

Cannot be used in completion rules.

http.stream.writeResponse(data)

Sends the data in the 'data' argument to the client. http.stream.writeResponse() can be called multiple times but http.stream.startResponse() must have been called beforehand.

Sample Usage

```
http.stream.startResponse(
  "200", "text/html", "", "Server: vTM");

while( 1 ) {
  # read full lines but at most 4k of data:
  $data = http.stream.readBulkResponse( 4096, "\n" );
  if( $data == "" )
    break; # server has finished the response
  # invert server's logic:
  $data = string.replaceAllI( $data, "yes", "no" );
  http.stream.writeResponse( $data );
}

http.stream.finishResponse();
```

See also: “http.stream.startResponse(resp_code, content_type, [content_length, headers])”, “http.stream.readBulkResponse(count, [delimiter])”, “http.stream.readResponse(count, [delimiter])”, “http.stream.finishResponse([data])”, “http.sendResponse(code, type, body, headers)”

Restrictions

Cannot be used in completion rules.

java.run(Java Extension class name, [options])

Runs a named Java Extension. The Java Extension class name must be given, and extra options can also be supplied to the Extensions. (These are supplied as the 'args' attribute in the Java Extension API). To use java.run(), first enable Java support through the Traffic Manager Admin UI or product APIs.

Sample Usage

```
java.run(
   "com.example.UserVerify", $user, $password
);
```

Restrictions

Can only be used with TCP-based protocols, Cannot be used in completion rules.

log.error(message)

Writes an error message to the traffic managers's event log file. This log can be viewed through the UI.

Sample Usage

```
log.error( "Insert coffee to continue" );
```

See also: “log.info(message)”, “log.warn(message)”, “event.emit(custom event name, message)”

© 2019 Pulse Secure, LLC.

log.info(message)

Writes an informational message to the traffic manager's event log file. This log can be viewed through the UI.

Sample Usage

```
log.info( "Everything is OK" );
```

See also: “log.warn(message)”, “log.error(message)”, “event.emit(custom event name, message)”

log.warn(message)

Writes a warning message to the traffic manager's event log file. This log can be viewed through the UI.

Sample Usage

```
log.warn( "There may be trouble ahead" );
```

See also: “log.info(message)”, “log.error(message)”, “event.emit(custom event name, message)”

net.dns.resolveHost(hostname)

Resolves a hostname into an IPv4 address, using the DNS name servers configured on the local system. If the lookup fails, an empty string is returned.

Sample Usage

```
# Do a double-dns lookup
$rip = request.getRemoteIP();
$rhost = net.dns.resolveIP( $rip );
$ip = net.dns.resolveHost( $rhost );
if( $ip != $rip ) {
   log.warn( "Double lookup failed" );
}
```

See also: “net.dns.resolveIP(IP address)”, “net.dns.resolveHost6(hostname)”

Restrictions

Cannot be used in completion rules.

net.dns.resolveHost6(hostname)

Resolves a hostname into an IPv6 address, using the DNS name servers configured on the local system. If the lookup fails, an empty string is returned.

Sample Usage

```
# Do a double-dns lookup for an IPv6 address
$rip = request.getRemoteIP();
$rhost = net.dns.resolveIP( $rip );
$ip = net.dns.resolveHost6( $rhost );
if( $ip != $rip ) {
   log.warn( "Double lookup failed" );
}
```

See also: “net.dns.resolveHost(hostname)”, “net.dns.resolveIP(IP address)”

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

net.dns.resolveIP(IP address)

Resolves an IP address to a hostname, using the DNS name servers configured on the local system.

Returns a hostname, or the IP address if the address cannot be resolved. An empty string is returned if the parameter is not a valid IP address.

Sample Usage

```
$rip = request.getRemoteIP();
$rhost = net.dns.resolveIP( $rip );
log.info( "Request from ".$rhost );
```

See also: “net.dns.resolveHost(hostname)”, “net.dns.resolveHost6(hostname)”

Restrictions

Cannot be used in completion rules.

pool.activenodes(Pool)

Returns the number of nodes that are alive in the named pool. This will not include any nodes that have been marked as 'draining'.

Sample Usage

```
# If the number of live nodes drops to below 3
# then only allow GET requests to be processed
# normally, other requests get an appropriate
# error message

if( pool.activeNodes( "database" ) < 3 ) {
   if ( http.getMethod() != "GET" ) {
      pool.use( "database_busy" );
   }
}
```

See also: “pool.listactivenodes(Pool)”, “pool.failednodes(Pool)”

pool.checknode(Pool, Host, Port)

Query the pool to determine the status of a node. Will return one of :
"NOTINPOOL","NOSUCHPOOL","NOSUCHHOST","DEAD" "ACTIVE","DISABLED","DRAINING"

Note: "DRAINING" is only returned for nodes present in the pool's configuration and in the 'draining' state. Nodes deleted from a pool with the 'drain to delete' behavior enabled via the node_delete_behavior configuration key will show as "NOTINPOOL".

© 2019 Pulse Secure, LLC.

Sample Usage

```
$status = pool.checknode("FTP Server","app1",21);
if($status != "ACTIVE") {
   log.warn("FTP Server app1 unavailable ".$status);
}
```

Restrictions

Cannot be used in completion rules.

pool.disablednodes(Pool)

Returns the number of nodes that are have been marked as disabled in the named pool.

Sample Usage

```
# Log the number of disabled nodes
$nodes = pool.disabledNodes("mypool");
log.info($nodes." nodes are disabled");
```

See also: “pool.activenodes(Pool)”, “pool.listdisablednodes(Pool)”

pool.drainingnodes(Pool)

Returns the number of nodes that are draining in the named pool.

Sample Usage

```
# Log connections to draining nodes

$pool = connection.getPool();
if( pool.drainingNodes( $pool ) ) {
   $node = connection.getNode();
   $ip_port = string.split( $node, ":" );
   if( pool.checknode( $pool,
                       $ip_port[0],
                       $ip_port[1] )
                           == "DRAINING" ) {
      log.info($node." is disabled");
   }
}
```

See also: “pool.activenodes(Pool)”, “pool.listdrainingnodes(Pool)”

pool.failednodes(Pool)

Returns the number of nodes that have failed in the named pool.

Sample Usage

```
# Emit a warning if the number of failed nodes
# exceeds the number of active nodes.

if( pool.failedNodes( "database" ) >
    pool.activeNodes( "database" ) ) {
```

```
  event.emit( "warning", "Too many failed nodes" );
}
```

See also: “pool.activenodes(Pool)”, “pool.listfailednodes(Pool)”

pool.listactivenodes(Pool)

Lists the IP addresses of the nodes that are alive in the named pool. This will not include any nodes that have been marked as 'draining'.

Sample Usage

```
# Post some data to every active node.
$nodes = pool.listActiveNodes("thepool");
foreach ( $node in $nodes ) {
   http.request.post("http://".$node."/someurl",
                     "1234");
}
```

See also: “pool.listdrainingnodes(Pool)”, “pool.listfailednodes(Pool)”, “pool.listdisablednodes(Pool)”, “pool.listallnodes(Pool)”

pool.listallnodes(Pool)

Lists the IP addresses of all the nodes including disabled nodes.

Sample Usage

```
# Log every node we know about using port 1234
$nodes = array.filter( pool.listAllNodes("thepool"),
                       ":1234$" );
foreach ( $node in $nodes ) {
   log.info( "Found ".$node );
}
```

See also: “pool.listactivenodes(Pool)”, “pool.listdrainingnodes(Pool)”, “pool.listfailednodes(Pool)”, “pool.listdisablednodes(Pool)”

pool.listdisablednodes(Pool)

Lists the IP addresses of the nodes that are disabled in the named pool.

Sample Usage

```
# Log every disabled node.
$nodes = pool.listDisabledNodes("thepool");
foreach ( $node in $nodes ) {
   log.warn( $node." is disabled" );
}
```

See also: “pool.listactivenodes(Pool)”, “pool.listdrainingnodes(Pool)”, “pool.listfailednodes(Pool)”, “pool.listallnodes(Pool)”

© 2019 Pulse Secure, LLC.

pool.listdrainingnodes(Pool)

Lists the IP addresses of the nodes that are draining in the named pool.

Sample Usage

```
# Search for a node in the draining list
$nodes = pool.listDrainingNodes("thepool");
$mynode = connection.getNode();
if ( array.contains( $nodes, $mynode ) ) {
   log.info( $mynode." is draining" );
}
```

See also: "pool.listactivenodes(Pool)", "pool.listdisablednodes(Pool)", "pool.listallnodes(Pool)", "pool.listallnodes(Pool)"

pool.listfailednodes(Pool)

Lists the IP addresses of the nodes that are marked as failed in the named pool.

Sample Usage

```
# Log every failed node.
$nodes = pool.listFailedNodes("thepool");
foreach ( $node in $nodes ) {
   log.warn( $node." has failed" );
}
```

See also: "pool.listactivenodes(Pool)", "pool.listdrainingnodes(Pool)", "pool.listdisablednodes(Pool)", "pool.listallnodes(Pool)"

pool.select(Pool, [Host, Port])

Selects a pool to load-balance this connection with. By default, the pool name should be a literal string (i.e. not dynamically generated and not containing any variables), however, if you enable the "trafficscript!variable_pool_use" global setting variables can be used too. Please refer to the Troubleshooting section of the vTM TrafficScript Overview and Reference Manual for more information about this setting. Unlike pool.use(), your traffic manager will continue to process further request rules after this function.

If the pool named does not exist, your traffic manager will log a warning message.

Optionally, a specific machine can be specified that will be used to forward the request on to. This machine does not have to be in the pool selected, or in fact in any pool. In this mode, the selected pool is used only for its configuration settings (e.g. timeout values, SSL encryption options, etc.) The request will bypass any queueing based on connection or transaction limits defined for the pool.

If the machine is specified as an empty string, a node will be chosen from the pool normally. However the connection will be made to the specified port instead of the port specified in the pool configuration.

Sample Usage

```
# Use the pool named 'Content'
pool.select( "Content" );
```

```
# Send this request to www.example.com:80,
# using config from pool 'proxy'
pool.select( "proxy", "www.example.com", 80 );

# Send this request to a node from the pool
# 'Content', but retain the original
# destination port
pool.select( "Content", "", request.getDestPort() );
```

See also: "pool.use(Pool, [Host, Port])"

Restrictions

Cannot be used in completion rules.

pool.use(Pool, [Host, Port])

Selects a pool to load-balance this connection with, and stops processing any more rules. It must only be used in request rules.

By default the pool name should be a literal string, however, if you enable the "trafficscript!variable_pool_use" global setting, variables can be used too. Please refer to the Troubleshooting section of the vTM TrafficScript Overview and Reference Manual for more information about this setting.

If the pool named does not exist, your traffic manager will log a warning message and use the default pool configured for the virtual server.

Optionally, a specific machine can be specified that will be used to forward the request on to. This machine does not have to be in the pool selected, or in fact in any pool. In this mode, the selected pool is used only for its configuration settings (e.g. timeout values, SSL encryption options, etc.) The request will bypass any queueing based on connection or transaction limits defined for the pool.

If the machine is specified as an empty string, a node will be chosen from the pool normally. However the connection will be made to the specified port instead of the port specified in the pool configuration.

Sample Usage

```
# Use the pool named 'Content'
pool.use( "Content" );

# Send this request to www.example.com:80,
# using config from pool 'proxy'
pool.select( "proxy", "www.example.com", 80 );

# Send this request to a node from the pool
# 'Content', but retain the original
# destination port
pool.select( "Content", "", request.getDestPort() );
```

See also: "pool.select(Pool, [Host, Port])"

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

rate.getbacklog(class_name, [context])

Returns the number of connections that are currently waiting to be released by the supplied rate class.

Sample Usage

```
$backlog = rate.getbacklog( "gold-user",
                            request.getRemoteIP() );
if( $backlog > 10 ){
   # Tell the customer to come back later
   http.sendResponse( "503 Service Unavailable",
                      "text/html", "Go away",
                      "Retry-After: 10" );
}
```

See also: “rate.use(class_name, [context])”

Restrictions

Cannot be used in completion rules.

rate.use(class_name, [context])

Immediately queues a connection using the named rate class.

The connection and the current TrafficScript rule is stalled until the rate class releases it, according to the rate limits defined in the class. When the connection is released, the rate.use() function returns and the TrafficScript rule continues to execute.

If rate.use() is called with the optional 'context' value, it uses a new rate class which inherits all of the rate settings from the named rate class. All connections called with the same 'context' value use the same new rate class. This allows you to shape connections based on arbitary data, such as a user name or source IP address, shaping connections from different users or source IPs independently.

If the connection has passed through the class successfully then the value 1 is returned. If the connection times out while it is queued, then the TrafficScript rule is abandoned. If the connection could not be queued because an invalid rate class name was provided, rate.use() returns 0.

Sample Usage

```
rate.use( "protect_database" );

rate.use( "limit_user",
          http.getCookie( "SessionID" ) );
```

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

rate.use.noQueue(class_name, [context])

Checks if this connection will exceed the rate limits of the named rate class. If connection is within rate limits, a value of 1 is returned and the connection is added to rate usage data. If usage has exceeded rate limits, a value of 0 is returned. If the rate class does not exist, a value of -1 will be returned.

Optionally a context value can be used to check rate limits based on a context, for example, rate limits for a specific client IP address. See rate.use for more details on context.

Unlike rate.use(), this will not queue connections if the rate limit is exceeded.

Note that calling rate.use() after rate.use.noQueue() will mean that the connection is counted twice, halving the allowed rate.

Sample Usage

```
$use = rate.use.noQueue( "protect_database" );
# usage is over rate limits.
if( $use == 0 ){
   http.sendResponse( "503 Service Unavailable",
                      "text/html", "Go away",
                      "Retry-After: 10" );
   connection.discard();
} else if( $use > 0 ){ # usage is within rate limits
   log.info( "No queueing" );
} else { # Rate class does not exist
   log.info( "Rate class doesn't exist" );
}
```

See also: “rate.use(class_name, [context])”

Restrictions

Cannot be used in completion rules.

recentconns.exclude()

Specifies that this connection should not be saved in the Recent Connections buffer upon completion. This function can be used when the virtual server has recent_conns!save_all set to Yes to filter out unwanted connections from the Recent Connections page.

Note that recent_conns!enabled must be set to Yes on the virtual server for any connections to be saved.

Sample Usage

```
# Don't bother saving connections that completed
# within 1 second and returned a 200.
if( http.getResponseCode() == 200
   && stats.getTransactionDuration() < 1000 ) {
   recentconns.exclude();
}
```

See also: “recentconns.include()”, “requestlog.exclude()”

© 2019 Pulse Secure, LLC.

recentconns.include()

Mark this connection for inclusion in the Recent Connections buffer. Details about this connection can then be viewed on the Activity > Connections page.

Note that recent_conns!enabled must be set to Yes on the virtual server for any connections to be saved.

Returns true if the connection will be added to the Recent Connections buffer, or false otherwise.

Sample Usage

```
# Show server errors on the recent connections page
if( http.getResponseCode() >= 500 ){
   recentconns.include();
}
```

See also: “recentconns.exclude()”, “requestlog.include()”

recentconns.markedForInclusion()

Returns whether or not this connection will be added to the Recent Connections page upon completion.

Connections will be included on the Recent Connections page only if recent_conn!enabled is set to Yes for this virtual server and the connection has been marked for inclusion either by default, or by the recentconns.include() function.

Sample Usage

```
# If this rule will be added to the recent
# connections page, add some additional info
if( recentconns.markedForInclusion() ) {
   # Add the user's name to the log
   $user = connection.data.get( "user" );
   http.addResponseHeader( "X-Username", $user );
}
```

See also: “recentconns.include()”, “recentconns.exclude()”, “requestlog.markedForInclusion()”

request.avoidNode(node)

Indicates that the named node should be avoided if at all possible.

When picking a node to use for a request, the traffic manager will not use any nodes that have been named by request.avoidNode() unless session persistence mandates it, or unless there are no other nodes available.

Sample Usage

```
# if we get a 503 Too Busy response, retry
if( http.getResponseCode() == 503 ) {
   if( request.getRetries() < 3 ) {
      request.avoidNode( connection.getNode() );
      request.retry();
   }
}
```

See also: “request.retry()”, “request.getRetries()”, “connection.getNode()”

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

request.endsAt(offset)

Marks the end of the current request. Any more data read in from the network is not handled until the next request has started to be handled.

This function is useful to synchronise requests and responses. An example of its use would be for a line-oriented protocol such as POP3, where you wish to process each command.

It returns the entire request.

This function allows you to program layer-7 intelligence to correctly parse and manage generic TCP protocols.

Sample Usage

```
# get one line from input
$req = request.getLine();

# this is the end of the current request
request.endsAt( string.len( $req ) );

# Note: request.endsAt will return the request,
# but we've already got this in $req
```

See also: “request.endsWith(regex)”

Restrictions

Cannot be used in completion rules, Not available when the virtual server's internal protocol is 'generic streaming'.

request.endsWith(regex)

Marks the end of the current request. Any more data read in from the network is not handled until the next request has started to be handled.

This function is useful to synchronise requests and responses. An example of its use would be for a line-oriented protocol such as POP3, where you wish to process each command.

It returns the entire request.

This function allows you to program layer-7 intelligence to correctly parse and manage generic TCP protocols.

Sample Usage

```
# this is the end of the current request
$req = request.endsWith( "\n" );
```

See also: “request.endsAt(offset)”

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules, Not available when the virtual server's internal protocol is 'generic streaming'.

request.get([count])

Returns the first 'count' bytes of data provided by the client in the current request. If no count parameter is provided, all data read so far is returned, which may be none unless request.get() has previously been called with a positive count. If you cannot determine how much data to read, use request.getLine or request.endsWith instead.

When called in a request rule for a virtual server configured to use a protocol based on UDP, such as DNS (UDP), request.get will return the contents of the most recently received request datagram. When called in a response rule for a virtual server using UDP, request.get will return an empty string.

Warning: you can stall a connection by asking it to read more data than the remote client will provide. Combine this with request.getLength() or request.getLine() to reliably read data from a connection. For HTTP, you are required to use the HTTP specific functions like http.getBody() to read the request.

Sample Usage

```
# Get a length
$buf = request.get( 4 );
$l = string.bytesToInt( $buf );
# Now we know how much more data to ask for
$data = request.get( 4 + $l );
```

See also: “request.getLength()”, “request.getLine([regex], [offset])”, “request.endsWith(regex)”, “request.set(request data)”, “response.get([count])”

Restrictions

Cannot be used in completion rules.

request.getBandwidthClass()

Returns the current bandwidth class for the connection to the backend node, or an empty string if no class is set.

Sample Usage

```
$class = request.getBandwidthClass();
```

See also: “request.setBandwidthClass(name)”, “response.setBandwidthClass(name)”, “response.getBandwidthClass()”

request.getDSCP()

Returns the Differentiated Service Code Point (DSCP) field from the IP header of traffic being sent to the server. The return value is a either the DSCP value, or -1 if it could be not obtained from the socket. The return value is a 6-bit value.

© 2019 Pulse Secure, LLC.

Sample Usage

```
if ( request.getDSCP() == 46 ) {
  log.info( "Processing Expedited traffic" );
  connection.setServiceLevelClass( "gold" );
}
```

See also: “request.setDSCP(6-bit DSCP field)”, “response.getDSCP()”, “response.setDSCP(6-bit DSCP field)”

request.getDestIP()

Returns the original IP address that the client attempted to connect to. This will be the same as request.getLocalIP() unless the connection was redirected via firewall rules (e.g. using iptables on Linux)

Sample Usage

```
# Get the local IP address, such as "10.1.4.21" or
# "2001:200::8002:203:47ff:fea5:3085"
$ip = request.getDestIP();
```

See also: “request.getDestPort()”, “request.getRemoteIP()”, “request.getRemotePort()”

request.getDestPort()

Returns the original network port number that the client attempted to connect to. This will be the same as request.getLocalPort() unless the connection was redirected via firewall rules (e.g. using iptables on Linux)

Sample Usage

```
# Get the port number on the traffic manager,
# such as 80
$port = request.getDestPort();
```

See also: “request.getDestIP()”, “request.getRemotePort()”, “request.getRemoteIP()”

request.getLength()

Returns the number of bytes of data already received from the client. This can be combined with request.get() to reliably read data from a connection without stalling if no data is available.

Sample Usage

```
$data = request.get( request.getLength() );
```

See also: “request.get([count])”, “response.getLength()”

request.getLine([regex], [offset])

Returns a line of request data provided by the client. The line is terminated by the supplied regular expression, or by '\n'. If 'offset' is provided, request.getLine() returns the data from that offset to the terminating expression. The terminating expression is included in the returned string.

When request.getline() returns, the variable $1 is updated to point to the start of the next line in the datastream.

© 2019 Pulse Secure, LLC.

You can iterate through the lines of request data by using $1 as the iterator variable.

To prevent excessive data usage, if the line ending is not found within trafficscript!memory_warning bytes (configurable on the Global Settings page), then that many bytes will be returned.

If the regular expression is constructed from client supplied data, see the security considerations in the "Administration System Security" chapter of the User's Guide.

Sample Usage

```
# Process the lines in the request until an empty
# line is found
$line = request.getLine( "\n" );
while( $line != "\n" ) {
   # process $line...
   $line = request.getLine( "\n", $1 );
}
```

See also: “request.get([count])”, “response.getLine([regex], [offset])”

Restrictions

Not available when the virtual server's internal protocol is 'generic streaming'.

request.getLocalIP()

Returns the IP address that the client connected to, i.e. the address local to this machine.

Sample Usage

```
# Get the local IP address, such as "10.1.4.21" or
# "2001:200::8002:203:47ff:fea5:3085"
$ip = request.getLocalIP();
```

See also: “request.getLocalPort()”, “request.getDestIP()”, “request.getRemoteIP()”, “request.getRemotePort()”, “response.getLocalIP()”, “response.getLocalPort()”, “response.getRemoteIP()”, “response.getRemotePort()”

request.getLocalPort()

Returns the network port number that the client connected to. (e.g. port 80 is normal for a web server)

Sample Usage

```
# Get the local port, such as 80
$port = request.getLocalPort();
```

See also: “request.getLocalIP()”, “request.getRemotePort()”, “request.getDestPort()”, “request.getRemoteIP()”, “response.getLocalPort()”, “response.getLocalIP()”, “response.getRemotePort()”, “response.getRemoteIP()”

request.getLogEnabled() - deprecated

This function has been deprecated. Use instead “requestlog.markedForInclusion()”.

Returns 1 if logging is enabled for this request, and 0 otherwise.

request.getMaxTransactionDuration()

Returns the current value of the allowed maximum duration for this transaction. If the duration has been set by a previous call to setMaxTransactionDuration, that value will be returned, otherwise the relevant virtual server's max_transaction_duration config key. A value of 0 means unlimited duration.

Sample Usage

```
$max_duration = request.getMaxTransactionDuration();
if( $max_duration == 0 || $max_duration > 60 ) {
   # enforce absolute maximum for all requests:
   request.setMaxTransactionDuration( 60 );
}
```

See also: “request.setMaxTransactionDuration(int)”, “request.setVirtualServerTimeout(seconds)”, “request.setMaxReplyTime(seconds)”

Restrictions

Can only be used with TCP-based protocols.

request.getRemoteIP()

Returns the remote IP address of the client.

Sample Usage

```
# Get the remote IP address, such as "10.1.4.21"
# or "2001:200::8002:203:47ff:fea5:3085"
$ip = request.getRemoteIP();
```

See also: “request.getRemotePort()”, “request.getLocalIP()”, “request.setRemoteIP(ipAddr)”, “request.getLocalPort()”, “response.getRemoteIP()”, “response.getRemotePort()”, “response.getLocalIP()”, “response.getLocalPort()”

request.getRemotePort()

Returns the remote network port of the client's connection.

Sample Usage

```
# Get the remote port, such as 20427
$port = request.getRemotePort();
```

See also: “request.getRemoteIP()”, “request.getLocalPort()”, “request.getLocalIP()”, “response.getRemotePort()”, “response.getRemoteIP()”, “response.getLocalPort()”, “response.getLocalIP()”

request.getRetries()

Returns the number of times that this request has been explicitly retried by request.retry().

© 2019 Pulse Secure, LLC.

Sample Usage

```
$code = http.getResponseCode();
if( $code == 404 ||
   ( $code >= 500 && $code != 503 ) ) {
   # Not retrying 503s here, because they get retried
   # automatically before response rules are run
   if( request.getRetries() < 3 ) {
      # Avoid the current node when we retry,
      # if possible
      request.avoidNode( connection.getNode() );
      request.retry();
   }
}
```

See also: “request.retry()”, “request.isResendable()”, “pool.select(Pool, [Host, Port])”

request.getToS() - deprecated

This function has been deprecated. Use instead “request.getDSCP()”.

NOTE: RFC 2474 has superseded IP ToS values with the DSCP field. Returns the Type of Service (ToS) of traffic going to the server. The return value is either "LOWDELAY", "THROUGHPUT", "RELIABILITY" or "NONE".

request.isResendable()

Test if it is possible to resend this request to a different node. It is only possible to resend a request if the entire request has been buffered up in the traffic manager, for example, by explicitly reading it in a request rule.

If the request was streamed through to the client, for example, as a large HTTP POST, it will not have been buffered and therefore cannot be resent.

Note that request.isResendable detects if it is possible to resend a request; request.setIdempotent can be used to tell the traffic manager not to automatically resend a request if it fails.

Sample Usage

```
if( request.isResendable() ) {
   log.info( "Retrying request" );
   request.retry();
}
```

See also: “request.retry()”, “request.getRetries()”, “request.setIdempotent(resend)”

request.retry()

Retry the request (using the currently selected pool). Load-balancing and session persistence decisions are recalculated, and the request is resubmitted - possibly to the same node as previously, although request.avoidNode() can prevent this.

If request.retry() is called, any request rules are not run again. When a new response is collected after request.retry(), the response rules are run again.

© 2019 Pulse Secure, LLC.

The response rule can modify the request in before resubmitting it.

It is only generally possible to resend a request if the entire request was read before the request rules completed. Otherwise, request data will have been streamed to the server and not cached. Use request.isResendable() to test for this.

request.getRetries() returns the number of times this request has already been tried.

On success, request.retry() does not return, but the response rules will be run again on the new response. On failure, request.retry() returns 0. request.retry() will do nothing if used in a request rule.

Sample Usage

```
$code = http.getResponseCode();
if( $code == 404 ||
   ( $code >= 500 && $code != 503 ) ) {
   # Not retrying 503s here, because they get retried
   # automatically before response rules are run
   if( request.getRetries() < 3 ) {
      # Avoid the current node when we retry,
      # if possible
      request.avoidNode( connection.getNode() );
      request.retry();
   }
}
```

See also: “request.isResendable()”, “request.getRetries()”, “pool.select(Pool, [Host, Port])”

Restrictions

Cannot be used in completion rules.

request.sendResponse(Data)

Writes the provided data directly back to the client.

Any data that has been read is discarded, and nothing is forwarded to the back-end node. Unlike its counterpart http.sendRequest(), this function does not terminate rule processing immediately. Instead, it stores the provided string for sending back to the client when rule processing has finished.

If you are managing HTTP traffic the http.sendResponse() function should be used instead.

Sample Usage

```
# Send a response
request.sendResponse( "530 Login incorrect\r\n" );
```

See also: “connection.discard()”, “http.sendResponse(code, type, body, headers)”, “response.set(response data)”, “response.append(response data)”, “connection.close(Data, [Read])”

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

request.set(request data)

Replaces the input data read from the client with the provided string.

This is a low-level routine that should be used with care. For protocols with their own higher-level protocol managers (e.g. HTTP), use the higher level routines to alter the input data (e.g. http.setHeader() and http.setBody()).

Sample Usage

```
$data = request.get();
$data = string.regexsub( $data, "From", "To", "g" );
request.set( $data );
```

See also: "request.get([count])", "response.set(response data)"

Restrictions

Cannot be used in completion rules.

request.setBandwidthClass(name)

Sets the bandwidth class for the current connection to the backend node. Providing an empty class name removes the bandwidth class from the connection. It returns zero if an error occurs (for example, if the bandwidth class does not exist), and 1 otherwise.

Sample Usage

```
request.setBandwidthClass( "gold customers" );
```

See also: "request.getBandwidthClass()", "response.setBandwidthClass(name)", "response.getBandwidthClass()"

Restrictions

Cannot be used in completion rules.

request.setDSCP(6-bit DSCP field)

Sets the Differentiated Service Code Point (DSCP) field in the IP packet header of traffic being sent to the server. A 6-bit value must be provided. If successful, this function returns true, otherwise it returns false. DSCP fields can be used by network equipment to change how they route network traffic.

Sample Usage

```
if ( http.getHeader( "X-Set-DSCP-Priority" != "" )) {
   request.setDSCP( 0x2c );
}
```

See also: "request.getDSCP()", "response.getDSCP()", "response.setDSCP(6-bit DSCP field)"

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

request.setIdempotent(resend)

Marks a request as resendable or non-resendable.

An idempotent request has no detrimental side effects, so it can safely be attempted multiple times. A non-idempotent request has a side effect - for example, it may update a database, or initiate a purchase.

By default, all non-HTTP requests are marked as idempotent. If a back-end node fails to generate a correct response when a request is initially forwarded to it, an attempt will be made to resend the request to another node. An exception to this is requests received through a virtual server using one of the generic-type protocols. In order to be idempotent by default, an end-point to the request must first be defined (using functions such as request.endsWith() or request.endsAt()). Only then can failures be measured and alternative nodes tried.

request.setIdempotent() can override this behaviour. If 'resend' is zero, this indicates that the request should only be attempted against one back-end node.

If 'resend' has a non-zero value, this indicates that if a request is made to a back-end node and a correct response is not received, the request should be retried against another back-end node.

Note that a request cannot be resent once it has begun streaming data between the client and the node. Additionally, UDP connections cannot be marked as resendable (the UDP client application should handle failed UDP responses).

Sample Usage

```
# Mark this request as resendable
request.setIdempotent( 1 );
```

See also: “http.setIdempotent(resend)”

Restrictions

Cannot be used in completion rules.

request.setLogEnabled(enabled) - deprecated

This function has been deprecated. Use instead “requestlog.include()”.

Enables or disables logging for the current request. Note that if logging for the current virtual server is disabled, then this function cannot currently enable it.

Returns 1 if logging is now enabled, and 0 if it is now disabled.

request.setMaxConnectionAttempts(int)

Overrides the pool's maximum number of connection attempts for this request. Connections to different nodes will be attempted until the maximum number of attempts is exceeded or the connection is successful. A value of 0 means unlimited attempts.

© 2019 Pulse Secure, LLC.

Sample Usage

```
if( http.getMethod() == "POST" ) {
   # This is the same as request.setIdempotent( 0 )
   request.setMaxConnectionAttempts( 1 );
}
```

See also: "request.setMaxReplyTime(seconds)", "request.setMaxTimedOutConnectionAttempts(int)"

Restrictions

Can only be used with TCP-based protocols, Cannot be used in completion rules.

request.setMaxReplyTime(seconds)

Overrides the pool's max reply time for this request, also overriding the virtual server timeout if necessary. A node must start its reply within this time or it will be timed out.

Sample Usage

```
if( http.getMethod() == "POST" ) {
   # Allow more time to send data to server.
   request.setMaxReplyTime( 60 );
}
```

See also: "request.setVirtualServerTimeout(seconds)"

Restrictions

Cannot be used in completion rules.

request.setMaxTimedOutConnectionAttempts(int)

Overrides the pool's maximum number of timed-out connection attempts for this request. If the connection to the node fails due to max_reply_time being exceeded, connections to other nodes will be attempted until the maximum number of attempts is exceeded or the connection is successful. A value of 0 means unlimited attempts.

Sample Usage

```
if( http.getMethod() == "POST" ) {
   request.setMaxTimedOutConnectionAttempts( 2 );
}
```

See also: "request.setMaxConnectionAttempts(int)"

Restrictions

Can only be used with TCP-based protocols, Cannot be used in completion rules.

request.setMaxTransactionDuration(int)

Overrides the virtual server's maximum duration for this transaction. A value of 0 means unlimited duration.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$ct_len = http.getHeader( "Content-Length" );
if( $ct_len > 1024 ) {
   # give client 10s per kB of uploaded body data
   $d = 10 * $ct_len / 1024;
   request.setMaxTransactionDuration( $d );
} else {
   # use virtual server default
}
```

See also: “request.getMaxTransactionDuration()”, “request.setVirtualServerTimeout(seconds)”, “request.setMaxReplyTime(seconds)”

Restrictions

Can only be used with TCP-based protocols, Cannot be used in completion rules.

request.setRemoteIP(ipAddr)

Sets the remote IP address of the client. This function should be used with care, as it will alter what is logged in request logs and the address seen by a back-end node in 'transparent' mode. 0 is returned if the IP address is invalid, and 1 otherwise.

Sample Usage

```
# Set the remote IP address, such as "10.1.4.21"
request.setRemoteIP( "10.1.4.21" );
request.setRemoteIP( "2001:200::3085" );
```

See also: “request.getRemoteIP()”

Restrictions

Cannot be used in completion rules.

request.setToS(Type of Service) - deprecated

This function has been deprecated. Use instead “request.setDSCP(6-bit DSCP field)”.

NOTE: RFC 2474 has superseded IP ToS values with the DSCP field. Sets the Type of Service (ToS) flags of traffic going to the server. Valid options are "LOWDELAY", "THROUGHPUT", "RELIABILITY" or "NONE". ToS flags may be used by network equipment to change how they route network traffic.

Restrictions

Cannot be used in completion rules.

request.setVirtualServerTimeout(seconds)

Overrides the virtual server's timeout for this request. The connection to the client will be closed after this period of time, regardless of what state it is in.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$path = http.GetPath();
if( string.startsWith( $path, "/downloads" ) ) {
   request.setVirtualServerTimeout( 60 );
}
```

See also: "request.setMaxReplyTime(seconds)"

Restrictions

Cannot be used in completion rules.

request.skip(count)

Removes the specified number of bytes from the start of the request provided by the client. This can be used in combination with request.get() and request.getLine() to stream data from a client, or to alter a request before passing it on to a server.

Successive calls to this function will remove further data.

Sample Usage

```
# Skip the first 1K of data
request.skip( 1024 );

# Now skip another 1K
request.skip( 1024 );
```

See also: "request.getLength()", "request.getLine([regex], [offset])", "request.set(request data)", "request.get([count])"

Restrictions

Cannot be used in completion rules.

requestlog.exclude()

Specifies that this connection should not be written to the request log upon completion. This function can be used when the virtual server has log!save_all set to Yes to filter out unwanted connections from the request log.

Note that log!enabled or syslog!enabled must be set to Yes on the virtual server for any connections to be logged.

Sample Usage

```
# Don't log requests from local IP addresses.
$ip = request.getRemoteIP();
if( string.ipmaskmatch( $ip, "10.0.0.0/8" ) ) {
   requestlog.exclude();
}
```

See also: "requestlog.include()", "recentconns.exclude()"

© 2019 Pulse Secure, LLC.

requestlog.include()

Mark this connection to be added to the request log for this virtual server when it has completed.

Note that log!enabled or syslog!enabled must be set to Yes on the virtual server for any connections to be logged.

Returns true if the connection will be logged, or false otherwise.

Sample Usage

```
# Record when the server is particularly slow to
# respond to a request
if( stats.getServerResponseTime() > 5000 ){
   requestlog.include();
}
```

See also: "requestlog.exclude()", "recentconns.include()"

requestlog.markedForInclusion()

Returns whether or not this connection will be written either to the disk-based request log or to the remote syslog upon completion.

Connections will be written to the request log only if log!enabled or syslog!enabled is set to Yes for this virtual server and the connection has been marked for inclusion either by default, or by the requestlog.include() function.

Sample Usage

```
if( requestlog.markedForInclusion() ) {
   http.addResponseHeader( "X-Logged", "Yes" );
}
```

See also: "requestlog.include()", "requestlog.exclude()", "recentconns.markedForInclusion()"

resource.exists(filename)

Checks whether or not the named file exists in ZEUSHOME/zxtm/conf/extra/. If it exists 1 is returned, 0 otherwise.

Sample Usage

```
# Test if the file exists
if( resource.exists( "myfile" ) ) {
   # ... process resource
}
```

See also: "resource.get(filename)", "resource.getMD5(filename)"

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

resource.get(filename)

Returns the contents of a named file stored in the ZEUSHOME/zxtm/conf/extra/ directory. If the file doesn't exist, then an empty string is returned. Note that subdirectories of conf/extra are not supported.

Resources are pre-loaded into memory, so this call does not cause the file to be reloaded.

Sample Usage

```
# Read the contents of the 'info' file and add them
# as a new header.
http.addheader( "X-Info", resource.get( "info" ));
```

Restrictions

Cannot be used in completion rules.

resource.getLines(filename)

Returns the contents of a named file stored in the ZEUSHOME/zxtm/conf/extra/ directory as an array. If the file doesn't exist, then an empty array is returned. Note that subdirectories of conf/extra are not supported.

Resources are pre-loaded into memory, so this call does not cause the file to be reloaded.

Sample Usage

```
# Read the contents of the 'info' file and add
# process them line-by-line
$info = resource.getLines( "info" );
foreach( $line in $info ) {
   # ...
}
```

Restrictions

Cannot be used in completion rules.

resource.getMD5(filename)

Returns the MD5 of the current contents of the file in ZEUSHOME/zxtm/conf/extra/. If the file doesn't exist, an empty string is returned.

File MD5s are cached to speed up this call.

Sample Usage

```
# Get the MD5 of the file
$md5 = resource.getMD5( "myfile" );
```

See also: “resource.exists(filename)”, “resource.get(filename)”

Restrictions

Cannot be used in completion rules.

resource.getMTime(filename)

Returns the time that the named file in ZEUSHOME/zxtm/conf/extra/ was last modified, in seconds since the epoch (i.e. UNIX time). If the file doesn't exist, 0 is returned.

Sample Usage

```
# Find out the time the file was last modified
$mtime = resource.getMTime( "myfile" );
```

See also: “resource.exists(filename)”, “resource.get(filename)”, “resource.getMD5(filename)”, “sys.timeToString(unixtime)”, “sys.time()”

Restrictions

Cannot be used in completion rules.

response.append(response data)

Appends the provided string to the response data.

This is a low-level routine that should be used with care. For protocols with their own higher-level protocol managers (e.g. HTTP), you must use the higher level routines to alter the input data (e.g. http.setResponseHeader() and http.setResponseBody()).

Sample Usage

```
response.append(
   "I always have to have the last word." );
```

See also: “response.set(response data)”

Restrictions

Cannot be used in completion rules.

response.close()

Immediately closes the connection to the back-end node. Any response data that has already been read from the server will be forwarded to the client, but no more response data will be read.

Your traffic manager will reconnect to a back-end node when it next needs to forward request data to it; the back-end node is specified by either calling pool.use() or pool.select() in a request rule, or by the default pool.

Sample Usage

```
if( $neednewnode ) {
   response.close();
   pool.use( "servers" );
}
```

See also: “connection.close(Data, [Read])”, “connection.discard()”, “pool.use(Pool, [Host, Port])”, “pool.select(Pool, [Host, Port])”

© 2019 Pulse Secure, LLC.

Restrictions

Can only be used with TCP-based protocols, Cannot be used in completion rules.

response.flush(count)

Transfers the first count bytes of the response back to the client. These bytes are removed from the underlying response buffer. If count is not specified, all current response data is flushed.

This function is useful in generic client- and server-first protocols, to synchronise responses with the next request. This may be necessary if your traffic manager is likely to respond directly to some requests, and the back-end node responds to others.

Sample Usage

```
# keep flushing response data until we get
# an empty line...

$res = response.getLine();

while( $res != "\n" ) {
   response.flush( string.len( $res ) );
   $res = response.getLine();
}

# the remainder of the response buffer will be
# flushed when all response rules complete
```

See also: “response.get([count])”, “response.getLine([regex], [offset])”

Restrictions

Can only be used with TCP-based protocols, Cannot be used in completion rules.

response.get([count])

Returns the first 'count' bytes of data provided by the server in the current response. If you do not supply a count parameter, then the entire response will be read in.

When called in a request rule for a virtual server configured to use a protocol based on UDP, such as SIP, response.get will return an empty string. When called in a response rule for a virtual server using UDP, response.get will return the most recently received request datagram. When called in a completion rule for a virtual server using UDP, response.get will return the most recently received request datagram, if any, or an empty string if a request datagram has been received more recently than the last response datagram, such as in a three way handshake.

Warning: you can stall a connection by asking it to read more data than the back-end server will provide. Combine this with response.getLength() or response.getLine() to reliably read data from a connection. For HTTP, you must use the HTTP specific functions like http.getResponseBody() to read the response.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Get the first 1K of data
$data = response.get( 1024 );
```

See also: “response.getLength()”, “response.getLine([regex], [offset])”, “response.set(response data)”, “request.get([count])”

response.getBandwidthClass()

Returns the current bandwidth class for the connection to the client, or an empty string if no class is set.

Sample Usage

```
$class = response.getBandwidthClass();
```

See also: “response.setBandwidthClass(name)”, “request.setBandwidthClass(name)”, “request.getBandwidthClass()”

response.getDSCP()

Returns the Differentiated Service Code Point (DSCP) field from the IP header of traffic being sent to the client. The return value is a either the DSCP value, or -1 if it could be not obtained from the socket. The return value is a 6-bit value.

Sample Usage

```
if ( response.getDSCP() == 46 ) {
  log.info( "Processing Expedited traffic" );
  connection.setServiceLevelClass( "gold" );
}
```

See also: “response.setDSCP(6-bit DSCP field)”, “request.getDSCP()”, “request.setDSCP(6-bit DSCP field)”

response.getLength()

Returns the amount of data already received from the server. This can be combined with response.get() to reliably read data from a connection without stalling if no data is available.

Sample Usage

```
$data = response.get( response.getLength() );
```

See also: “response.get([count])”, “request.getLength()”

response.getLine([regex], [offset])

Returns a line of response data provided by the server. The line is terminated by the supplied regular expression, or by '\n'. If 'offset' is provided, response.getLine() returns the data from that offset to the terminating expression.

When response.getLine() returns, the variable $1 is updated to point to the start of the next line in the datastream.

You can iterate through the lines of response data by using $1 as the iterator variable.

© 2019 Pulse Secure, LLC.

If the regular expression is constructed from client supplied data, see the security considerations in the "Administration System Security" chapter of the User's Guide.

Sample Usage

```
# Process the lines in the response until an empty
 # line is found
 $line = response.getLine( "\n" );
 while( $line != "\n" ) {
    # process $line...
    $line = response.getLine( "\n", $1 );
 }
```

See also: “response.get([count])”, “request.getLine([regex], [offset])”

Restrictions

Can only be used with TCP-based protocols, Not available when the virtual server's internal protocol is 'generic streaming'.

response.getLocalIP()

Returns the local IP address of the connection to the node in use, i.e. an IP address on the local machine that your traffic manager connected from. It returns the empty string if no connection exists.

Sample Usage

```
# Find the IP address we connected from, such as
# "10.1.4.21" or "2001:200::8002:203:30:40:3085"
$ip = response.getLocalIP();
```

See also: “response.getLocalPort()”, “response.getRemoteIP()”, “response.getRemotePort()”, “request.getLocalIP()”, “request.getLocalPort()”, “request.getRemoteIP()”, “request.getRemotePort()”

response.getLocalPort()

Returns the local port of the connection to the node in use, i.e. the port number on the local machine that the traffic manager connected from. It returns 0 if there is no current connection to a node.

Sample Usage

```
$port = response.getLocalPort();
```

See also: “response.getLocalIP()”, “response.getRemotePort()”, “response.getRemoteIP()”, “request.getLocalPort()”, “request.getLocalIP()”, “request.getRemotePort()”, “request.getRemoteIP()”

response.getRemoteIP()

Returns the remote IP address of the node used. If there is no current connection, it will return an empty string.

Sample Usage

```
# Get the IP address of the node used, such as
```

```
# "10.1.4.21" or "2001:200::8002:203:a:1:3085"
$ip = response.getRemoteIP();
```

See also: “response.getRemotePort()”, “response.getLocalIP()”, “response.getLocalPort()”, “request.getRemoteIP()”, “request.getRemotePort()”, “request.getLocalIP()”, “request.getLocalPort()”

response.getRemotePort()

Returns the network port number on which the traffic manager connected to a node. (e.g. port 80 is normal for a web server). If there is no current connection, it will return 0.

Sample Usage

```
$port = response.getRemotePort();
```

See also: “response.getRemoteIP()”, “response.getLocalPort()”, “response.getLocalIP()”, “request.getRemotePort()”, “request.getRemoteIP()”, “request.getLocalPort()”, “request.getLocalIP()”

response.getToS() - deprecated

This function has been deprecated. Use instead “response.getDSCP()”.

NOTE: RFC 2474 has superseded IP ToS values with the DSCP field. Returns the Type of Service (ToS) of traffic going to the client. The return value is either "LOWDELAY", "THROUGHPUT", "RELIABILITY" or "NONE".

response.set(response data)

Sets the server response to the provided string.

This is a low-level routine that should be used with care. For protocols with their own higher-level protocol managers (e.g. HTTP), you must use the higher level routines to alter the input data (e.g. http.setResponseHeader() and http.setResponseBody()).

Sample Usage

```
$data = response.get();
$data = string.regexsub( $data,
   "From: ", "To: ", "g" );
response.set( $data );
```

See also: “request.sendResponse(Data)”, “response.append(response data)”, “response.get([count])”, “request.set(request data)”

Restrictions

Cannot be used in completion rules.

response.setBandwidthClass(name)

Sets the bandwidth class for the current connection to the client. Providing an empty class name removes the bandwidth class from the connection. It returns zero if an error occurs (for example, if the bandwidth class does not exist), and 1 otherwise.

© 2019 Pulse Secure, LLC.

Sample Usage

```
response.setBandwidthClass( "gold customers" );
```

See also: “response.getBandwidthClass()”, “request.getBandwidthClass()”, “request.setBandwidthClass(name)”

Restrictions

Cannot be used in completion rules.

response.setDSCP(6-bit DSCP field)

Sets the Differentiated Service Code Point (DSCP) field in the IP packet header of traffic being sent to the client. A 6-bit value must be provided. If successful, this function returns true, otherwise it returns false. DSCP fields can be used by network equipment to change how they route network traffic.

Sample Usage

```
if ( http.getHeader( "X-Set-DSCP-Priority" != "" )) {
   response.setDSCP( 0x2c );
}
```

See also: “response.getDSCP()”, “request.setDSCP(6-bit DSCP field)”, “request.getDSCP()”

Restrictions

Cannot be used in completion rules.

response.setToS(Type of Service) - deprecated

This function has been deprecated. Use instead “response.setDSCP(6-bit DSCP field)”.

NOTE: RFC 2474 has superseded IP ToS values with the DSCP field. Sets the Type of Service (ToS) flags of traffic going to the client. Valid options are "LOWDELAY", "THROUGHPUT", "RELIABILITY" or "NONE". ToS flags may be used by network equipment to change how they route network traffic.

Restrictions

Cannot be used in completion rules.

rtsp.addRequestHeader(name, value)

Adds an RTSP header. If the header already exists, via a case-insensitive lookup, this value will be appended to the existing value.

Sample Usage

```
# Add a transport header
$my_header = "RTP/AVP/TCP; interleaved=0-1";
rtsp.addRequestHeader( "Transport", $my_header );
```

See also: “rtsp.setRequestHeader(name, value)”, “rtsp.getRequestHeader(name)”, “rtsp.removeRequestHeader(name)”

Restrictions

Cannot be used in completion rules.

rtsp.addResponseHeader(name, value)

Adds an RTSP header to the RTSP response that will be sent back to the client. If the header already exists in the response, via a case-insensitive lookup, this value will be appended to the existing value.

Sample Usage

```
# Add a new transport header
$my_header = "RTP/AVP/TCP; interleaved=0-1";
rtsp.addResponseHeader( "Transport", $my_header );
```

See also: “rtsp.setResponseHeader(name, value)”, “rtsp.getResponseHeader(name)”, “rtsp.removeResponseHeader(name)”, “rtsp.addRequestHeader(name, value)”

Restrictions

Cannot be used in completion rules.

rtsp.getMethod()

Returns the RTSP method that was used to make the request, such as SETUP or PLAY.

Sample Usage

```
# Direct DESCRIBE requests to separate pool
if( rtsp.getMethod() == "DESCRIBE" ) {
   # set pool to Describe pool
}
```

See also: “rtsp.setMethod(method)”

rtsp.getPath()

Returns the %-decoded path in the RTSP request URL

Sample Usage

```
# Retrieve the requested file
$file = rtsp.getPath();
```

See also: “rtsp.setPath(url)”, “rtsp.getRawURL()”

rtsp.getRawURL()

Returns the raw URL data provided by the client in the first line of the RTSP request, after the method and before the RTSP version specifier. No %-decoding is performed on the URL.

Sample Usage

```
$rawurl = rtsp.getRawURL();
if( string.contains( $rawurl, "../" ) ) {
```


© 2019 Pulse Secure, LLC.

```
   # Something suspicious here ...
   connection.discard();
}
```

See also: "rtsp.getPath()"

rtsp.getRequest()

Returns the full RTSP request and headers, but does not include any body data.

Sample Usage

```
# Get the complete rtsp request
$request = rtsp.getRequest();
```

See also: "rtsp.listRequestHeaderNames()", "rtsp.getRequestHeader(name)", "rtsp.getRequestBody([count])"

rtsp.getRequestBody([count])

Returns the body data of the request.

If the optional 'count' parameter is supplied, rtsp.getRequestBody() will only read and return this number of bytes. If count is 0, rtsp.getRequestBody() returns the entire request.

If the request has no body, then this returns an empty string. This function is not useable in response rules, as the body data of the request will no longer be accessible.

Sample Usage

```
# Read the entire request body
$body = rtsp.getRequestBody();
```

See also: "rtsp.getResponseBody([count])", "rtsp.setRequestBody(body)", "rtsp.getRequestBodyLines(count)"

Restrictions

Cannot be used in completion rules.

rtsp.getRequestBodyLines(count)

Splits the body data of the RTSP request into individual lines and returns an array of the data.

If the request has no body, then this returns an empty array.

Sample Usage

```
# Read the entire request body
$body = rtsp.getRequestBodyLines();

# Process it line-by-line
foreach( $line in $body ) {
   # ...
}
```

See also: “rtsp.getRequestBody([count])”, “rtsp.getResponseBodyLines(count)”, “rtsp.setRequestBody(body)”

Restrictions

Cannot be used in completion rules.

rtsp.getRequestHeader(name)

Returns the value of a named RTSP header in the RTSP request, or the empty string if the header does not exist. The header name lookup is case-insensitive.

Sample Usage

```
# Get the transport detail
$transport = rtsp.getRequestHeader( "Transport" );
```

See also: “rtsp.setRequestHeader(name, value)”, “rtsp.addRequestHeader(name, value)”, “rtsp.removeRequestHeader(name)”, “rtsp.getRequestHeaders()”

rtsp.getRequestHeaderNames() - deprecated

This function has been deprecated. Use instead “rtsp.listRequestHeaderNames()”.

Returns a list of all the headers that are present in the request.

The headers are returned as a single string, separated by spaces.

rtsp.getRequestHeaders()

Returns a hash containing all the header names in the request mapped to their values.

Sample Usage

```
# Show all the headers in the request
$headers = rtsp.getRequestHeaders();

foreach( $header in hash.keys( $headers ) ) {
   log.info( $header . "=" . $headers[$header] );
}
```

See also: “rtsp.getRequestHeader(name)”, “rtsp.setRequestHeader(name, value)”, “rtsp.addRequestHeader(name, value)”, “rtsp.removeRequestHeader(name)”, “rtsp.requestHeaderExists(names)”, “rtsp.getResponseHeaders()”

rtsp.getResponse()

Returns the full RTSP response and headers, but does not include any body data.

Sample Usage

```
# Get the complete rtsp response
$request = rtsp.getResponse();
```


© 2019 Pulse Secure, LLC.

See also: “rtsp.listResponseHeaderNames()”, “rtsp.getResponseHeader(name)”, “rtsp.getResponseBody([count])”

rtsp.getResponseBody([count])

Returns the body of the RTSP response. This could be an SDP response.

If the optional 'count' parameter is provided, rtsp.getResponseBody() will read and return the first 'count' bytes of the response. If count is 0, rtsp.getResponseBody() will return the entire response.

Sample Usage

```
# Read the entire response body
$body = rtsp.getResponseBody();
```

See also: “rtsp.getRequestBody([count])”, “rtsp.setResponseBody(body)”, “rtsp.getResponseBodyLines(count)”

Restrictions

Cannot be used in completion rules.

rtsp.getResponseBodyLines(count)

Splits the body data of the RTSP request into individual lines and returns an array of the data.

If the request has no body, then this returns an empty array.

Sample Usage

```
# Read the entire request body
$body = rtsp.getResponseBodyLines();

# Process it line-by-line
foreach( $line in $body ) {
   # ...
}
```

See also: “rtsp.getResponseBody([count])”, “rtsp.getRequestBodyLines(count)”, “rtsp.setResponseBody(body)”

Restrictions

Cannot be used in completion rules.

rtsp.getResponseCode()

Returns the status code from the first line of the RTSP response.

Sample Usage

```
# Is the status '200'
if( rtsp.getResponseCode() == 200 ) {
   # ...
```

```
}
```
See also: “rtsp.setResponseCode(code, [message])”

rtsp.getResponseHeader(name)

Returns the value of a RTSP header in the RTSP response, or the empty string if the header does not exist. The header name lookup is case-insensitive.

Sample Usage

```
# Get the transport header
$transport = rtsp.getResponseHeader( "Transport" );
```
See also: “rtsp.setResponseHeader(name, value)”, “rtsp.addResponseHeader(name, value)”, “rtsp.removeResponseHeader(name)”, “rtsp.responseHeaderExists(name)”, “rtsp.getRequestHeader(name)”, “rtsp.getResponseHeaders()”

rtsp.getResponseHeaderNames() - deprecated

This function has been deprecated. Use instead “rtsp.listResponseHeaderNames()”.

Returns a list of all the headers that are present in the response.

The headers are returned as a single string, separated by spaces.

rtsp.getResponseHeaders()

Returns a hash containing all the header names in the response mapped to their values.

Sample Usage

```
# Show all the headers in the response
$headers = rtsp.getResponseHeaders();

foreach( $header in hash.keys( $headers ) ) {
   log.info( $header . "=" . $headers[$header] );
}
```
See also: “rtsp.getResponseHeader(name)”, “rtsp.setResponseHeader(name, value)”, “rtsp.addResponseHeader(name, value)”, “rtsp.removeResponseHeader(name)”, “rtsp.responseHeaderExists(name)”, “rtsp.getRequestHeaders()”

rtsp.getVersion()

Returns the version of the RTSP protocol being used. It returns the version string in the RTSP/version specifier in the first line of the RTSP request, such as 'RTSP/1.0'.

Sample Usage

```
# Get the RTSP version
$version = rtsp.getVersion();
```


© 2019 Pulse Secure, LLC.

rtsp.listRequestHeaderNames()

Returns a list of all the headers that are present in the request.

The headers are returned as a an array.

Sample Usage

```
# Log all of the header names and values
$headers = rtsp.listRequestHeaderNames();
foreach( $header in $headers ) {
   log.info($header . "=" .
            rtsp.getRequestHeader($header));
}
```

See also: “rtsp.listResponseHeaderNames()”, “rtsp.getRequestHeader(name)”, “rtsp.getRequestHeaders()”

rtsp.listResponseHeaderNames()

Returns a list of all the headers that are present in the response.

The headers are returned as an array.

Sample Usage

```
# Log all of the header names and values
$headers = rtsp.listResponseHeaderNames();
foreach( $header in $headers ) {
   log.info($header . "=" .
            rtsp.getRequestHeader($header));
}
```

See also: “rtsp.getResponseHeader(name)”, “rtsp.removeResponseHeader(name)”, “rtsp.getResponseHeaders()”, “rtsp.listRequestHeaderNames()”

rtsp.redirect(path)

Sends back an RTSP 302 redirect response, which will send the client to a different URL. This is equivalent to rtsp.sendResponse("302 Moved Temporarily", "", "Location: " . $url);

Sample Usage

```
# Redirect requests for a particular file elsewhere
$path = rtsp.getPath();
if( $path == "specialfile" ) {
   rtsp.redirect( "rtsp://otherserver/" . $path );
}
```

See also: “rtsp.sendResponse(code, body, headers)”

Restrictions

Cannot be used in completion rules.

rtsp.removeRequestHeader(name)

Removes an RTSP header if it exists in the request. The header name lookup is case-insensitive.

Sample Usage

```
# Remove the transport header
rtsp.removeRequestHeader( "Transport" );
```

See also: "rtsp.setRequestHeader(name, value)", "rtsp.getRequestHeader(name)", "rtsp.addRequestHeader(name, value)"

Restrictions

Cannot be used in completion rules.

rtsp.removeResponseHeader(name)

Removes a RTSP header from the RTSP response. The header name is automatically translated to the correct case.

Sample Usage

```
# Remove the GET_PARAMETER header
rtsp.removeResponseHeader( "GET_PARAMETER" );
```

See also: "rtsp.setResponseHeader(name, value)", "rtsp.getResponseHeader(name)", "rtsp.addResponseHeader(name, value)", "rtsp.removeRequestHeader(name)"

Restrictions

Cannot be used in completion rules.

rtsp.requestHeaderExists(names)

Reports if a header exists or not. It is similar to rtsp.getRequestHeader(), but makes it possible to distinguish between a header not being present and a header having no value.

The header name lookup is case-insensitive.

It returns 1 if the header exists, and 0 if it does not.

Sample Usage

```
# Check for the Transport header
if( rtsp.requestHeaderExists( "Transport" ) ) {
   # Modify the transport options
}
```

rtsp.responseHeaderExists(name)

Reports if a named header exists in the RTSP response. It is similar to rtsp.getResponseHeader(), but makes it possible to distinguish between a header not being present and a header having no value.

© 2019 Pulse Secure, LLC.

The header name lookup is case-insensitive.

It returns 1 if the header exists, and 0 if it does not.

Sample Usage

```
# Test for the 'Transport' response header
if( rtsp.responseHeaderExists( "Transport" ) ) {
   # modify the parameters
}
```

See also: "rtsp.getResponseHeader(name)"

rtsp.sendResponse(code, body, headers)

Sends back an RTSP response to the client instead of balancing the request via a pool onto a node. It generates a correct RTSP response from the response code, body data and headers supplied. Multiple headers should be separated with \r\n.

Sample Usage

```
# Discard SET_PARAMETER requests if the server
# does not support it
if( rtsp.getMethod() == "SET_PARAMETER" ) {
   rtsp.sendResponse( "401 Unauthorised", "", "" );
}
```

Restrictions

Cannot be used in completion rules.

rtsp.setMethod(method)

Sets the RTSP method to use when forwarding the request via a pool to a node.

Sample Usage

```
# Force the client to send an OPTIONS packet
rtsp.setMethod( "OPTIONS" );
```

See also: "rtsp.getMethod()"

Restrictions

Cannot be used in completion rules.

rtsp.setPath(url)

Replaces the Path portion of the request URL with the supplied value.

Sample Usage

```
# Make customer view specific file:
rtsp.setPath( "myvideo.rm" );
```

See also: "rtsp.getPath()"

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

rtsp.setRequestBody(body)

Sets the request body for this RTSP request, replacing any request body already present.

This also updates the 'Content-Length' header in the request to the length of the new body data.

Sample Usage

```
$body = rtsp.getRequestBody( 0 );
$body = string.regexsub( $body, "Buy", "Sell", "g" );
rtsp.setRequestBody( $body );
```

See also: “rtsp.getRequestBody([count])”, “rtsp.setResponseBody(body)”

Restrictions

Cannot be used in completion rules.

rtsp.setRequestHeader(name, value)

Sets the value of a RTSP header, replacing any existing value if the header already exists.

Note that this function should not be used with the Connection header, i.e. setRequestHeader("Connection", value) since it may not give the expected results.

The header name lookup is case-insensitive.

Sample Usage

```
# Replace the transport header
$my_header = "RTP/AVP/TCP; interleaved=0-1";
rtsp.setRequestHeader( "Transport", $my_header );
```

See also: “rtsp.getRequestHeader(name)”, “rtsp.addRequestHeader(name, value)”, “rtsp.removeRequestHeader(name)”

Restrictions

Cannot be used in completion rules.

rtsp.setResponseBody(body)

Sets the response body for this RTSP response, replacing any response body already present.

This also updates the 'Content-Length' header in the response to the length of the new body data. If the server is still sending the original response body when this function is called, the connection to the server will be harmlessly dropped.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$body = rtsp.getResponseBody( 0 );
$body = string.regexsub( $body, "Buy", "Sell", "g" );
rtsp.setResponseBody( $body );
```

See also: "rtsp.getResponseBody([count])", "rtsp.setRequestBody(body)"

Restrictions

Cannot be used in completion rules.

rtsp.setResponseCode(code, [message])

Sets the status code and message in the first line of the RTSP response.

Sample Usage

```
# Stop clients receiving a particular file
$path = rtsp.getPath();
if( $path == "specialfile" ) {
   rtsp.setResponseCode( "401", "Unauthorised" );
}
```

See also: "rtsp.getResponseCode()"

Restrictions

Cannot be used in completion rules.

rtsp.setResponseHeader(name, value)

Sets a RTSP header in the RTSP response that will be sent back to the client. If the header already exists in the response, via a case-insensitive lookup, it will be replaced with this new value.

Note that this function should not be used with the Connection header, i.e. setResponseHeader("Connection", value) since it may not give the expected results.

Sample Usage

```
# Replace the transport header
$my_header = "RTP/AVP/TCP; interleaved=0-1";
rtsp.setResponseHeader( "Transport", $my_header );
```

See also: "rtsp.addResponseHeader(name, value)", "rtsp.getResponseHeader(name)", "rtsp.removeResponseHeader(name)", "rtsp.setRequestHeader(name, value)"

Restrictions

Cannot be used in completion rules.

rule.getName()

Returns the name of the currently executing rule.

Sample Usage

```
$rulename = rule.getname();
```

rule.getState()

Returns the state of the currently executing rule, either "REQUEST", "RESPONSE", "COMPLETION" or "GLBRESPONSE".

Sample Usage

```
$rulestate = rule.getstate();
```

sip.addRequestHeader(name, value, at_top)

Modifies the current SIP request, adding a SIP header with the supplied value. If the header already exists, then this value will be appended to the existing value. If at_top is set then the value will be prepended to the header. The header name lookup is case-insensitive.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

Sample Usage

```
# Add a priority header if it is missing
if( !sip.requestHeaderExists( "Priority" ) ) {
   sip.addRequestHeader( "Priority", "normal", 0 );
}
```

See also: “sip.setRequestHeader(name, value)”, “sip.getRequestHeader(name)”, “sip.removeRequestHeader(name)”, “sip.requestHeaderExists(name)”

Restrictions

Cannot be used in completion rules.

sip.addResponseHeader(name, value, at_top)

Adds a header to the SIP response that will be sent back to the client. If the header already exists in the response, via a case-insensitive lookup, then this value will be appended to the existing value. If at_top is set then the value will be prepended to the existing value.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

Sample Usage

```
# Use an internal webpage to see if the callee is
#  logged into their system
$loggedin = http.request.get(
                  "http://internal.example.com/".
                  "lookupuser=bob" );
if( $loggedin = "No" ) {
   # Set a warning if they are not
   sip.addResponseHeader( "Warning",
```


© 2019 Pulse Secure, LLC.

```
      "399 example.com User is not logged in. ".
      " Your call might not be answered.", 0 );
}
```

See also: “sip.setResponseHeader(name, value)”, “sip.getResponseHeader(name)”, “sip.removeResponseHeader(name)”, “sip.addRequestHeader(name, value, at_top)”

Restrictions

Cannot be used in completion rules.

sip.getMethod()

Returns the SIP method that was used to make the request, such as INVITE or REGISTER.

Sample Usage

```
# Direct REGISTER requests to Registrar
if( sip.getMethod() == "REGISTER" ) {
   # set pool to Registrar pool
}
```

See also: “sip.setMethod(method)”

sip.getRequest()

Returns the full SIP request and headers, but does not include any body data.

Sample Usage

```
# Get the full SIP headers
$request = sip.getRequest();
```

See also: “sip.listRequestHeaderNames()”, “sip.getRequestHeader(name)”

sip.getRequestBody()

Returns the data contained in the body of the request.

Sample Usage

```
# Get the inbound Session Description to check for
#  unsupported content
if( sip.getMethod() == "INVITE" ) {
   $body = sip.getRequestBody();
   # inspect packet body
}
```

See also: “sip.setRequestBody(body)”, “sip.getResponseBody()”, “sip.getRequestBodyLines()”

Restrictions

Cannot be used in completion rules.

sip.getRequestBodyLines()

Splits the body data of the SIP request into individual lines and returns an array of the data.

If the request has no body, then this returns an empty array.

Sample Usage

```
# Read the request body
$body = sip.getRequestBodyLines();

# See if it's an SDP
if( string.startswith( $body[0], "v=0" ) ) {
   # Process SDP data
}
```

See also: “sip.getRequestBody()”, “sip.getResponseBodyLines()”, “sip.setRequestBody(body)”

Restrictions

Cannot be used in completion rules.

sip.getRequestHeader(name)

Returns the named SIP header in the SIP request, or the empty string if the header does not exist. The header name lookup is case-insensitive.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

Sample Usage

```
# Get information about the UAC
#  originating this request
$uac = sip.getRequestHeader( "User-Agent" );

# this returns the same value
$uac = sip.getRequestHeader( "user-agent" );
```

See also: “sip.setRequestHeader(name, value)”, “sip.addRequestHeader(name, value, at_top)”, “sip.removeRequestHeader(name)”, “sip.requestHeaderExists(name)”, “sip.getRequestHeaders()”

sip.getRequestHeaderNames() - deprecated

This function has been deprecated. Use instead “sip.listRequestHeaderNames()”.

Returns a list of all the headers that are present in the request.

The headers are returned as a single string, separated by spaces.

sip.getRequestHeaders()

Returns a hash containing all the header names in the request mapped to their values.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Show all the headers in the request
$headers = sip.getRequestHeaders();

foreach( $header in hash.keys( $headers ) ) {
   log.info( $header . "=" . $headers[$header] );
}
```

See also: “sip.getRequestHeader(name)”, “sip.setRequestHeader(name, value)”, “sip.addRequestHeader(name, value, at_top)”, “sip.removeRequestHeader(name)”, “sip.requestHeaderExists(name)”, “sip.getResponseHeaders()”

sip.getRequestURI()

Returns the target of the SIP request.

Sample Usage

```
# Check a status file to see if this user
#  wants to accept calls.
if( sip.getRequestURI() == "sip:bob@example.com" ) {
   # see if this user is available
   $status = http.request.get(
      "http://www.example.com/status.cgi?user=bob" );
   if( $status != "available" ) {
      sip.sendResponse( "486",
                     "User is currently " . $status );
   }
}
```

See also: “sip.setRequestURI(uri)”

sip.getResponse()

Returns the full SIP response and headers, but does not include any body data.

Sample Usage

```
# Get the full SIP headers
$request = sip.getResponse();
```

See also: “sip.listResponseHeaderNames()”, “sip.getResponseHeader(name)”

sip.getResponseBody()

Returns the session description of the SIP response.

Sample Usage

```
# Read the entire response body
$sdp = sip.getResponseBody();
```

See also: “sip.setResponseBody(body)”, “sip.getRequestBody()”, “sip.getResponseBodyLines()”

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

sip.getResponseBodyLines()

Splits the body data of the SIP response into individual lines and returns an array of the data.

If the response has no body, then this returns an empty array.

Sample Usage

```
# Get the response body
$sdp = sip.getResponseBodyLines();

# Process it line-by-line
foreach( $line in $sdp ) {
   # ...
}
```

See also: “sip.getResponseBody()”, “sip.setResponseBody(body)”, “sip.getRequestBodyLines()”

Restrictions

Cannot be used in completion rules.

sip.getResponseCode()

Returns the status code from the first line of the SIP response.

Sample Usage

```
# Is the status '200'
if( sip.getResponseCode() == 200 ) {
   # ...
}
```

See also: “sip.setResponseCode(code, [message])”

sip.getResponseHeader(name)

Returns the value of a header in the SIP response, or the empty string if the header does not exist. The header name lookup is case-insensitive.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

Sample Usage

```
# Get the route all future requests will take
$rr = sip.getResponseHeader( "Record-Route" );
```

See also: “sip.setResponseHeader(name, value)”, “sip.addResponseHeader(name, value, at_top)”, “sip.removeResponseHeader(name)”, “sip.responseHeaderExists(name)”, “sip.getRequestHeader(name)”, “sip.getResponseHeaders()”

© 2019 Pulse Secure, LLC.

sip.getResponseHeaderNames() - deprecated

This function has been deprecated. Use instead "sip.listResponseHeaderNames()".

Returns a list of all the headers that are present in the response.

The headers are returned as a single string, separated by spaces.

sip.getResponseHeaders()

Returns a hash containing all the header names in the response mapped to their values.

Sample Usage

```
# Show all the headers in the response
$headers = sip.getResponseHeaders();

foreach( $header in hash.keys( $headers ) ) {
   log.info( $header . "=" . $headers[$header] );
}
```

See also: "sip.getResponseHeader(name)", "sip.setResponseHeader(name, value)", "sip.addResponseHeader(name, value, at_top)", "sip.removeResponseHeader(name)", "sip.responseHeaderExists(name)", "sip.getRequestHeaders()"

sip.getVersion()

Returns the version of the SIP protocol being used. It returns the version string in the SIP/version specifier in the first line of the SIP request, such as 'SIP/2.0'.

Sample Usage

```
# Get the SIP version
$version = sip.getVersion();
```

sip.listRequestHeaderNames()

Returns a list of all the headers that are present in the request.

The headers are returned as an array.

Sample Usage

```
# Log all of the header names and values
$headers = sip.listRequestHeaderNames();
foreach( $header in $headers ) {
   log.info($header . "=" .
            sip.getRequestHeader($header));
}
```

See also: "sip.getRequestHeader(name)", "sip.removeRequestHeader(name)", "sip.getRequest()", "sip.listResponseHeaderNames()", "sip.getRequestHeaders()"

sip.listResponseHeaderNames()

Returns a list of all the headers that are present in the response.

The headers are returned as an array.

Sample Usage

```
# Log all of the header names and values
$headers = sip.listResponseHeaderNames();
foreach( $header in $headers ) {
   log.info($header . "=" .
            sip.getResponseHeader($header));
}
```

See also: “sip.getResponseHeader(name)”, “sip.removeResponseHeader(name)”, “sip.listRequestHeaderNames()”, “sip.getResponseHeaders()”

sip.redirect(contact)

Sends back a 302 Moved Temporarily response. This response instructs the client to retry the request at the new address(es) given in the 'contact' parameter. This is equivalent to sip.sendResponse("302", "Moved Temporarily", "Contact: " . $uri, "");

Sample Usage

```
# Example's offices are closed, redirect all their
# calls to voicemail.
$user = sip.getRequestURI();
if( string.EndsWith( $user, "@example.com" ) ) {
   $username = string.left( $user,
                      string.find( $user, "@" ) );
   $contact = $username . "@voicemail.example.com";
   sip.redirect( $contact );
}
```

See also: “sip.sendResponse(code, reason, [headers], [body])”

Restrictions

Cannot be used in completion rules.

sip.removeRequestHeader(name)

Removes a header if it exists in the request. The header name lookup is case-insensitive.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

Sample Usage

```
# Filter out any custom alert tones that have
#  been specified by the client
sip.removeRequestHeader( "Alert-Info" );
```

See also: “sip.getRequestHeader(name)”, “sip.addRequestHeader(name, value, at_top)”, “sip.setRequestHeader(name, value)”

© 2019 Pulse Secure, LLC.

Restrictions

Cannot be used in completion rules.

sip.removeResponseHeader(name)

Removes a header from the SIP response.

The header name lookup is case-insensitive. You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

Sample Usage

```
# Remove the server header, if it exists, to avoid
#  application-specific exploits being used
sip.removeResponseHeader( "Server" );
```

See also: "sip.setResponseHeader(name, value)", "sip.getResponseHeader(name)", "sip.addResponseHeader(name, value, at_top)", "sip.removeRequestHeader(name)"

Restrictions

Cannot be used in completion rules.

sip.requestHeaderExists(name)

Reports if a named header exists or not. It is similar to sip.getRequestHeader(), but makes it possible to distinguish between a header not being present and a header having no value.

The header name lookup is case-insensitive.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

It returns 1 if the header exists, and 0 if it does not.

Sample Usage

```
# Add a Priority header if it is missing
if( !sip.requestHeaderExists( "Priority" ) ) {
   sip.addRequestHeader( "Priority", "normal", 0 );
}
```

See also: "sip.getRequestHeader(name)"

sip.responseHeaderExists(name)

Reports if a named header exists in the SIP response. It is similar to sip.getResponseHeader(), but makes it possible to distinguish between a header not being present and a header having no value.

The header name lookup is case-insensitive.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

It returns 1 if the header exists, and 0 if it does not.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Test for the 'Warning' response header
if( sip.responseHeaderExists( "Warning" ) ) {
   # ...
}
```

See also: "sip.getResponseHeader(name)"

sip.sendResponse(code, reason, [headers], [body])

Sends back a SIP response to the client instead of balancing the request via a pool onto a node. The Statue-Line of the response has the form: SIP/2.0 code reason Via, Record-Route, From, To, CSeq, Call-ID and Content-Length headers are automatically added to the response. Any headers supplied in the headers parameter will also be added to the response. Multiple headers must be separated by \r\n. Any body data specified is appended to the response.

Sample Usage

```
# Send Forbidden response if the user is blacklisted
$contact = sip.getRequestHeader( "Contact" );
if( string.contains( $contact, "10.234.12.42" ) ) {
   sip.sendResponse( "403", "Forbidden" );
}
```

See also: "sip.redirect(contact)"

Restrictions

Cannot be used in completion rules.

sip.setMethod(method)

Sets the SIP method to use when forwarding the request via a pool to a node.

Sample Usage

```
# Force non-standard PING requests to become OPTIONS
if( sip.getMethod() == "PING" ) {
   sip.setMethod( "OPTIONS" );
   sip.setRequestHeader( "Max-Forwards", "1" );
}
```

See also: "sip.getMethod()"

Restrictions

Cannot be used in completion rules.

sip.setRequestBody(body)

Sets the request body for this SIP request to the supplied string, replacing any request body already present.

This also updates the 'Content-Length' header in the request to the length of the new body data.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$body = sip.getRequestBody();
# Forward all data from the server through an
#  intermediate machine
$body = string.regexsub($body, request.getRemoteIP(),
                        "192.168.9.100", "g");
sip.setRequestBody( $body );
```

See also: "sip.getRequestBody()"

Restrictions

Cannot be used in completion rules.

sip.setRequestHeader(name, value)

Sets a SIP header, replacing any existing value if the header already exists.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

Sample Usage

```
# Add a reference to an information page about
#  a known company when a call is received from
#  them, and an icon to help identify them.
if( sip.getRequestHeader( "Organization" )
                                 == "Zeus" ) {
   sip.setRequestHeader( "Call-Info",
      "&lt;http://www.example.com/" .
      "assets/img/logo.gif&gt; ;purpose=icon,".
      "&lt;http://www.example.com/" .
      "about/&gt; ;purpose=info"
   );
}
```

See also: "sip.addRequestHeader(name, value, at_top)", "sip.getRequestHeader(name)", "sip.requestHeaderExists(name)", "sip.removeRequestHeader(name)"

Restrictions

Cannot be used in completion rules.

sip.setRequestURI(uri)

Sets the target of the SIP request.

Sample Usage

```
# If the user has recently changed username, rewrite
#  requests that address their old username.
if( sip.getRequestURI() == "sip:jond@example.com" ) {
   sip.setRequestURI( "sip:jdoe@example.com" );
}
```

See also: "sip.getRequestURI()"

Restrictions

Cannot be used in completion rules.

sip.setResponseBody(body)

Sets the response body for this SIP response, replacing any response body already present.

This also updates the 'Content-Length' header in the response to the length of the new body data. If the server is still sending the original response body when this function is called, the connection to the server will be harmlessly dropped.

Sample Usage

```
$body = sip.getResponseBody();
# Forward all data from the client through an
#  intermediate machine
$body = string.regexsub( $body,
                         "c=.*",
                         "c=IN IP4 "
                        ."192.168.9.100",
                         "g");
sip.setResponseBody( $body );
```

See also: “sip.getResponseBody()”, “sip.setRequestBody(body)”

Restrictions

Cannot be used in completion rules.

sip.setResponseCode(code, [message])

Sets the status code and message in the first line of the SIP response.

Sample Usage

```
# Redirect client to a backup server if the proxy
#  has an internal error.
if( sip.getResponseCode() == "500" ) {
   sip.setResponseHeader( "Contact",
               "sip:backup.example.com" );
   sip.setResponseCode( "305", "Use Proxy" );
}
```

See also: “sip.getResponseCode()”

Restrictions

Cannot be used in completion rules.

sip.setResponseHeader(name, value)

Sets a header in the SIP response that will be sent back to the client. If the header already exists in the response, via a case-insensitive lookup, then it will be replaced with this new value.

© 2019 Pulse Secure, LLC.

You can specify the long or short form of the header name, so 'Via' and 'v' will both match the Via header field.

Sample Usage

```
# Change the server string
sip.setResponseHeader( "Server",
        "Pulse Secure vTM 10.1 (Linux/i386)" );
```

See also: “sip.addResponseHeader(name, value, at_top)”, “sip.getResponseHeader(name)”, “sip.removeResponseHeader(name)”, “sip.setRequestHeader(name, value)”

Restrictions

Cannot be used in completion rules.

slm.conforming([class name])

Returns the current percentage of requests that are meeting the Service Level Monitoring objectives. If no class name is provided, it will use the SLM class currently associated with this connection; if no SLM class is associated with this connection, it returns 100.

Sample Usage

```
# If the Gold customers are starting to get slow,
# gradually reroute other services...
$conforming = slm.conforming( "gold requests" );
if( ( $level == "bronze" && $conforming < 70 ) ||
    ( $level == "silver" && $conforming < 50 )) {
  # tell lower value customers to come back later to
  # reduce load on back-end node to ensure premium
  # customers get good response
  http.sendResponse( "302", "text/html", "",
     "Location: /toobusy.html" );
}
if( $conforming < 80 && $level != "gold" ) {
  # slow down rate of responding to non-premium
  # customers
  connection.sleep( 500 );
}
```

See also: “slm.threshold([class_name])”, “slm.isOK([class_name])”

slm.isOK([class_name])

Returns whether a particular Service Level Monitoring class is meeting its objectives. If no class name is provided, it will use the SLM class currently associated with this connection; if no SLM class is associated with the connection, it returns 1. This function is a convenience shorthand for 'slm.conforming() > slm.threshold()'.

Sample Usage

```
# If the search nodes are under-utilised, use this
# spare capacity to process web page requests too
if( slm.isOK( "search-nodes" )) {
  pool.use( "search+web nodes");
```

```
} else {
  pool.use( "web nodes" );
}
```

See also: "slm.conforming([class name])", "slm.threshold([class_name])"

slm.threshold([class_name])

returns the value of the serious_threshold setting in the given SLM class. If no class name is provided, it will use the SLM class currently associated with this connection; if no SLM class is associated with the connection, it returns 0.

Sample Usage

```
# If we are within 10% of our threshold, divert a
# portion of traffic elsewhere.  If we are less than
# 25% below of our threshold, take evasive action to
# get our site's end user experience under control!

if( slm.conforming() < slm.threshold()*0.75 ) {
   # evasive action!
   log.warn(
     "Site performance requires evasive action" );
   if( $level == "bronze" ) {
      # send away low priority traffic
      http.sendResponse( "302", "text/html", "",
        "Location: /toobusy.html" );
   } else if( $level == "silver" ) {
      # slow down processing of medium traffic
      connection.sleep( 500 );
   } else if( $level == "gold" ) {
      # use reserved bandwidth QoS
      response.setBandwidthClass(
        "premium reserved bandwidth" );
  }
} else if( slm.conforming() < slm.threshold()*1.1 ) {
   # getting slow to the danger level; start
   # proactive traffic management
   if( $customer != "gold" ) {
     pool.use( "non-priority-nodes" );
  }
}
```

See also: "slm.conforming([class name])", "slm.isOK([class_name])"

ssl.clientCert()

Returns the PEM encoded client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate data
$cert = ssl.clientCert();
log.info( "Certificate: " . $cert );
```


© 2019 Pulse Secure, LLC.

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertAlgorithm()

If the connection is SSL-encrypted and the client has supplied a valid certificate, then this returns either "rsaEncryption", "md2withRSAEncryption", "md5withRSAEncryption", "sha1withRSAEncryption", "RSA", "sha256withRSAEncryption", "sha384withRSAEncryption", "sha512withRSAEncryption", "DSA with SHA-1", "DSA with SHA-224", "DSA with SHA-256", "ECDSA with SHA-1", "ECDSA with SHA-224", "ECDSA with SHA-256", "ECDSA with SHA-384", or "ECDSA with SHA-512" depending on the certificate's encryption and hash algorithms. Otherwise, it returns the empty string.

Sample Usage

```
# Display the client certificate algorithm
$alg = ssl.clientCertAlgorithm();
log.info( "Certificate alg: ".$alg );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertChain()

Returns a multi-line string, with each line being the PEM encoding of a certificate in the chain from the client, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate data
$cert = ssl.clientCertChain();
log.info( "Certificate Chain: " . $cert );
```

See also: "ssl.isSSL()", "ssl.clientCert()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertCommonName()

Returns the common name of the client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
$common_name = ssl.clientCertCommonName();
log.info( "Certificate common name: ".$common_name );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertEndDate()

If the connection is SSL-encrypted and the client has supplied a valid certificate, then this returns the date when the certificate is no longer valid. The date is an integer, representing seconds since the epoch.

Otherwise, this function returns 0.

Sample Usage

```
# Display the client certificate end date
$end = ssl.clientCertEndDate();
log.info( "Certificate is valid until ".
   sys.timeToString( $end ) );
```

See also: "ssl.isSSL()", "ssl.clientCertStartDate()", "sys.timeToString(unixtime)"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertHash()

Returns a hex-encoded MD5 hash of the client certificate, or the empty string if the connection was not SSL-encrypted, if a certificate was not supplied or if the MD5 algorithm is unavailable (e.g. when operating in FIPS Mode).

Sample Usage

```
# Display the client certificate hash
$hash = ssl.clientCertHash();
log.info( "Certificate hash: ".$hash );
```

See also: "ssl.isSSL()", "ssl.clientCertSHA1()", "sys.isFIPS()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertIssuer()

Returns a string representing the issuer of the client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate issuer
$issuer = ssl.clientCertIssuer();
log.info( "Certificate issuer: ".$issuer );
```


© 2019 Pulse Secure, LLC.

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertPublicKey()

Returns a string representation of the public key of the client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate key
$key = ssl.clientCertPublicKey();
log.info( "Certificate key: ".$key );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertSHA1()

Returns a hex-encoded SHA1 fingerprint of the client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate SHA1 fingerprint
$sha1 = ssl.clientCertSHA1();
log.info( "Certificate SHA1 fingerprint: " . $sha1 );
```

See also: "ssl.isSSL()", "ssl.clientCertHash()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertSerial()

Returns the serial (in hex) of the client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate serial
$serial = ssl.clientCertSerial();
log.info( "Certificate serial: ".$serial );
```

See also: "ssl.clientCertSerialDec()"

© 2019 Pulse Secure, LLC.

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertSerialDec()

Returns the serial (in decimal) of the client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate serial in decimal
$serial = ssl.clientCertSerialDec();
log.info( "Certificate serial: ".$serial );
```

See also: “ssl.clientCertSerial()”

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertStartDate()

If the connection is SSL-encrypted and the client has supplied a valid certificate, then this returns the date when the certificate became valid. The date is an integer, representing seconds since the epoch.

Otherwise, this function returns 0.

Sample Usage

```
# Display the client certificate start date
$start = ssl.clientCertStartDate();
log.info( "Certificate is valid from ".
   sys.timeToString( $start ) );
```

See also: “ssl.isSSL()”, “ssl.clientCertEndDate()”, “sys.timeToString(unixtime)”

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertStatus()

Returns 'OK' if the client certificate is valid, or 'NoClientCert' if it was missing or not valid. It returns the empty string if the connection was not SSL-encrypted.

Sample Usage

```
if( ssl.clientCertStatus() != "OK" ) {
   # Handle missing client certificate ...
}
```

See also: “ssl.isSSL()”

© 2019 Pulse Secure, LLC.

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertSubject()

Returns a string representing the subject of the client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate subject
$subject = ssl.clientCertSubject();
log.info( "Certificate subject: ".$subject );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCertVersion()

Returns "1", "2" or "3" denoting the version of the client certificate, or the empty string if the connection was not SSL-encrypted or if a certificate was not supplied.

Sample Usage

```
# Display the client certificate version
$version = ssl.clientCertVersion();
log.info( "Certificate version: ".$version );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.clientCipher()

Returns the cipher used by the client to SSL-encrypt the connection. It returns an empty string if the connection was not SSL-encrypted.

The string returned contains the cipher algorithm, SSL version and effective cipher strength, such as:

SSL_RSA_WITH_AES_128_GCM_SHA256, version=TLSv1.2, bits=128

Sample Usage

```
# Get the encryption cipher
$cipher = ssl.clientCipher();
log.info( "Encrypted with ".$cipher );
```

See also: "ssl.isSSL()"

© 2019 Pulse Secure, LLC.

Restrictions

Can only be used with TCP-based protocols.

ssl.clientSupportsSecureRenegotiation()

Returns true if the client is RFC 5746 compliant, else false.

Sample Usage

```
if( ssl.isSSL() ) {
   # gather statistics about client-side support
   # for RFC 5746
   if( ssl.clientSupportsSecureRenegotiation() ) {
      counter.increment( 1 );
   } else {
      counter.increment( 2 );
   }
}
```

See also: “ssl.requireCert()”, “ssl.requestCert()”, “ssl.isSSL()”

Restrictions

Can only be used with TCP-based protocols.

ssl.getClientCloseAlert()

Check whether your traffic manager will send the SSL client an SSL close alert prior to terminating the TCP connection. The function will return a value of 1 if close alerts are enabled, and 0 if they are disabled. The function will return -1 if the client-side connection is not established, or is not an SSL connection.

Sample Usage

```
if( ssl.isSSL() ) {
   $alert = ssl.getClientCloseAlert();
   log.info( "client close alert: " . $alert );
}
```

See also: “ssl.isSSL()”, “ssl.setClientCloseAlert(alertflag)”, “ssl.setServerCloseAlert(alertflag)”, “ssl.getServerCloseAlert()”

Restrictions

Can only be used with TCP-based protocols.

ssl.getServerCloseAlert()

Check whether your traffic manager will send the SSL server an SSL close alert prior to terminating the TCP connection. The function will return a value of 1 if close alerts are enabled, and 0 if they are disabled. The function will return -1 if the connection is not an SSL connection, or if the server-side connection is not yet established (i.e. it should typically only be used in response rules.)

© 2019 Pulse Secure, LLC.

Sample Usage

```
if( ssl.isSSL() ) {
   $alert = ssl.getServerCloseAlert();
   log.info( "server close alert: " . $alert );
}
```

See also: "ssl.isSSL()", "ssl.setServerCloseAlert(alertflag)", "ssl.setClientCloseAlert(alertflag)", "ssl.getClientCloseAlert()"

Restrictions

Can only be used with TCP-based protocols.

ssl.getTLSServerName()

Returns the hostname provided by the client using the TLS 1.0 'server_name' extension. If this connection is not using SSL decryption or the client did use the extension then this function returns the empty string.

Sample Usage

```
$name = ssl.getTLSServerName();
log.info( "The client provided " .
          "the server name: " . $name );
```

See also: "ssl.isSSL()", "ssl.setTLSServerName(servername)"

Restrictions

Can only be used with TCP-based protocols.

ssl.isSSL()

Returns 1 if this connection from the remote client was SSL encrypted and your traffic manager has decrypted the traffic. Otherwise, it returns 0.

Sample Usage

```
if( ssl.isSSL() ) {
   # This is an SSL-encrypted connection ...
}
```

Restrictions

Can only be used with TCP-based protocols.

© 2019 Pulse Secure, LLC.

ssl.requestCert()

Initiates an SSL renegotiation with the client, requesting a certificate. Even if the client fails to provide a certificate, rule processing continues after the re-handshake is complete. If the client had already provided a certificate, this function does nothing and rule processing continues. If the underlying connection is not SSL, the transaction is aborted (i.e., this is interpreted as a failed re-handshake). If the underlying connection is HTTP/2, renegotiation is not attempted. (i.e. this function has no effect). Secure renegotiation is forbidden by HTTP/2. Note that using this function overrides the setting ssl!allow_rehandshake. It is strongly recommended to check for client-side support of RFC 5746 before using it, see ssl.clientSupportsSecureRenegotiation.

Sample Usage

```
$path = http.getPath();
if( string.startsWith( $path, "/wikileaks" ) ) {
   # Only let certain people see this data ...
   if( ssl.clientSupportsSecureRenegotiation() ) {
      ssl.requestCert();
      if( ssl.clientCertStatus() != "OK" ) {
         http.sendResponse( 403, "text/plain",
            "Valid SSL credentials are required",
            "" );
      } else {
         # allow the original request to
         # continue, client provided an
         # acceptable certificate.
         break;
      }
   } else {
      http.sendResponse( 403, "text/plain",
         "Please upgrade your browser", "" );
   }
}
```

See also: “ssl.clientSupportsSecureRenegotiation()”, “ssl.requireCert()”

Restrictions

Can only be used with TCP-based protocols, Cannot be used in completion rules.

ssl.requireCert()

Initiates an SSL renegotiation with the client, requiring a certificate. If the client fails to provide a valid cert, the connection is closed with an SSL alert of type handshake_failure. Otherwise, rule processing continues after the re-handshake is complete. If the client had already provided a certificate, this function does nothing and rule processing continues. If the underlying connection is not SSL, the transaction is aborted (i.e., this is interpreted as a failed re-handshake). If the underlying connection is HTTP/2, the transaction is aborted, and the client is sent the error HTTP_1_1_REQUIRED. Secure renegotiation is forbidden by HTTP/2. Note that using this function overrides the setting ssl!allow_rehandshake. It is strongly recommended to check for client-side support of RFC 5746 before using it, see ssl.clientSupportsSecureRenegotiation.

© 2019 Pulse Secure, LLC.

Sample Usage

```
$path = http.getPath();
if( string.startsWith( $path, "/wikileaks" ) ) {
   # Only let certain people see this data ...
   if( ssl.clientSupportsSecureRenegotiation() ) {
      ssl.requireCert();
   } else {
      http.sendResponse( 403, "text/plain",
         "Please upgrade your browser", "" );
   }
}
```

See also: "ssl.clientSupportsSecureRenegotiation()", "ssl.requestCert()"

Restrictions

Can only be used with TCP-based protocols, Cannot be used in completion rules.

ssl.serverCert()

Returns a PEM encoded version of the entire certificate being used by your traffic manager for this connection. If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$cert = ssl.serverCert();
log.info( "Server certificate: " . $cert );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertAlgorithm()

Returns a description of the algorithms being used by the virtual server's current certificate. This will either be "rsaEncryption", "md2withRSAEncryption", "md5withRSAEncryption", "sha1withRSAEncryption", "RSA", "sha256withRSAEncryption", "sha384withRSAEncryption", "sha512withRSAEncryption", "DSA with SHA-1", "DSA with SHA-224", "DSA with SHA-256", "ECDSA with SHA-1", "ECDSA with SHA-224", "ECDSA with SHA-256", "ECDSA with SHA-384" or "ECDSA with SHA-512". If this virtual server is not using SSL decryption (or the algorithm type is not recognised) then this function will return the empty string.

Sample Usage

```
$alg = ssl.serverCertAlgorithm();
log.info( "Server cert algorithm: " . $alg );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

© 2019 Pulse Secure, LLC.

ssl.serverCertCommonName()

Returns the common name of the certificate being used by your traffic manager for this connection. If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$common_name = ssl.serverCertCommonName();
if( http.getHostHeader() != $common_name ) {
   log.warn( "Client browser may report certificate".
      " as invalid." );
}
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertEndDate()

Returns the date that the certificate being used by your traffic manager for this connection became valid. The date is an integer, representing the number of seconds since the epoch. If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$end = ssl.serverCertEndDate();
if( $time < sys.time() ) {
   log.warn( "Using out of date certificate!" );
}
```

See also: "ssl.isSSL()", "ssl.serverCertStartDate()", "sys.time()", "sys.timeToString(unixtime)"

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertHash()

Returns the hex-encoded MD5 hash of the certificate being used by your traffic manager for this connection. If this virtual server is not using SSL decryption or if the MD5 algorithm is unavailable (e.g. when operating in FIPS Mode) then this function will return the empty string.

Sample Usage

```
$md5 = ssl.serverCertHash();
log.info( "Server Cert Hash: " . $md5 );
```

See also: "ssl.isSSL()", "ssl.serverCertSHA1()", "sys.isFIPS()"

Restrictions

Can only be used with TCP-based protocols.

© 2019 Pulse Secure, LLC.

ssl.serverCertIssuer()

Returns a string with the issuer data of the certificate being used by your traffic manager for this connection (each field is separated by commas). If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$issuer = ssl.serverCertIssuer();
if( ssl.serverCertSubject() == $issuer ) {
   log.info( "Certificate is self-signed" );
}
```

See also: "ssl.isSSL()", "ssl.serverCertSubject()"

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertName()

Returns the name of the certificate being used by the virtual server for this connection. This is the name used to identify the certificate in the UI. If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$name = ssl.serverCertName();
log.info( "Using server cert: " . $name );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertPublicKey()

Returns information about the public key of the certificate being used by your traffic manager for this connection. If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
# Returns public key information,
# e.g.  "RSA (1024 bit)"
$public_key = ssl.serverCertPublicKey();
log.info( "Certificate public key: " . $public_key );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

© 2019 Pulse Secure, LLC.

ssl.serverCertSHA1()

Returns the hex-encoded SHA1 fingerprint of the certificate being used by your traffic manager for this connection. If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$sha1 = ssl.serverCertSHA1();
log.info( "Server Cert SHA1 Fingerprint: " . $sha1 );
```

See also: "ssl.isSSL()", "ssl.serverCertHash()"

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertSerial()

Returns a string with the serial number (in hex) of the certificate being used by your traffic manager for this connection. If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$serial = ssl.serverCertSerial();
log.info( "Server certificate serial: " . $serial );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertStartDate()

Returns the date the certificate being used by your traffic manager for this connection became valid. The date is an integer, representing the number of seconds since the epoch. If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$start = ssl.serverCertStartDate();
log.info( "Server Certificate is valid from ".
    sys.timeToString( $start ) );
```

See also: "ssl.isSSL()", "ssl.serverCertEndDate()", "sys.time()", "sys.timeToString(unixtime)"

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertSubject()

Returns the subject data of the certificate being used by your traffic manager for this connection (each field is separated by commas). If this virtual server is not using SSL decryption then this function will return the empty string.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Returns the subject information of the cert,
# e.g. C=GB, L=Cambridge, O=Zeus, CN=foo.com
$subject = ssl.serverCertSubject();
log.info( "Server cert subject: " . $subject );
```

See also: “ssl.isSSL()”, “ssl.serverCertIssuer()”

Restrictions

Can only be used with TCP-based protocols.

ssl.serverCertVersion()

Returns the version of the certificate being used by your traffic manager for this connection (either "1", "2" or "3"). If this virtual server is not using SSL decryption then this function will return the empty string.

Sample Usage

```
$version = ssl.serverCertVersion();
log.info( "Server Cert Version: " . $version );
```

See also: “ssl.isSSL()”

Restrictions

Can only be used with TCP-based protocols.

ssl.serverSiteName()

Returns the hostname or IP address that was used to select the current server certificate. If the default certificate was used, or the current connection is not encrypted, the empty string is returned.

Sample Usage

```
$site_name = ssl.serverSiteName();
if( ssl.isSSL() && $site_name == "" ) {
   # The default certificate was used
}
```

See also: “ssl.isSSL()”

Restrictions

Can only be used with TCP-based protocols.

ssl.setClientCloseAlert(alertflag)

Sets whether your traffic manager will send the SSL client an SSL close alert prior to terminating the TCP connection. An value of 0 will disable close alerts, a non-zero value will enable close alerts. This setting also applies to related connections, such as data transfer channels related to FTP command channels. If the connection with the client is not an SSL connection then calling this function will do nothing.

© 2019 Pulse Secure, LLC.

Sample Usage

```
if( ssl.isSSL() ) {
   # Ensure close alerts are enabled.
   ssl.setClientCloseAlert( 1 );
}
```

See also: "ssl.isSSL()", "ssl.getClientCloseAlert()", "ssl.setServerCloseAlert(alertflag)", "ssl.getServerCloseAlert()"

Restrictions

Can only be used with TCP-based protocols.

ssl.setServerCloseAlert(alertflag)

Sets whether your traffic manager will send the SSL server an SSL close alert prior to terminating the TCP connection. A value of 0 will disable close alerts, a non-zero value will enable close alerts. This setting also applies to related connections, such as data transfer channels related to FTP command channels. Note that this function will only work on SSL connections and the server side connection must be established (i.e. it should typically only be used in response rules.)

Sample Usage

```
if( ssl.isSSL() ) {
   # Ensure close alerts are enabled.
   ssl.setServerCloseAlert( 1 );
}
```

See also: "ssl.isSSL()", "ssl.getServerCloseAlert()", "ssl.setClientCloseAlert(alertflag)", "ssl.getClientCloseAlert()"

Restrictions

Can only be used with TCP-based protocols.

ssl.setTLSServerName(servername)

Instructs your traffic manager to use the specified hostname when using the TLS 1.0 server_name extension. This method only works if the back end pool has SSL encryption and the server_name option enabled. Using the empty string as this function's parameter will make your traffic manager use the hostname of the node that it is connecting to.

Sample Usage

```
# Use the hostname foo in the TLS 1.0 server_name
# extension
ssl.setTLSServerName( "foo" );
```

See also: "ssl.isSSL()", "ssl.getTLSServerName()"

Restrictions

Can only be used with TCP-based protocols.

© 2019 Pulse Secure, LLC.

ssl.sslKeyLogLine()

If the ssl!log_keys global config key is enabled the function will return a key log line (or lines for TLS 1.3) suitable for use in Wireshark's SSL key log. If ssl!log_keys is disabled, the empty string will always be returned.

Sample Usage

```
# Get the SSL key log
$keylogline = ssl.sslKeyLogLine();
log.info( "SSL key log: " . $keylogline );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

ssl.sslSessionID()

Returns the session ID of the current SSL connection as a hex-encoded string. An empty string is returned if the connection is not SSL-encrypted, or if no session ID is available. For example, no session id is available if SSL resumption is disabled (pre-TLS 1.3) or if the protocol version is TLS 1.3 or later. This function can also be used with SSL pass-through traffic. When used in a request rule the traffic manager will look for a session ID provided in the ClientHello. Clients provide session IDs only if they are trying to resume a previously used SSL session. If the client did not provide a session ID, this function will return an empty string.

When used in a response rule with SSL pass-through traffic, this function will return the session ID provided in the ServerHello. This might be the same as the session ID sent by the client If the client attempted to resume the session, but this is not guaranteed.

Sample Usage

```
# Get the SSL session ID
$sessionid = ssl.sslSessionID();
log.info( "SessionID: " . $sessionid );
```

See also: "ssl.isSSL()"

Restrictions

Can only be used with TCP-based protocols.

stats.getQueueTime()

Returns the amount of time, in milliseconds, that the client spent queueing to connect to the server due to backend connection limits.

Sample Usage

```
# Keep a record of how many connections
# queue for longer than 1 second.
if( stats.getQueueTime() > 1000 ) {
   # Increment user counter 1
   counter.increment( 1 );
```

```
}
```

stats.getServerResponseTime()

Returns the amount of time, in milliseconds, between sending the first byte of the request to the server, and receiving a response.

In the case of HTTP, SIP and RTSP this is the time until the traffic manager has received a full set of headers from the server. For all other protocols, it is the time until the traffic manager has received the first byte of the response from the server.

If the server has not sent a response, this function will return 0.

Sample Usage

```
# Emit an event when the server takes a
# particularly long time to respond to a
# request.
if( stats.getServerResponseTime() > 30000 ) {
   event.emit( "slowresponse", http.getRawURL() );
}
```

stats.getTransactionDuration()

Returns the amount of time, in milliseconds, for which the current transaction has been active, from the time the first byte was received from the client to the current time.

If used in a transaction completion rule, this will be the total duration of the transaction.

Sample Usage

```
# Record this connection on the recent connections
# page if it took more than 5 seconds to complete
if( stats.getTransactionDuration() > 5000 ) {
   recentconns.include();
}
```

sys.getRestApiPort()

Returns the TCP port number used by the REST API, such as 9070.

Sample Usage

```
# get the REST API port
$restport = sys.getRestApiPort();

# use that to get the root node from the REST API
$body = http.request.get(
              "http://localhost:".$restport."/");

# in this case an 'error: authentication required'
# type response
# will be returned
```


© 2019 Pulse Secure, LLC.

sys.isFIPS()

Returns 1 if the traffic manager is operating in FIPS Mode, otherwise it returns 0.

Sample Usage

```
if( sys.isFIPS() ) {
   # FIPS Mode is in operation
}
```

tcp.close(sock)

Close a previously opened TCP socket. A non-zero return value indicates a successful close, if 0 is returned then an error occurred and an error string will be in '$1'.

If a rule fails to close a TCP socket, it will be closed automatically when the Virtual Server connection finishes.

Sample Usage

```
$sock = tcp.connect( "10.100.1.5", 7 );
tcp.close( $sock );
```

See also: “tcp.connect(ip, port, [timeout])”, “tcp.write(socket, data, [timeout])”, “tcp.read(socket, maximum, [timeout])”

Restrictions

Cannot be used in completion rules.

tcp.connect(ip, port, [timeout])

Create a new TCP socket to the supplied IP address and port. This function will return a socket handle that can be used by other tcp.* functions. The created TCP socket is unique to this connection, and can't be used by other connections.

An optional timeout can be specified in milliseconds. If a connection has not been established within this time the function will return 0 and $1 will be set to "timeout".

Note that if the Virtual Server connection times out before this socket has been established then the connection will be terminated.

Returns 0 on error (with an error message in $1).

Sample Usage

```
$sock = tcp.connect( "::1", 3306 );
if( ! $sock ) {
   log.error( "Error: " . $1 );
}
```

See also: “tcp.write(socket, data, [timeout])”, “tcp.read(socket, maximum, [timeout])”, “tcp.close(sock)”

Restrictions

Cannot be used in completion rules.

tcp.read(socket, maximum, [timeout])

Read data from a previously opened TCP connection. The function waits until data is available on the TCP socket, and will then return the available data (up to the specified 'maximum'). If an error occurs an empty string will be returned and $1 will contain an error message, or 0 if the connection has been closed.

An optional timeout can be specified in milliseconds. If a no data has been read within this time the function will return an empty string and $1 will be set to "timeout".

Note that if the Virtual Server connection times out before any data has been read then the connection will terminated.

Sample Usage

```
# Read from a TCP socket, ensuring we get 1024 bytes
$amount = 1024;
$buf = '';
while( string.len( $buf ) != $amount ) {
   $data = tcp.read( $sock, $amount -
                     string.len( $buf ) );

   if( $data == "" ) {
       $buf = '';
       break;
   }
   $buf .= $data;
}
```

See also: “tcp.connect(ip, port, [timeout])”, “tcp.write(socket, data, [timeout])”, “tcp.close(sock)”

Restrictions

Cannot be used in completion rules.

tcp.write(socket, data, [timeout])

Writes all of the supplied data to a TCP socket. The function will return the number of bytes written. If an error occurs then -1 will be returned, and $1 will contain an error message.

An optional timeout can be specified in milliseconds. After this time the number of bytes written will be returned, and $1 will contain the string 'timeout'.

Note that if the Virtual Server connection times out before the data has been written then the connection will be terminated.

Sample Usage

```
$sock = tcp.connect( "10.100.1.5", 7 );
tcp.write( $sock, "Ping\n" );
```

See also: “tcp.connect(ip, port, [timeout])”, “tcp.read(socket, maximum, [timeout])”, “tcp.close(sock)”

Restrictions

Cannot be used in completion rules.

© 2019 Pulse Secure, LLC.

udp.sendTo(host, port, data)

Send some data via UDP to ip:port.

If successful, returns the number of bytes written. A return value of -1 indicates a failure; in this case an error string will be stored in '$1'.

Sample Usage

```
$res = udp.sendTo( "10.100.1.5", 82, "ping" );
if( $res == -1 ) {
   log.warn( "Error sending data: " . $1 );
}
```

See also: "tcp.connect(ip, port, [timeout])", "tcp.write(socket, data, [timeout])", "tcp.read(socket, maximum, [timeout])"

xml.validate(document, DTD) - deprecated

This function has been deprecated. Use instead "xml.validate.dtd(document, DTD)".

Validates an XML document against a DTD. It returns 1 if the document validated correctly, and 0 if it did not. It returns -1 if there was an error parsing the XML document or the DTD.

The XML processing functionality must be enabled by the software license.

xml.validate.dtd(document, DTD)

Validates an XML document against a DTD. It returns 1 if the document validated correctly, and 0 if it did not. It returns -1 if there was an error parsing the XML document or the DTD.

The XML processing functionality must be enabled by the software license.

Sample Usage

```
# Validate the HTTP body against the DTD stored in
# the resource 'mydtd'
$theDoc = http.getBody( 0 );
$theDTD = resource.get( "mydtd" );
if( xml.validate.dtd( $theDoc, $theDTD ) != 1 ) {
   # validation failed ...
}
```

See also: "xml.validate.xsd(document, schema)"

xml.validate.xsd(document, schema)

Validates an XML document against an XML schema. It returns 1 if the document validated correctly, and 0 if it did not. It returns -1 if there was an error parsing the XML document or the schema.

If the schema against which the document is being validated needs to import another schema file, it will search for it inside Catalog > Extra Files > Miscellaneous Files.

The XML processing functionality must be enabled by the software license.

© 2019 Pulse Secure, LLC.

Sample Usage

```
# Validate the HTTP body against the schema stored in
# the resource 'myschema'
$theDoc = http.getBody( 0 );
$theSchema = resource.get( "myschema" );
if( xml.validate.xsd( $theDoc, $theSchema ) != 1 ) {
   # validation failed ...
}
```

See also: "xml.validate.dtd(document, DTD)"

xml.xpath.matchNodeCount(doc, nspacemap, query)

Applies an XPath query to the supplied XML document. It returns the number of entries in the result node set.

The function will return -1 if there was an error parsing the XML document, XML namespace or XPath query.

Namespaces can be used in the XPath query by defining them in the "nspacemap" parameter. This parameter is specified as a space-separated list of identifiers mapped to namespaces. For example, specifying 'xmlns:myns=http://www.example.com/mynamespace' allows you to use the identifier 'myns' in the XPath query to find elements in the 'http://www.example.com/mynamespace' namespace within the XML document.

The XML processing functionality must be enabled by the software license.

Sample Usage

```
# A sample XML document
$theDoc =
'<?xml version="1.0" encoding="UTF-8"?>
 <mybook xmlns="http://www.example.com/book"
   xmlns:recent="http://www.example.com/newitems">

    <item>Item 1</item>
    <recent:item>Item 2</recent:item>
    <item>Item 3</item>
    <recent:item>Item 4</recent:item>
 </mybook>
';

# How many items are in the 'newitems' namespace
# under the 'mybook' element of the 'book' namespace?
# Note: namespace identifiers in the XPath query
# do not have to be the same as those used in the XML
# document itself.
$items = xml.xpath.matchNodeCount(
   $theDoc,
   "xmlns:book=http://www.example.com/book ".
   "xmlns:new=http://www.example.com/newitems",
   "//book:mybook/new:item"
);

# $items = 2
```

See also: "xml.xpath.matchNodeSet(doc, nspacemap, query)"

© 2019 Pulse Secure, LLC.

xml.xpath.matchNodeSet(doc, nspacemap, query)

Applies an XPath query to the supplied XML document. It returns a string representation of the result node set.

The function will return an empty string if there was an error parsing the XML document, XML namespace or XPath query.

Namespaces can be used in the XPath query by defining them in the "nspacemap" parameter. This parameter is specified as a space-separated list of identifiers mapped to namespaces. For example, specifying 'xmlns:myns=http://www.example.com/mynamespace' allows you to use the identifier 'myns' in the XPath query to find elements in the 'http://www.example.com/mynamespace' namespace within the XML document.

The XML processing functionality must be enabled by the software license.

Sample Usage

```
# A sample XML document
$theDoc =
'<?xml version="1.0" encoding="UTF-8"?>
 <mybook xmlns="http://www.example.com/book"
   xmlns:recent="http://www.example.com/newitems">

    <item>Item 1</item>
    <recent:item>Item 2</recent:item>
 </mybook>
';

# Get the value of the 'item' node in the 'newitems'
# namespace under the 'mybook' element of the 'book'
# namespace.
# Note: namespace identifiers in the XPath query
# do not have to be the same as those used in the XML
# document itself.
$set = xml.xpath.matchNodeSet(
   $theDoc,
   "xmlns:book=http://www.example.com/book ".
   "xmlns:new=http://www.example.com/newitems",
   "//book:mybook/new:item/text()"
);

# $set = 'Item 2'
```

See also: “xml.xpath.matchNodeCount(doc, nspacemap, query)”

xml.xslt.transform(document, stylesheet)

Performs an XSLT transformation on a XML document. It returns the transformed document, or -1 on failure.

The XML processing functionality must be enabled by the software license.

Sample Usage

```
# Perform XSLT transformation on supplied POST XML
```

```
# data, and return the HTML result to the client.
$response = xml.xslt.transform( http.getbody( 0 ),
   resource.get( "people.xsl" ));
http.sendResponse( "200 OK", "text/html",
   $response, "" );
```


© 2019 Pulse Secure, LLC.