FAQ

Policy Engine (PE)
Frequently Asked Questions

Notice:

The information in this publication is subject to change without notice.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. CITRIX SYSTEMS, INC. ("CITRIX"), SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR DIRECT, INCIDENTAL, CONSEQUENTIAL OR ANY OTHER DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS PUBLICATION, EVEN IF CITRIX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES IN ADVANCE.

The exclusive warranty for Citrix products, if any, is stated in the product documentation accompanying such products. Citrix does not warrant products other than its own.

Table of Contents

Overview of Policy Engine (PE)

The Policy Engine (PE) provides a common framework for creating policy expressions that can be utilized by any of the features of the Citrix NetScaler Application Switch. The Policy Engine refers to the architecture in the Citrix NetScaler Application Switch for versions up to 8.x. The architecture for Policy Engine and the manner in which it operates is presented in this guide.

The features that use policies are:

- Load Balancing
- Content Switching
- Content Filtering
- AppCompress
- Cache Redirection
- SSL VPN
- Priority Queuing
- DoS Protection
- Sure Connect

Policy expressions are applied to content that enters the system. Expressions are shared among features, but actions are feature-specific. For example, you can create an expression to identify .pdf files being sent through the system. You can then create a compression policy that uses this expression to compress those files.

An expression is generally made up of 3 parts: The Qualifier, Operator and Operand. For instance in the policy expression,

```
add expression not_nsip “DESTIP neq 10.102.12.20”
```

“DESTIP” is the Qualifier specifying the information in the IP Packet that needs to be tested, “neq” is the Operator specifying the condition, and “10.102.12.20” is the Operand that the Qualifier is compared against.

Traffic flowing through a NetScaler can be evaluated against an expression at any of the following 4 (protocol) levels:

- IP
- TCP
- HTTP
- SSL

An example of each type of policy expression that belongs to these 4 protocols are given below:

IP:

```
add expression not_nsip “DESTIP neq 10.102.12.20”
```

TCP:

```
add policy expression ldap_traffic “req.tcp.destport == 389”
```

HTTP:

```
add policy expression finance_url “URL CONTAINS ‘finance’”
```

SSL:

```
add policy expression ssl_v3 “CLIENT.SSL.VERSION EQ SSLV3”
```

Compound expressions can be used to match traffic patterns more precisely and apply the required action only on desired traffic. For example, to make web pages in the Finance Department only accesible to the Finance IP Subnet:

```
add policy expression not_finance “SOURCEIP neq 10.102.1.0 -netmask 255.255.255.0”

add policy expression finance_url “URL CONTAINS ‘finance’”

add policy expression ext_html “URL == /*.htm || URL == /*.html”

add policy expression secure_finance “not_finance && finance_url && ext_html”
```

Now you can define a filtering policy to drop packets that match the above criteria:

```
add filter policy drop_finance_attack -rule secure_finance -reqAction DROP
```

Compound expressions can have a combination of inline expressions along with named/simple expressions that are already created. Thus the above compound expression “secure_finance” can also be defined as:

```
add policy expression secure_finance “not_finance && URL CONTAINS ‘finance’ && ext_html”
```

Another class of policy expressions is used to define Command Policy when creating custom roles/users for the NetScaler system. Here you are allowed to use regular expressions to define Command Policies against which fully expanded form of the command issued by the user is checked to determine whether the command is permitted or not.

The general form of the command policy is:

```
add system cmdPolicy >policyName> <action> <cmdSpec>
```

where the action would be ALLOW or DENY and cmdSpec is the regular expression against which the user input will be evaluated.

Components/keywords that make up the regular expression language is as given below:

Character Match:

? Optional character

. Any Character

Repetition Characters:

+1 or more of previous character

*0 or more of previous character

Text Anchors:

^ Begin match at start of line

$ Match at end of line

Negative Look Ahead:

(?!*pattern*)

The following regular expression matches ‘add vserver’ followed by any parameters:

```
“^add/s+vserver.*”
```

The following matches ‘add lb vserver’ with any parameters:

```
“^add\s+(lb\s+vserver).*”
```

The following matches any show command except show system commands:

```
“(^show\s+(?!system).*)”
```

Command policies can be bound to both (user created) groups as well as users. Priorities can also be assigned to command policies to influence the order of evaluation. The lower the numerical value, the higher will be the priority. Thus command policy with priority 1 will be evaluated before command policy with priority 10.

In case a user has policies bound on itself and at group level the user level command policies take precedence and in case of multiple matches, the higher priority command policy will be matched and action will be taken accordingly.

A sample configuration is given below:

```
add system cmdPolicy deny_all_rm DENY “^rm.*”

add system cmdPolicy deny_all_sh DENY “^shell”

add system cmdPolicy deny_system_cmnd DENY “*.system.*”

bind system user johnd deny_system_cmnd 1

bind system user johnd deny_all_rm 5

bind system user johnd deny_all_sh 10

add system cmdPolicy default_deny_override ALLOW “^.*”

bind system group nocusers -policyName default_deny_override 100
```

Note again that the policy has been bound to the nocusers group with a priority of 100. This will ensure the ordering of the priority among any other policies that may later be bound against this group.

Now that all of the group and user command policies are in place, the complete order of policy evaluations for johnd can be explained. The user johnd’s direct policies will be evaluated first, preventing access to system command group commands, remove actions and access to shell, in that order of priority.

Due to his group membership, the user will otherwise have access to remaining commands because of the group’s default deny override policy. The next section explains how the NetScaler’s command policy evaluation procedure causes this overall policy order to achieve the desired level of user access for johnd.

Policy Engine (PE) Capabilities

How many policies can be configured on a NetScaler?

There is a built-in maximum of 1024 expressions in the NetScaler system and each new policy adds one expression internally. A few expressions are added by default, thus approximately 1000 policies can be added which is the limit shared by all the policies.

We further limit the Cache Redirection (CR) policies to a maximum of 128 and Content Switching (CS) policies to 512, which are hard coded limits and cannot be changed.

If we want to allow addition of more policies/expressions, we can change the max configurable expressions limit through nsapimgr:

```
nsapimgr -ys maxexpr=New_Limit_Number
```

In summary, we have:

- 128 CR policies limit.
- 512 CS policies limit.
- 1024 expression limit (which can be changed via maxexpr).
- otherwise, the policies are limited by max expression limit.

How do I choose between the operators “==” and “CONTAINS”?

Consider the following facts when deciding between “==” and “CONTAINS”

The operator “==” is:

- Case-sensitive
- Accepts wildcards
- Is less CPU intensive

The operator “CONTAINS” is:

- Case-insensitive
- Does not accept wildcards
- Is more CPU intensive

Thus the policy “URL = =/Admin/*” will match “Admin” but will not match “ADMIN” or “admin”, while “URL CONTAINS admin” will match “admin”, “Admin” and “ADMIN” but does so by consuming more CPU resources.

Rule based vs. URL based polices?

When traffic has to be matched using both URL related parameters (like REQ.HTTP.URL = =/*.html) and HTTP header related parameters (like REQ.HTTP.HEADER Cookie EXISTS) rule based policies should be used. URL based polices should be used only when HTTP header based conditions are not involved and most specific URL match is needed.

You will be able to configure priority only for rule based policies and not for URL based policies.

URL based policies are less CPU intensive than rule based policies. Rule based polices do allow the use of inline expressions.

Error message “Invalid rule”?

The rule that I am entering is perfectly valid. What should I do?

```
add cs policy cs_pol_10 -rule “(REQ.HTTP.URL == /*.html && (REQ.HTTP.URLLEN LT 256 && REQ.HTTP.HEADER Cookie EXISTS))
```

```
ERROR: Invalid Rule
```

You are encountering this problem because you are using too many inline expressions. Please configure unary expressions and then create compound expressions where required. Use these compound expressions as rules to match traffic. Please refer earlier sections on creating compound expressions and CPE limitations.

Policy expression that contains escape sequences?

Like question mark, single quotes or double quotes?

You can do this by preceding the special character with a “backslash” as shown in the examples below:

```
    add policy expression spl_char_qm “URL == /blahblahblah\”

    add policy expression spl_char_qm “URL == /blahblahblah?123”

    add cs policy cs_pol1 -rule “url CONTAINS sports || http_port || (REQ.HTTP.HEADER Cookie CONTAINS ‘abc pqr)

    add cs policy cs_pol1 -rule “url CONTAINS sports || http_port || (REQ.HTTP.HEADER Cookie CONTAINS \”abc pqr)

    add cs policy cd_pol1 -rule “url CONTAINS sports || http_port || (REQ.HTTP.HEADER Cookie CONTAINS \’abc pqr)

    add cs policy cs_pol2 -rule “url CONTAINS ‘a \\”b\\” \\’c\\’ d’”

    add cs policy cs_pol2 -rule “url CONTAINS \”a \\”b\\” \\’c\\\’ d\””

    add cs policy cs_pol2 -rule “url CONTAINS \’a \\”b\\” \\’c\\’ d\’”
```

For the next two Questions, Presently configured policy expressions and CS policy are as follows:

```
add policy expression imdb_boards_expr “URL CONTAINS /board”

add cs policy imdb_boards -rule imdb_boards_expr
```

I need to add a new policy to this setup:

```
add policy expression imdb_keyword_expr “URL == /keyword/*”
```

This policy needs to run *before* imdb_boards_expr to avoid something like /keyword/boards tripping the boards_expr before the keyword_expr sees the URL.

Linear Ordering of Policies?

Is there any way to insert a policy in a particular place in the linear order (e.g. edit ns.conf directly and reload, or via the Web GUI)?

As long as the policies are rule-based you will be able to re-order the polices without reload or even unbinding the policies.

Priority Ordering of Policies?

Is there any way to add priorities to existing policies without removing and readding them?

You are allowed to set precedence between URL-based and rule-based polices which are the two kinds of policies configurable on a NetScaler.

URL-based polices are of atomic nature and can match traffic based on domain and URL.

Rule-based polices are more powerful, can match on many conditions and can be of compound nature.

If you want to create a policy of the form “contains A and doesn’t contains B” it would be a rule-base policy. Within rule-based polices you are allowed to assign priorites to different polices. The lower the numerical value, higher the priority.

The following sample configuration will achieve the desired effect:

```
add cs policy rule1 -rule “URL CONTAINS /board”

add cs policy rule2 -rule “URL == /keyword/*”

bind cs vserver cs1 lb1 -policyName rule1

bind cs vserver cs1 lb2 -policyName rule2

bind cs vserver cs1 lb1 -policyName rule1 -priority 10

sh cs vserver cs1
```

No support for compound expression?

I am trying to modify a compound expression that I created but I am getting an error message stating that there is “no support for compound expression”. What am I doing wrong?

You are only allowed to modify a simple expression using the “set policy expression” command. For compound expressions you will have to delete and add it again.

```
add policy expression not_finance “SOURCEIP neq 10.102.1.0 -netmask 255.255.255.0”

add policy expression finance_url “URL CONTAINS ‘finance’”

add policy expression ext_html “URL == /*.htm || URL == /*.html”

add policy expression secure_finance “not_finance && finance_url && ext_html”

set policy expression secure_finance “not_finance || finance_url || ext_html”

ERROR: No Support For set compound expression
```

Error: CS policy limit reached?

We are a website hosting company and rely heavily on Content Switching and have a large number of content switching polices qualified. Recently while trying to add a new CS policy we got the following error message:

```
add cs policy epol347 -rule “(url_eq_ecol && REQ.IP.SOURCEIP == 64.4.184.0 -netmask 255.255.255.0)”

ERROR: CS policy limit reached
```

What is the cause of this error message and what is the workaround?

You have reached the limit in terms of the number of content switching policies that can be configured (512 at present). A workaround would be to try and combine different content switching policies into one if possible.

Convert a simple expression into a compound expression?

This again is not possible just like modification of existing compound expressions.

You will have to delete and re-create the expressions as required.

System user password modification?

Is it possible to allow a system user to change only their own password?

As an example let us take the case of two system users meggie and simon, both users belong to group “unix” which has built-in policy “network” binded:

```
#define ALLOW policy for changing specific user’s password

add system user meggie

add system user simon

add system group unix

add system cmdPolicy sim-pol ALLOW “^set\\s+((system\\s+user)\\s+simon).*”

add system cmdPolicy meg-pol ALLOW “^set\\s+((system\\s+user)\\s+meggie).*”

#define ALLOW policy for “show system user” command since the “set system user” command needs the privilege for “show system user” command as well

#while the build-in “network” policy (the policy you are using) does not have the privilege for any system command

add system cmdPolicy showuser-pol ALLOW “^show\\s+(system\\s+user).*”

#bind the “set system user” policy to different system users with high priority

bind system user meggie meg-pol 1

bind system user simon sim-pol 1

#bind the “show system user” policy and “network” policy to “unix” group

bind system group unix -policyName showuser-pol 10

bind system group unix -policyName network 100

#bind user “meggie” and “simon” to “unix” group

bind system group unix -username meggie

bind system group unix -username simon
```

When you test it, please log out the user after adding those policies, then log back in to try changing password, you can use the following shell command to trace:

```
>shell

#tail -f /var/log/auth.log
```

Load balancing with token value?

Customer is load balancing HTTP traffic across multiple back-end servers and would like to make the load balancing decision based on the 5 byte token “serverid” which is set by the server and is returned by the client in the query part of the URL.

We will have to define a policy expression that will search for the token “serverid” in the URLQUERY and extract the value associated with it. Once this is done the value can be used to make load balancing decision such that requests with the same Server Id will always return to the same server.

```
enable ns feature lb

add policy expression serverid “REQ.HTTP.URLQUERY CONTAINS serverid -length 5”

add Service websvc1 10.102.12.204 http 80

add Service websvc2 10.102.12.205 http 80

add lb vserver HTTP_VIP http 10.102.12.121 80

bind lb vserver HTTP_VIP websvc1

bind lb vserver HTTP_VIP websvc2

set lb vserver HTTP_VIP -lbmethod token -rule serverid
```

Content switching clients?

Customer would like requests that have the string “mobile” to be directed to the VIP “WAP” and all other requests to be directed to the VIP “Web”.

This can be achieved by defining a content switching policy that will direct all traffic that contains the string “mobile” to desired VIP. A default LB VIP can be bound to forward all other traffic to other server farm.

```
enable feature lb cs

add Service wapsvc1 10.102.12.203 http 80

add Service websvc1 10.102.12.204 http 80

add Service websvc2 10.102.12.205 http 80

add lb vserver WAP http 0.0.0.0 0

add lb vserver Web http 0.0.0.0 0

bind lb vserver WAP wapsvc1

bind lb vserver Web websvc1

bind lb vserver Web websvc2

add pol expression wap_url “URL contains mobile”
add cs policy wap_pol -rule “wap_url”
add cs vserver csvip http 10.102.12.120 80
bind cs vserver csvip WAP -pol wap_pol
bind cs vserver csvip Web
```

Content Filtering?

Customer wants to configure a content filtering policy that can protect against the computer worm Nimda.

The Nimda worm attempts to access cmd.exe and root.exe and this characteristic can be used to reset suspicious connections.

```
enable feature cf
add policy expression root “URL contains root.exe”
add policy expression cmd “URL contains cmd.exe”
add policy expression nimda “cmd || root”
add filter policy nimda_filter -rule nimda -reqaction RESET
bind filter global nimda_filter -priority 1
```

Compress/No Compress?

Customer would like to configure a compression policy that will ensure that CSS files are not compressed.

CSS content can be identified by the value in the “Content-Type” header and this will allow us to setup the necessary no compression action.

```
enable ns feature cmp
add policy expression css_rule “RES.HTTP.HEADER Content-Type CONTAINS text/css”
add cmp policy css_pol -rule css_rule -resaction nocompress
bind cmp global css_pol -priority 1
```

Cache Redirect ASP pages?

Configure the NetScaler to fetch ASP pages from the Cache Server.

The below cache redirection configuration will forward all ASP requests to the cache virtual server.

```
enable ns feature lb cr

add policy expression asp_rule “URL == /*.asp”

add cr policy asp_cache -rule asp_rule

add cr vserver cr_vip HTTP * 80 -cacheType TRANSPARENT -redirect POLICY

add lb vserver cache_vip HTTP 0.0.0.0 0 -lbmethod URLHASH

add service cache_svc1 10.102.12.204 HTTP 80 -cacheType TRANSPARENT

add service cache_svc2 10.102.12.205 HTTP 80 -cacheType TRANSPARENT

bind lb vserver cache_vip cache_svc1

bind lb vserver cache_vip cache_svc2

set cr vserver cr_vip -cacheVserver cache_vip

bind cr vserver cr_vip -policy asp_cache

bind cr vserver cr_Vip -policy bypass-non-get

bind cr vserver cr_Vip -policy bypass-cache-control

bind cr vserver cr_Vip -policy bypass-dynamic-url

bind cr vserver cr_Vip -policy bypass-urltokens

bind cr vserver cr_Vip -policy bypass-cookie
```

SSL VPN with LDAP?

Customer is using SSL VPN with LDAP authentication and would like to enforce authorization policies as follows: Engineering should have access to Database server but should not have access to SAP server. HR has access to SAP server but no access to Database server. Both groups have access to the Intranet server and Mail server. Customer is having LDAP group extraction in place.

In order to achieve the above given objective one will have to create expressions that can match each of the mentioned servers. Using these expression and based on the above given security requirement authorization policies will have to be created and bound to the different groups.

```
add aaa group ENGINEERING

add aaa group HR

add policy expression DB_Server “destip == 10.102.1.100”

add policy expression SAP_Server “destip == 10.102.7.53”
add policy expression Web_Server “destip == 10.102.0.78”

add policy expression Mail_Server “destip == 10.102.0.121”

add authorization policy Eng_DB DB_Server ALLOW

add authorization policy Eng_SAP SAP_Server DENY

add authorization policy HR_DB DB_Server DENY

add authorization policy HR_SAP SAP_Server ALLOW

add authorization policy All_Web Web_Server ALLOW

add authorization policy All_Mail Mail_Server ALLOW

bind aaa group ENGINEERING -policy Eng_DB -priority 10

bind aaa group ENGINEERING -policy Eng_SAP -priority 20

bind aaa group ENGINEERING -policy All_Web -priority 30

bind aaa group ENGINEERING -policy All_Mail -priority 40

bind aaa group HR -policy HR_DB -priority 10

bind aaa group HR -policy HR_SAP -priority 20

bind aaa group HR -policy All_Web -priority 30

bind aaa group HR -policy All_Mail -priority 40
```

Priority Queueing for users?

An ISP provides both paid as well as free internet access. They would like to make sure that when the paid users access the mail server, their access is prioritized above the free users. The two groups of users can be identified by their IP ranges; paid users have IPs in the range 145.17.20.0/24 and free users have IPs in the range 145.17.21.0/24.

Assuming the users accesses the VIP HTTP_Mail, the desired objective can be met by configuring policy expressions that match the IP range and Priority Queuing policy that assign the proper weight and queue depth. NetScaler’s Priority queuing configuration allows the use of 3 priority levels: level1, level 2 and level 3. Level 1 has highest priority and level 3 has lowest priority. In addition weights can also be assigned to determine whether the traffic that is queued under a particular priority will/will not be processed when traffic is queued in other priorities. Weight 0 means that this priority level will be served only if no requests are stored in the other queues. The maximum weight that can be assigned is 101.

A weight of 101 means that this priority level will be served, regardless of the requests waiting in other queues. In other words, this queue starves all other priority queues. Weights should be assigned in the increasing order of priority levels (for example, a weight with priority 1 is greater to a weight with a priority of 2).

```
enable ns feature lb pq

add Service mailsvc1 10.102.12.204 http 80
add Service mailsvc2 10.102.12.205 http 80
add lb vserver HTTP_Mail http 10.102.12.121 80
bind lb vserver HTTP_Mail mailsvc1
bind lb vserver HTTP_Mail mailsvc2
add policy expression paid_user “SOURCEIP eq 145.17.20.0 -netmask 255.255.255.0”
add pq policy paid_access -rule paid_user -priority 1 -weight 10 -qdepth 1000
bind lb vserver HTTP_Mail -policyname paid_access
set lb vserver HTTP_Mail -pq on
```

SureConnect?

Customer would like to make sure that when the back-end server is unable to respond to heavy traffic in response to the Thanksgiving sale, SureConnect sale kicks in and a progress bar is displayed. Requests directed to the Thanksgiving sale can be uniquely identified by the string “turkey.asp”.

Assuming that the VIP in use is called HTTP_VIP and that the SureConnect policy is to kick in if the delay exceeds average response time by 1 minute and that the alternate content is being served from the NetScaler, the following configuration will help us achieve the desired objective.

```
enable ns feature lb sc
add Service websvc1 10.102.12.204 http 80
add Service websvc2 10.102.12.205 http 80
add lb vserver HTTP_VIP http 10.102.12.121 80
bind lb vserver HTTP_VIP websvc1
bind lb vserver HTTP_VIP websvc2
add policy expression thanksgiving “URL == /*turkey.asp”
add sc policy thanksgiving_sc -rule thanksgiving -delay 1000000 -action NS
bind lb vserver HTTP_VIP -policy thanksgiving_sc
```

Abbreviated Qualifiers?

Are there any abbreviated Qualifiers?

Yes there are a few, however they only apply to Request based flows and are listed below. For best practices it is better to specify the complete Qualifier, REQ.HTTP.HEADER as opposed to HEADER.

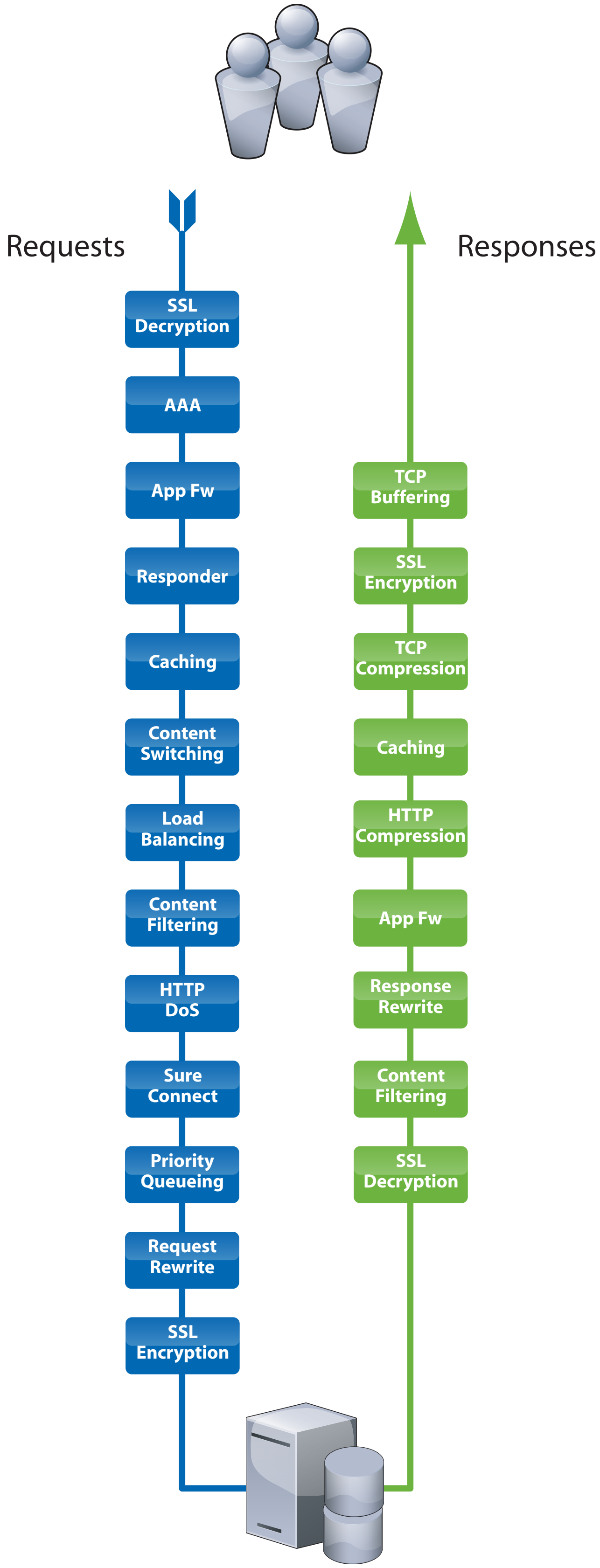
Abbreviated Qualifier	Full Qualifer
VERSION	REQ.HTTP.VERSION
METHOD	REQ.HTTP.METHOD
URL	REQ.HTTP.URL
URLTOKENS	REQ.HTTP.URLTOKENS
URLQUERY	REQ.HTTP.URLQUERY
URLLEN	REQ.HTTP.URLLEN
URLQUERYLEN	REQ.HTTP.URLQUERYLEN
HEADER	REQ.HTTP.HEADER
SOURCEIP	REQ.IP.SOURCEIP
DESTIP	REQ.IP.DESTIP
SOURCEPORT	REQ.TCP.SOURCEPORT
DESTPORT	REQ.TCP.DESTPORT

SSL Qualifiers?

Are there any SSL abbreviated Qualifiers?

Yes there are a few, listed in the table below.

Abbreviated Qualifier
CLIENT.SSL.VERSION
CLIENT.CIPHER.BITS
CLIENT.CIPHER.TYPE
CLIENT.CERT
CLIENT.CERT.VERSION
CLIENT.CERT.SERIALNUMBER
CLIENT.CERT.SIGALGO
CLIENT.CERT.SUBJECT
CLIENT.CERT.ISSUER
CLIENT.CERT.VALIDFROM
CLIENT.CERT.VALIDTO



Important Policy Behavior - Policy Engine (PE)

Policies get evaluated in the order that they are classified in, that is with their priority numbers. Policies operate on a first-match principle. In a policy classification, the action associated with the first policy that matches gets applied. Once a match is determined, the policy evaluation exits the evaluation logic tree and no more policies are evaluated.

If there is no match, the GOTO expression is evaluated, which can be goto the 'END' of the logic tree, or go to the 'NEXT' priority number, or goto a specific priority number.

Each Feature has it's own set of priority numbers for it's own set of policies. Policy priority numbers don't overlap between feature sets. Having a policy for rewrite with priority 20 doesn't interfere with a policy for caching with priority 20 or 10 or 30. Request flow policy priorities come before (lower numbers) Response flow policy priorities (higher numbers).

Priority numbers increment in units of 10.

Citrix Worldwide

Worldwide headquarters

Citrix Systems, Inc.
851 West Cypress Creek Road
Fort Lauderdale, FL 33309
USA
T +1 800 393 1888
T +1 954 267 3000

Regional headquarters

Americas
Citrix Silicon Valley
4988 Great America Parkway
Santa Clara, CA 95054
USA
T +1 408 790 8000

Europe
Citrix Systems International GmbH
Rheinweg 9
8200 Schaffhausen
Switzerland
T +41 52 635 7700

Asia Pacific
Citrix Systems Hong Kong Ltd.
Suite 3201, 32nd Floor
One International Finance Centre
1 Harbour View Street
Central
Hong Kong
T +852 2100 5000

Citrix Online division
5385 Hollister Avenue
Santa Barbara, CA 93111
USA
T +1 805 690 6400

www.citrix.com

AppExpert
Tap into the power.

About Citrix

Citrix Systems, Inc. (Nasdaq:CTXS) is the global leader and the most trusted name in application delivery infrastructure. More than 200,000 organizations worldwide rely on Citrix to deliver any application to users anywhere with the best performance, highest security and lowest cost. Citrix customers include 100% of the Fortune 100 companies and 98% of the Fortune Global 500, as well as hundreds of thousands of small businesses and prosumers. Citrix has approximately 6,200 channel and alliance partners in more than 100 countries. Annual revenue in 2006 was $1.1 billion.

CiTRiX®