SAM Office B.V.

Venrayseweg 16
5961 AG Horst

+31 77 398 22 88
www.samoffice.com

SAM Office

Whitepaper

NetScaler Rocks

Core Logic

22-2-2016

Roel Schreibers, Jan Tytgat

info@samoffice.com

INTRODUCTION

This document is intended as a whitepaper and a guide with in-depth information for all Citrix NetScaler enthusiasts out there. Whether you are a SAM Office customer who already has the Core-Logic installed, or a fellow Citrix NetScaler community member, we hope this whitepaper will provide you with some insights and ideas.

This document shows the Core-Logic as a black box:

- How to setup Core-Logic
- How to add a new Content Switching Virtual Server to the NetScaler
- How to add a new Application to a Content Switching Virtual Server
- How to manage the Control Plane

The SAM Office Challenge:

Based on this document, we challenge **you** to figure out the policy expressions and the logic behind the Core-Logic. Let us know what you think, or give us your input on how we can make it even better! Rather start working with the SAM Office Core-Logic immediately? No problem, send an e-mail to info@samoffice.com and we will send you the files for free.

SAM Office started with the Core-Logic, we believe in it, let's make it even better together!

Need help? That's what we at SAM Office do!

Kind regards,

Roel Schreibers, Jan Tytgat.

CONTENT

BUILD TO MAINTAIN

HISTORY

It's not clear when we started to think about a central steering mechanism for content switching, which eventually led to this whitepaper. Somewhere in 2014, we first published a blog on NetScalerRocks.com[1] introducing the idea of utilizing Citrix NetScaler's strength of building dynamic expressions to steer requests to the correct Load Balancing Virtual Server. Until then, each request was steered using individual policies, causing configurations to become large and complex; and therefore hard to maintain.

The complexity and diversity of many configurations was very time-consuming in terms of figuring out how things were actually processed, and what needed to be changed in order to make the requested change work without altering or disrupting the whole environment.

Version 11 of Citrix NetScaler also emphasized the possibility to use content switching in combination with the authentication possibilities of the platform. Using Unified Gateway as a AAA server, while being integrated into the Content Switching Virtual Server, enables us to be even more flexible in deploying applications. Though, this flexibility also emphasizes the need for a unified method to configure and maintain the growing complexity of a configuration.

CONCEPT

The "Build to Maintain" concept is older, however. Finding ***a manageable and unified method*** to configure and maintain the NetScaler Configuration for a customer is an on-going quest:

- *First of all,* ***Visualization*** of the actual flow through different NetScaler components helped us communicating with the customer and support engineers, as it is imperative that both parties understand what is happening. At the same time, it helped defining our intents to solve a given problem while making it easier to acquire a quick insight into what was happening at the customer.
  Even today, visualization is a major area of interest to us and we are still looking to improve on communicating about the mechanics of the grey area between Networking and Application, called Application Delivery.
- *Second*, besides the fact that ***Monitoring*** is already one of the primary services Citrix NetScaler offers, the complete ***Service Chain*** became an important part of the concept. Although the NetScaler appliance might be running flawlessly, we should also monitor the services and applications which are being load-balanced for their health, throughput, etc.
  Bridging the gap between Networking and Applications also requires monitoring to happen throughout the whole organization. As such, a specialized monitoring system like Command Center does not suffice, as it is accessible by NetScaler engineers only. It is imperative the customer's IT department takes ownership of the (SNMP) monitoring as well, and Citrix NetScaler can be of great assist in this area.
  In result, monitoring Citrix NetScaler has become an intricate part of the "Build to maintain" concept and has an impact on how an application is load-balanced on the platform.

[1] https://netscalerrocks.com/netscaler/contentswitching-quick-dirty/

- *Third,* ***standardized implementation methodology***, starting with naming conventions, simple redirects to HTTPS, rewrites etc.… Standardization is key to a maintainable environment.

The "build to maintain" concept, combined with our ideas around the ***Core-Logic*** gave us new insights on how to build a manageable and unified Citrix NetScaler configuration for a customer, while starting a new quality cycle in improving our services for our customers.

CORE-LOGIC

Overview

Core-Logic does not specifically target the implementation of a single application on a Citrix NetScaler. However, it is considered as an ***integrated strategy*** to get consistency between different applications implemented on a NetScaler platform.

The focal point of the Core-Logic is to centralize all application steering across multiple HTTP/HTTPS Content Switching Virtual Servers by using a single String Map, which we call the ***Control Plane***.

Much of this was inspired by https://www.citrix.com/blogs/2011/07/29/how-string-maps-help-simplify-and-reduce-configuration/ (thank you Neha).

In short, the Core-Logic is a collection of Advanced Policy Expressions and non-addressable Load Balancing Virtual Servers. The policy expressions are static and therefore version able within a configuration's lifecycle, so new features should/can be implemented in a controlled manner.

Expressions

Currently, the **Core-Logic** takes care of the following things on a HTTP and/or HTTPS Content Switch:

- ***Select the correct (non-addressable) Load Balancing Virtual Server***, based on:
  - FQDN
  - FQDN + 1st path of the URL
  - FQDN domain (wildcard)
- ***Redirect the request***:
  - From HTTP to HTTPS or vice versa
  - 301/302 Redirect based on:
    - FQDN
    - FQDN + 1st path of the URL
    - FQDN domain (wildcard)
- ***Drop or Reset the request***, based on:
  - FQDN
  - FQDN + 1st path of the URL
  - FQDN domain (wildcard)

In most cases, reducing the number of content switching policies bound to a Content Switching Virtual Server also reduces the "***time-to-decision***" on how to process a request.

Control Plane

The **Control Plane** is a single String Map, which results in the following properties when used for the Core-Logic:

- ***Provide a centralized configuration*** for specific flows through the different Content Switching Virtual Servers.
- ***Minimize changes*** to the content switching policies by using the Core-Logic.
- ***Improve performance***, especially for large configurations, as string maps are indexed on Citrix NetScaler.

Advantages

With the Core-Logic implemented, adding a new application should only require the creation of a non-addressable load balancing virtual server for the application and adding a corresponding entry to the Control Plane.

The advantages are clear:"

- Changes have a lower impact on the current configuration.
- Changes are easier to automate.
- Changes take less time to be implemented.
- Lower time-to-decision
- Improved performance

Looking Forward

Currently, Core-Logic is at version 9.
We could continue knocking ourselves out in adding new features or fancier possibilities to this single string map. However, this version delivers the necessary flexibility for most (current) implementations.

In the coming period, we tend to spend more time on automation of the entire process.

IMPLEMENTATION

REQUIREMENTS

A typical implementation of Citrix NetScaler is based on having a number of applications that need to be made accessible from the internet. Microsoft ADFS, Microsoft Exchange, Microsoft SharePoint, Citrix Storefront, etc. Needless to say this can be any web application.

Possible extra requirements:

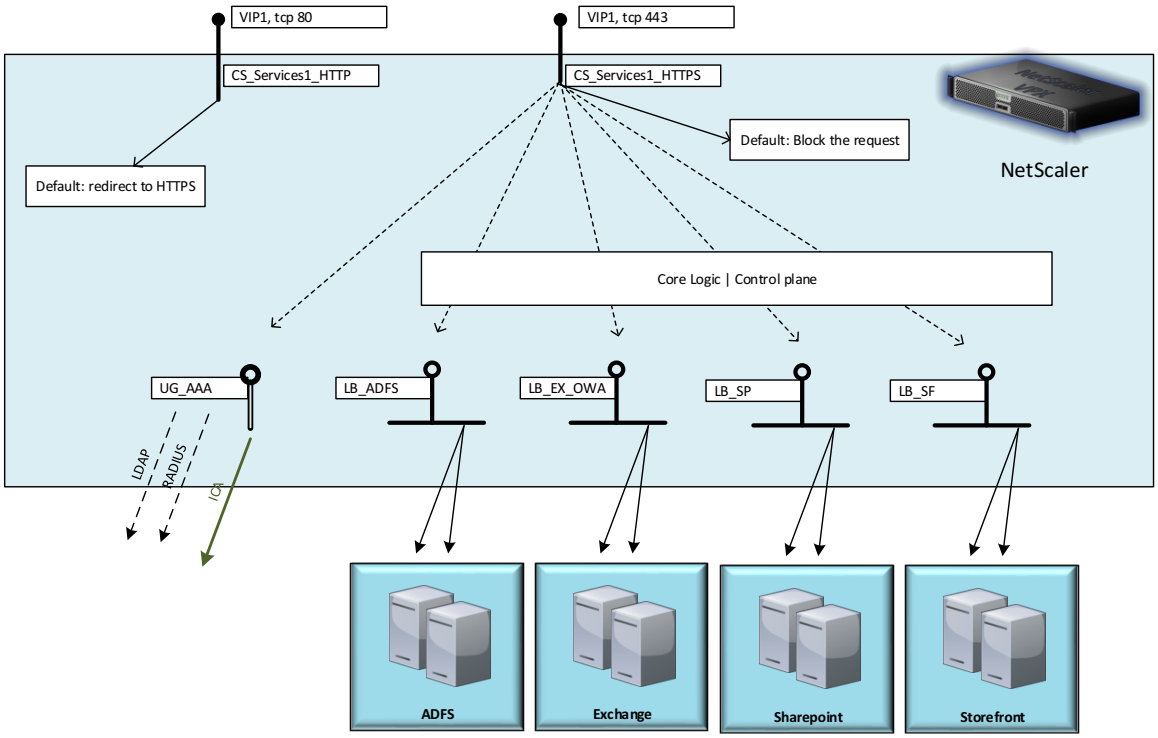
- Some applications require two-factor authentication.
- Some applications should be accessible anonymously.
- We really want to use only 1 IP address per “tenant”.

BASIC DESIGN

A deployment normally/regularly has the following basic ingredients:

- A logon point (AAA / Universal Gateway)
- A HTTP and a HTTPS version of a Content Switching Virtual Server
- A default Redirect to HTTPS
- Redirect capabilities (example: redirect an empty path to some sub-path)
- Content Switching Policies which define the steering.

With the exception of the logon point, the Core-Logic will take care of all basic ingredients. This leads to the following typical design:



Other features like Rewrites, Application Firewall, Caching, etc. are application specific and must be configured on the individual Load Balancing Virtual Servers for an application.

INSTALLATION

The installation of the Core-Logic code is very easy, as outlined below.

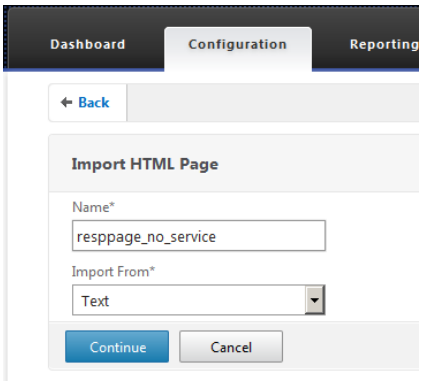
STEP-BY-STEP

Responder HTML pages

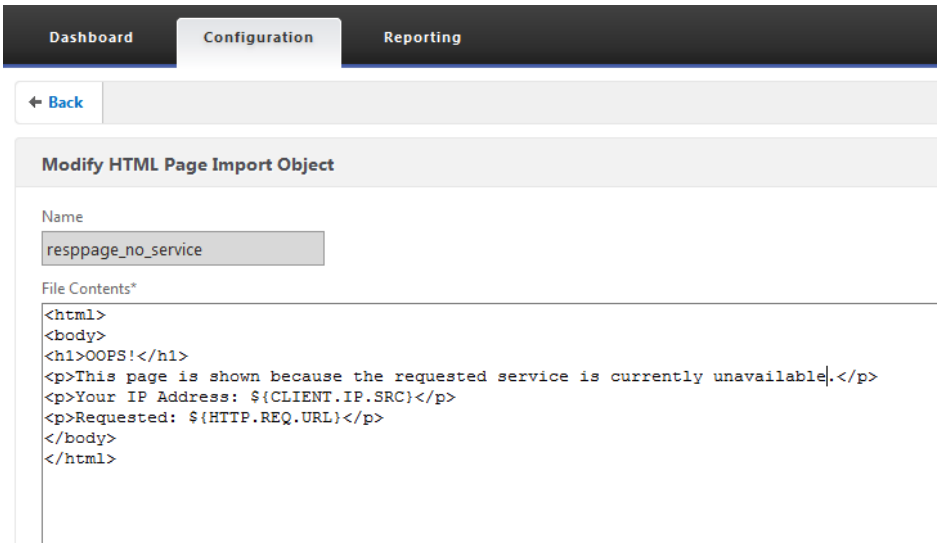
resppage_no_service

Choose: **AppExpert | Responder | HTML Page Imports**, Click **Add**.

Make sure the import page is named "***resppage_no_service***", the core logic will refer to this name later on.



Click **Continue**.



This page will be shown if the Core-Logic detects a service has been configured, but the actual virtual server is currently not available.

Click **Done.**

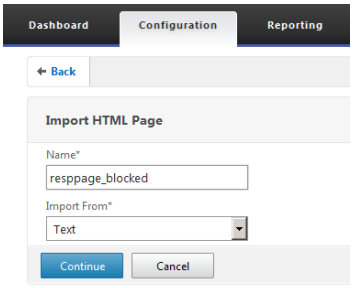
```
<html>
<body>
<h1>OOPS!</h1>
<p>This page is shown because the requested service is currently unavailable.</p>
<p>Your IP Address: ${CLIENT.IP.SRC}</p>
<p>Requested: ${HTTP.REQ.URL}</p>
</body>
</html>
```

Note: *You might want to adjust this basic html code to reflect standard messages within your organization.*

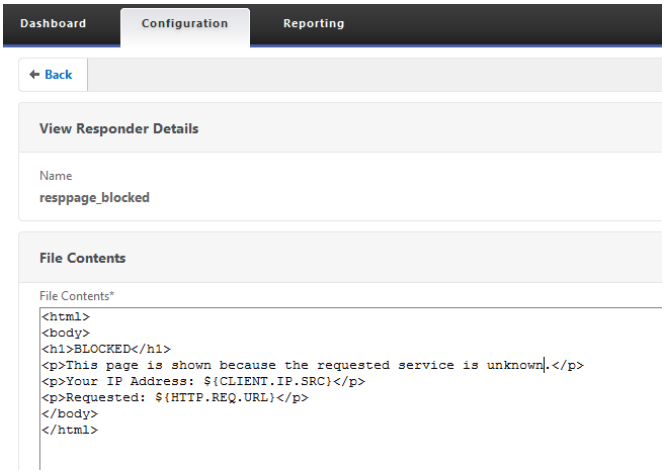
resppage_blocked

Choose: **AppExpert | Responder | HTML Page Imports**, Click **Add**.

Make sure the import page is named "**resppage_blocked**", the core logic will refer to this name later on.



Click **Continue**.



This page will be shown if the Core-Logic detects a request for a service which is not configured.

Click **Done.**

```
<html>
<body>
<h1>BLOCKED</h1>
<p>This page is shown because the requested service is unknown.</p>
<p>Your IP Address: ${CLIENT.IP.SRC}</p>
<p>Requested: ${HTTP.REQ.URL}</p>
</body>
</html>
```

Note: *You might want to adjust this basic html code to reflect standard messages within your organization.*

Core-Logic Module

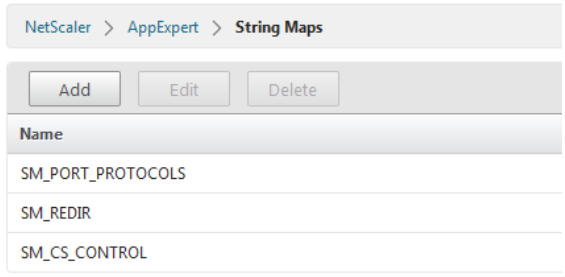
To install the Core-Logic, deploy the scripts through the command-line interface:

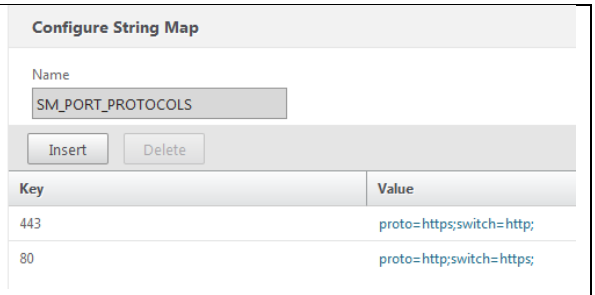
- Open an SSH shell to the NetScaler appliance.
- Copy/paste the code into the CLI
    - Note: Make sure you paste the different files in the correct order!
- Save the configuration!!

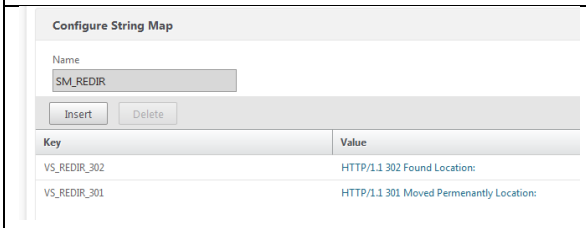
Post-Installation

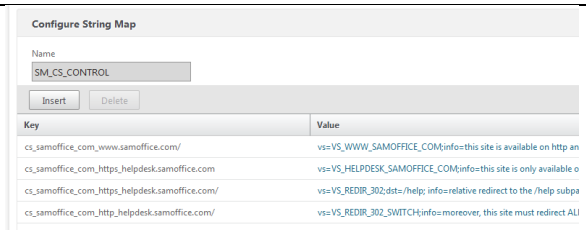
The following items should be visible in the GUI after installing the core-logic files:

String Maps



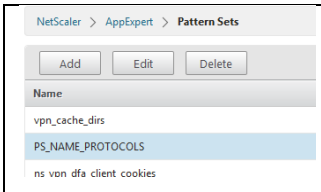


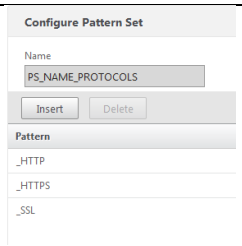




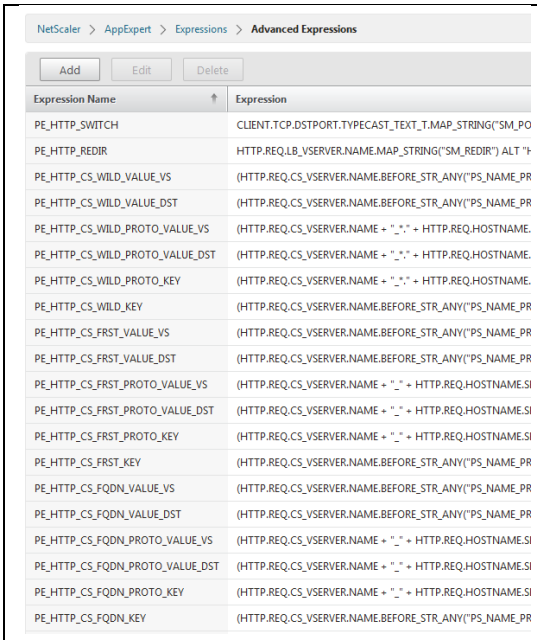
(filled with sample data)

Pattern Sets





Policy Expressions


NetScaler > AppExpert > Expressions > Advanced Expressions

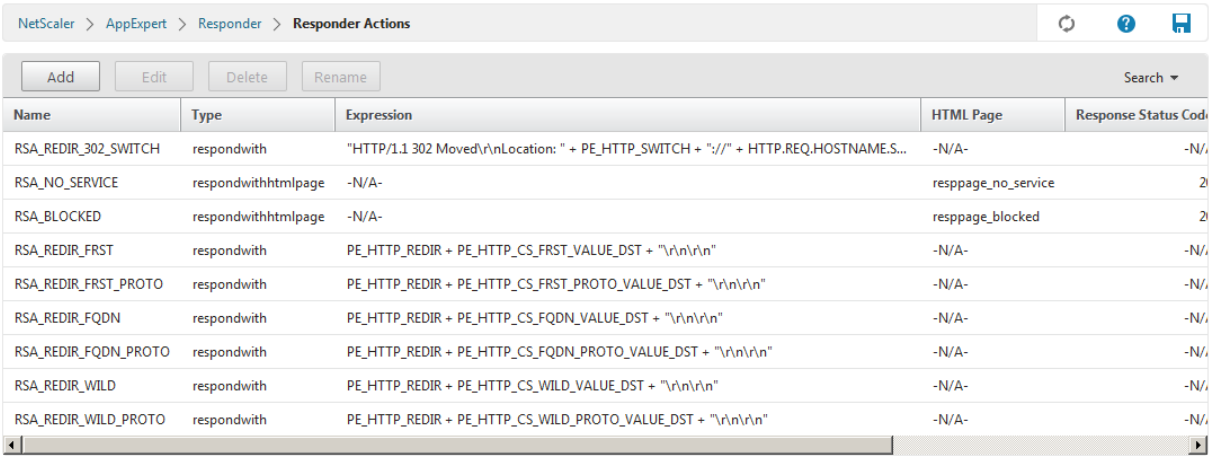
Add | Edit | Delete

Expression Name	Expression
PE_HTTP_SWITCH	CLIENT.TCP.DSTPORT.TYPECAST_TEXT_T.MAP_STRING("SM_PO
PE_HTTP_REDIR	HTTP.REQ.LB_VSERVER.NAME.MAP_STRING("SM_REDIR") ALT "H
PE_HTTP_CS_WILD_VALUE_VS	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR
PE_HTTP_CS_WILD_VALUE_DST	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR
PE_HTTP_CS_WILD_PROTO_VALUE_VS	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.
PE_HTTP_CS_WILD_PROTO_VALUE_DST	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.
PE_HTTP_CS_WILD_PROTO_KEY	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.
PE_HTTP_CS_WILD_KEY	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR
PE_HTTP_CS_FRST_VALUE_VS	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR
PE_HTTP_CS_FRST_VALUE_DST	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR
PE_HTTP_CS_FRST_PROTO_VALUE_VS	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.SI
PE_HTTP_CS_FRST_PROTO_VALUE_DST	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.SI
PE_HTTP_CS_FRST_PROTO_KEY	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.SI
PE_HTTP_CS_FRST_KEY	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR
PE_HTTP_CS_FQDN_VALUE_VS	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR
PE_HTTP_CS_FQDN_VALUE_DST	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR
PE_HTTP_CS_FQDN_PROTO_VALUE_VS	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.SI
PE_HTTP_CS_FQDN_PROTO_VALUE_DST	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.SI
PE_HTTP_CS_FQDN_PROTO_KEY	(HTTP.REQ.CS_VSERVER.NAME + "_" + HTTP.REQ.HOSTNAME.SI
PE_HTTP_CS_FQDN_KEY	(HTTP.REQ.CS_VSERVER.NAME.BEFORE_STR_ANY("PS_NAME_PR

These expressions can be considered the "core-logic"

Responder

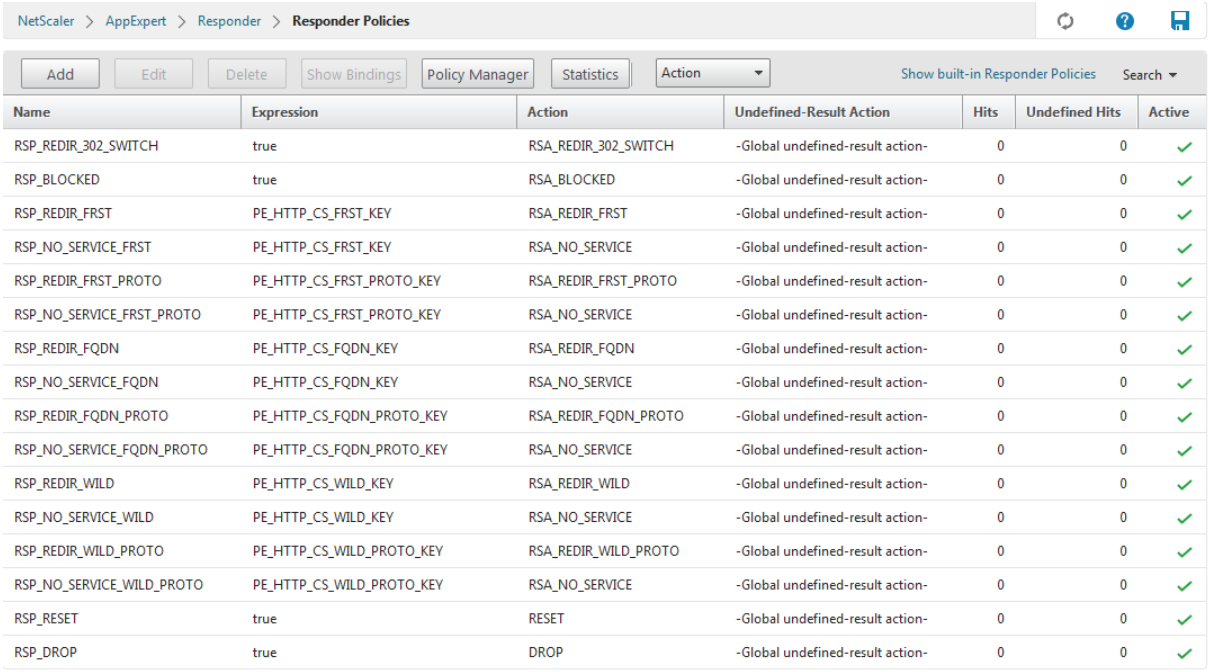
Responder Actions


NetScaler > AppExpert > Responder > Responder Actions

Add | Edit | Delete | Rename | Search

Name	Type	Expression	HTML Page	Response Status Cod
RSA_REDIR_302_SWITCH	respondwith	"HTTP/1.1 302 Moved\r\nLocation: " + PE_HTTP_SWITCH + "://" + HTTP.REQ.HOSTNAME.S...	-N/A-	-N/
RSA_NO_SERVICE	respondwithhtmlpage	-N/A-	resppage_no_service	2
RSA_BLOCKED	respondwithhtmlpage	-N/A-	resppage_blocked	2
RSA_REDIR_FRST	respondwith	PE_HTTP_REDIR + PE_HTTP_CS_FRST_VALUE_DST + "\r\n\r\n"	-N/A-	-N/
RSA_REDIR_FRST_PROTO	respondwith	PE_HTTP_REDIR + PE_HTTP_CS_FRST_PROTO_VALUE_DST + "\r\n\r\n"	-N/A-	-N/
RSA_REDIR_FQDN	respondwith	PE_HTTP_REDIR + PE_HTTP_CS_FQDN_VALUE_DST + "\r\n\r\n"	-N/A-	-N/
RSA_REDIR_FQDN_PROTO	respondwith	PE_HTTP_REDIR + PE_HTTP_CS_FQDN_PROTO_VALUE_DST + "\r\n\r\n"	-N/A-	-N/
RSA_REDIR_WILD	respondwith	PE_HTTP_REDIR + PE_HTTP_CS_WILD_VALUE_DST + "\r\n\r\n"	-N/A-	-N/
RSA_REDIR_WILD_PROTO	respondwith	PE_HTTP_REDIR + PE_HTTP_CS_WILD_PROTO_VALUE_DST + "\r\n\r\n"	-N/A-	-N/

Responder Policies



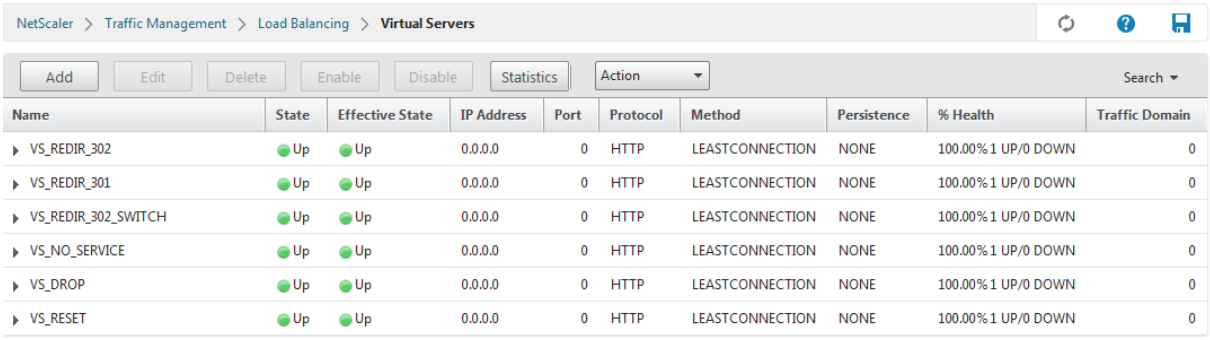
NetScaler > AppExpert > Responder > **Responder Policies**

Add | Edit | Delete | Show Bindings | Policy Manager | Statistics | Action | Show built-in Responder Policies | Search

Name	Expression	Action	Undefined-Result Action	Hits	Undefined Hits	Active
RSP_REDIR_302_SWITCH	true	RSA_REDIR_302_SWITCH	-Global undefined-result action-	0	0	✓
RSP_BLOCKED	true	RSA_BLOCKED	-Global undefined-result action-	0	0	✓
RSP_REDIR_FRST	PE_HTTP_CS_FRST_KEY	RSA_REDIR_FRST	-Global undefined-result action-	0	0	✓
RSP_NO_SERVICE_FRST	PE_HTTP_CS_FRST_KEY	RSA_NO_SERVICE	-Global undefined-result action-	0	0	✓
RSP_REDIR_FRST_PROTO	PE_HTTP_CS_FRST_PROTO_KEY	RSA_REDIR_FRST_PROTO	-Global undefined-result action-	0	0	✓
RSP_NO_SERVICE_FRST_PROTO	PE_HTTP_CS_FRST_PROTO_KEY	RSA_NO_SERVICE	-Global undefined-result action-	0	0	✓
RSP_REDIR_FQDN	PE_HTTP_CS_FQDN_KEY	RSA_REDIR_FQDN	-Global undefined-result action-	0	0	✓
RSP_NO_SERVICE_FQDN	PE_HTTP_CS_FQDN_KEY	RSA_NO_SERVICE	-Global undefined-result action-	0	0	✓
RSP_REDIR_FQDN_PROTO	PE_HTTP_CS_FQDN_PROTO_KEY	RSA_REDIR_FQDN_PROTO	-Global undefined-result action-	0	0	✓
RSP_NO_SERVICE_FQDN_PROTO	PE_HTTP_CS_FQDN_PROTO_KEY	RSA_NO_SERVICE	-Global undefined-result action-	0	0	✓
RSP_REDIR_WILD	PE_HTTP_CS_WILD_KEY	RSA_REDIR_WILD	-Global undefined-result action-	0	0	✓
RSP_NO_SERVICE_WILD	PE_HTTP_CS_WILD_KEY	RSA_NO_SERVICE	-Global undefined-result action-	0	0	✓
RSP_REDIR_WILD_PROTO	PE_HTTP_CS_WILD_PROTO_KEY	RSA_REDIR_WILD_PROTO	-Global undefined-result action-	0	0	✓
RSP_NO_SERVICE_WILD_PROTO	PE_HTTP_CS_WILD_PROTO_KEY	RSA_NO_SERVICE	-Global undefined-result action-	0	0	✓
RSP_RESET	true	RESET	-Global undefined-result action-	0	0	✓
RSP_DROP	true	DROP	-Global undefined-result action-	0	0	✓

Load Balancing

Virtual Servers

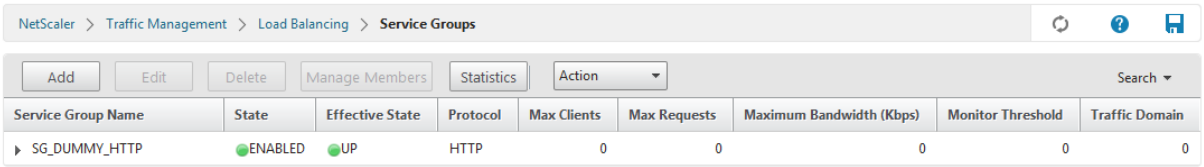


NetScaler > Traffic Management > Load Balancing > **Virtual Servers**

Add | Edit | Delete | Enable | Disable | Statistics | Action | Search

Name	State	Effective State	IP Address	Port	Protocol	Method	Persistence	% Health	Traffic Domain
VS_REDIR_302	Up	Up	0.0.0.0	0	HTTP	LEASTCONNECTION	NONE	100.00% 1 UP/0 DOWN	0
VS_REDIR_301	Up	Up	0.0.0.0	0	HTTP	LEASTCONNECTION	NONE	100.00% 1 UP/0 DOWN	0
VS_REDIR_302_SWITCH	Up	Up	0.0.0.0	0	HTTP	LEASTCONNECTION	NONE	100.00% 1 UP/0 DOWN	0
VS_NO_SERVICE	Up	Up	0.0.0.0	0	HTTP	LEASTCONNECTION	NONE	100.00% 1 UP/0 DOWN	0
VS_DROP	Up	Up	0.0.0.0	0	HTTP	LEASTCONNECTION	NONE	100.00% 1 UP/0 DOWN	0
VS_RESET	Up	Up	0.0.0.0	0	HTTP	LEASTCONNECTION	NONE	100.00% 1 UP/0 DOWN	0

Service Groups

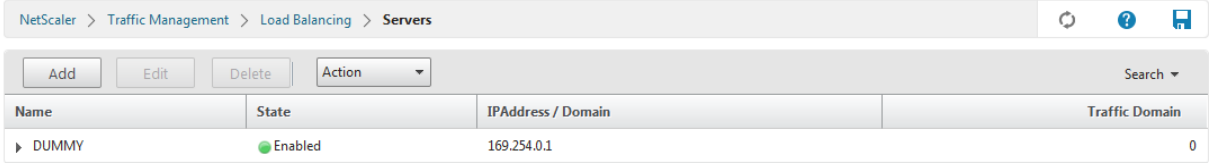


NetScaler > Traffic Management > Load Balancing > **Service Groups**

Add | Edit | Delete | Manage Members | Statistics | Action | Search

Service Group Name	State	Effective State	Protocol	Max Clients	Max Requests	Maximum Bandwidth (Kbps)	Monitor Threshold	Traffic Domain
SG_DUMMY_HTTP	ENABLED	UP	HTTP	0	0	0	0	0

Servers



NetScaler > Traffic Management > Load Balancing > **Servers**

Add | Edit | Delete | Action | Search

Name	State	IPAddress / Domain	Traffic Domain
DUMMY	Enabled	169.254.0.1	0

Content Switching

Content Switching Actions

NetScaler > Traffic Management > Content Switching > **Content Switching Actions**

Add | Edit | Delete | Rename | Search

Name	Target Virtual Server	Hits	Undefined Hits	Reference Count	Comment
CSA_FRST_VALUE_VS	**Expression :** PE_HTTP_CS_FRST_VALUE_VS	0	0	1	
CSA_FRST_PROTO_VALUE_VS	**Expression :** PE_HTTP_CS_FRST_PROTO_VALUE_VS	0	0	1	
CSA_FQDN_VALUE_VS	**Expression :** PE_HTTP_CS_FQDN_VALUE_VS	0	0	1	
CSA_FQDN_PROTO_VALUE_VS	**Expression :** PE_HTTP_CS_FQDN_PROTO_VALUE_VS	0	0	1	
CSA_WILD_VALUE_VS	**Expression :** PE_HTTP_CS_WILD_VALUE_VS	0	0	1	
CSA_WILD_PROTO_VALUE_VS	**Expression :** PE_HTTP_CS_WILD_PROTO_VALUE_VS	0	0	1	

Content Switching Policies

NetScaler > Traffic Management > Content Switching > **Content Switching Policies**

Add | Edit | Delete | Action | Search

Name	Action	Log Action	URL	Expression	Domain	Hits
CSP_FRST	CSA_FRST_VALUE_VS			PE_HTTP_CS_FRST_KEY		0
CSP_FRST_PROTO	CSA_FRST_PROTO_VALUE_VS			PE_HTTP_CS_FRST_PROTO_KEY		0
CSP_FQDN	CSA_FQDN_VALUE_VS			PE_HTTP_CS_FQDN_KEY		0
CSP_FQDN_PROTO	CSA_FQDN_PROTO_VALUE_VS			PE_HTTP_CS_FQDN_PROTO_KEY		0
CSP_WILD	CSA_WILD_VALUE_VS			PE_HTTP_CS_WILD_KEY		0
CSP_WILD_PROTO	CSA_WILD_PROTO_VALUE_VS			PE_HTTP_CS_WILD_PROTO_KEY		0

User Guide

The following section will provide you with detailed instructions on how to use the Core-Logic.

First of all, the actual name of the content switch is bound to the following limitations:

- The HTTP version of the content switch should end with _HTTP
- The HTTPS version of the content switch should end with _HTTPS (or _SSL)
- Both versions of the Content switch need to start off using the same name

Samples:

- A Content Switching Virtual Server for HTTP:
  - CS_Tenant1_HTTP
- A Content Switching Virtual Server for HTTPS:
  - CS_Tenant1_HTTPS

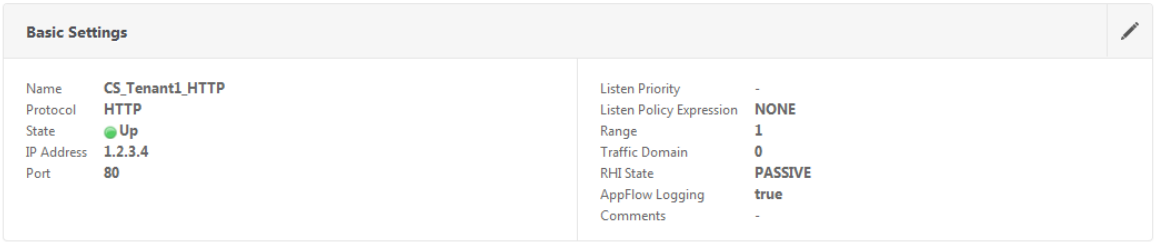
In-housing a Tenant

In-housing a new tenant equals the creation of two new Content Switching Virtual Servers and binding the Core-Logic policies with their correct priorities.

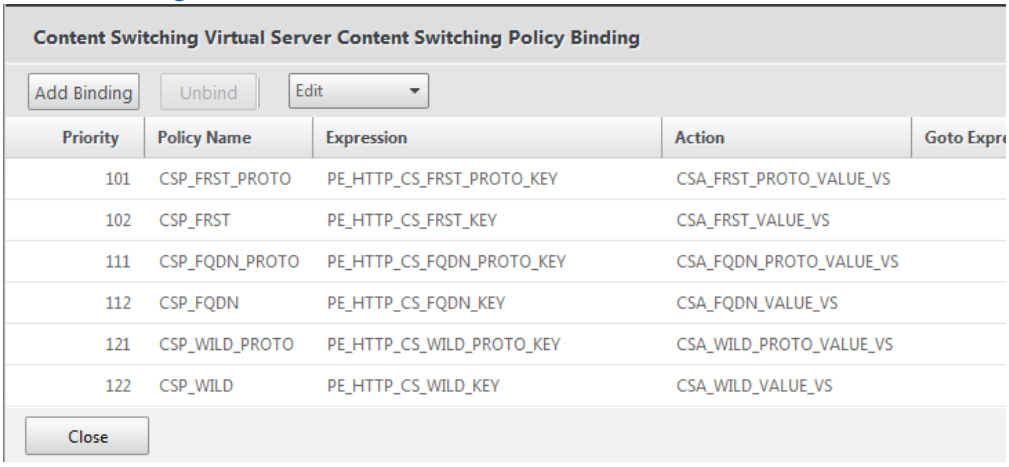
Content Switching Virtual Servers

CS_Tenant1_HTTP

Content Switching Virtual Server



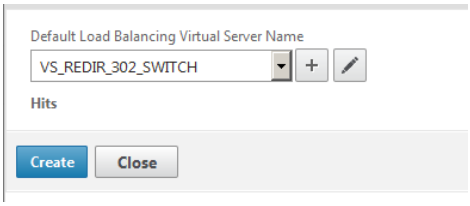
Policy Bindings



***Note**: Make sure the Content Switching Policy bindings have the correct priority.*

Default Load Balancing Virtual Server

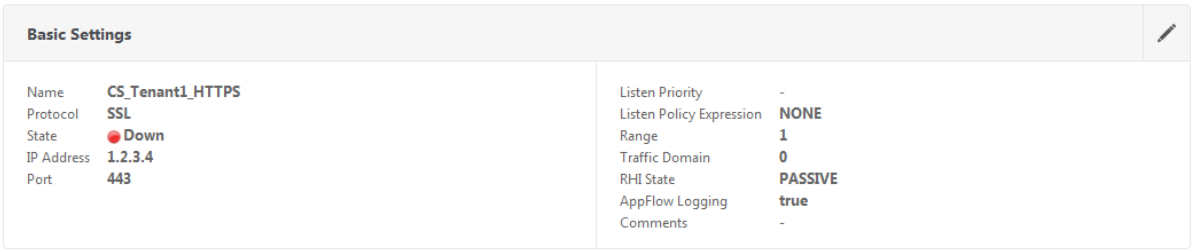
The default Load Balancing Virtual Server for this Content Switching Virtual Server is **VS_REDIR_302_SWITCH**, as we wish to redirect all traffic from HTTP to HTTPS.



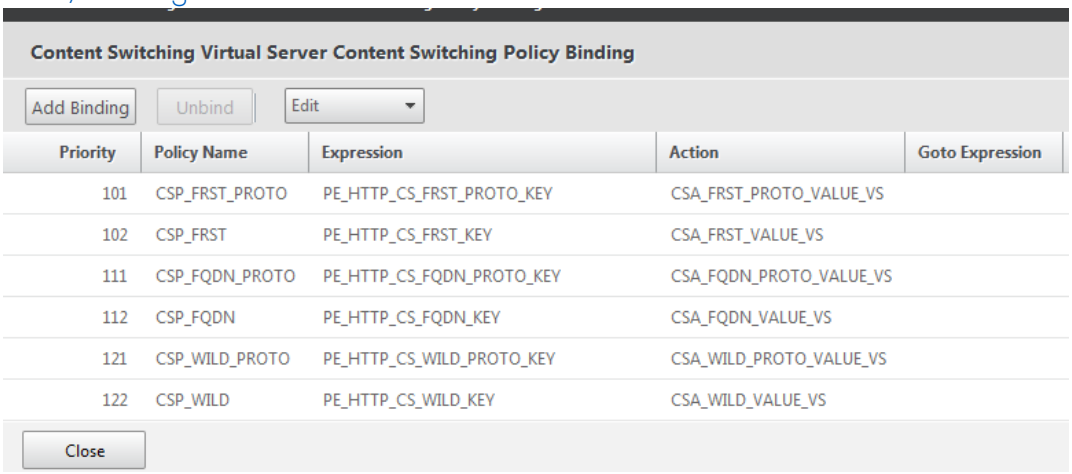
The **VS_REDIR_302_SWITCH** Load Balancing Virtual Server redirects the user in this situation to HTTPS.

CS_Tenant1_HTTPS

Content Switching Virtual Server



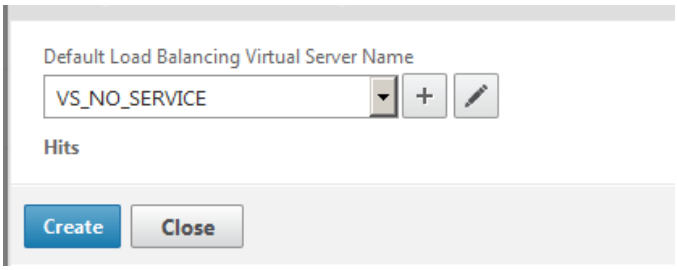
Policy Bindings



***Note**: This Content Switching Virtual Server has the same Content Switching Policy bindings, using the same priorities.*

Default Load Balancing Virtual Server

The default Load Balancing Virtual Server for this Content Switching Virtual Server is **VS_NO_SERVICE**



The **VS_NO_SERVICE** Load Balancing Virtual Server informs the user that:

- The requested application is currently unavailable (down)
- The requested application is unknown to the Control Plane

Extra Configuration

Additional resources will be bound to the Content Switching Virtual Server **CS_Tenant1_HTTPS**:

- One, or more valid certificates (using Wildcard/SAN certificates, optionally using SNI)
- An AAA or Universal Gateway authentication virtual server.

Deployment Script

Content Switching Virtual Server for HTTP

```
add cs vserver CS_[TENANTNAME]_HTTP HTTP [VIP-Address] 80 -cltTimeout 180
bind cs vserver CS_[TENANTNAME]_HTTP -policyName CSP_FRST_PROTO -priority 101
bind cs vserver CS_[TENANTNAME]_HTTP -policyName CSP_FRST -priority 102
bind cs vserver CS_[TENANTNAME]_HTTP -policyName CSP_FQDN_PROTO -priority 111
bind cs vserver CS_[TENANTNAME]_HTTP -policyName CSP_FQDN -priority 112
bind cs vserver CS_[TENANTNAME]_HTTP -policyName CSP_WILD_PROTO -priority 121
bind cs vserver CS_[TENANTNAME]_HTTP -policyName CSP_WILD -priority 122
bind cs vserver CS_[TENANTNAME]_HTTP -lbvserver VS_REDIR_302_SWITCH
```

Content Switching Virtual Server for HTTPS

```
add cs vserver CS_[TENANTNAME]_HTTPS SSL [VIP-Address] 443 -cltTimeout 180
bind cs vserver CS_[TENANTNAME]_HTTPS -policyName CSP_FRST_PROTO -priority 101
bind cs vserver CS_[TENANTNAME]_HTTPS -policyName CSP_FRST -priority 102
bind cs vserver CS_[TENANTNAME]_HTTPS -policyName CSP_FQDN_PROTO -priority 111
bind cs vserver CS_[TENANTNAME]_HTTPS -policyName CSP_FQDN -priority 112
bind cs vserver CS_[TENANTNAME]_HTTPS -policyName CSP_WILD_PROTO -priority 121
bind cs vserver CS_[TENANTNAME]_HTTPS -policyName CSP_WILD -priority 122
bind cs vserver CS_[TENANTNAME]_HTTPS -lbvserver VS_NO_SERVICE
```

***Note**: Do not forget to bind Certificates to this Content Switching Virtual Server.*

DEPLOYING APPLICATIONS FOR A TENANT

Web Application X

Web application X is configured as a non-addressable Load Balancing Virtual Server: ***VS_T1_Web***

- The basic FQDN for the web application X is: ***www.tenant1.com***
- All FQDN using the ***tenant1.com*** domain should be redirected to "**www.tenant1.com**"
    - Redirect using a 301, moved permanently
- If the path is **empty**, we should redirect the user to ***/app1***
- The application should always run on HTTPS

We add to the string map **SM_CS_CONTROL**:

Key	Value
cs_tenant1_https_www.tenant1.com	vs=VS_T1_Web;
cs_tenant1_tenant1.com	vs=VS_REDIR_301;dst=https://www.tenant1.com;
cs_tenant1_*.tenant1.com	vs=VS_REDIR_301;dst=//www.tenant1.com;
cs_tenant1_https_www.tenant1.com/	vs=VS_REDIR_302;dst=/app1;

Note*: the redirect to HTTPS is performed by default due to the configuration of CS_Tenant1_HTTP.*

Sub-path with different configuration:

If the tenant has ***/app2*** added on their webserver in a later stage, and should run on ***HTTP*** only, we add the following entries to the string map **SM_CS_CONTROL**:

Key	Value
cs_tenant1_http_www.tenant1.com/app2	vs=VS_T1_Web;
cs_tenant1_https_www.tenant1.com/app2	vs=VS_REDIR_302_SWITCH;

Microsoft Exchange 2013

Using the deployment guide provided by Citrix, following Load Balancing Virtual Servers are created:

Application Component	Load Balancing Virtual Servers
Outlook web access	Vs_ t1_ex_owa
Ecp	Vs_ t1_ex_ecp
Ews	Vs_ t1_ex_ews
Eas	Vs_ t1_ex_eas
Oab	Vs_ t1_ex_oab
RPC	Vs_ t1_ex_rcp
Mapi	Vs_t1_ex_mapi
Autodiscover	Vs_t1_ex_autod

Note: The deployment guide for Microsoft Exchange 2013 can be found at the following *url*[2].

[2] Microsoft Exchange 2013 – Deployment Guide: *https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/microsoft-exchange-2013-citrix-netscaler-deployment-guide.pdf*

After the creation of the Load Balancing Virtual Servers we only need to edit the Control Plane, by adding the necessary entries:

Key	Value
cs_tenant1_https_mail.tenant1.com/owa	vs=Vs_t1_ex_owa;
cs_tenant1_https_mail.tenant1.com/eas	vs=Vs_t1_ex_eas;
cs_tenant1_https_mail.tenant1.com/ews	vs=Vs_t1_ex_ews;
cs_tenant1_https_mail.tenant1.com/ecp	vs=Vs_t1_ex_ecp;
cs_tenant1_https_mail.tenant1.com/autodiscover	vs=Vs_t1_ex_autod;
cs_tenant1_autodiscover.tenant1.com	vs=VS_REDIR_302;dst=https://mail.tenant1.com/AutoDiscover/AutoDiscover.xml;
cs_tenant1_https_mail.tenant1.com/	vs=VS_REDIR_302;dst= /owa;
cs_tenant1_http_mail.tenant1.com	vs=VS_REDIR_SWITCH;info=[3];

Note: *there are multiple ways to implement "autodiscover" for outlook, depending on the configuration of Exchange 2013.*

[3] This one is needed because we redirected the wildcard *.tenant1.com to www.tenant.com earlier

Control Plane – Coding

The **SM_CS_CONTROL** entries are designed to be human-readable, even without a deeper understanding of NetScaler or Core-Logic.

A string map consists key-value pairs, which are being used by the Core-Logic.
In order for the Control-Pane to work, some rules must be kept in mind when editing the string map.

Keys

The key describes when the core logic should take action:

- The key is always in **lowercase!**
- A key cannot be used twice (it is the index for the String Map)
- The key consists of 2 parts separated by a single underscore (_):
  - The full name of the content switching virtual server (e.g. cs_tenant1_https) **or** the common part of the name for HTTP|HTTPS content switching virtual servers (e.g. cs_tenant1).
  - The url we want to take action on:
    - FQDN (www.tenant1.com)
    - FQDN/1stpath (www.tenant1.com/app2)
    - Wildcard Domain (*.tenant1.com)[4]

Values

The value describes what action should be taken:

- **vs=**[a load balancing virtual server name]**;**
  - ***Mandatory!***
  - **Do not forget the semicolon ";" at the end!**
  - Special VServers:
    - VS_REDIR_302_SWITCH (redirect http->https or https->http)
    - VS_REDIR_301 (redirect "301 moved permanently" to the **dst value**)
    - VS_REDIR_302 (redirect "302 found" to the **dst value**)
    - VS_DROP (drops the request)
    - VS_RESET (resets the request)

- **dst=**[a destination reference]**;**
  - ***Mandatoy for VS_REDIR_301 and VS_REDIR_302!***
  - **Do not forget the semicolon ";" at the end!**
  - Both VS_REDIR_301 and VS_REDIR_302 perform relative redirects when using the **dst** entry.

- **info=**[some remark on the entry]**;**
  - **Optional**
  - **Do not forget the semicolon ";" at the end!**

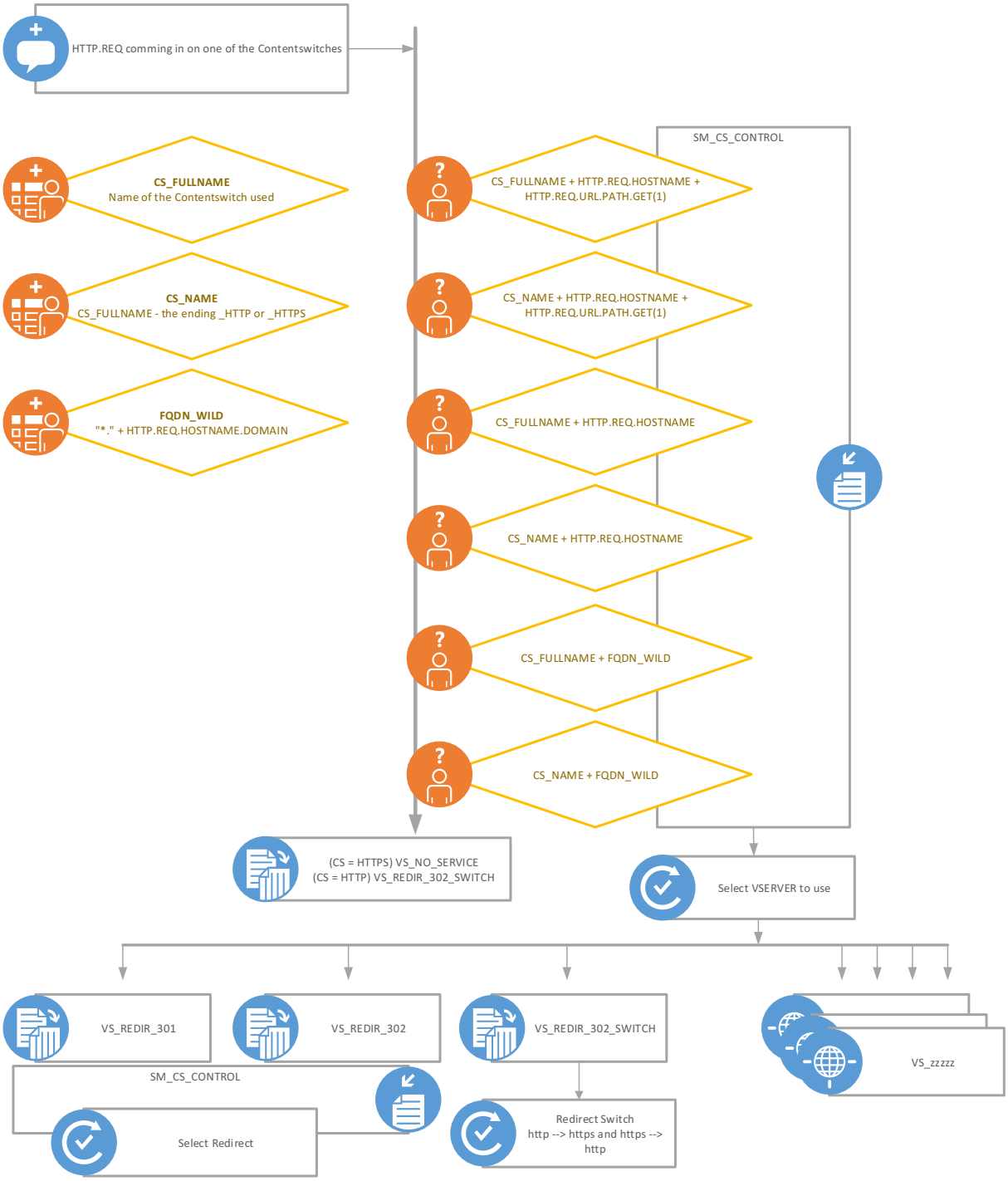
[4] The entry tenant1.com refers tot he FQDN, *.tenant1.com refers to all the subdomains!!

Control Plane – Result

The Control Plane for our tenant1 would look like this:

```
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_www.tenant1.com" "vs=VS_T1_Web;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_tenant1.com"
"vs=VS_REDIR_301;dst=https://www.tenant1.com;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_*.tenant1.com"
"vs=VS_REDIR_301;dst=//www.tenant1.com;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_www.tenant1.com/" "vs=VS_REDIR_302;dst=/app1;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_http_www.tenant1.com/app2" "vs=VS_T1_Web;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_www.tenant1.com/app2" "vs=VS_REDIR_302_SWITCH;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_mail.tenant1.com/owa" "vs=Vs_t1_ex_owa;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_mail.tenant1.com/eas" "vs=Vs_t1_ex_eas;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_mail.tenant1.com/ews" "vs=Vs_t1_ex_ews;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_mail.tenant1.com/ecp" "vs=Vs_t1_ex_ecp;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_mail.tenant1.com/autodiscover"
"vs=Vs_t1_ex_autod;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_autodiscover.tenant1.com"
"Vs=VS_REDIR_302;dst=https://mail.tenant1.com/AutoDiscover/AutoDiscover.xml;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_https_mail.tenant1.com/" "Vs=VS_REDIR_302;dst= /owa;"
Bind stringmap SM_CS_CONTROL "cs_tenant1_http_mail.tenant1.com"
"Vs=VS_REDIR_SWITCH;info=*.tennant1.com goes to www.tenant1.com";"
```

Control Plane – Processing Flow



CONTROL PLANE – CLI MANAGEMENT

The string map **SM_CS_CONTROL** can be managed through the GUI. Although with a bit of practice, using the command line interface is generally easier and faster for larger configurations.

Adding Entries

The basic command to add an entry to the SM_CS_CONTROL string map:

```
bind stringmap SM_CS_CONTROL [key] [value]
```

Tip: always put the key and value between "[value]"

Deleting Entries

The basic command to remove an entry from the SM_CS_CONTROL string map:

```
unbind stringmap SM_CS_CONTROL [key]
```

Showing Entries

The command to get all entries for cs_tennant1 (http and https):

```
show run | grep SM_CS_CONTROL | grep cs_tenant1
```

CONCLUSIONS

The Core-Logic is an attempt at creating a **unified way to integrate applications** into one or more content switching virtual servers. The Core-Logic links the Content Switching Virtual Server(s) to the applications using the Control Plane.

It generalizes the most comment Content Switching Policies and Responder Policies into a **single set of code**. In addition, the Core-Logic code itself is **not specific** for a Content Switching Virtual Server, a Load Balancing Virtual Server or a given redirect.

Creating a new Content Switching Virtual Server can easily be automated, since the policies bound to a Content Switching Virtual Server are static. A new tenant can be deployed by having 3 parameters:

- [Name]
- [VIP]
- [Certificate]

The steering is done through a **single Control Plane**, which can also easily be automated. Changes to this string map can be considered a **lower-impact change** to the configuration.

The Control Plane uses the “more restrictive” principal to determine the flow of the requests, resulting in the following list of keys from least restrictive to most restrictive:

- cs_tenant1_[wildcarddomain]
- cs_tenant1_[protocol]_[wildcarddomain]
- cs_tenant1_[fqdn]
- cs_tenant1_[protocol]_[fqdn]
- cs_tenant1_[fqdn+1st path]
- cs_tenant1_[protocol]_[fqdn+1st path]

For DTAP (development-test-acceptance-production) situations this unification of code is helpful.

For Multi-tenant / hosting providers it can help keeping control of application delivery for their customers and an easier deployment of new tenants and/or applications.