



# Implementing NetScaler VPX™

Leverage the features of NetScaler VPX™ to optimize and deploy responsive web services and applications on multiple virtualization platforms

**Marius Sandbu**

**[PACKT]** enterprise   
PUBLISHING professional expertise distilled

[www.allitebooks.com](http://www.allitebooks.com)

# Implementing NetScaler VPX™

Leverage the features of NetScaler VPX™ to optimize and deploy responsive web services and applications on multiple virtualization platforms

**Marius Sandbu**



BIRMINGHAM - MUMBAI

# Implementing NetScaler VPX™

Copyright © 2014 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: April 2014

Production Reference: 1170414

Published by Packt Publishing Ltd.  
Livery Place  
35 Livery Street  
Birmingham B3 2PB, UK.

ISBN 978-1-78217-267-3

[www.packtpub.com](http://www.packtpub.com)

Cover Image by Maruf Ahmed Dhali ([ahmed.maruf@hotmail.com](mailto:ahmed.maruf@hotmail.com))

# Credits

**Author**

Marius Sandbu

**Project Coordinator**

Melita Lobo

**Reviewers**

Kees Baggerman

Anton van Pelt

Daniel Wedel

**Proofreaders**

Maria Gould

Lawrence A. Herman

**Commissioning Editor**

Pramila Balan

**Indexer**

Hemangini Bari

**Acquisition Editor**

Harsha Bharwani

**Graphics**

Disha Haria

**Content Development Editor**

Sriram N

**Production Coordinator**

Nilesh R. Mohite

**Technical Editors**

Taabish Khan

Nikhil Potdukhe

**Cover Work**

Nilesh R. Mohite

**Copy Editor**

Laxmi Subramanian

# Notice

The statements made and opinions expressed herein belong exclusively to the author and reviewers of this publication, and are not shared by or represent the viewpoint of Citrix Systems®, Inc. This publication does not constitute an endorsement of any product, service, or point of view. Citrix® makes no representations, warranties or assurances of any kind, express or implied, as to the completeness, accuracy, reliability, suitability, availability, or currency of the content contained in this publication or any material related to this publication. Any reliance you place on such content is strictly at your own risk. In no event shall Citrix®, its agents, officers, employees, licensees, or affiliates be liable for any damages whatsoever (including, without limitation, damages for loss of profits, business information, or loss of information) arising out of the information or statements contained in the publication, even if Citrix® has been advised of the possibility of such loss or damages.

Citrix®, Citrix Systems®, XenApp®, XenDesktop®, and CloudPortal™ are trademarks of Citrix Systems®, Inc. and/or one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries.

# About the Author

**Marius Sandbu** is a Consultant, Advisor, and Trainer working at the Value Added Distributor (VAD) Commaxx in Norway. He has worked with Microsoft technology for over nine years and has been awarded an MVP title from Microsoft because of his great dedication to the Microsoft community. He is also a board member of the local Microsoft technology user group and has spoken at many public events at both Microsoft and other events. He has always had a high interest in technology. Over the past few years, he has taken over 30 certifications in different areas of technology, and also had a role within Microsoft as an Infrastructure Ranger. He is also a certified Microsoft trainer and has held different courses on System Center and Windows Server. As an experiment to improve his learning skills, he started blogging in 2012 and now has over 2,000 visitors to date. He also contributes to Born To Learn, which is a Microsoft community website for training and certification.



# About the Reviewers

**Kees Baggerman** works for Inter Access as a Senior Technical Consultant. His main areas of work are migrations and implementations of Microsoft and Citrix® infrastructures, writing functional/technical designs for Microsoft infrastructures, Microsoft Terminal Server, or Citrix® (XenApp®, XenDesktop®, and NetScaler®) in combination with RES Workspace Manager and/or RES Automation Manager.

He is a Citrix® Certified Integration Architect, Microsoft Certified IT Professional, RES Certified Professional, and RES Certified Trainer. RES Software also named him RES RSVP in 2010, 2011, 2012, and 2013. He was named the RES Software Most Valuable Professional of 2011.

In 2013, he received the VMware vExpert title. This title is given to individuals who have significantly contributed to the community of VMware users over the preceding year. The title is awarded to individuals (not employers) for their commitment to sharing their knowledge and passion for VMware technology above and beyond their job requirements.

He is a co-founder and member of the Board of the Dutch Citrix® User Group and writes on his website and on the ITVCE Community blog.

**Anton van Pelt** is a consultant with over 10 years of Citrix® experience. His focus is primarily on Enterprise Mobility solutions such as Citrix® XenMobile®, ShareFile®, and NetScaler®. Nevertheless, his interests go much further than this, thus giving him a broad knowledge in complex IT environments. He is active in presenting his technical knowledge throughout the community (Citrix® IRC channel, Citrix® support forums, and NetScaler® KB, among others) and at various congresses. He is also the co-author of PQR's *Enterprise Mobility Management Smackdown* and *User Environment Management Smackdown*. You can contact him at [ape@pqr.nl](mailto:ape@pqr.nl) or follow him on Twitter @antonvanpelt.

**Daniel Wedel** is the Senior Consultant and Founder of Wedel IT, a company specializing in Citrix® and Microsoft technology. With more than 10 years of experience in the Citrix® field, he has extensive knowledge about products. He is passionate about new technologies and uses his expertise to ensure that customer solutions are built to order. In recent years, he has combined consulting and Citrix® training for customers across Norway. He was awarded CCI of the year 2010 – Nordic region. He is also a popular speaker at events, such as VirtualPower, E2E, and the Norwegian Citrix® User Group.

Wedel IT is a consulting company based in Norway that specializes in virtualization technology, primarily Citrix®. The company was founded in 2010. The employees are known for their expertise in the field and work with a range of customers in both the private and the public sector.

---

I would like to thank my nephews Leon and Emanuel; they are true inspirations in my daily life and remind me that it's the little things in life that matter.

---



# www.PacktPub.com

## Support files, eBooks, discount offers and more

You might want to visit [www.PacktPub.com](http://www.PacktPub.com) for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.PacktPub.com](http://www.PacktPub.com) and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at [service@packtpub.com](mailto:service@packtpub.com) for more details.

At [www.PacktPub.com](http://www.PacktPub.com), you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

## Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

## Free Access for Packt account holders

If you have an account with Packt at [www.PacktPub.com](http://www.PacktPub.com), you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

## Instant Updates on New Packt Books

Get notified! Find out when new books are published by following @PacktEnterprise on Twitter, or the *Packt Enterprise* Facebook page.

# Table of Contents

<b>Preface</b>	<b>1</b>
<b>Chapter 1: NetScaler VPX™ 10.1 Basics and Setup</b>	<b>5</b>
<b>Getting started with NetScaler®</b>	<b>5</b>
MPX	7
SDX	8
VPX	8
<b>Licensing</b>	<b>10</b>
<b>Setup scenarios</b>	<b>11</b>
<b>Creating our first setup</b>	<b>12</b>
Dashboard	14
Reporting	15
Configuration	15
<b>NetScaler® modes and features</b>	<b>18</b>
<b>NetScaler® networking</b>	<b>20</b>
NSIP	20
MIP	21
SNIP	21
<b>Summary</b>	<b>24</b>
<b>Chapter 2: NetScaler Gateway™</b>	<b>25</b>
<b>A brief history</b>	<b>25</b>
<b>Understanding the features</b>	<b>26</b>
<b>Deploying ICA Proxy</b>	<b>29</b>
StoreFront integration	38
<b>Deploying VPN</b>	<b>41</b>
<b>Deploying clientless access</b>	<b>43</b>
<b>Binding the features together</b>	<b>44</b>
<b>Tuning</b>	<b>48</b>
Redirection	48

Profiles	50
<b>Testing</b>	<b>51</b>
<b>Summary</b>	<b>52</b>
<b>Chapter 3: Load Balancing</b>	<b>53</b>
<b>Load balancing a generic web application</b>	<b>55</b>
Assigning weights to a service	61
Redirect URL	62
Backup vServer and failover	62
<b>Load balancing StoreFront</b>	<b>63</b>
<b>Load balancing Web Interface</b>	<b>65</b>
<b>Load balancing XML Broker</b>	<b>65</b>
<b>Load balancing Desktop Delivery Controller</b>	<b>66</b>
<b>Load balancing TFTP for provisioning servers</b>	<b>66</b>
<b>Load balancing SharePoint 2013</b>	<b>67</b>
<b>Load balancing Exchange 2013</b>	<b>70</b>
IMAP	71
<b>Load balancing MSSQL</b>	<b>72</b>
<b>Summary</b>	<b>76</b>
<b>Chapter 4: Compression and Caching</b>	<b>77</b>
<b>Compression</b>	<b>78</b>
Implementing compression policies	79
Defining global compression settings	81
Creating custom compression policies	82
Testing our compression policies	84
<b>Caching</b>	<b>85</b>
Enabling caching	85
Creating a content group	86
Creating a caching policy	86
Fine-tuning caching	89
<b>Summary</b>	<b>90</b>
<b>Chapter 5: High Availability and Traffic Analysis</b>	<b>91</b>
<b>Setting up high availability</b>	<b>91</b>
Differences between clustering, HA, and GSLB	95
<b>Using AppFlow® to monitor traffic with NetScaler Insight Center™</b>	<b>99</b>
<b>Traffic analysis with NetScaler® tools and Wireshark</b>	<b>104</b>
Analyzing encrypted content with Wireshark	108
<b>Maintaining security using NetScaler AppFirewall™</b>	<b>110</b>
<b>Summary</b>	<b>116</b>
<b>Index</b>	<b>117</b>

---

# Preface

NetScaler® is becoming more essential in many environments and is often crucial for many of the services it offers. *Implementing NetScaler VPX™* is a book that covers all the basics on how to get started with NetScaler VPX™ in a virtual environment and how to deliver highly available services and remote access to a Citrix® environment.

The book starts with an easy introduction on what the product is, what it can offer, and how to do an initial setup using the command line and the graphical user interface.

Later it goes into some of the more advanced features such as remote access functionality against Citrix® environments, use of different VPN features, and how to set up clientless access.

It also covers high availability features such as active/passive, and clustering and how to load balance much of the commonly used platforms such as SharePoint, Exchange, SQL, and other Citrix® components. It will also show how to optimize web services with features such as caching and compression and many of the built-in optimization features in NetScaler®.

## What this book covers

*Chapter 1, NetScaler VPX™ 10.1 Basics and Setup*, goes through the initial setup of NetScaler VPX™ in a virtual environment. It also describes the different deployment types and different features and settings and what they can do.

*Chapter 2, NetScaler Gateway™*, explains how to set up the NetScaler Gateway™ feature against a XenApp®/XenDesktop® environment, and also covers how to set up SSL-based VPN and use the NetScaler Gateway™ plugin.

*Chapter 3, Load Balancing*, tells us how to set up load balancing against generic web services as well as many of the most used platforms such as Exchange, SharePoint, MSSQL, and other Citrix® products.

*Chapter 4, Compression and Caching*, explains how to set up and configure compression and caching on NetScaler® in order to increase the performance on websites.

*Chapter 5, High Availability and Traffic Analysis*, explains the different high availability features and how to configure them. It will also give a walkthrough on how you can do traffic analysis to troubleshoot network issues with Wireshark. Lastly, it gives an introduction on how to secure web applications using Application Firewall.

## What you need for this book

You can download a trial of the NetScaler® virtual appliance from Citrix® at <https://secureportal.citrix.com/MyCitrix/login/EvalLand.aspx?downloadid=1857216&LandingFrom=1005>.

You should also have a virtual environment with either VMware, Citrix® XenServer®, or Hyper-V. If you do not have a virtual environment, you can test it out on a client hypervisor.

For example, if you are using Windows 8, you can use Client Hyper-V, which is an add-on that needs to be added from programs and features under the Control Panel. Or you can use the VMware player from [https://my.vmware.com/web/vmware/free#desktop\\_end\\_user\\_computing/vmware\\_player/6\\_0](https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/6_0).

## Who this book is for

This book is intended for system administrators who are working with either Citrix® or networking and want to learn how to implement NetScaler VPX™ in a virtual environment for use with, for example, remote access for Citrix® environments, CVPN, and load balancing different services.

## Conventions


In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.


Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "We then use the general `ns_true` expression to apply to the rest and bind a session policy for the rest of the devices."

Any command-line input or output is written as follows:

```
Get-NetworkAdapter -VM NameofVM
```

**New terms** and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Here, click on **Add** and enter the IP address of our DNS server, and leave the rest as default values."

[  Warnings or important notes appear in a box like this. ]

[  Tips and tricks appear like this. ]

## Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to [feedback@packtpub.com](mailto:feedback@packtpub.com), and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on [www.packtpub.com/authors](http://www.packtpub.com/authors).

## Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

## Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

## Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at [copyright@packtpub.com](mailto:copyright@packtpub.com) with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

## Questions

You can contact us at [questions@packtpub.com](mailto:questions@packtpub.com) if you are having a problem with any aspect of the book, and we will do our best to address it.



# 1


## NetScaler VPX™ 10.1 Basics and Setup

Welcome to the first chapter of this book. Throughout the course of this book, we will cover most of the different areas where NetScaler serves its purpose. The first chapter will cover a little introduction of what NetScaler is and some of its features. Throughout this book, we will be focusing mostly on how to set up and deploy a NetScaler VPX in a Hyper-V and System Center environment. This is because in the Nordic market, most of the deployments run on Hyper-V; however, the process is not so different for other hypervisors. So to sum it up, here's what we will cover throughout this chapter:

- Introduction to NetScaler
- The definition of Application Delivery Controller
- NetScaler Gateway
- Differences between VPX, MPX, and SDX
- Editions and models
- Setup and configuring the basics
- Some deployment scenarios

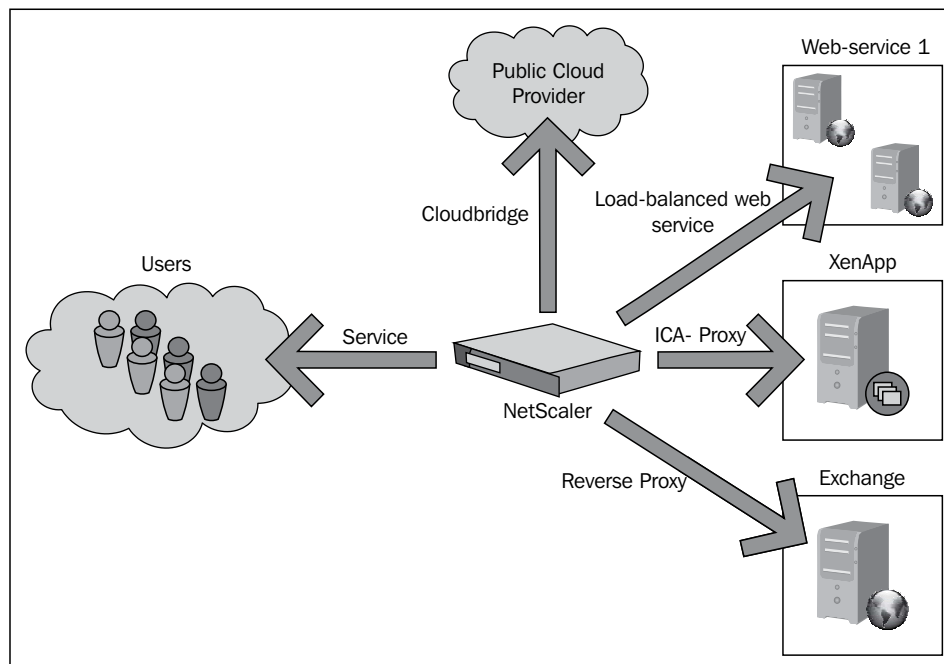
### **Getting started with NetScaler®**

NetScaler was an acquisition that Citrix made back in 2005, and it is one of the best selling products in their portfolio today and is pivotal in many large enterprises. Today, many of the largest IT organizations, such as Microsoft, Google, and eBay to mention a few, are using NetScaler in front of their websites and services to ensure availability.

 We can check the kind of solution an organization is using on their website by using a free web tool from [www.netcraft.com](http://www.netcraft.com). For example, for eBay, go to <http://searchdns.netcraft.com/?restriction=site+contains&host=ebay.com>.

NetScaler can be defined as a network appliance with the primary role of delivering services to the end clients who are connecting to it. It does this through the use of different features, such as load balancing, proxy, gateway solutions, and so on. The commonly used term for it is **Application Delivery Controller (ADC)**, as users in many cases connect to their services through, for example, a load-balanced web service such as NetScaler. It also has many features to optimize network traffic, such as web caching, compression, and SSL offloading, to give a service optimal performance. It also includes features such as application firewall, URL rewrite and responder, global server load balancing, and gateway function for XenApp/XenDesktop to name a few. We will cover some of these features in greater detail in a later chapter.

So its whole purpose is to ensure that a service or an application is delivered through different availability and performance features. The following diagram is an example of some of the different uses of NetScaler, and how users can access their different applications and services:



As we can see in the diagram, there are many ways in which we can deliver and ensure content is delivered to the users. Also, there are features that allow us to bridge different infrastructures such as public cloud providers. We will delve into some of the features throughout the rest of the chapters.

There are a variety of features included in NetScaler; some information about the different features and the product itself can be found in the Citrix eDocs available at <http://support.citrix.com/proddocs/topic/netscaler/ns-gen-netscaler-wrapper-con.html>. eDocs is an ideal place for knowledge and support documentation about setup and configuration of the different features included in NetScaler.

NetScaler comes in three different types of appliances. They are:

- MPX
- SDX
- VPX

## MPX

The MPX is a physical appliance of the NetScaler, which again comes in different models. As an example, the MPX 5550 is the starting platform that consists of an Intel CPU with 8 GB of RAM, and can handle up to 5,000 concurrent SSL VPN sessions and up to 175,000 HTTP requests every second. The MPX 5550 has a maximum throughput of 0.5 Gbps, but it can be upgraded to the 5650 which has 1 Gbps throughput. This only requires a change of license as it still runs on the same hardware. There is a long list of different models that suit most business needs depending on how many users, what kind of services, and what kind of bandwidth are required. The largest physical appliance available is the MPX 21550, which has up to 50 Gbps of throughput.



One of the benefits of NetScaler is that if we need better performance or more bandwidth, we can, in many cases, just upgrade the platform license to the next edition. You can refer to the NetScaler datasheet to see which platforms can be upgraded and also check the specifications of the different platforms at [http://www.citrix.com/content/dam/citrix/en\\_us/documents/products-solutions/netscaler-data-sheet.pdf](http://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/netscaler-data-sheet.pdf).

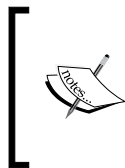
All of the MPX models come with special SSL chips, which are specifically used to handle encrypted traffic (SSL traffic). The NetScaler uses an architecture called nCore, which allows it to intelligently load balance the SSL operations among the chips available on the hardware. This allows for faster handling of the SSL traffic on the regular load balancers. Also, an important point to remember is that each platform has a limit on how many SSL-based operations and throughput it can handle each second, which can be viewed in the earlier mentioned datasheet.

## SDX

The SDX is a special kind of platform available on many of the same models as the MPX as it uses the same underlying hardware. The difference is that the SDX itself cannot do load balancing or any other NetScaler functions as it is just a virtualization platform that runs a virtual NetScaler (VPX) on top of itself. By default, when purchasing an SDX, it ships with five VPXs. SDX runs a customized version of XenServer at the bottom of the appliance, and there we can create multiple VPX instances running on top of it, which have the NetScaler features. This platform is better suited for multitenant environments or when we want to isolate the traffic into separate instances.

## VPX

The VPX is the virtual edition of NetScaler. It has the same features as the MPX; the only difference is that it runs as a virtual appliance instead of as a hardware appliance. There are four different editions of this platform, VPX 10, VPX 200, VPX 1000, and VPX 3000, where the number stands for the throughput of the device in Mbps.



There is also a free edition of the VPX called VPX Express. The VPX Express has the same functionality as VPX standard, but has a limit of 5 Mbps of throughput and is valid for one year at a time. It also gives you access to running up to five users with NetScaler Gateway, which we will go through in the next chapter.


The VPX is available for XenServer, VMware, and Hyper-V, or as an instance on the SDX platform. There is a minor difference between running VPX in a regular virtual environment or as a part of an SDX environment. In an SDX environment, the VPX has access to the onboard SSL chips and is able to handle SSL traffic accordingly. In a regular virtual environment, the VPX can handle only limited SSL traffic as it is dependent on the virtualization host CPUs. Regular CPUs are not designed to handle SSL offload very well as compared to SSL chips; therefore, they have a soft limit on how many SSL connections they can handle. This can be seen in the NetScaler datasheet mentioned earlier.

Barry Schiffer has written an excellent article regarding NetScaler sizing and what model to choose, which I would recommend taking a look at if you are unsure of what to use. It is available at <http://www.barryschiffer.com/citrix-NetScaler-platform-sizing-guide/>.

NetScaler also has different types of editions, and depending on the level will grant access to the different features. The three editions are Standard, Enterprise, and Platinum.

Standard is the most basic edition, and contains most of the basic features, such as load balancing, SQL load balancing, NetScaler Gateway (formerly known as Access Gateway), network optimization, HTTP/URL rewrite, and more. The Enterprise edition gives us **Global Server Load Balancing (GSLB)**, HTTP compression, AAA management, and surge protection. Lastly, the Platinum edition gives us CloudBridge, full NetScaler Insight Center functionality, application firewall, and more. An important point to note here is that on an SDX appliance, all the VPX appliances have Platinum edition features.

Now, many of these features may be unfamiliar to you, but these will be covered throughout the later chapters.

 The complete feature set of NetScaler and its different editions can be found in the NetScaler datasheet available at [http://www.citrix.com/content/dam/citrix/en\\_us/documents/products-solutions/netscaler-datasheet.pdf?accessmode=direct](http://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/netscaler-datasheet.pdf?accessmode=direct). There is also another edition called NetScaler Gateway VPX, which is a virtual appliance containing only the gateway feature.

One of the things that I mentioned earlier was that in case we needed more bandwidth or better performance, we could just upgrade the license to another platform. The same goes for features as well; if we need features that are available in the Enterprise edition and we have only the Standard edition, we just have to buy a license upgrade to access those features. If, for example, we are in a situation where we need more bandwidth for a period of time, we can also purchase something called burst licenses. Burst licenses allow us to extend our bandwidth on the appliance, for example, for 90 days.

## Licensing

When we want to set up or deploy a NetScaler, we need a license in place in order to access the features we want to use. An important point to note here is that there are three types of licenses available for NetScaler:

- **Platform license:** This license is used for NetScaler features and defines the bandwidth.
- **Universal license:** This license is used for NetScaler Gateway features such as SSL VPN, CVPN, SmartAccess, and Endpoint analysis.
- **Feature license:** This license is used for features such as clustering, caching, and so on.

Licenses can be allocated and downloaded from [www.mycitrix.com](http://www.mycitrix.com) under **Licensing**. Here, we need to enter our hardware information, so that the license can be bound to the appliance. An important point to remember is that you need to have a valid Citrix account to be given access to the licenses.



If you do not have access to a regular license, you can download a trial version of the latest NetScaler VPX Platinum edition from Citrix, available at <http://www.citrix.com/products/netScaler-application-delivery-controller/try.html>.

If you want to download a platform license for NetScaler from [www.mycitrix.com](http://www.mycitrix.com), you need to enter the MAC address of the first NIC on your appliance in the **Host ID** field on the website.



If you are deploying a NetScaler Gateway VPX, and you want to download a platform license for it, or generate universal licenses, both of these can be created with the hostname of the appliance instead of the MAC address. These licenses can be generated from the same website.

The MAC address can be found either via the CLI of the appliance, or by using a hypervisor. We will learn more about CLI throughout this chapter. To get the hardware information from the CLI of the appliance, we have to first log in to the NetScaler System CLI, and then switch to the FreeBSD shell by typing `shell` and running the following command:

```
lmutil lmhostid
```

When using a hypervisor, such as the virtual machine manager PowerShell, run the following command:

```
Get-VM | Where { $_.Name -match "VM" } | Get-SCNetworkAdapter | Select  
MACAddress
```

If you are using VMware and have PowerCLI available, we can use a similar command, as follows, to get the same result:

```
Get-NetworkAdapter -VM NameofVM
```

This will give you the host ID/MAC address of the appliance, which needs to be entered on [mycitrix.com](http://mycitrix.com) to generate a platform license. We will cover installing the license a bit later.

## Setup scenarios

When thinking about the deployment of NetScaler, there are a couple of things that need to be taken into consideration, which are listed here:

- How is the network layout between the users and the service?
- What kind of network security is in place?
- Is the business using NAT or any other kind of firewall that requires configuration to allow traffic?
- What service or application is going to be published?

A common scenario is load balancing some sort of a web service to external users. In such a scenario, a business might have a DMZ zone and an intranet zone. One topology that can be used here is that NetScaler can be placed with one interface in the DMZ zone and one interface in the intranet zone. This is also known as a two-armed setup. It is important to note that a two-armed setup is not necessarily two NICs connected to different networks; it might also be multiple VLANs trunked to the same NIC. This is practical for load balancing internal resources as well because the traffic does not need to flow back and forth through the firewall multiple times.

In some cases, because of business requirements, you might have NetScaler attached to only one interface or only one VLAN, which resides in the same zone. This is known as a one-armed setup. Here, NetScaler is placed, for example, in only the DMZ zone and routing tables are in place to allow NetScaler to access the backend services. This type of topology emphasizes security. We will cover a sample scenario later in this chapter.

Now that we have gone through the different editions, features, and licensing, let us continue with the initial setup.



## Creating our first setup

Before setting up the VPX, we need to make sure that we have the following resources available in our virtual environment:

- 2 GB RAM
- Two vCPUs
- 20 GB disk space
- One vNIC

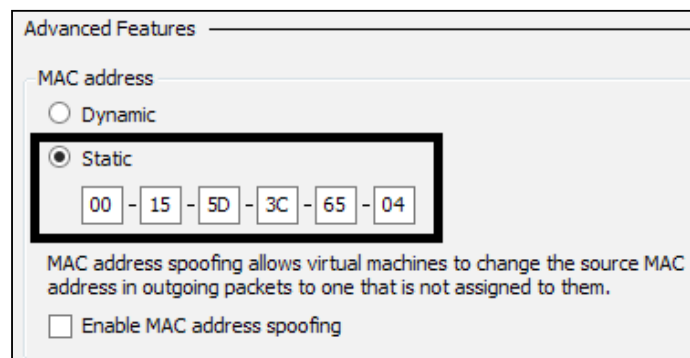


NetScaler VPX supports a maximum of eight virtual network interfaces, and as of now it supports Windows Server Hyper-V 2008 R2 and Windows Server Hyper-V 2012. It also supports XenServer 6.0, XenServer 6.1, and VMware Vsphere from version 4.0 up to 5.1.

After downloading NetScaler from [www.mycitrix.com](http://www.mycitrix.com), we can import the virtual machine using the Hyper-V manager by selecting **Import Virtual Machine...** and browsing to the download location of NetScaler VPX.

After the appliance is imported, we should change the MAC address of the network adapter to static, as the license is based on the MAC address. Hyper-V manages MAC allocation for virtual machines, and in some scenarios, a virtual machine might generate a new MAC address. Therefore, it is important to set the MAC address as static.

This can be done by navigating to **Virtual Machine | Network | Advanced Features**, as shown in the following screenshot:



Note that the same applies for VMware and XenServer as well.

After we are done changing the MAC address to static, we can boot the virtual appliance. The initial setup needs to be done using the CLI to connect the virtual machine console to the appliance console. The first thing we need to enter is the **NetScaler IP Address (NSIP)**, which is used for management purposes, then a subnet mask, and finally a default gateway. Now we can press 4 to save the settings. After this is done, we can then access the console using HTTP through the NSIP address that we entered earlier. The default username and password for the web administration GUI is `nsroot` and `nsroot`. Prior to logging in, make sure that the deployment type is set to NetScaler ADC.

Before continuing with more configuration using the web interface, we need to make sure that we have **Java Runtime Engine (JRE)** installed. This can be downloaded from <http://java.com/en/download/>. Also make sure that our client computer or management computer has firewall opened for TCP port 3010 and TCP port 3008 for a secure session because the web interface uses these ports to parse commands via the Java applet to the NetScaler appliance. Citrix has made a list of all the ports and functions used in their products, which you can view at [http://support.citrix.com/servlet/KbServlet/download/2389-102-704421/CTX101810\\_28th\\_June\\_2013.pdf](http://support.citrix.com/servlet/KbServlet/download/2389-102-704421/CTX101810_28th_June_2013.pdf).



Throughout the last few years, there have been some issues related to the NetScaler GUI and the use of Java. If you are having issues such as the Java applet not loading when you want to do some configuration inside NetScaler, then there are a couple of things that you can do. They are:

- Disable **Keep temporary files on my computer** in the Java settings under the control panel.
- Lower the security settings from **Medium** to **Low** in the same menu
- Add a site exception under the **Edit site** list

When logging in to the web console for the first time after the initial setup, we are presented with a wizard that allows us to enter information such as DNS, time zone, and SNIP, and to change password settings. We can enter that information or we can click on the **Skip** button in the upper right-hand corner of the window. This will bring us to the main dashboard. For the purpose of this book, I am going to show you how to add different configurations using regular GUI and CLI instead of using the wizard. An important point to note here is that the initial setup wizard will always pop up until we have added a platform license, subnet IP, and NetScaler IP.

You can restart the initial setup in the CLI by typing the following command:

**Configns**

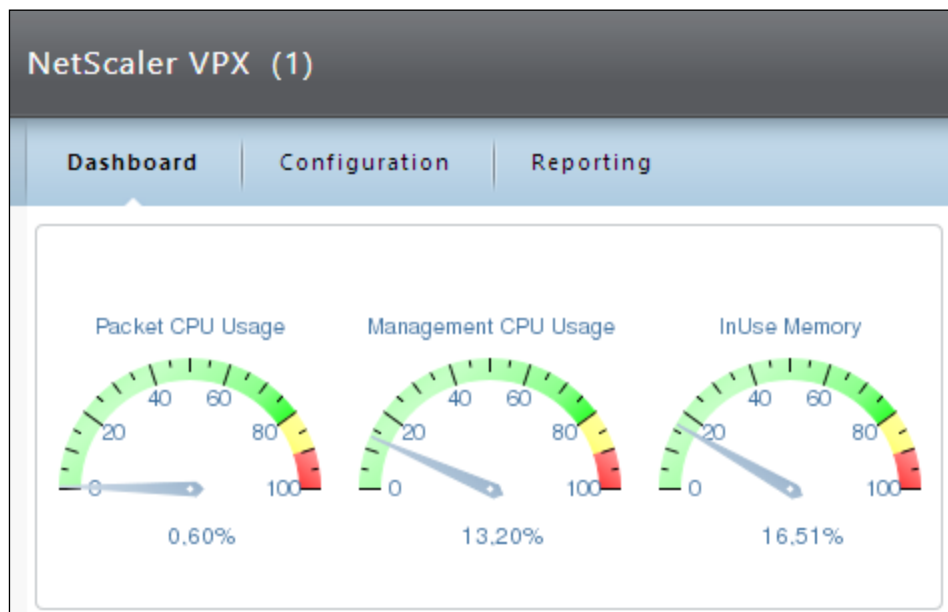


When altering the configuration of NetScaler, the configurations are put into the running configuration file. If we do not save the configuration, the settings that we changed will be lost when we restart. Make sure to save the configuration using the CLI command `save config`, or by clicking on the **Save** button (represented as a disk drive) in the GUI, after performing the changes to the configuration.

Now, inside the main administration GUI, we are presented with three main panes:

- **Dashboard**
- **Configuration**
- **Reporting**

We are directly transferred to the **Dashboard** pane, as shown in the following screenshot:



## Dashboard

The **Dashboard** pane gives us an overview of what is happening in NetScaler, how much CPU is used, how much memory is in use, what the throughput is, and so on. We can also view how many active sessions are using our services such as load-balanced web services or VPN connections.

## Reporting

We also have the **Reporting** pane, where we can run different built-in reports or create our own reports based upon different criteria. There are more than 100 built-in reports that we can use, for example, to see how many SSL connections have been used in the last day. We also have a link for documentation that redirects us to eDocs on Citrix, and a **Downloads** pane where we can download the SNMP MIB files, Nitro SDK, and some other files such as integrations for System Center.

## Configuration

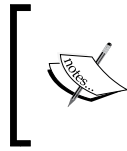
The **Configuration** pane is where we do our configuration of services and also of NetScaler; this is where we will spend most of our time.

Now, there are some basic features we should set up before deploying any services to NetScaler.

- **DNS:** This feature allows for name resolution.
- **NTP:** This feature allows for time synchronization.
- **Syslog:** This feature allows for central logging of states, auditing, and status information.
- **SNMP:** This feature allows NetScaler to send alarms to a designated SNMP server.

Syslog and SNMP features are not needed but should be evaluated in larger deployments, and for auditing and monitoring purposes. For example, NetScaler can be monitored using SNMP with System Center Operations Manager. You can read more about it at <http://msandbu.wordpress.com/2013/04/02/monitoring-netscaler-with-operations-manager-2012/>.

The first thing we are going to add is a DNS server to allow for name resolution. This setting can be found by navigating to **Configuration | Traffic Management | DNS | Name Servers**. Here, click on **Add** and enter the IP address of our DNS server, and leave the rest as default values. After we have added the DNS server, NetScaler will automatically start monitoring it. Make sure that ICMP is also opened in the firewall to the DNS servers; NetScaler uses ICMP with UDP to monitor, if the DNS servers are available. For redundancy, you should add more than one DNS server to the list. After you have added the DNS servers, you can verify the state of the servers by going back to the **Name Servers** pane.



DNS using TCP is only needed for zone transfers and therefore it is not used for regular name resolution. We also have the ability to use both UDP and TCP; this is used for TCP-enabled DNS systems.

After each configuration, I am going to show the CLI-based option to perform the same action. To add a DNS server using the CLI, we can use the following command:

```
add dns nameServer IPaddress
```

Next, you should add an NTP server; this is important because of logging purposes, timestamps, certificates, reporting, and so on. The NTP server's configuration can be found by navigating to **System | NTP Servers**. Here, click on **Add** and enter the IP information and a key if you are using authentication. If you do not have an NTP server available in your network, you can use a public one. For example, you can find a public NTP server at <http://www.pool.ntp.org/en/>.

We can also add an NTP server using the following command:

```
add ntp server IPaddress
```

After we have added the NTP server, we have to do a sync using the following CLI command:

```
enable ntp sync
```

We also need to change the time zone of NetScaler to reflect our own time zone. This can be done by navigating to **System | Settings | Change time zone**.

Another important feature that we should look closer at is Syslog. Syslog is a common open standard logging feature, which allows us to place logs on a central host instead of on NetScaler itself; this makes it easier to view logs from different devices that use Syslog from a single repository. This is not something that I consider as required, but it makes it easier to access and view logs. If we do not set up Syslog, we have to view the different logs locally on NetScaler. The Syslog feature can be enabled by navigating to **System | Auditing | Servers**. This requires that we have a central Syslog server in place.

If we have a central monitoring solution, we should consider configuring SNMP. SNMP consists of alarms and traps. If any abnormalities happen, such as high usage of RAM or, for example, Syslog, an alarm will trigger on NetScaler and the SNMP agent on it would send the alarm to an SNMP trap listener (which could be a central SNMP solution such as Microsoft System Center Operations Manager).

In order to allow NetScaler to be queried by an SNMP server for information, we need to enter the following information. This can be added in the GUI by navigating to **System | SNMP**.

- **SNMP manager:** This is the IP address of the host that is allowed access.
- **SNMP community string:** This is used for authentication of the appliance.

In order for NetScaler to send traps whenever a critical event occurs, we need to enter the following information:

- **Enable/Disable SNMP alarms:** This defines which alarms should create a trap.
- **SNMP traps:** This defines which host should get the traps and the conditions for the traps.

We can also change the hostname of the appliance; by default, it comes with the name ns. We can change it using the following CLI command:

```
set ns hostname
```

We should also change the default password as nsroot is the default password for all NetScaler appliances. This can be done using the following CLI command:

```
set system user nsroot password
```

This can also be done through the GUI by navigating to **System | User Administration | Users | nsroot | Choose Action** and clicking on **Change password**.

After we are done with this setup, we also need to add our platform license to the appliance. This can be done through the GUI by navigating to **System | Licenses**. Here, just click on **Add license** and upload the license that was generated from [www.mycitrix.com](http://www.mycitrix.com).

After we have added the license, we need to reboot the appliance. We can verify that the license is properly applied by checking under the **Licenses** tab or by using the CLI command `show license`, as this will list all the features that are licensed along with the model type, as shown in the following screenshot:

```
AAA: YES
OSPF Routing: YES
RIP Routing: YES
BGP Routing: YES
Rewrite: YES
IPv6 protocol translation: YES
Application Firewall: YES
  Responder: YES
  HTML Injection: YES
  NetScaler Push: YES
Web Interface on NS: YES
  AppFlow: YES
  CloudBridge: YES
  ISIS Routing: YES
  Clustering: NO
  CallHome: NO
  AppQoE: YES
Appflow for ICA: YES
  Upath: NO
Model Number ID: 1000
License Type: Platinum License
```

## NetScaler® modes and features

Now that we have added the license and configured most of the basic features, such as DNS, NTP, and SNMP, we need to take a closer look at the different modes through which NetScaler can process traffic. The different modes can be found by navigating to **System | Settings | Change Modes**.

Here, there are modes that we can configure depending on the following parameters:

- How do we want NetScaler to process network traffic such as L2 and L3?
- Where is NetScaler placed?

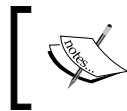
Not all the advanced features are covered here as some of them are not relevant for every environment. Information about the remaining features can be found in the Citrix article located at <http://support.citrix.com/article/CTX121149>. The different modes here decide how NetScaler should handle different kinds of traffic. So, a quick overview of the different modes is as follows:

- **Fast Ramp:** This mode bypasses the slow-start mechanism of the TCP protocol and allows for a faster increment of TCP windowing, thereby allowing for faster packet transmission. This feature is enabled by default.



- **Layer 2 mode:** This mode allows NetScaler to behave as a switch and should only be used if servers are directly attached to NetScaler, or if it is being used as a transparent bridge, for example, CloudBridge.
- **Use Source IP:** By default, when NetScaler connects to a backend server, it uses one of its own addresses to establish a connection. By enabling the Use Source IP mode, the end client IP address is used to connect to the backend server. This should only be used in deployments where you need direct connections from the clients, or when you have an IDS environment. Make sure that when this feature is enabled, the backend servers need to have one of NetScaler's IP addresses to be used as the Gateway IP address.
- **Client Keep-Alive:** This feature is mostly useful when the backend server or service does not support client keep-alive. It allows clients to maintain connectivity to the appliance even if the backend server closes the connection. This eliminates the need to reestablish the connection between the client and the backend server, and will reduce the time needed for a client to reopen the connection. This feature should only be enabled if there are performance issues with a service.
- **TCP buffering:** This feature allows the adjustment of speed between a high-speed server and a slow client. If a backend server responds too fast for a client, the appliance will buffer the packets and adjust the sending based upon the speed of the client. This allows the backend server to devote the CPU resources to other tasks. This mode should be enabled if there are performance issues or if the TCP window scaling is not working.
- **MAC based forwarding:** This mode allows NetScaler to return packets based upon the MAC address of the received packet. For example, in environments where you have multiple routers, and you need to make sure that the packets are returned through the same path, we need to enable the MAC-based forwarding mode. If this feature is disabled, the return path is based upon the route look up.
- **Edge Configuration:** We need to enable this feature if clients are using the link load balancing feature.
- **Use Subnet IP:** This feature allows for the use of subnet IP addresses.
- **Layer 3 mode:** When the Layer 3 mode is enabled, the NetScaler appliance performs route table look ups and forwards all packets that are not destined for any NetScaler-owned IP addresses. This mode is enabled by default, but should be disabled if not used for security purposes.
- **Path MTU Discovery:** This mode allows network devices to share information to determine the largest layer size that can be allowed on a network. This mode is enabled by default.

- **Static Route Advertisement:** This mode allows for the advertisement of static routes when using dynamic routing protocols.
- **Direct Route Advertisement:** This mode allows for the advertisement of direct routes when using dynamic routing protocols.
- **Intranet Route Advertisement:** This mode allows for the advertisement of intranet routes when using dynamic routing protocols.
- **IPv6 Static Route Advertisement:** This mode allows for the advertisement of IPv6 static routes when using dynamic routing protocols.
- **IPv6 Direct Route Advertisement:** This mode allows for the advertisement of IPv6 direct routes when using dynamic routing protocols.
- **Bridge BDPUs:** This mode is used for the Spanning Tree Protocol, allowing NetScaler to participate or not participate in the STP state.



When using NetScaler at the edge of the network as a firewall, we should uncheck all the boxes regarding route advertisement and Path MTU discovery.

## NetScaler® networking

We have gone through the basic setup of NetScaler, its different modes, and its basic features. Now, we will go deeper into the different IP addresses that can be used in NetScaler and how they operate. NetScaler can have the following different IP addresses:

- **NSIP:** This is the NetScaler IP address.
- **MIP:** This is the mapped IP address.
- **SNIP:** This is the subnet IP address.
- **VIP:** This is the virtual IP address.
- **GSLBIP:** This is the Global Server Load Balancing site IP address.
- **CLIP:** This is the cluster IP address.

We will not cover the GSLBIP and CLIP addresses as part of this book.

### NSIP

As we have discussed earlier, this IP address is used for management purposes in the local NetScaler and when authenticating against services such as AD, LDAP, and Radius. We need to make sure that the NSIP address is allowed to talk to these services in the firewall.

By default, the NSIP address is allowed to be used for management services using several protocols such as SSH, HTTP, and HTTPS. We can restrict the security level to only allow secure access by navigating to **System | Network | IPs | NSIP**, and then choosing **Secure Access**. Remember that this requires that we import a trusted certificate, as by default it uses a self-signed certificate. If we try to connect it with a browser when running a self-signed certificate, we will get browser warnings stating it cannot verify the publisher.

## MIP

Next we have the MIP address, which is used for backend server connectivity. When we add an MIP address to a network, it automatically creates a route entry with its address as the gateway to reach that particular network.

## SNIP

The SNIP address is also used for backend server connectivity. When setting up a NetScaler appliance, the startup wizard requires you to enter an SNIP address. The SNIP address also creates a route entry with its address as the gateway to reach that particular network. The SNIP address is also used for connectivity against DNS/WINS servers. In order to use an SNIP address, the Use Subnet IP (USNIP) feature must be enabled.

The common feature of both these addresses is that they are used for proxy connections by users connecting to a service via a VIP address to a backend server. Most of the time, MIP was used to set up an address on the same subnet in which the NSIP was placed, and the SNIP address was used to contact backend servers, which were located on another subnet. But with the latest releases of NetScaler, there is no need to use the MIP address feature. So, in common deployments, we can use the SNIP address.

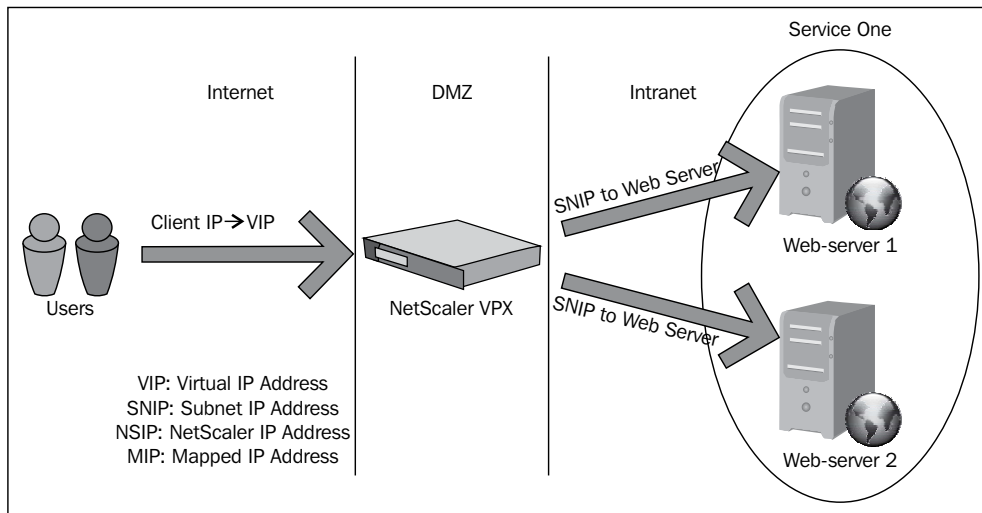
When we want to add an SNIP or an MIP address to NetScaler, we can do this from the same pane where we saw the NSIP address, that is, by navigating to **System | Network | IP addresses | Add**. If we want, we can also use the following CLI command:

```
add ns ip 10.0.0.0 255.255.255.0 -type SNIP
```

We can change the type name depending on what we need. Valid parameters here are SNIP, VIP, MIP, and NSIP.

VIP is a virtual IP address. It represents a service or different services by an IP address, port, and a protocol, and depending on the configuration, it might be a load-balanced service. This is the IP address that clients connect to for accessing a service. We will have a detailed look at how the VIP address works in *Chapter 2, NetScaler Gateway™*, and *Chapter 3, Load Balancing*.

Now, let us tie this together to understand the concept of how NetScaler processes traffic for a service. In this example, we have a web service running on a couple of web servers located on our intranet subnet *10.0.0.x*. We want this service to be accessible to our external users by using NetScaler. We will place it in the DMZ with a two-arm topology, with one NIC in the intranet, and define the different IP addresses to be used. In this example, we set up an SNIP with the address as *10.0.0.2*, which is used for server connectivity at the backend. Our users are placed on the Internet and will access the service using *www.service1.company.com*. This FQDN resolves into the VIP address on NetScaler, which is *80.80.80.80*. Remember that VIP is a virtual address, and in our example it is used to load balance the connection between the two web servers that are placed on the intranet.



So, when a client connects to the VIP of NetScaler, it terminates the connection and establishes a connection with the backend web server using its SNIP client connection to the VIP address *www.service1.company.com*, as shown in the earlier example. The following table shows how the packets are routed.

HTTP request	Source	Destination
IP	Client IP address	NetScaler VIP address
MAC	Default router	NetScaler MAC

From here, NetScaler establishes a connection to the backend server on behalf of the client requesting the content.

HTTP request	Source	Destination
IP	NetScaler SNIP address	Backend web server 1
MAC	NetScaler MAC	Backend web server 1

And the return traffic goes in the same direction back to the client.

This is a simple overview of how the traffic flow might be with a load-balanced service. There are, of course, many factors here that decide how the traffic flows, and it is also dependent on how the network is configured.

One thing that is important to note is that the IP addresses are not associated with an interface as they are with a regular network appliance. They are active on all the interfaces so NetScaler behaves more like a hub. This might be a problem in some cases, where TCP packets are sent and received on different interfaces, and it might cause a loop. This is where VLANs come in. We can associate an IP address with a VLAN, which we can again associate with an interface. First, we need to create a VLAN. This can be done through the GUI by navigating to **Network | VLANs | Add**. From here, we can enter an ID for the VLAN and give it an alias name. Then, we can bind an interface and an IP address to the VLAN. This allows an IP address to be bound to a specific virtual interface.

We can also do this via the CLI by using the following commands. First, we need to create the VLAN.

```
add vlan 20 -aliasName "Network 1"
```

Next, we need to bind it to an interface.

```
bind vlan 2 -ifnum 1/8
```



We have an option to choose the Tagged VLAN. This uses the 802.1 standard, but it is not supported by NetScaler VPX, and it is recommended to leave this to the hypervisor layer. If we need to tag a particular VLAN to NetScaler, we can do this under the network settings for NetScaler VPX in the Hyper-V manager. To define a tagged VLAN, enable the option for virtual LAN identification for a management operation system and define a VLAN ID.

## Summary

We have now gone through the basics of NetScaler, covering the basics and the definition of an ADC, how it works, and also a bit on the different models and editions we can choose from. We also went through some advanced feature modes, and how NetScaler processes traffic for a sample web service. Lastly, we looked at how NetScaler handles traffic for a load-balanced service, and how we can add VLANs.

So, to sum it up, this is what we did to get NetScaler up and running:

- Imported the virtual machine on the virtual environment
- Initial setup of NetScaler using CLI by setting the NSIP
- Changed the default password from `nsroot`
- Added a platform license to enable more features
- Added additional IP addresses such as SNIP to enable backend communications
- Added a DNS server for name look up and an NTP server for time synchronization
- Configured modes depending on the network topology
- Saved the configuration

In the next chapter, we will look more into the NetScaler Gateway feature, which is commonly used for XenApp/XenDesktop environments, and we will also have a look at the different modes it can operate in.

# 2

## NetScaler Gateway™

NetScaler Gateway is one of the most commonly used features of NetScaler. In earlier versions, Citrix had products such as Secure Gateway or Access Gateway, but with the 10.1 release of NetScaler, it was renamed NetScaler Gateway. This feature does the same task as it did before; it grants users remote access through a gateway to the corporate network. It has multiple features and ways in which it can give end users access to the corporate network, which we will cover throughout this chapter. So here is a quick overview of what we will be covering in this chapter:

- Basics of NetScaler Gateway
- Different connection methods and settings
- Sample setup scenarios
- Integrating with XenDesktop/XenApp and XenMobile
- Configuring StoreFront

### **A brief history**

The NetScaler Gateway feature has been available in many different forms. The first release of the Gateway feature came with Citrix Secure Gateway, which was first released in 2001. This was a Windows-based software, which gave users remote access to their Citrix solutions. Later on, Citrix released Access Gateway, which gave us more features such as SSL VPN, Endpoint analysis, and VPN. It is has now been named NetScaler Gateway and it is available in two different forms, either as a subfeature of NetScaler, or as its own appliance, which further comes in two editions, either a VPX or an MPX edition. This appliance only has the NetScaler Gateway feature and nothing more. More information about these two editions and what their specifications are can be found in the NetScaler Gateway datasheet located at <http://bit.ly/1kGLZS0>.



Many companies use the NetScaler Gateway feature on a regular NetScaler appliance as it gives the benefit of incorporating many features on the same appliance, instead of having scattered features on many different appliances.

## Understanding the features

NetScaler Gateway has a set of features, which can be used to grant end users remote access, such as:

- **ICA Proxy:** This feature allows the gateway to proxy ICA traffic from the backend XenApp or XenDesktop solution to the user through the TCP 443 port.
- **SSL VPN:** This is a browser-based VPN solution also known as clientless access.
- **VPN:** This is a virtual private network feature that gives users access to the corporate network using the NetScaler Gateway plugin.
- **Endpoint analysis:** This is a network access control feature, which scans clients to see whether they fulfill corporate security policy before they are allowed to connect to the network.
- **SmartAccess:** SmartAccess allows us to control access to applications and desktops on a server through the use of NetScaler session policies. You can read more about SmartAccess at <http://support.citrix.com/proddocs/topic/access-gateway-92/agee-smartaccess-how-it-works-c.html>.
- **MDX:** It is basically an application-level-based VPN solution, used for integrating NetScaler with XenMobile application management.

There are more features of NetScaler Gateway, which we will cover as we go through the chapter. One important thing to note is that all of these features require that we have a legitimate license installed on our NetScaler. For the use of the regular ICA Proxy feature, we only need a regular platform license, which we have covered in the previous chapter. If we want to use any of the other features in the preceding list, we also need a license called the universal license.

When we purchase a regular NetScaler platform license, either for an MPX or a VPX, we are given five universal licenses. We can verify this from the GUI by navigating to **System | Licenses**.

Content Filtering	✓
Integrated Caching	✓
NetScaler Gateway	✓
<b>Maximum NetScaler Gateway Users Allowed</b>	<b>5</b>
Cache Redirection	✓
Sure Connect	✓

We can also verify this through the CLI using the following command:

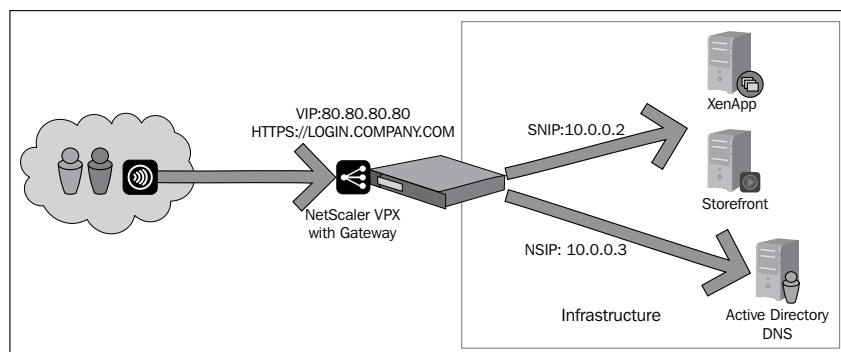
```
show license
```

These licenses are concurrent, which means that we can have five users using a VPN-based solution at any given time. If we need more concurrent users, we have to buy more universal licenses.

The most commonly used feature of NetScaler Gateway is ICA Proxy, which allows remote access for users to XenApp or XenDesktop solutions, and requires only that the users have Citrix Receiver installed. It requires no additional licenses. The solution is quite simple as it tunnels all ICA traffic through the gateway and back to the user via port 443. This port is commonly used for secure HTTP traffic, is open on most firewalls, and is allowed on remote locations such as hotels and airports.

Let us take a look at a sample scenario to see how ICA Proxy operates, and how a user can access their applications or desktops. This scenario describes an example, and might be different from deployment to deployment depending on the network layout and infrastructure.

We have a company that has the NetScaler Gateway feature set up to allow remote users to access their XenApp solution. The example gateway is available at <https://login.company.com>, and the NSIP and SNIP are set up according to the design shown in the following figure:



When a user tries to access the solution, for example, using the web portal, the following happens:

1. User 1 accesses the solution through `https://login.company.com`, which is accessible via the VIP on NetScaler.
2. User 1 authenticates on the site with his/her credentials.
3. NetScaler uses its NSIP to contact the **Active Directory (AD)** to validate the user. If we have configured a load-balanced Active Directory service, then it would use the SNIP.
4. The user is validated and the credentials are forwarded to the StoreFront Authentication service using the SNIP of NetScaler.
5. The StoreFront Authentication service forwards the credentials to the Store service, which in turn contacts the XenApp farm using its XML service to validate the user against the STA and get a list of available resources.
6. The XML service from XenApp contacts StoreFront and delivers a secure ticket and information regarding available resources.
7. The user is validated and StoreFront contacts the callback URL to NetScaler using its FQDN and generates a list of resources available for the user.

Now, the user has a portal which shows all the resources that are available. If the user was to click on an application/desktop, the following would happen:

1. When User 1 clicks on Application 1, NetScaler sends an HTTP or HTTPS request depending on the setup to the StoreFront server, which indicates what resource is being requested.
2. StoreFront connects using the XML service to the XenApp farm.
3. The STA service queries the IMA service for a resource that can launch Application 1.
4. The STA service returns a server that has available resources with its IP address to StoreFront.
5. StoreFront generates an ICA file, which contains the ticket issued by the STA and sends it to the client web browser. The ICA file that is generated contains the full FQDN name of the NetScaler Gateway VIP. The IP address of the backend server is never revealed as the resource is bound up by the STA ticket.
6. Citrix Receiver launches a session with the FQDN and the STA ticket.


We have now seen how a sample scenario might look like and how the different Citrix components communicate and generate an ICA connection with an external user. Even though we looked at how ICA Proxy operates, the procedure is not so much different for a regular VPN connection.

Now, let us look closer at the configuration of the Gateway feature within NetScaler, and how we can set it up to reflect the sample scenario.

## Deploying ICA Proxy

First of all, we need to enable the NetScaler Gateway feature. This can be done either in the GUI by navigating to **System | Settings | Configure Basic Modes** and choosing **Access Gateway** or by using the following CLI command:

```
enable ns feature SSLVPN
```

 After build 10.118 was released, Citrix introduced a quick configuration wizard that makes it easy to set up and configure the NetScaler Gateway feature. In the steps covered in this book, we do not make use of the wizard, but we are going for the same configuration. This wizard can be started from within the GUI by navigating to **NetScaler Gateway Pane | NetScaler Gateway Wizard**.

Then, we go into the NetScaler Gateway feature of the GUI. Here, we choose **Virtual Servers**, and right-click and choose **Add Server**. This will open a menu that allows us to create a virtual server, which the clients are going to access.

There are multiple configuration items that are required in order to create a fully functional vServer. Remember that first we need to configure the default settings such as NSIP, SNIP, and a functional license. Following are some of the settings we need to set:

- **Name of the server:** This is purely for description purposes.
- **Protocol:** This is always SSL.
- **IP Address:** This is going to be the VIP address.
- **Port:** The default port is port 443.
- **Basic mode / SmartAccess mode:** For configuration of ICA Proxy, we need to set this to Basic mode. If we need a SSL VPN-based vServer, we should create another vServer and set it to SmartAccess mode.
- **Certificate:** A trusted certificate is required by the clients and the StoreFront server. It should be issued by a public certificate provider, such as GlobalSign or Go Daddy. This certificate is added under the certificate pane.
- **Authentication Policy:** This defines the connection strings to the Active Directory for authentication and, for example, to the RADIUS two-factor authentication method.

- **Session Policy:** This defines where and how to contact the StoreFront server, and how credentials are forwarded to the StoreFront server.
- **STA:** This is the Secure Ticket Authority server.

We enter the information in the window, such as **Name** and **IP Address** and set the vServer to Basic mode, and leave the rest at default for now, as shown in the following screenshot:

**Create NetScaler Gateway Virtual Server**

Name\* Netscaler Gateway IP Address\* 192 . 168 . 60 . 111 ☐ IPv6

Protocol\* SSL Port\* 443

☐ Network VServer Range 1 Failed Login Timeout Max Users

☐ SmartAccess Mode ☒ Basic Mode ☐ AppFlow Logging ☒ Down state flush ☐ Double Hop Max Login Attempts

☒ Enable Virtual Server

Certificates Authentication Bookmarks Policies Intranet Applications Intranet IPs Published Applications Advanced

SSL Parameter... Ciphers...

Available:

Certificates
ns-server-certificate

Add >

< Remove

Install...

Configured:

Certificates	Type	Check	Skip CA
ns-server-certificate	Server Certificate		

To bind a Certificate to NetScaler Gateway Virtual Server, select a certificate on the left and click 'Add'. To bind a Certificate to NetScaler Gateway Virtual Server as CA, select a certificate on the left and click 'Add as CA'. To configure SSL parameters, click 'SSL Parameters'. To configure ciphers, click 'Ciphers'.

Comments



This section assumes that you are familiar with how a digital certificate works, and what the prerequisites are to create one. In order to create a certificate, we first need a private key and a **Certificate Signing Request (CSR)**, which is sent to a third-party certificate issuer such as Go Daddy. In NetScaler, go to the **SSL** pane under **Traffic Management** in the GUI. This menu also allows us to import the certificate itself, and the intermediate and root certificates. It also has an option to import PFX certificates, which often bundle the private key, the intermediate certificate, and the PFX certificate itself in one file. You can also use OpenSSL from NetScaler to convert a CRT certificate to a PFX certificate using the guide located at <http://bit.ly/11zMvEq>.

Next, we need to add a digital certificate for the vServer. The typical way is by going to the **Certificate** pane, choosing **Install**, and adding a certificate. Now, depending on how the public certificate provider issues certificates, the steps may vary. Following are the cases:

- If we have a PFX certificate, it usually contains the private key, the intermediate certificate, and the PFX certificate itself. In this case, we need to choose PFX as the certificate filename and the private key filename. If the intermediate certificate is part of the file, we should also choose the certificate bundle.
- If we have a CER certificate, it usually contains a single certificate file and a key file, which contains the private key we need to add to it. If we have an additional intermediate certificate that we need to add to the vServer, we first have to import the certificate and choose **Add as CA**. In many cases, you would need to link the main certificate to an intermediate certificate to validate the chain. You can see how to configure it at <http://support.citrix.com/article/CTX114146>.


There is an excellent blog written by Derek Seaman describing how to add a certificate to NetScaler, which can be viewed at <http://www.derekseaman.com/2013/05/import-iis-ssl-certificate-to-citrix-netscaler.html>. Also, I have written a post on how to use OpenSSL to convert a CRT file to a PFX file. This can be viewed at <http://msandbu.wordpress.com/2012/10/15/convert-from-crt-to-pfx-with-openssl/>.

After we have added a certificate, we need to add an authentication policy. There are two types of authentication policies that we need to think about, primary and secondary. The primary authentication type is typical LDAP (Active Directory), and the secondary might be a RADIUS solution. When the users try to log in, they will be presented with an interface that requires them to enter a username and password for the Active Directory, and a second password for the secondary authentication solution.

Now, this is very dependent on how the RADIUS provider delivers the authentication process. An example is SafeNet, which we just need to add as a secondary RADIUS authentication provider. There are also others that completely replace the LDAP authentication process. I have added links to different vendors and their setup for a two-factor authentication. For RSA SecurEnvoy, the link is <https://www.securenvoy.com/integrationguides/citrix%20net%20scaler%20with%20ipad.pdf>. For SafeNet, the link is [http://www2.safenet-inc.com/cryptocard/implementation-guides/Citrix/Citrix\\_Access\\_Gateway\\_Implementation\\_Guide\\_1111.pdf](http://www2.safenet-inc.com/cryptocard/implementation-guides/Citrix/Citrix_Access_Gateway_Implementation_Guide_1111.pdf).


For mobile devices, such as an iPad, we need to enter the RADIUS solution as the primary authentication. In order to sort the different authentication policies, based upon which device is connecting, we need to use the appropriate expression. We are going to cover this later in the chapter. You can also read more about this at <http://support.citrix.com/article/CTX125364>.

As this scenario focuses only on a single Active Directory authentication, we only need to create a single primary authentication policy. Click on the **Authentication** pane under **vServer Settings | Primary**. Go to **Insert Policy | Policy Name**, and choose **New**. This will open a policy window. First, we need to enter a name for this policy and then choose **LDAP** as the authentication type.

 If we want, it is also possible to allow users to change their Active Directory password. This requires that the domain controllers are responding on LDAPS, and that the authentication policy is configured with the option to allow for password change.

It is also important that we understand how policies work in NetScaler, as this not only applies to authentication, but to other features in NetScaler as well. When we create a policy, we define an expression. That expression defines what must happen to the policy it is bound to for it to run. There are multiple options that we can choose here. If we click on the **Add** button, we get numerous options that allow us to granularly define who this policy should apply to.

In the following scenario, we want a policy to apply to all users who want to access this solution. Therefore, we are going to add the `ns_true` expression. This can either be typed directly into the **Expression** field or found through the GUI under **General | True Value Expression | Add Expression**. This expression now states that this authentication policy should apply to all users that are connecting via NetScaler.

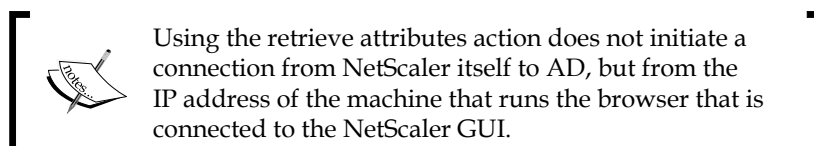
 You can read more about expressions and how to create sample expressions in the Citrix article at <http://support.citrix.com/proddocs/topic/ns-main-appexpert-10-map/ns-pi-config-classic-expr-tsk.html>.

After we have added the expression, we need to define a server that NetScaler can contact to verify the credentials. Under the **Server** pane, choose **New**. Here, we need to enter information about the Active Directory server, such as the following:

- **IP Address:** This is the IP address.
- **Base DN:** This defines the organizational unit in Active Directory where the users are located. For example, a base DN for an organizational unit called users in the domain `test.com` would look like `CN = users, DC = test, DC = com`.

- **Administrator Bind DN:** This is the username for an AD user that is allowed to query the domain. The username can be written in the domain\username form.
- **Administrator Password:** This is the password for the account and needs to be written twice.
- **Server Log on Name Attribute:** This should be set to `samAccountName`. It is also possible to use `UserPrincipalName`. This allows users to log in with their e-mail addresses.

After this is done, you can check the connection to the domain controller, and verify that the user has the correct rights by clicking on the **Retrieve Attributes** button.



Also, you might encounter a case where you would need the user to choose from different AD domains, for example, in a migration situation, or if you have multiple domains from which the users need to choose a domain to log in to. By default, there is no built-in feature that allows users to choose from different domains. We can add multiple authentication policies that point to different domains, or we can use expressions to filter domains based upon different criteria, such as the IP address. We could also add a drop-down menu on the login portal that allows the users to choose from different domains. Citrix has published an article that covers how you can create a drop-down menu to choose a domain. You can read it at <http://support.citrix.com/article/CTX118657>.

If you are having issues with authentication from NetScaler to Active Directory, there is a built-in tool that can be run from the CLI. Start by typing `shell`, and then change the directory to `tmp cd /tmp`. Then run the following command:

```
cat aaad.debug
```

This will automatically generate events in the console for every authentication attempt. The task can be killed by pressing `Ctrl + Z`.

After we have finished adding an Active Directory server, adding an expression, and naming the policy, we can click on **Create**. This will then add the Active Directory policy to the vServer. Notice that the policy has a priority of 100. If we have multiple policies here, the one with the lowest priority will win as long as the expression is the same. Therefore, if we have three primary authentication policies, all with the same configuration, but with different priorities of 80, 90, and 100, and all try to connect to the service, they would be bound to the policy that has the priority of 80.

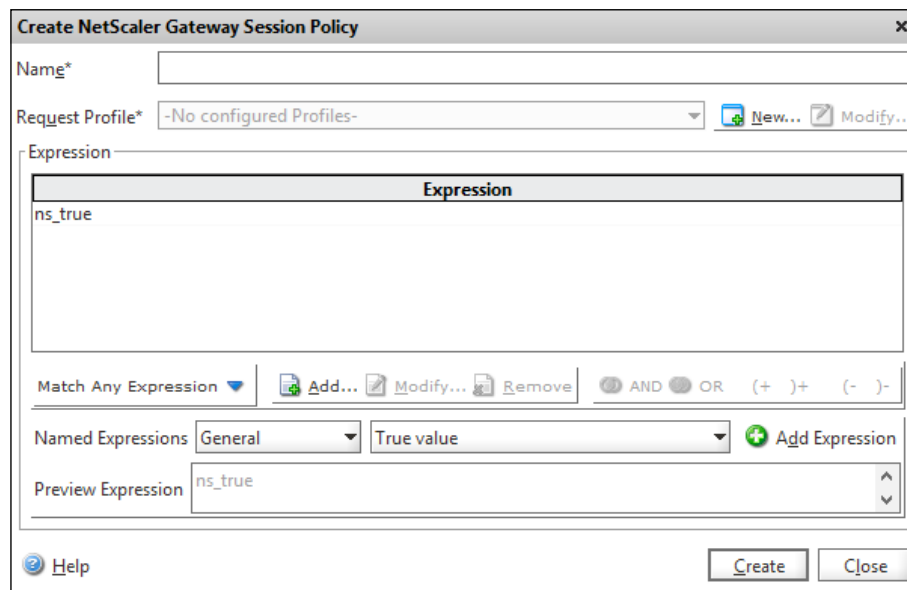


Let us look at another example of how policies work. Suppose we have three authentication policies bound to a vServer, each with different priorities, and each of them has an expression that requires the client to have a particular IP address such as the following:

- **Policy 1:** In this policy, the priority is 80. The expression is `REQ.IP.SOURCEIP == 200.0.0.0 -net mask 255.0.0.0`.
- **Policy 2:** In this policy, the priority is 90. The expression is `REQ.IP.SOURCEIP == 210.0.0.0 -net mask 255.0.0.0`.
- **Policy 3:** In this policy, the priority is 100. The expression is `REQ.IP.SOURCEIP == 220.0.0.0 -net mask 255.0.0.0`.

When a client connects, NetScaler will look at the list of policies starting with the one having the lowest priority and see if the clients match the criteria in the expression. If they do not match, it will go down the list until it gets a match. This is also the case if we have multiple RADIUS servers listed under secondary authentication.

Now, the most important part that is needed to finish the configuration is the session policy. This can be found under **Policies | Session** on the vServer. Click on **Insert Policy**, and then choose **New Policy**. We are presented with a policy window. As we want all users to be bound to this policy, we use the same general expression `ns_true` as shown in the following screenshot:



We give this session policy a name, and then we add a request profile to the policy. The request profile stores information about where the StoreFront server is located, how it should forward credentials to the StoreFront server, whether ICA Proxy should be used, and so on.

Click on **New** under **Request Profile**, and we are taken to another window. Here, there are multiple configurations that we need to take a closer look at. They are shown in the following screenshot:

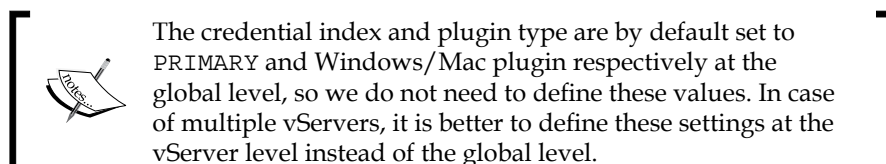


Notice that we have the option to choose **Override Global** for every item. Instead of defining settings at the vServer level, we can also define settings at a global level. These settings would then be default for all vServers created in NetScaler. If we have settings set at a global level, and we wish to override these, we need to choose **Override Global**.

The **Network Configuration** pane is used for regular VPN profiles, so we will skip that for now. Under the **Client Experience** pane, we have multiple settings that we can define. We are going to cover all of these as some are used for ICA Proxy and some are used for SSL VPN or VPN. Following are the settings:

- **Home Page:** This is mostly used for XenMobile deployments and when you want to define a homepage for an SSL VPN session.
- **URL for Web-Based Email:** This is used for logging in to web-based e-mail solutions, such as Exchange OWA.
- **Split Tunnel:** This is used to define whether all client traffic or only traffic meant for destined servers in the network should go through the gateway in a VPN connection.
- **Session Time-out (mins):** This defines how long NetScaler waits before it disconnects the session when there is no network traffic. This applies to all type of clients.
- **Client Idle Time-out (mins):** This defines how long NetScaler waits before it disconnects the session when there is no user activity. This applies only to NetScaler plugins.
- **Clientless Access:** This defines whether the SSL-based VPN should be enabled or disabled.
- **Clientless Access URL Encoding:** This defines if the URLs of internal web applications are obscured, or are in clear text and visible to the users.
- **Clientless Access Persistent Cookie:** This is needed for access to certain features in SharePoint, such as opening and editing documents.
- **Plug-in Type:** This defines what kind of plugin is offered to the user, whether it is Windows/Mac-based or Java-based.
- **Single Sign-on to Web Applications:** This allows NetScaler to do SSO either for the web interface/StoreFront or for a custom homepage that is set to be the SharePoint site.
- **Credential Index:** This defines which authentication credentials are forwarded to the web application. Here, we can choose from the primary or the secondary authentication set.
- **Single Sign-on with Windows:** This allows the Gateway plugin to authenticate to NetScaler using the Windows credentials.
- **Client Cleanup Prompt:** This is used to control the display of the client cleanup prompt after exiting an SSL VPN session. This feature is available for regular Windows/Mac-based clients, but not for mobile devices.

All we need to do in the **Network Configuration** pane is check the **Single Sign-on to Web Applications** option, and set the **Credential Index** field to `PRIMARY`. This allows NetScaler Gateway to forward LDAP credentials (which are defined as the primary authentication) to the StoreFront server. We should also define the plugin type to be Windows/Mac, as we want users to connect using the Windows plugin.



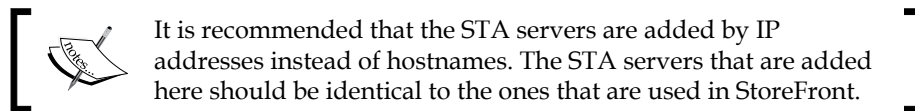
The plugin will be offered to users after they log in to the site, if they do not have it installed.

In the **Security** pane, all we need to do is make sure that the **Default Authorization Action** option is set to `Allow`. This ensures that the users are actually allowed to log in and access the resources.

Next, we have the **Published Applications** pane. This is where we enter the information needed to access our Citrix environment. Following are the settings:

- **ICA Proxy:** Here, we define if NetScaler should tunnel ICA traffic through port 443.
- **Web Interface Address:** Here, we define the URL to StoreFront Citrix web receiver.
- **Web Interface Portal Mode:** This defines if the web interface should appear with full graphical experience or use the compact view.
- **Single Sign-on Domain:** This defines which AD domain should be used for single sign-on.
- **Citrix Receiver Homepage:** This is used to define the URL for clients connecting to a Citrix receiver that does not support StoreFront.
- **Account Services Address:** This is used for e-mail based account discovery for Citrix Receiver. The URL must be in the form of `https://<StoreFront/AppController URL>/Citrix/Roaming/Accounts`. This requires that the DNS is properly configured. More information can be found at <http://blogs.citrix.com/2013/04/01/configuring-email-based-account-discovery-for-citrix-receiver/>.

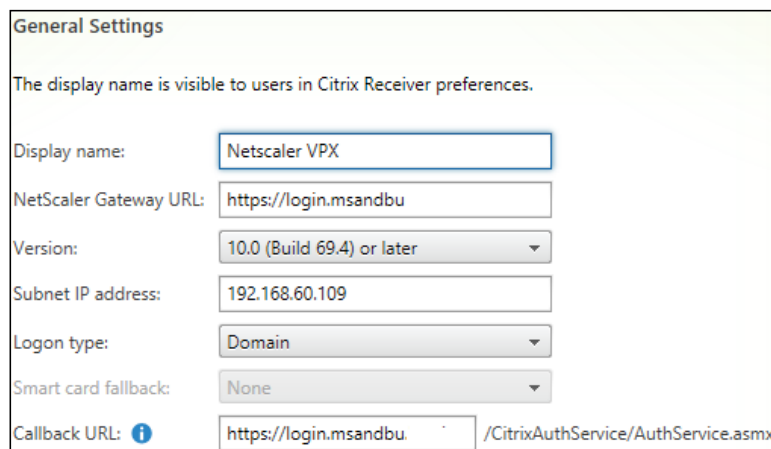
Here, we need to set ICA Proxy to **on**, and define the URL to the StoreFront address. We can also define a single sign-on domain. The domain name entered here has to match the one set in StoreFront. After this is set, we are done with the session policy and we can click on **Create**. Now, the final part that is remaining is to add the STA servers. In the **vServer Creation** window, go to the **Published Applications** pane and click on **Secure Ticket Authority** to add the servers. Also, if we need to add the StoreFront server to the session policy using HTTPS, we need to import the trusted root certificate of the StoreFront server to NetScaler or else the call back authentication will not function properly. This can be done under **Traffic Management | SSL | Import PKCS #12** with the root certificate from the StoreFront server.



After this is done, we have now fully configured the Gateway function on NetScaler. Now, we have to configure StoreFront to allow remote connections to pass through authentication.

## StoreFront integration

These steps require an existing StoreFront and XenApp/XenDesktop deployment in place. First, we need to add a gateway to StoreFront. This can be done from the GUI by navigating to **StoreFront Administration Console | Gateways**. On the right-hand side here, click on **Add Gateway Server** and then add the information as shown in the following screenshot:



The screenshot shows a 'General Settings' form with the following fields and values:

Field	Value
Display name:	Netscaler VPX
NetScaler Gateway URL:	https://login.msandbu
Version:	10.0 (Build 69.4) or later
Subnet IP address:	192.168.60.109
Logon type:	Domain
Smart card fallback:	None
Callback URL:	https://login.msandbu /CitrixAuthService/AuthService.asmx

Enter the relevant information. The **Callback URL** field needs to point to the VIP address of NetScaler Gateway. This is needed so that StoreFront can send validation back to the Gateway authentication service. The Subnet IP address is used by StoreFront to identify if a user is connecting from NetScaler. This field is optional for newer releases of NetScaler Gateway. This can also cause an issue if we are using NetScaler to load balance StoreFront, so it is advisable not to do so if not necessary.

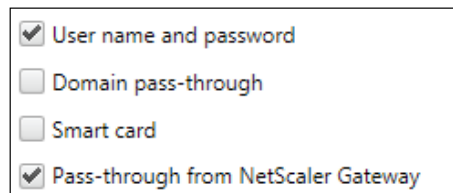
Now, there are some issues that might occur when communicating back to the callback URL via the external VIP in DMZ. Here, the problem might be twofold. One of the problems we face here is certificate mismatch. First, open a web browser on the StoreFront server and verify the certificate chain by connecting to the VIP. If it is unsuccessful, we need to add the root or intermediate certificate to the StoreFront server to verify the chain. Also, if DNS is pointing you to the external VIP, you can add an entry to the location in the local `hosts` file on the StoreFront server, where we enter the FQDN of the VIP server and the internal VIP. The second problem that might occur is a firewall issue. For example, if StoreFront is located in the intranet zone and is not allowed to communicate back to the VIP, which is located in DMZ.

If we are unable to communicate back to the internal VIP because of network restrictions, we can add a dummy VIP to NetScaler, which resides on the internal network with the SNIP. All we need to do to create a new vServer is add STAs and a certificate, and alter the StoreFront `hosts` file to point to the dummy VIP instead of the external VIP, which resides in DMZ.

Now for the final part, we need to go to the **Stores** node, and on the right-hand side of the console click on the **Enable Remote Access** option of the store we want to have remote access enabled for. Here, we have to specify how and when resources will be available. Following are the settings:

- **None:** This option means only local users on the internal network will be able to access the store.
- **No VPN Tunnel:** This option makes resources available directly to NetScaler Gateway and does not require the use of the NetScaler Gateway plugin.
- **Full VPN Tunnel:** This option makes resources only available through an SSL VPN. This requires a NetScaler Gateway plugin.

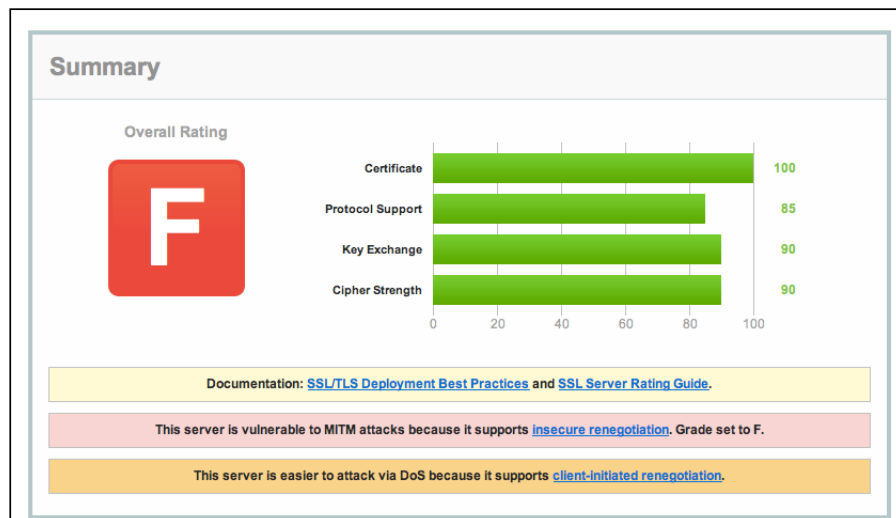
As ICA Proxy requires only Citrix Receiver, we choose **No VPN Tunnel**, and we mark the NetScaler appliance we added earlier. Lastly, we need to enable pass-through authentication from the **Authentication** pane in StoreFront. This is needed as users are entering their credentials in NetScaler Gateway, and we want them to automatically sign-on to the StoreFront site. This is shown in the following screenshot:



<input checked="" type="checkbox"/>	User name and password
<input type="checkbox"/>	Domain pass-through
<input type="checkbox"/>	Smart card
<input checked="" type="checkbox"/>	Pass-through from NetScaler Gateway

One last thing we need to configure in StoreFront are beacons. Beacons are used to identify if a user is coming externally or internally. You can read the Citrix article on how to set up beacons at <http://support.citrix.com/proddocs/topic/dws-storefront-21/dws-configure-beacon.html>. Now, we have successfully set up and configured ICA Proxy.

One thing that is important to note is that ICA Proxy allows for client-initiated renegotiation by default after establishing an SSL connection. This applies to every virtual server that uses SSL and is created in NetScaler. Because of a security vulnerability in the SSL and TLS protocols, this would allow an attacker to inject a custom packet inside a secure session when it starts. There is a free tool available on <https://www.ssllabs.com>, which we can use to find out if our service is vulnerable to such an attack. In the following screenshot, we can see that our solution is vulnerable to a client renegotiation attack.



There is a workaround that we can deploy in NetScaler to disable this attack. We can do this by using the following CLI command:

```
Set ssl parameter -denySSLrenegNONSECURE
```

## Deploying VPN

Deploying a full VPN solution might, in some cases, be needed for some users, as it allows a client to become a part of the internal network using the NetScaler Gateway plugin. With the use of the NetScaler Gateway plugin, we also have Endpoint analysis that allows us to scan a client for specific processes or, for example, to find out if antivirus software is running. Endpoint analysis is not covered as part of this book. For more information about Endpoint analysis, I suggest reading the Citrix article located at <http://support.citrix.com/article/CTX135136>.

Configuring the use of regular VPN with NetScaler Gateway is not so much different from ICA Proxy. We need the following:

- A NetScaler Gateway vServer with a name, a port, and an IP address
- A vServer set to SmartAccess
- A trusted certificate
- An authentication policy
- A session policy

The only difference here is that we need to set the vServer to SmartAccess, and we need to change the session policy.

When we are creating the session policy, there are attributes that should be configured in the **Client Experience** pane. There are multiple settings that define how a VPN tunnel should behave, so it is important to note what each feature does. Following are the settings:

- **Split Tunnel:** This is used to define if all client traffic or only traffic for destined servers in the network should go through the gateway in a CVPN connection.
- **Client Idle Time-out (mins):** This defines how long NetScaler waits before it disconnects the session when there is no user activity. This only applies to NetScaler plugins.
- **Plug-in Type:** This defines what kind of plugin is offered to the user, either Windows/Mac-based or Java-based.



- **Single Sign-on to Web Applications:** This allows NetScaler to do SSO either for the web interface/StoreFront or if we have set a custom homepage to be the SharePoint site.
- **Credential Index:** This defines which authentication credentials are forwarded to the web application. Here, we can choose from the primary or the secondary authentication set.
- **Single Sign-on with Windows:** This allows the Gateway plugin to authenticate to NetScaler using Windows credentials.

There are some exceptions here. If the **Split Tunnel** option is disabled, it means that all Endpoint traffic is routed through NetScaler. If it is enabled, it means the only traffic that is destined for the internal network is routed through NetScaler. This also means that we need to define which IP address or range of addresses the gateway should intercept. This is done using something called an intranet application, which is available in the vServer. Here, we can define a range of IP addresses or a single IP address that the gateway will intercept for the client. There are some differences here. For example, for Java-based clients, we need to define intranet applications as proxy-based, and for regular Windows/Mac clients we need to define the intranet applications as transparent. We cannot combine these two types of intranet applications.

Also, should we require giving a connected client a dedicated IP address, we need to define intranet IPs that will act like a DHCP server. Here, we can define a range of IP addresses that can be given to users. These intranet IPs can be bound to an AAA user, an AAA group, a vServer, or at a global level. IP addresses that are bound to a user take priority over the other options.

Now, under the **Published Applications** pane, we need to define the following:

- **ICA Proxy:** This should be set to `off` as the client will have a full VPN connection and does not need the gateway to proxy traffic.
- **Web Interface Address:** This should be defined to allow clients to connect to the StoreFront site.
- **Single Sign-on Domain:** This defines which AD domain can be used for single sign-on.

When users connect to this vServer now, they will be presented with a download option that allows them to download the NetScaler Gateway plugin and they will be redirected to the StoreFront site.

Now, even though this uses the NetScaler Gateway plugin to establish a connection, we still need Citrix Receiver to establish a connection to the Citrix environment. It is possible to integrate these two clients. If a user has Citrix Receiver installed previously, and then installs the NetScaler Gateway plugin, they will get a new option under **Citrix Receiver Applet | About | Advanced | Access Gateway Settings**. From here, users can start a full VPN session directly using Citrix Receiver when they are connected by clicking on **Login** from the **Access Gateway Settings** option.

So after these settings are configured, we will have successfully configured a VPN solution on our Gateway. An important point to note is that in some cases you might get issues with building a VPN tunnel. In that case, you might consider using MAC-based forwarding. This issue happens with some firewalls.

## Deploying clientless access

Clientless access is a basic browser-based VPN solution, where users are presented with a homepage when they log in and from there they can access file shares, web applications, and other settings depending on what is defined in the session policy.

In order to allow clientless access, we need to define some settings in the session policy part of the **Client Experience** pane. Following are the settings:

- **URL for Web-Based Email:** This is used for logging into web-based e-mail solutions, such as Exchange OWA. This will appear as a pane within the clientless access session policy window.
- **Session Time-out (mins):** This defines how long NetScaler waits before it disconnects the session when there is no network traffic.
- **Clientless Access:** This defines if the SSL-based VPN should be enabled or disabled.
- **Clientless Access URL Encoding:** This defines whether the URL of internal web applications are obscured or are in clear text and visible to the users.
- **Clientless Access Persistent Cookie:** This is needed for accessing certain features in SharePoint such as opening and editing documents.
- **Client Cleanup Prompt:** This is used to control the display of the client cleanup prompt after exiting a SSL VPN session.
- **Single Sign-on to Web Applications:** This allows NetScaler to do SSO either for the web interface/StoreFront or if we have set a custom homepage to be the SharePoint site.
- **Credential Index:** This defines which authentication credentials are forwarded to the web application. Here, we can choose from the primary or the secondary authentication set.

In the **Published Applications** pane, we define the following settings in the request profile:

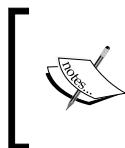
- **Web Interface Address:** Here, we define the URL to the StoreFront receiver.
- **Web Interface Portal Mode:** This defines if the web interface should appear with full graphical experience or use the compact view.
- **Single Sign-on Domain:** This defines which AD domain should be used for single sign-on.

Now, in order to activate clientless access, we only need to set the vServer to SmartAccess and set clientless access to ON under the session policy, but there are other settings as well that can affect this feature. For example, if we add a URL for web-based e-mail, users will have an e-mail pane after they log in, which is going to be proxied via NetScaler. This allows them to log in to their e-mail. URL encoding determines if the URL should be masked so that the users will never see the real URL when they are browsing on a web application. Persistent cookie is needed for some cases, such as using SharePoint. Also, adding the web interface address and defining single sign-on allows NetScaler to display the user's applications within the same clientless access session. If a user clicks on an application here, it will start a Citrix Receiver session. We also have the option to add file shares and web applications to the portal. This can be done either within the vServer under the **Bookmarks** pane or as a user-based policy. Note that when you add a bookmark and bind it to a user or a vServer, it will appear under the **Enterprise** pane of the portal. Users also have the option to add their own bookmarks. We will go through how to add settings to a specific user or group later in this chapter.

When we add a bookmark, we also have the option to use NetScaler Gateway as a reverse proxy. If this is enabled, it means that when a user clicks on the bookmark, the connection will go from the user to NetScaler, and from there to the application. If this is not enabled, the connection will go from the user to the specified address in the bookmark.

## Binding the features together

We have three different features that we have gone through, and each of them gives us some functionality. For example, if we want all of the different features to be available to the user at the same time, and give the users the option to choose between different features.




An important point to note is that using all these features on the same vServer requires universal licenses for all the users connecting to it. So, in some cases, it might be more beneficial to create three vServers, where you have one for each service.

If we want to have a single web portal where we want to give the users the ability to choose the kind of resource they need, we need to make a change to the default session policy that they use. Under the session policy, go to the request profile that is bound to it and then click on the **Client Experience** pane. Here, click on the **Advanced** button. In the menu, we have an option called **Client Choices**. By enabling this, the users will get an option to choose what type of feature they need when logging in to the web portal.

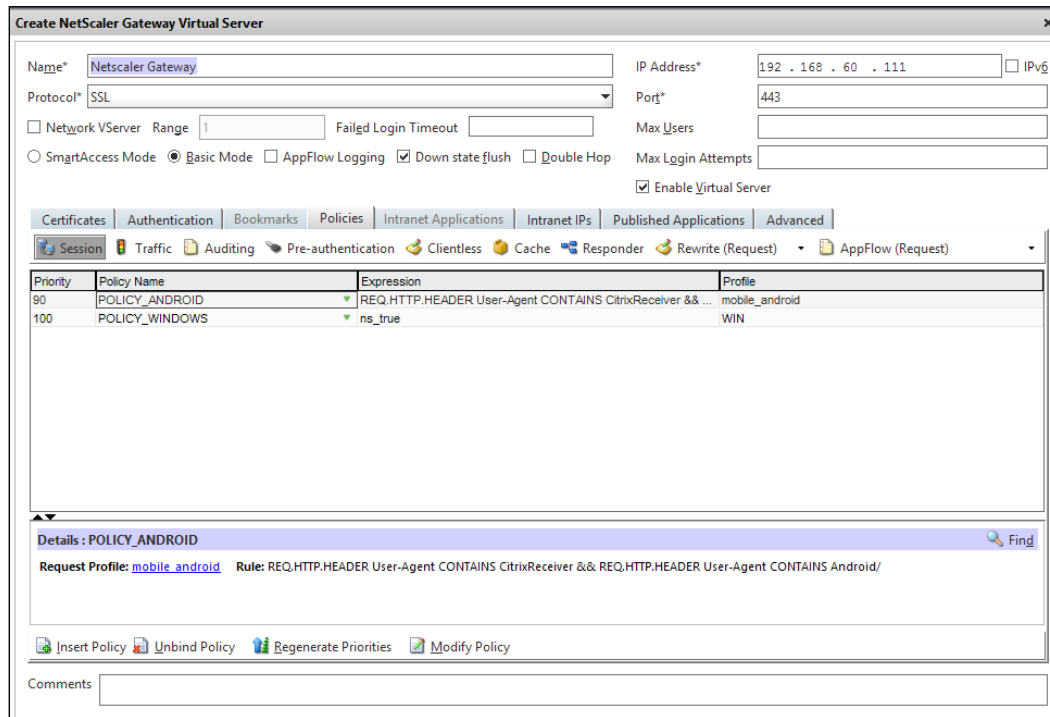


The options that are presented here are dependent on what is configured in the session policy. For example, if clientless access is not defined, it will not show up as an option here. If we have not entered a web interface address, Citrix XenApp will not show up as an option. Lastly, if we have set the vServer to basic mode, it will automatically go to the StoreFront server. Another option we have is to use expressions. We can use expressions to filter sessions based on user agents, IP addresses, and so on. For example, suppose we want to create a dedicated session policy to be applied only to Android devices, and the default session policy to be applied to all other devices that are connecting. For Android devices, the following CLI expression declares that the connecting client must have a user agent string, which contains Citrix Receiver and Android:

```
REQ.HTTP.HEADER User-Agent CONTAINS CitrixReceiver&&REQ.HTTP.HEADER
User-Agent CONTAINS Android/
```

[  A list of different expressions that can be used for different client types can be found at <http://support.citrix.com/proddocs/topic/access-gateway-10/agee-clg-session-policies-overview-con.html>. ]

Then, we create a custom session policy that is bound to that expression containing the specific configuration for our Android devices. We then use the general `ns_true` expression to apply to the rest and bind a session policy for the rest of the devices. Also, remember that the Android policy needs to have a lower priority than the other one, as `ns_true` applies to all clients that are connecting to the vServer, as shown in the following screenshot:

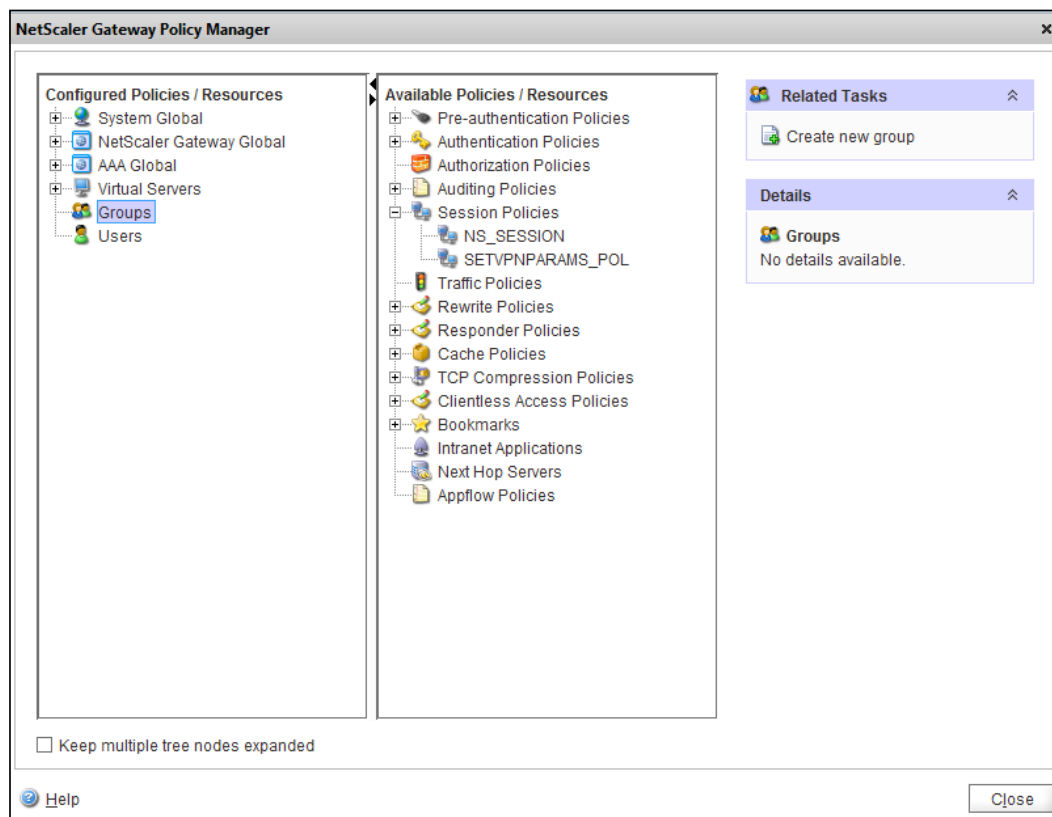


One last feature that we can use is the ability to filter based upon the Active Directory group. For example, suppose we want users who are part of the executives group to gain access to everything in the corporate network, and the regular users to gain access to some of the network. The way this operates is that when a user connects to the web portal, we can use NetScaler to get the list of the AD groups that the user is a member of from Active Directory, and find the first policy that is bound to one of the AD groups. It is important to note that user policies are processed before vServer and global policies. Therefore, if we have two session policies, one bound for the vServer and another for a user group, the user group policy will win.

In order to use this feature, we must first enable the authentication policy to get the list of the AD groups that the user who is connecting is a part of. This can be done by making sure that the **memberOf** attribute is entered in the authentication policy in the **Group Attribute** field. This is shown in the following screenshot:

Other Settings	
Server Logon Name Attribute	samAccountName
Search Filter	
Group Attribute	memberOf
Sub Attribute Name	CN
SSO Name Attribute	sAMAccountName

After that is done, we need to go into the policy manager to create the AD groups. This can be found in the **NetScaler Gateway** pane. Here, we must first create an AAA group under **Groups**. The group name must be identical to the group name in the Active Directory. Now, we can start binding policies to the group by dragging them from the **Available Policies/Resources** pane to the **Configured Policies/Resources** pane. We can also create new custom policies as shown in the following screenshot and bind them accordingly:



## Tuning

Now that we have gone through some of the different deployment types available in NetScaler Gateway, we should also take a closer look at fine-tuning the deployment setup with some simple changes.

## Redirection

First, let us take a closer look at redirection, as by default NetScaler Gateway answers only on port 443, which uses HTTPS. Many users, most likely, forget to type `https` when accessing the portal, and therefore are not able to locate it. The NetScaler Gateway wizard has the option to set up redirection from `http` to `https` automatically.

By default, a vServer consists of an IP address and a port. The Gateway vServer responds to port 443. So in order to perform a redirect, we need to set up a vServer using port 80, as it has the option to redirect.

This can be done with the help of the following steps:

1. Go to **Traffic Management | Load Balancing | Virtual Servers**, and click on **Add**.
2. Here, we enter the same IP address as the regular Gateway vServer, and choose the regular HTTP in the **Port** field.
3. Next, go to the **Advanced** pane. There, we have an option called **Redirect URL**, where we enter the FQDN of the vServer with HTTPS in the front.

**Create Virtual Server (Load Balancing)**

Name\*  ☒ IP Address Based ☐ IP Pattern Based

Protocol\*  IP Address\*  ☐ IPv6

☐ Network VServer Range  Port\*

☒ Directly Addressable ☒ State ☒ AppFlow Logging Traffic Domain ID

☐ Enable DNS64 ☐ Bypass AAAA Requests

Services | Service Groups | Policies | Method and Persistence | **Advanced** | Profiles | SSL Settings

Redirect URL  Client Time-out(secs)

Backup Virtual Server  ICMP VServer Response

Minimum Autoscale Members  Maximum Autoscale Members

VServer IP Port Insert...

Redirection Mode ☒ IP Based ☐ MAC Based ☐ IP Tunnel Based ☐ TOS Based TOS Id

Spillover

Method  Threshold

☐ Persistence Persistence Time-out (min)  Backup Action

Comments

Help

When this is done, we can click on **OK**. You will notice that the service is listed as **DOWN**. This is because no services are attached to the vServer; therefore, any connections made to the IP address will be forwarded to the **Redirect URL**. An important point to note is that if you have a regular NetScaler Gateway VPX, you cannot run this setup manually. You would need to run the remote access wizard.





Make sure that the redirect URL has the full prefix as `https://url.com/`, with the forward slash at the end, or else some security scanners might see it as an XSS vulnerability.

## Profiles

NetScaler can optimize connections using different TCP parameters, HTTP parameters, and net profiles. Many of the optimization parameters are set in different types of TCP profiles, and these can be viewed under **System | Profiles | TCP Profiles**.

Different profiles are useful in different scenarios and have different advantages depending on the network they are used in. By default, all vServers that are created in NetScaler use `nstcp_default_profile`. This profile makes sure that it works properly in all networks. Let us take a look at some of the different profiles.

- `nstcp_default_tcp_lfp`: This profile is best suited for high-bandwidth WAN and low packet loss environments.
- `nstcp_default_tcp_lnp`: This profile is best suited for low-bandwidth WAN and high packet loss environments.
- `nstcp_default_tcp_lan`: This profile is best suited for internal networks that are connected using LAN.
- `nstcp_internal_apps`: This profile should only be used for internal services on NetScaler, and should not be used on any other network services.
- `nstcp_default_xa_xd_profile`: This profile is best suited for ICA Proxy solutions.

Now, for a NetScaler Gateway vServer using ICA Proxy, it is highly recommended to use `nstcp_default_xa_xd_profile`. You can add this setting in NetScaler Gateway through the vServer, by going into the **Advanced** pane and clicking on **TCP Profiles**. When using a TCP profile, which is not the general one, you should test it properly before implementing it.

Another option where we can use profiles is if we are in a situation where we have different internal zones and different VIPs for our end users, and we need to make sure that a particular VIP uses a particular SNIP address. There is no direct relation between a VIP and an SNIP. If a NetScaler Gateway server was to connect to a StoreFront server, it would use the SNIP closest to it. If we have multiple internal zones, this would not work properly. This is where net profiles come in.

The network profile feature allows us to define the use of a specific source IP address or multiple addresses. In order to use network profiles, we must first create a network profile that contains the source IP address we wish NetScaler to use when initiating a connection to the backend servers. This can be done under **System | Network | Net Profiles**; then click on **Add**. Now we can enter a name for the profile, choose an IP address or multiple addresses (IP set), and then click on **Create**.

After we have created a network profile, we have to attach it to the different features that use an SNIP for backend connectivity. They are as follows:

- Virtual Server
- Service
- Service Groups
- Monitor

So, if we need to bind a specific network profile to a NetScaler Gateway vServer, we need to go to **vServer | Advanced | Net Profile**, and choose the network profile we created earlier.

## Testing

Now, we are done with the setup and configuration of NetScaler Gateway. Before putting it into production, make sure that you have tested it properly. A quick checklist that we should go through is as follows:

- Different clients
- Different features such as clientless access, VPN, and ICA Proxy
- Trusted certificates on different devices
- Authentication

## Summary

We have now gone through many of the different deployment types of the NetScaler Gateway function. We went through ICA Proxy, VPN using the NetScaler Gateway plugin, SSL-based VPN, and the different scenarios based on them.

There are many deployments that require the use of NetScaler Gateway, not just in XenDesktop but also in the newly released XenMobile. We cannot cover all of the different deployment types, so I recommend heading over to eDocs on Citrix for more information about the different deployment types. This eDoc is located at <http://support.citrix.com/proddocs/topic/netscaler-gateway-101/ng-deployment-wrapper-con.html>.

In the next chapter, we will explore load balancing, and how we can use load balancing for different technologies such as the roles feature in Citrix, and other products such as Exchange, SharePoint, and other generic web services.

# 3

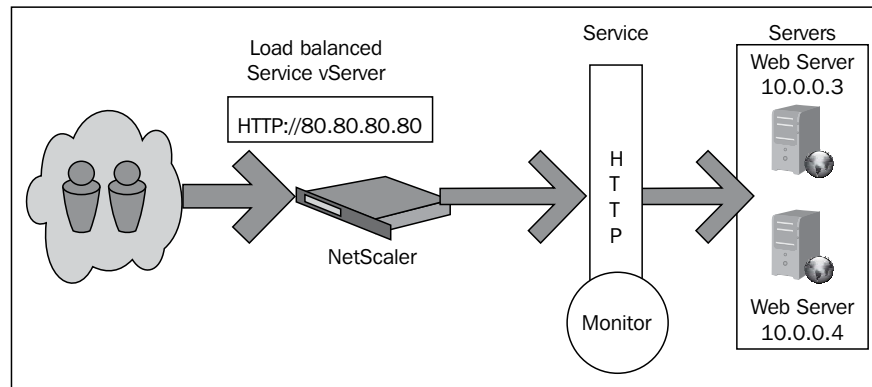
## Load Balancing

A load-balanced service is accessed by users every day, when they book a plane ticket on a website, watch the news, or access social media. With the use of load balancing, we have the ability to distribute user requests or client requests for content and applications across multiple backend servers where the content is located. In this chapter, we will cover the following topics:

- How load balancing works
- How to load balance a generic web application
- How to load balance Citrix services such as the XML service and DDC servers
- How to load balance Microsoft products such as SharePoint, Exchange, and MSSQL

A load-balanced service within NetScaler allows us to distribute user requests from different sources based upon different parameters and algorithms, such as least bandwidth or least connections. It also provides persistency, which allows us to maintain a session to the same server. These features allow us to redirect clients to a backend server, for example, a server with the least connections used.

A regular generic load-balanced service might look a bit like the one shown in the following figure. We have two backend web servers, which answer on port 80, and they are publicly accessible via a VIP address, which is the load-balanced service.



So, in essence, a load-balanced service in NetScaler consists of the following:

- **Servers:** These are the backend servers that host a service.
  - The IP address and server name are 10.0.0.3 and server1, respectively
- **Service:** Here, we define what service is hosted on the backend servers. We also define a monitor, which is used to check if the service is responding on the backend server.
  - The service name is IIS1
  - The IP address and server name are 10.0.0.3 and server1, respectively
  - The protocol and port are HTTP and 80, respectively
  - By defining a monitor, HTTP allows the service object to check if the server is responding to HTTP traffic
- **Virtual server:** Here, we define the IP, port, and protocol on which the load-balanced service will answer, what kind of service is attached in the backend, and how it will load balance between the different services at the back.
- **vServer name:** The vServer name is IIS. This field is only for description.
- **VIP address:** This is the external address of the load-balanced service. Here, it is 80.80.80.80.

- **Protocol and Port:** This is the protocol and port on which this service should respond. Here, it is HTTP and 80, respectively.
- **Services or Service Groups:** This defines what backend services are going to be included in the load-balanced object. Here, they are IIS1 and IIS2.
- **Load-balancing method:** This defines what kind of load balancing method is chosen. Here, it is the least connection method.

If we have multiple backend servers hosting the same service, it is much more convenient to use service groups. This allows us to easily bind a service against multiple servers simultaneously.



When starting the deployment of load-balanced services, we need to have the basic configuration in place, such as placement of SNIP to allow communication with backend servers. A quick rule of thumb is:

- Initial setup with NSIP
- Platform license in place
- SNIP defined for backend connectivity
- Backend servers added to server list
- Services bound to the backend server

First, we need to enable the load balancing feature. This can be done either by right-clicking on the load balancing menu under **Traffic Management** in the GUI and clicking on **Enable**, or by using the following CLI command:

```
Enable ns feature lb
```

## Load balancing a generic web application

In order to deploy a load-balanced web application, we first need to have servers in place that respond to some sort of a network service. In this example, we have two **Internet Information Servers (IIS)** running on Windows Server. These are accessible via the IPs 10.0.0.2 and 10.0.0.3 internally, and they respond to traffic HTTP on port 80.

First, we need to add the IP addresses to the server list. This can be done by going to **Traffic Management | Load Balancing | Servers**, and clicking on **Add**. Here, we just enter the IP address of the backend servers and click on **Create**. We have to do this for every backend server. After that is done, we have to add a service to the servers. This can be done by going to **Traffic Management | Load Balancing | Services**, and clicking on **Add**. Here, we have many different options. First, we need to choose the server we entered earlier, choose a type of protocol, enter a port number, and give the service a name.

**Create Service**

Service Name\* WebServer on IIS Server\* IIS (192.168.60.111)

Protocol\* HTTP Port\* 80

Traffic Domain

☒ Enable Service Number of Active Clients

☒ Enable Health Monitoring ☒ AppFlow Logging

**Monitors** Policies Profiles Advanced SSL Settings

Available


Monitors
arp
nd6
ping
tcp
tcp-ecv
http-ecv
udp-ecv
dns
ftp
tcps

Configured

Monitors	Weight	State	Passive
http	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Comments

Now, we add a monitor to the service, and click on **Create**. It will automatically start using the monitor to check the state of the backend server. If we open the service again, the monitor will show statistics about the response time and the status.

 Monitors are what makes load balancing unique. Unlike regular round-robin solutions, they will probe the backend servers, and provide NetScaler with backend server health status.

We have other types of monitors that we can use as well. All of the default monitors are listed under **Traffic Management | Monitors**. There are many types of built-in monitors that we can use. They are explained as follows:

- **TCP**: This monitor checks the availability using the TCP three-way handshake.
- **HTTP**: This monitor uses HTTP's `GET` request.
- **PING**: This monitor uses ICMP.
- **TCPS**: This monitor checks the availability using the TCP three-way handshake and a successful SSL handshake.
- **HTTPS**: This monitor uses HTTP's `GET` request and a successful SSL handshake.

All of the monitors have parameters that define how often they should probe a service before they set it as offline. Some monitors also have extended parameters. This can be viewed by opening a monitor and going into the **Special Parameters** pane.

These monitors listed here are just some examples. We also have monitors that have the suffix of `ecv`. These are used when we need to send a specific payload with a monitor. For example, if we want to check for custom headers on a web server, we can use the **http-ecv** monitor. The same can be done with other monitors as well.



There are also some monitors that are not built-in by default. We can add custom monitors for the Citrix web interface, XML service, DDC, and so on. These can be added by going to **Monitors | Add**. On the right-hand side under **Types**, there are different Citrix services that we can add a custom monitor to. For example, if we choose **CITRIX-XML-SERVICE**, we need to specify an application name in the **Special Parameters** pane. If we click on **Create**, we can use this monitor when setting up a load-balanced XML service.

Now, we need to create a service in NetScaler for each of the backend servers that are hosting the service. Also, a service is bound to a port. That means we cannot create another service on a server that is bound to another service.

If we want to limit the amount of bandwidth or number of clients that can access the backend service, we can add thresholds to the service. This can be done by going to **Service | Advanced | Thresholds**. This is useful if you have some backend servers that have limited bandwidth, or when you wish to guard yourself against a DDoS attack.



After we have created a service for each of our servers, we can go on to create the load-balanced virtual server. Go back to **Traffic Management | Load Balancing | Virtual Servers**, and click on **Add**. There are multiple settings that we need to set here. First, we need to enter a name, IP address, port, and protocol. Now, what kind of protocol we choose here is essential. For example, if we choose SSL, and the backend servers are responding on regular HTTP traffic, NetScaler will automatically do SSL offloading. This means that NetScaler will terminate the SSL connection at the VIP, and then fulfill regular HTTP requests to the backend servers. The advantage of this is that the backend web servers do not need to use CPU cycles for handling SSL traffic.

When we enable SSL as a protocol on the vServer, the **SSL Settings** pane is enabled and here we need to add an SSL certificate for our service. It is important that DNS is configured properly. If the DNS name and the subject name in the certificate do not match, we will get a warning, as NetScaler will not be able to validate the certificate. Also, it is important that we have the full SSL chain in place. If not, NetScaler cannot validate the certificate.

If company requirements are that all traffic needs to be encrypted from client to server, we can use SSL bridging. This enables NetScaler to bridge traffic from the clients to the backend servers. When we enable SSL bridging, NetScaler disables some features as it cannot see into the packets because the traffic is encrypted. For example, features such as content switching, SureConnect, or cache redirection will not work. Also, with SSL bridging, we do not need to add a certificate, as it is already available in the backend servers. So for this example, we will use SSL and add a certificate in the **SSL Settings** pane. After we have done this, we have to bind the backend services or service groups to the vServer. If we do not add a service to the vServer, it will be listed as **DOWN** until one has been added and assigned. After we have added the required information, the vServer should look something like the following screenshot:

Active	Service Name	IP Address	Port	Protocol	State	Weight	Dynamic Weight
<input checked="" type="checkbox"/>	server1-http	10.0.0.2	80	HTTP	DOWN	1	

Now, we should define the load balancing methods and persistency. There are multiple ways to load balance between the different services. They are explained as follows:

- **Least connection:** In this method, the backend service with the fewest active connections is used. This is the default method.
- **Round robin:** In this method, the first session is handed to the service that is on top of the list, and the next connection goes to the second service on the list. This continues down the list and then starts over again.
- **Least response time:** In this method, the service that has the fastest response time is used.
- **URL hash:** In this method, when a connection is made to a URL for the first time, NetScaler creates a hash to that URL and caches it. So frequent connections to the same URL will go to the same service.
- **Domain hash:** In this method, when a connection is made to a domain name for the first time, NetScaler creates a hash for that name and caches it. So, frequent connections to the same domain will go to the same service. The domain name is fetched either from the URL or from the HTTP headers.
- **Destination IP hash:** In this method, when a connection is made to a specific IP address for the first time, NetScaler creates a hash for that destination IP, and redirects all connections to that IP address through the same service.
- **Source IP hash:** In this method, when a connection is made from an IP address for the first time, for example 10.0.0.1, NetScaler creates a hash out of the source IP. Frequent connections made from the IP and/or subnet will go to the same service.
- **Source destination IP hash:** In this method, NetScaler creates a hash based upon the source and destination IP. This ensures that a client will be connected to the same server.
- **Call ID hash:** In this method, NetScaler creates a hash based upon the Call ID in the SIP header. This makes sure that an SIP session is directed to the same server.
- **Source IP source port hash:** In this method, NetScaler creates a hash based upon the source and source port. This ensures that a particular connection will go to the same server.
- **Least bandwidth:** This method is based upon the service with the least amount of bandwidth usage.
- **Least packets:** This method is based upon the service with the fewest packets.

- **Custom load:** This method allows us to set custom weights.
- **Token:** In this method, NetScaler selects a service based upon a value from the client request using expressions.
- **LRTM:** In this method, NetScaler selects a service based upon the one with the least response time.

Some of the load balancing methods are explicitly used for some special services and protocols to make sure that when we set up load balancing, and want to use a custom load balancing method, the method is supported for the service. For example, Lync 2013 uses a special NetScaler monitor, which is listed in the setup guide.

Here, we will use the round-robin method. After we have chosen a way to load balance, we can choose how the connection will persist to the service. Again, there are different methods for a connection to persist. They are listed as follows:

- **Source IP:** In this method, connections from the same source IP are persisted to the same server.
- **Cookie insert:** In this method, each client is given a cookie, which contains the IP address and the port of the service that the client is accessing. The client uses the cookie to maintain a persistent connection to the same service.
- **SSL session:** This method bases persistency upon the SSL session ID of the client.
- **Rule:** This method is based upon custom made rules.
- **URL passive:** This method bases persistency upon URL queries.
- **Custom server ID:** In this method, servers can be given a custom server ID, which can be used in URL queries.
- **Destination IP:** In this method, connections to the same destination IP are persisted to the same server.
- **Source and destination IPs:** In this method, connections from the same Source IP to the same destination IP are persistent.
- **RTSP session ID:** This method bases persistency upon the RTSP session ID.
- **SIP call ID:** In this method, persistency is based upon the same SIP call ID.

Some of the persistency types are specific to a particular type of vServer, and all persistency types have a timer attached to it, which defines how long a connection should persist to a service. You can view more about the different persistency types, and what kind of protocol they can be used for at <http://support.citrix.com/proddocs/topic/netScaler-load-balancing-93/ns-lb-persistence-about-con.html>.



We also have the option to set a backup persistence. This is used when a connection does not support the primary type.

Now, let us explore a bit about the more advanced configurations that we can configure on a vServer.

## Assigning weights to a service

Assigning weights to a service allows us to distribute load evenly, based upon parameters such as hardware. If we have many backend web servers that have 4 GB RAM, and we have newly set up vServers, that have 8 GB RAM, then the new ones should have a higher weight. This can be done when we attach a service to a load-balanced vServer. The higher weight we set on a service, the more user-defined traffic/connections it can handle. This is shown in the following screenshot:

The screenshot shows the 'Configure Virtual Server (Load Balancing)' window. The 'Services' tab is selected, displaying a table of services. The 'Weight' column for the 'MSSL2' service is highlighted with a red box, showing a value of 2.

Active	Service Name	IP Address	Port	Protocol	State	Weight	Dynamic Weight
<input checked="" type="checkbox"/>	MSSQL	192.168.88.10	1433	MSSQL	UP	4	0
<input checked="" type="checkbox"/>	MSSL2	192.168.88.11	1433	MSSQL	UP	2	0

However, it is important to remember that not all load balancing methods support weighing. For example, all the hashing load balancing methods and the token load balancing method do not support weighing.

## Redirect URL

Redirect URL is a function that allows us to send a client to a custom web page if the vServer is down. This only works if the vServer is set up using the HTTP or HTTPS protocols. This can be useful for instances where we have a planned maintenance or some unplanned failures, and we want to redirect users to a specific web page, where we have posted information about what is happening. This feature can be configured under **vServer | Advanced | Redirect URL**.

## Backup vServer and failover

Backup vServer allows us to failover to another vServer, in case the main vServer should go down. This can be configured under **vServer | Advanced | Backup vServer**.



An important point to note is that a NetScaler Gateway vServer can also be configured to be used as a backup vServer.

In addition to handling failover, we can also use the Backup vServer to handle excessive traffic, in case the primary vServer is flooded. This is known as a spillover. We can define spillover based upon different criteria, such as bandwidth connections. We can then define what the vServer should do if there are too many connections to the vServer, for example, if it should drop new connections, accept them, or redirect them to the backup vServer. These settings can also be configured in the same pane as the failover settings. Here, we need to configure the method and what kind of action we want it to take.




If we have configured a backup vServer and a redirect URL for the same vServer, then the backup vServer takes precedence over the redirect URL.

We have now gone through the basics of setting up a load-balanced service, and some of the advanced configuration that we can set. Now, let us continue with this and use the basics to set up load balanced services for Citrix XenApp and XenDesktop.

Now, there are only certain particular services that we can set up as load-balanced in a Citrix environment. They are listed as follows:

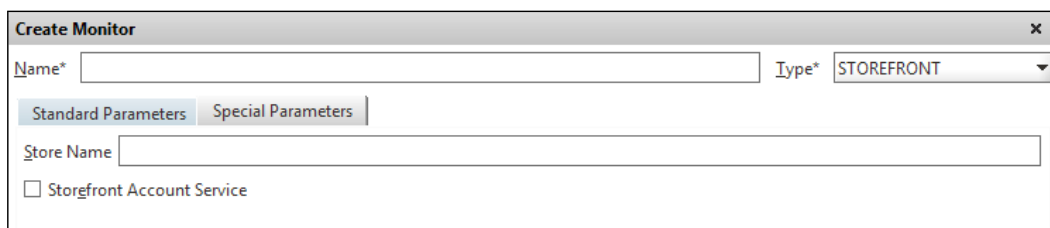
- Storefront/Web Interface
- Desktop delivery controllers
- XML service
- TFTP for provisioning servers

 NetScaler includes a set of finished wizards, which allow us to easily create a load-balanced service for Citrix services such as WI, DDC, and XML. The examples in this book are not going to use the wizards, so as to give you a clear understanding of what lies underneath.

## Load balancing StoreFront

StoreFront is the replacement for Web Interface, and is included by default in XenDesktop 7. This deployment should be in place before starting to set up a load-balanced service for it. We should configure StoreFront as part of a server group. More information about this can be found in the eDoc located at <http://support.citrix.com/proddocs/topic/dws-storefront-21/dws-deploy-join.html>.

Before we start configuring a load-balanced service for StoreFront, we need a monitor in place to verify that the StoreFront store is functioning and not just checking if it is responding on HTTP or HTTPS. NetScaler has a built-in monitor that we can use for this purpose, but we have to specify some additional parameters and create it first. Go to **Load Balancing | Monitors**, and click on **Add**. In the right-hand part of the window, choose **STOREFRONT** from the list, and then go to the **Special Parameters** pane. Here, we need to enter the name in the **Store Name** field, as shown in the following screenshot. This can be found in the StoreFront administration console.



**Create Monitor**

Name\*  Type\* **STOREFRONT**

**Standard Parameters** **Special Parameters**

Store Name

☐ Storefront Account Service

After this is done, give the monitor a name, and click on **Create**. Now, we can continue on with the setup as follows:

1. First, we need to add the backend servers that are running StoreFront to the server list.
2. Next, we need to bind a service to the servers. The difference from when we configured the generic web application is that we need to choose the custom-made monitor we created. This will make sure that the StoreFront service is functioning before a client connects to it. Another option we could configure here is to allow NetScaler to insert the client IP into the HTTP header. Because of the way NetScaler operates, the StoreFront server will never actually see the client IP, which is sometimes needed for troubleshooting and logging purposes. We can configure this while setting up the services by going to **Advanced | Settings | Enable Client IP**. Under the header box, we enter, for example `X-Forwarded-For`, to distinguish the name in the web server logs. When we have created the service for each of the StoreFront servers, it is time to create the vServer.
3. Go to **Virtual Servers** and click on **Add**. Enter an IP address, a port, and a protocol. Lastly, bind the backend services to the vServer. Now, depending on how we want the users to access the StoreFront resource, we need to consider what kind of protocol we set here. For example, if all users are accessing Citrix using the Gateway function, we should choose the **HTTP** protocol, and change the URL in the session policy to point to the new VIP created by the load-balanced server. If Citrix access is also available for internal users directly without using Gateway, we should choose **SSL** and add a certificate, and use NetScaler to handle the SSL traffic. If we want the traffic from the client to StoreFront to be encrypted, we should choose **SSL\_bridge** under protocol type.
4. Most regular deployments use SSL here. After we have defined this, we should add a persistency to the vServer, which allows us to stay connected to the same StoreFront server. The recommended setting here is to use **COOKIEINSERT**, and the timeout value should be set to **0**, which means no expiry. This will allow users to reconnect to the same StoreFront server as long as it is responding to requests. Also, if there are browsers that do not allow or are configured to not allow cookies, we should set up a backup persistency such as **SOURCEIP**.

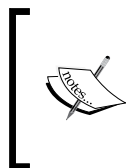
## Load balancing Web Interface

Setting up a load-balanced Web Interface is not very different from setting up StoreFront. The main difference is the monitor that is used for the services. NetScaler also has a predefined Web Interface monitor, which is called **CITRIX-WEB-INTERFACE**, but we have to create it before we can use it. We also have to enter a site path in the **Special Parameters** pane of the monitor window. This monitor checks for resource availability by authenticating an Active Directory user. You can read more about setting up Web Interface monitoring at <http://support.citrix.com/proddocs/topic/netScaler-load-balancing-93/ns-lb-monitors-builtin-wi-tsk.html>. Other than that, the configuration is the same as it would be with StoreFront. Enter the servers, bind the services, and create the load-balanced server.

## Load balancing XML Broker

The XML Broker service is needed for communication between the web interface/StoreFront, and the data collector. Web Interface communicates using XML Broker to get information, such as application availability for a user, and available resources.

The XML Broker service is needed in a XenApp/XenDesktop environment, and can be deployed as a load-balanced service. Again, Citrix has made a custom monitor available, which we can use to monitor whether the XML Broker service is responding or not. To add the custom monitor, go to **Load Balancing | Monitors**, and click on **ADD**. Here, choose **CITRIX-XML-SERVICE**, and in the **Special Parameters** pane, enter an application name. The default application is Notepad. This monitor will open a connection to the service, and probe the XML Broker service to which it is bound. If the server responds as expected within the configured time period, the monitor marks the service as up. After this is added, we can start adding servers that have the XML service running under servers. Then, we add the XML service under services by choosing **HTML** as the protocol, and adding the port on which the XML service is running. Next, we need to create the load-balanced vServer, by choosing protocol **HTTP** and port **80**, and binding the services to the vServer.



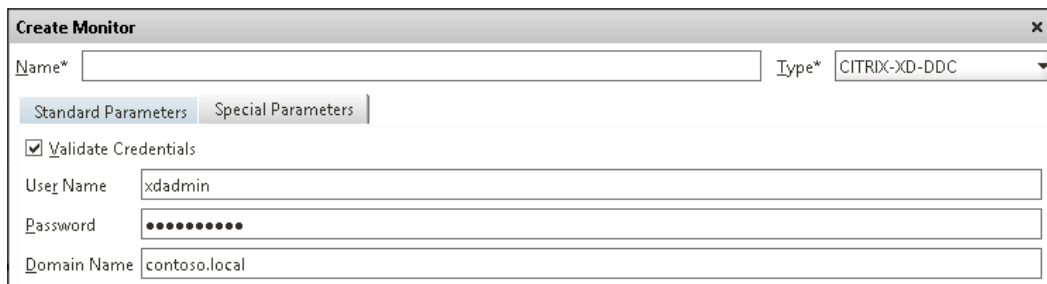
Even though we can choose to create the XML service with HTTP, it is always considered a best practice to use SSL so as to secure communication, even if the traffic is internal. Also, if you intend to use HTTP for XML, a best practice is to use another port instead of port 80.



When we are finished with the configuration of the vServer, we can now use this IP when connecting to StoreFront or Web Interface. For example, if we want to use the vServer with Storefront, we can add the load-balanced server under **Add Delivery Controllers | XenApp | Servers**. Here, we can enter the IP address of the load-balanced XML service.

## Load balancing Desktop Delivery Controller

With the release of XenDesktop 7 and the combining of XenApp/XenDesktop architecture, this has become a more crucial part to load balance. Yet again, there is a custom monitor that needs to be created called **CITRIX-XD-DDC**. In the **Special Parameters** pane, we need to enter an AD user, which can be used to validate credentials. This is shown in the following screenshot:



The screenshot shows the 'Create Monitor' dialog box. At the top, there's a title bar 'Create Monitor' with a close button. Below it, there's a 'Name\*' text box and a 'Type\*' dropdown menu set to 'CITRIX-XD-DDC'. There are two tabs: 'Standard Parameters' (selected) and 'Special Parameters'. Under 'Standard Parameters', there's a checked checkbox for 'Validate Credentials'. Below that are three text boxes: 'User Name' with 'xdadmin', 'Password' with masked characters, and 'Domain Name' with 'contoso.local'.

Now, add the servers, bind them to the service, and then create a load-balanced vServer, as we have done for other load-balanced services.

## Load balancing TFTP for provisioning servers

This is a new feature, which came with NetScaler 10.1. It is the ability to load balance TFTP servers. Before the 10.1 release of NetScaler, this required a great deal of work including the use of **Direct Server Return (DSR)** and other options. However, they are no longer required.

An important point to remember is that when you boot a virtual machine using PVS, it uses either PXE or the DHCP options, including options 66 and 67. This guide uses DHCP options to distribute the link to the bootstrap file.

Now, in order to set up load balancing for TFTP properly, we need a monitor that we can use to verify if load balancing is operational. To get a monitor for TFTP, we can follow the guidelines located at <http://blogs.citrix.com/2011/01/06/monitoring-tftp-with-citrix-NetScaler-revisited-pvs-edition/>.

After we have created the monitor, we need to add the servers where the TFTP service resides. Next, we need to create a service for each TFTP server. Here, we need to choose **TFTP** as the protocol, and in the **Port** field we need to enter 69. Then, we need to bind the custom-made monitor to the service. After this is done, we create the load-balanced vServer, where we enter an IP address, the name, the protocol (TFTP), and the port (69). When this is complete and we have created the vServer, we can alter the DHCP option 66 to point to the new VIP address that we created in NetScaler.

As a side note, it is also possible to deliver the bootstrap using HTTP instead of TFTP. This scenario is only viable for XenServer as it uses gPXE, which allows for extra features such as HTTP. This makes it a lot easier to load balance, as we only need to load balance a simple HTTP server, and change the option 67 boot filename to point to `http://serverip/ARDBP32.BIN`. However, this is not supported by Citrix and should only be used in environments where HTTP is a more suited protocol. As always, remember to save the configuration using the GUI or the `save config` command in CLI.

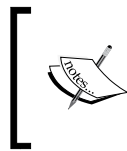


Make sure you are running build 120 of NetScaler or higher before setting up TFTP load balancing. If you have build 118 or 119, make sure that you do not create a load-balanced TFTP server before setting up high-availability, or else NetScaler will crash. This is a known issue that has been fixed in build 120. You can read more about it at <http://support.citrix.com/proddocs/topic/ns-rn-main-release-10-1-map/ns-rn-known-issues-10-1-118-x-con.html>.

## Load balancing SharePoint 2013

SharePoint has become quite a complex product in its latest releases, from starting out as a portal solution to becoming a complete collaboration platform for businesses. SharePoint can be seen as a web application, and it primarily uses HTTP and HTTPS protocols for delivering content to the users. With SharePoint 2013, there have also been some changes in how it operates. For example, Microsoft has introduced a new distributed cache system, which allows a frontend web server to store a login token in memory. This token is also available for other frontend web servers in the farm. This means that we do not need to set up persistency, as all of the authentication tokens are stored in the cache of the web servers. Also, SharePoint 2013 supports SSL offloading, which means that we can use NetScaler to handle SSL traffic and thereby reduce the load on the SharePoint servers by allowing them to respond only on HTTP.

Lastly, as SharePoint has a conception of what is seen as an internal and a public URL (known as alternate access mappings), we need to configure this as well when we set up load balancing so that SharePoint knows that we have set up a new public URL for the site. We start by adding the frontend SharePoint web servers to the server list by going to **Traffic Management | Load Balancing | Servers**. Next, we need to create a service or service group, where we add the servers and bind them to port 80 and protocol HTTP. Then, we add an HTTP monitor to the services, and click on **Create**.



There is also a way to create a custom monitor that actually monitors if a user can authenticate on the SharePoint site. You can read more about this at <http://support.citrix.com/article/CTX126201>.

Now, we need to create a load-balanced service. Here, we need to bind the services we created earlier, and assign a name, IP address, protocol (SSL), and port (443). We also need to bind a certificate, which can be done in the **SSL Settings** pane.

After we are done creating the load-balanced service, we need to make some changes in SharePoint. We have to configure public URLs under **Farm Settings | Alternate Access Mappings** in SharePoint. You can read more about this at <http://technet.microsoft.com/en-us/library/cc263208.aspx>. Now, we have successfully created a load-balanced SharePoint service.

One thing that is important to note is that NetScaler has a feature called AppExpert. AppExpert allows for simplified deployment of service. This includes load balancing rules, caching rules, compression, redirect, and so on. Citrix has created a template for SharePoint 2013 that we can use with AppExpert. The template can be found at <http://www.citrix.com/static/appexpert/appexpert-template.html>.

An important point to note is that AppExpert templates are based upon default settings from Citrix and require tuning before being used in a production environment. This feature is available under the **AppExpert** pane. From there, we need to import the template by clicking on **Import AppExpert Template**, which we have downloaded from the URL mentioned earlier. So, we have to import the deployment and the template file, and then give it an application name. By default, it is called SharePoint 2013. Next, go to **AppExpert | Applications**. It will list the application name that was created earlier and also the different services that make up the SharePoint 2013 site, as shown in the following screenshot:

**Applications View** x

	Compression	Caching	Rewrite	Responder	Application Firewall
<b>Applications</b>					
<b>SharePoint_2013</b>					
FrontPage_Services					
SOAP_Services					
Portal_Management					
Document_Management					
Editable_Image_Management					
ReadOnly_Image_Management					
Video_Management					
Audio_Management					
Styles_and_Scripts					
Web_Service_Definitions					
Web_Service_Schemas					
default					

**Details: SharePoint\_2013** Find

Public Endpoints: [Not configured \(Click to configure\)](#) Backend Services: [Not configured \(Click to configure\)](#)

Add... Remove Configure Public Endpoints... Configure Backend Services... Configure Application Firewall Enable EdgeSight Monitoring  
 Disable EdgeSight Monitoring Export... Visualizer... Statistics Manage Permissions... Authentication... Authorization... Auditing...  
 Turn Off AAA Configure Persistency Groups... Hits...

Help Close

The different services are listed as a subgroup of the application. We can define public endpoint for each of the services or for the whole application. The same goes for backend services. We can define which backend servers should have each of the services. For larger deployments, we will have the different services scattered across different backend servers; however, for a small deployment, it might be the same. So, all we need to do is create a public endpoint on the SharePoint 2013 application level (which is going to create a load-balanced VIP server), and configure backend services, which bind the load-balanced service to the backend servers. The only thing we need to do first is add the backend servers to the server list.

## Load balancing Exchange 2013

Exchange has always been difficult to load balance because of the way it works, but with the release of Exchange 2013, it has become a lot easier to load balance as the architecture in Exchange 2013 has been dramatically simplified with only two roles, the **Client Access server (CAS)** and the Mailbox server. The CAS now only serves as a stateless proxy to the Mailbox server. This means that we can load balance on layer 4 as it does not matter which CAS a user is sent to. Also, RPC has been removed as a protocol and now HTTPS is used by default with Outlook Anywhere, which makes it a lot easier to load balance. When configuring Exchange, we need to set up CAS using an external URL, which is available only in NetScaler.



An important point to note is that Exchange 2013 does not support SSL offloading, even with the latest release of CU3 and service pack 1. Microsoft has not stated that this would change or be added. Even though it places an extra load on the Exchange servers, they still benefit from NetScaler's ability to do SSL session multiplexing and health checking.

Now, there are multiple features and protocols that we can load balance using NetScaler. They are listed as follows:

- Outlook Web Access (OWA)
- Outlook Anywhere
- ActiveSync
- IMAP4

OWA, Outlook Anywhere, and ActiveSync all use the same port, and can be load balanced using the same vServer. The only difference is that they are available on different URL paths. First, we need to add the servers that are running as CAS to the list of servers. Next, we need to create a service or service group, which we will bind to the server on port 443 and protocol HTTPS. After we have chosen **HTTPS** as a protocol for the service, the **SSL Settings** pane will become active, and there we need to add the digital signed certificate that is attached to CAS. This can be done by going to **Traffic Management | SSL**. From there, we can import the certificate, and then install it for the service.



The purpose of the certificate is to ensure that NetScaler can enable a complete connection to the OWA server backend as the use of certificates requires that both parties have a trusted root certificate in place in order to trust the connection.

Next, we need to create a vServer to set up a load-balanced service. Then, we need to bind it to a virtual IP address, port (443), and protocol (SSL), and bind a new certificate to the vServer. Under **Method** and **Persistence**, we choose **Least Connection** and **COOKIEINSERT** respectively and a timeout of 2 minutes, and then click on **Create**. Also, it is important to set the external domain URL in CAS. This needs to be set from the exchange management console, which you can read more about at <http://technet.microsoft.com/en-us/library/jj218640%28v=exchg.150%29.aspx>.

The external domain URL in the Exchange management console needs to point to the VIP address of the load-balanced service we created.

## IMAP

IMAP is also a protocol that is commonly used in conjunction with Exchange, even though it does not provide many of the same features, such as calendar and public folders. IMAP is primarily used by a client to access e-mail on an Exchange server. Note that IMAP is not enabled by default on Exchange 2013. If you want to use this feature, you can read more about it at [http://technet.microsoft.com/en-us/library/bb124489\(v=exchg.150\).aspx](http://technet.microsoft.com/en-us/library/bb124489(v=exchg.150).aspx).

IMAP primarily uses two ports, TCP 143 for non-secure connections and TCP 993 for secure connections. Again, if we already have CAS on the server list, we do not need to add them again. If they are not added, add them to the list. Before we set up a service, we need to create a custom monitor. Go to **Traffic Management | Load Balancing | Monitors**, and click on **Add**. Here, we need to enter a name, define an interval of 30 seconds, and define port 143 as the destination port. As type, choose **TCP-ECV** and then go to the **Special Parameters** pane. Here, we need to type `The Microsoft Exchange IMAP4 service is ready as the received string`. This monitor queries CAS on that particular port and expects the text in response. Next, we need to create a service or service group. Add CAS to the list and bind them to the service, using protocol TCP and port 143. Then, bind the custom-made monitor we just created.

Now, we need to create a vServer. To this, we bind the service we created earlier, protocol `SSL_TCP`, port 993, and define a virtual IP address. Then, we need to add a digital certificate in the **SSL Settings** pane of the vServer to ensure that clients can use the IMAP service securely.



As we have seen in the SharePoint part, Citrix has the AppExpert feature, which simplifies deployment of a service and configures optimization such as caching and redirection. As of now, this is only available for Exchange 2010 but stay tuned on <http://www.citrix.com/static/appexpert/appexpert-template.html> for newer releases of Exchange 2013.

## Load balancing MSSQL

NetScaler is the only certified load balancer that can load balance the MySQL and MSSQL services. It can be quite complex and there are many requirements that need to be in place in order to set up a proper load-balanced SQL server.

Let us go through how to set up a load-balanced Microsoft SQL Server running on 2008 R2. Now, it is important to remember that using load balancing between the end clients and SQL Server requires that the databases on the SQL server are synchronized. This is to ensure that the content that the user is requesting is available on all the backend servers. Microsoft SQL Server supports different types of availability solutions, such as replication. You can read more about it at [http://technet.microsoft.com/en-us/library/ms152565\(v=sql.105\).aspx](http://technet.microsoft.com/en-us/library/ms152565(v=sql.105).aspx). Using transactional replication is recommended, as this replicates changes to different SQL servers as they occur.

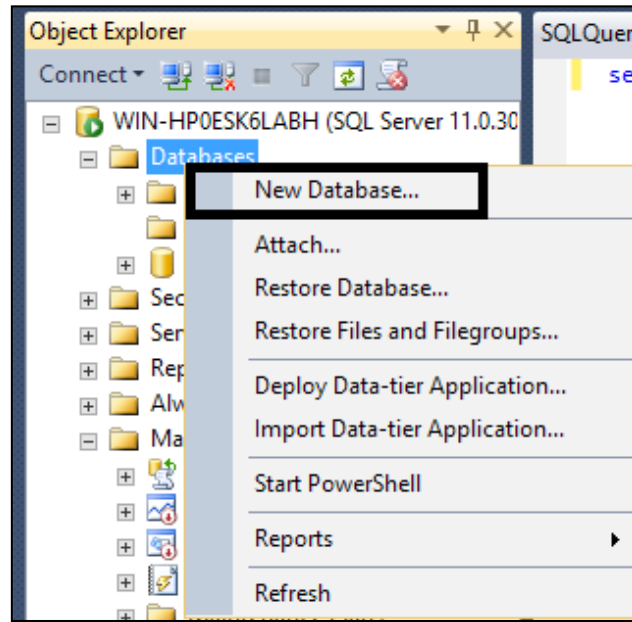


As of now, the load balancing solution for MSSQL, also called DataStream, supports only certain versions of SQL Server. They can be viewed at <http://support.citrix.com/proddocs/topic/netScaler-traffic-management-10-map/ns-dbproxy-reference-protocol-con.html>. Also, only certain authentication methods are supported. As of now, only SQL authentication is supported for MSSQL.

The steps to set up load balancing are as follows:

1. We need to add the backend SQL servers to the list of servers.
2. Next, we need to create a custom monitor that we are going to use against the backend servers.
3. Before we create the monitor, we can create a custom database within SQL Server that NetScaler can query.

4. Open **Object Explorer** in the SQL Management Studio, and right-click on the **Database** folder. Then, select **New Database**, as shown in the following screenshot:



5. We can name it `ns` and leave the rest at their default values, and then click on **OK**. After that is done, go to the **Database** folder in **Object Explorer**.
6. Then, right-click on **Tables**, and click on **Create New Table**. Here, we need to enter a column name (for example, `test`), and choose **nchar(10)** as the data type. Then, click on **Save Table** and we are presented with a dialog box, which gives us the option to change the table name. Here, we can type `test` again.
7. We have now created a database called `ns` with a table called `test`, which contains a column also called `test`. This is an empty database that NetScaler will query to verify connectivity to the SQL server.

Now, we can go back to NetScaler and continue with the set up. First, we need to add a DBA user. This can be done by going to **System | User Administration | Database Users**, and clicking on **Add**. Here, we need to enter a username and password for a SQL user who is allowed to log in and query the database.

After that is done, we can create a monitor. Go to **Traffic Management | Load Balancing | Monitors**, and click on **Add**. As the type, choose **MSSQL-ECV**, and then go to the **Special Parameters** pane.



Here, we need to enter the following information:

- **Database:** This is `ns` in this example.
- **Query:** This is a SQL query, which is run against the database. In our example, we type `select * from test`.
- **User Name:** Here we need to enter the name of the DBA user we created earlier. In my case, it is `sa`.
- **Rule:** Here, we enter an expression that defines how NetScaler will verify whether the SQL server is up or not. In our example, it is `MSSQL.RES.ATLEAST_ROWS_COUNT(0)`, which means that when NetScaler runs the query against the database, it should return zero rows from that table.
- **Protocol Version:** Here, we need to choose the one that works with the SQL Server version we are running. In my case, I have SQL Server 2012.

So, the monitor now looks like the following screenshot:

The screenshot shows the 'Configure Monitor' window. At the top, the title is 'Configure Monitor'. Below it, the 'Name' field contains 'MSSQL'. There are two tabs: 'Standard Parameters' (selected) and 'Special Parameters'. Under 'Standard Parameters', the following fields are visible: 'Database' with value 'ns', 'Query' with value 'select \* from test', 'User Name\*' with value 'sa', 'Rule\*' with value 'MSSQL.RES.ATLEAST\_ROWS\_COUNT(0)', 'Protocol Version' with a dropdown set to '2012', and 'KCD Account Name' with a dropdown. At the bottom, there is an unchecked checkbox labeled 'Store DB'. Below the 'Rule\*' field, there are icons for 'Prefix', 'Add...', and 'Operators'.

It is important that the database we created earlier should be created on all the SQL servers we are going to load balance using NetScaler. So, now that we are done with the monitor, we can bind it to a service. When setting up the services against the SQL servers, remember to choose **MSSQL** as the protocol and 1433 as the port, and then bind the custom-made monitor to it. After that, we need to create a virtual load-balanced service. An important point to note here is that we choose **MSSQL** as the protocol and use the same port nr as we used before 1433.

We can use NetScaler to proxy connections between different versions of SQL Server. As our backend servers are not set up to connect the SQL 2012 version, we can present the vServer as a 2008 server. For example, if we have an application that runs on SQL Server 2008, we can make some custom changes to the vServer. To create the load-balanced vServer, go to **Advanced | MSSQL | Server Options**. Here, we can choose different versions, as shown in the following screenshot:

The screenshot shows the 'Configure Virtual Server (Load Balancing)' window in NetScaler. The 'Name' field is 'MSSQL'. The 'Protocol' is 'MSSQL'. The 'IP Address' is '192.168.88.15' and the 'Port' is '1433'. The 'State' is 'UP'. The 'Advanced' tab is selected, showing 'Connection Failover' as 'DISABLED'. The 'Server Version' is set to '2012'.

After we are done with the creation of the vServer, we can test it by opening a connection using the SQL Management Server to the VIP address. We can verify whether the connection is load balancing properly by running the following CLI command:

```
Stat lb vserver nameofvserver
```

## **Summary**

We have now gone through the different load-balanced services, such as a generic web application, different Citrix components, Exchange, SharePoint, and SQL Server using the DataStream service. Load balancing is essential for many businesses, and it is important to understand how NetScaler can load balance between different backend servers.

In the next chapter, we will go through how to use compression and caching to improve the performance of web services and websites.

# 4

## Compression and Caching

Compression and caching are two important features of an ADC, as they allow for optimization of a service without involving the backend servers and without having to do any configuration on the clients that are connecting. So, we are going to cover the following in this chapter:

- What is caching and compression?
- How do they work?
- How to configure compression and caching for web services?

Let us explore what caching and compression is and how they work as follows:

- **Compression:** This is the ability to reduce the number of bits within data. More specifically, it is the ability to use fewer bits than in the original data, as it is compressed.
- **Caching:** This allows for a NetScaler unit to store commonly accessed data in the RAM, which allows for faster fetching of data.

Both these features allow the client to get hold of the content faster, as it saves bandwidth between the service and the client. It can also reduce traffic to backend servers, and it can protect the backend servers from traffic storms. An important point to note is that these two features are not included by default in the Standard edition. In order to use these features, we either need to buy a feature license or upgrade to the Enterprise or Platinum edition. In order to use caching, we must upgrade to either NetScaler Platinum or NetScaler Enterprise and then buy a feature license.

So, let us start by taking a look at the compression feature of NetScaler.

## Compression

The compression feature enables a NetScaler vServer to compress HTTP data that is going to or from the client. Another benefit of this feature is that the HTTP compression algorithm encrypts the data going from the client to the server and therefore adds another layer of security.



Using compression does not encrypt the data as efficiently as using a digital signed certificate, so do not think of replacing certificates with compression.

This feature requires that the client who is requesting the content has a browser that supports compression. The newest and most common browsers, such as Firefox 4 and above, Google Chrome 20 and above, and Internet Explorer 7 and above, support HTTP compression. So, when a client connects to a vServer, it will announce what capabilities it has to the server. This allows NetScaler to choose the best type of algorithm.



HTTP compression is based on the GZIP and DEFLATE algorithms. These are defined in the RFC 1950/1951/1952 formats. People interested in its technical aspects can read more at <http://www.ietf.org/rfc/rfc1952.txt>.

Now, the HTTP compression feature of NetScaler will compress data inside HTML, XML, CSS, text, and Microsoft Office documents. It does not compress any picture format files, JavaScript files, or other web files that are not text related. In order to configure compression in NetScaler, we first have to enable the feature globally in the appliance. This can be done using the following CLI command:

```
enable ns feature cmp
```


Here, `cmp` stands for compression. After we have enabled this feature in NetScaler, we have to activate it for a service. A service in this context can be a load-balanced service. This can be done using the following CLI command:

```
Set service nameofservice -CMP yes
```

This can also be done through the GUI under **Traffic Management**. Then, click on **Service** and go to the **Advanced** pane. Go to the **Settings** part of the window and enable **Compression**, as shown in the following screenshot:


The screenshot shows the 'Configure Service' window with the following details:

- Service Name:** web
- Server:** okok (192.168.88.100)
- Protocol:** HTTP
- Port:** 80
- Traffic Domain:** 0
- Service State:** UP (indicated by a green dot)
- Number of Active Clients:** (empty field)
- Buttons:** Disable, Enable Health Monitoring (checked), AppFlow Logging (checked)
- Tabs:** Monitors, Policies, Profiles, Advanced (selected), SSL Settings
- Thresholds:**
  - Max Requests: 0
  - Max Clients: 0
  - Max Bandwidth (kb/s): (empty field)
  - Monitor Threshold: (empty field)
- Idle Time-out (secs):**
  - Client: 180
  - Server: 360
- Settings:**
  - Use Source IP: (unchecked)
  - Client Keep-Alive: (unchecked)
  - TCP Buffering: (unchecked)
  - Compression: (checked)

 How to use Wireshark to analyze network traffic using filters and looking into different HTTP headers will be covered as part of *Chapter 5, High Availability and Traffic Analysis*.

## Implementing compression policies

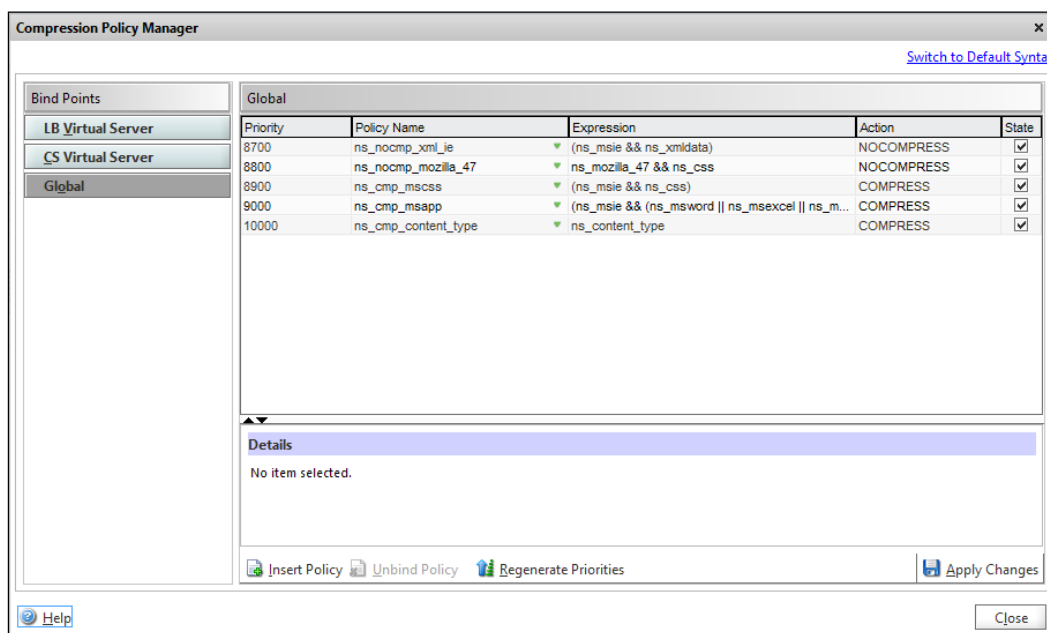
Now, after compression has been enabled, NetScaler will use the default policies that are set at a global level. We can see that, after we enable this for a service, it will automatically start compressing data for that service. If we go to the **Compression Policy Manager** window under **HTTP Compression** and click on **Switch to Classic Syntax**, then on **Global**, we can see the policies that are applied on a global level.

 The reason why we need to go for the classic syntax here is that, in the global settings of the compression feature, we have a configuration that defines which policies are processed and which are not. By default, this is configured to be policy type Classic. This is covered later in this chapter.

By default, there are five global policies. Each of the policies has an action attached to it; they are explained as follows:

- `ns_nocomp_xml_ie`: This policy does not compress when a request is sent from Internet Explorer. The content type is either text or XML.
- `ns_nocomp_mozilla_47`: This policy does not compress when a request is sent from Firefox. The content type is either text or XML.
- `ns_cmp_mscss`: This policy compresses the CSS file when the request is sent from Internet Explorer.
- `ns_cmp_msapp`: This policy compresses files that are generated by Microsoft Word, Excel, or PowerPoint.
- `ns_cmp_content_type`: This policy compresses data when the response contains text.

These policies can be viewed in the following screenshot:



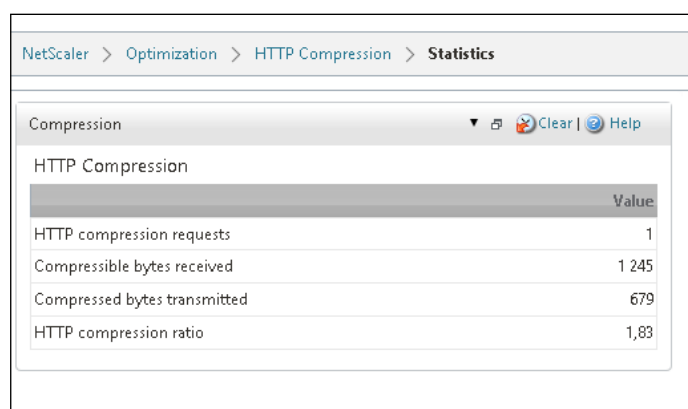
What these policies do is that they do not compress data coming from the client to the services, but they will compress data that is generated from the servers, which contains either CSS files, Microsoft Office documents, or text.

After we have enabled compression for a service, we can test it by running a few HTTP requests against a service, for example, by opening a web browser to a service we defined in NetScaler. In my example, I have a simple IIS server setup, where I query the index page.

To view statistics, we can use the following CLI command:

```
Show cmp stats
```

We can also go through the GUI under **HTTP Compression | Statistics**, as shown in the following screenshot:



Compression	
	Value
HTTP compression requests	1
Compressible bytes received	1 245
Compressed bytes transmitted	679
HTTP compression ratio	1,83

We can see that it has already managed to compress about 50 percent of the data. This feature uses the CPU of the appliance. So make sure that you do not enable compression if you have large amount of services, as NetScaler uses a large amount of CPU to perform compression.

## Defining global compression settings

We can define some global settings to make sure that the compression feature does not run if NetScaler exceeds a particular CPU usage. Go to **Optimization | HTTP Compression | Settings | Change Compression Settings**. Here, we can define the following parameters:

- **Quantum Size:** This parameter defines the amount of data that has to go through before NetScaler starts to compress data. The default value here is 57344 or 57 KB.
- **Compression Level:** Here, we can define at what level NetScaler should compress data. Best performance equals less compression and less CPU usage. Best compression equals more CPU usage.



- **Minimum HTTP Response Size:** This parameter defines what the minimum size of an HTTP response must be before it starts to compress. This should be set at a minimum of 100 KB so that NetScaler does not use a lot of CPU to compress small responses.
- **Bypass Compression On CPU Usage:** Here, we can define a percentage of CPU usage before NetScaler bypasses the compression feature. By default, this is set at 100 percent, which means that if NetScaler has 100 percent CPU usage, the compression feature is bypassed.
- **Policy Type:** Here, we can define a classic policy or an advanced policy. When we enable compression at a global level, it means that all classic policies will be enabled. If we create an advanced policy and bind it globally, it will not be processed if the policy type is classic.
- **Allow Server-side Compression:** If we have this feature enabled, it allows the backend web servers to enable compression. This feature makes NetScaler remove the Accept-Encoding header on all requests going to the web server, which makes the web server respond with an uncompressed response. This makes NetScaler respond back to the client with a compressed response.
- **Compress Push Packet:** This allows NetScaler to avoid waiting until it reaches the quantum size before it can start to compress data, when it receives a TCP packet with the PUSH flag enabled.
- **Vary Header:** This feature adds the Vary header to HTTP responses that are being compressed. This is mostly used in caching scenarios.
- **External Cache:** If private caching is enabled, NetScaler will add private cache-control to the HTTP header to make sure that the data is intended for a single user.

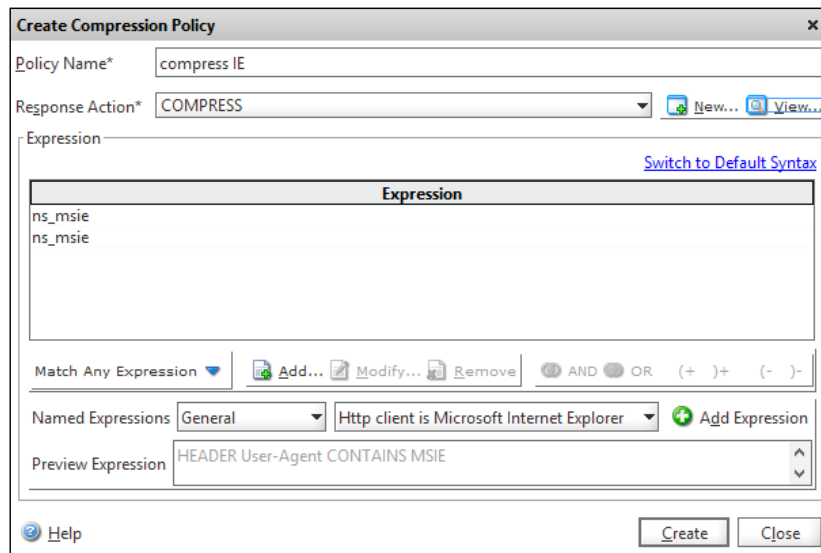
Now, most of these settings will be at their default values, but if you have a scenario where you, for example, have lots of large services and web servers with backend-enabled compression, you will need to change some settings here to make sure that it works properly.

## Creating custom compression policies


We have gone through the different settings; it is now time to create our own compression policies. A policy is built up of a rule and an action. The rule can contain a query, for example, a client who is connecting using Firefox version 4.7. The default action for this rule would be to compress data.

1. Go to **Optimization | HTTP Compression | Policies | Add**. First, we need to change to classic syntax because of the global configuration of the policy type.

2. Next, add an expression in the **Expression** window. Note that we can choose from a list of predefined expressions.
3. As an example, we can choose **General** and **HTTP client is Microsoft Internet Explorer**, and add the action **Compress**.
4. The expression should be similar to the one shown in the following screenshot:



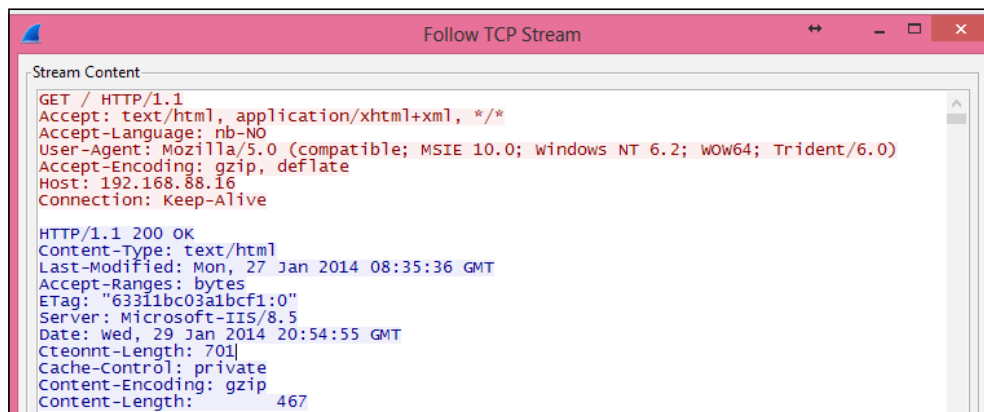
5. Click on **Add Expression**, and then on **Create**.
6. If we choose **Add** here, we can define a custom expression based upon the URL, header, IP, and so on. I'm not going to cover all the different types of expressions that we can use. Citrix has a good overview of the different expressions we can use in classic syntax at <http://support.citrix.com/proddocs/topic/ns-main-appexpert-10-map/ns-pi-gen-expr-ref.html>.
7. After we have created the policy, we have to either bind the policy to a service or bind it globally. Go to **Compression Policy Manager | Switch to Classic Syntax | Global**. Then, click on **Insert Policy** and choose the newly created policy.

 An important point to note is that classic and default syntaxes can perform the same type of evaluations, but the default syntax policies can also analyze deeper within the data, for example, the body of a HTTP request. It is therefore recommended to start using default syntax policies.

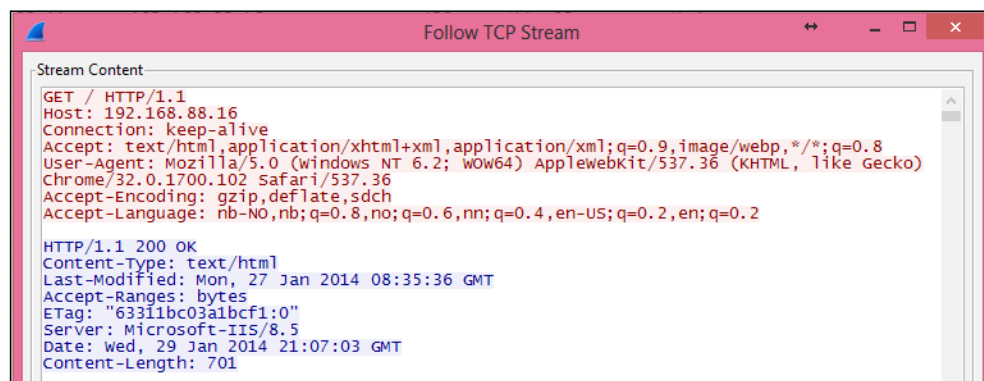
## Testing our compression policies

If you wish to test a policy against a service, bind it to the service and define a low-numbered priority to make sure that it applies before other policies. In this example, we've added the newly created policy to the global level, and set it at priority 100 so that we can make sure that the policy is applied for all connections made from Internet Explorer. We can also unbind all the other policies to make sure that no other policies interfere with the one for Internet Explorer.

So, when we try to open a connection from Internet Explorer, we will be able to see from the packets that the traffic is compressed from the HTTP request header. We can see this in the following screenshot in the **Content-Encoding** field, which says it is compressed with **gzip**.



If we do the same for Google Chrome and analyze the traffic in Wireshark, we can see in the following screenshot that the traffic is not compressed and the data is sent in clear text, as there is no policy that involves an expression containing an action for Google Chrome:



We have now created a custom policy for Internet Explorer users and explored the different options for compression and how it works. It is time to continue on with the other optimization feature, that is, caching.

## Caching

With the use of caching, we allow NetScaler to store frequently accessed objects in its cache area, which uses the RAM of the appliance as storage. Note that the MPX utilizes the SSD drive as the cache drive. This allows NetScaler to serve clients directly without requiring a trip to the backend servers. This offloads the backend servers and improves the overall performance of the services.

Now, caching is not something new; it is used everywhere. Most of the common web browsers have some sort of a local cache; many of the ISP vendors also have some sort of caching involved for frequently accessed services. Also, it is important to note that this feature requires a license in place, either a NetScaler Platinum license, or an add-on to Enterprise. Note that the caching feature is available only when using the HTTP protocol, and it can cache the following two types of data content:

- **Static data:** This includes CSS files, JavaScript files, images, static HTML pages, and so on.
- **Dynamic data:** This includes the dynamic catalog view, automatically generated files, and so on.

## Enabling caching

Before we process any configuration, we first need to enable the caching feature. This can either be done via the GUI by right-clicking on **Integrated Caching** under **Optimization** and enabling this feature, or by using the following CLI command:

```
Enable ns feature IC
```

Before we do any other configuration, we also need to define some of the global settings for caching. This can be done under **Optimization | Integrated Caching | Change Cache Settings**. After we have enabled integrated caching, the default value for memory usage limit is set to 0, which means that NetScaler will not cache anything. Therefore, we need to put a value here. The maximum value that we can set is half of the memory available on the appliance. After we have put a value here, we can leave the rest of the settings at their default values and continue with the creation of a policy.

The caching feature consists of the following:

- A policy
- An action
- A content group

In the policy, we define an expression (rule) that needs to be evaluated, and then an action that defines what to do with the data defined in the rule, for example, to cache the objects or not. Then, we need to define a content group. This is where NetScaler is going to store the cached data for the objects.

## Creating a content group

As an example, let us set up a basic caching policy that caches PDF files. Then, we are going to bind that policy to a load-balanced service.

First, we need to create a content group. This can be done by going to **Optimization | Integrated Caching | Content | Groups** and clicking on **Add**. Here, we enter the name of the content group, for example, PDF group, define an expiration timer (60 seconds), then go into the **Memory** pane, and define that NetScaler does not have to cache if the file is over 5,000 KB. By adding these parameters, we allow cached content to live in the content group for 60 seconds before it expires, and then be removed from the content group. We also define that NetScaler is not going to cache objects larger than 5 MB. We can leave the rest of the settings at their default values.

If you wish to know more about the different parameters in the content group, you can view the eDocs article at <http://support.citrix.com/proddocs/topic/ns-optimization-10-map/ns-IC-setbasiccntgrp-tsk.html>.

## Creating a caching policy

Now that we have created a content group, we can create a policy. Go to **Policies** and click on **Add**. Now, we need to define an expression that only includes PDF files, attach an action, and configure where NetScaler is going to store the files. Give the policy a name, choose **CACHE** as the action, change **Store in Group**, and choose the content group that was created earlier.

In the **Expressions** field, type the expression `HTTP.REQ.URL.ENDSWITH("pdf")`. This expression allows NetScaler to cache all objects where the URL ends with pdf. We could also use the `HTTP.REQ.URL.CONTAINS("\pdf\")` expression if we had pages containing links to PDF reports.

If you wish to combine multiple expressions in a policy, you can see the example at <http://support.citrix.com/proddocs/topic/ns-optimization-10-map/ns-IC-configcacheobjectandstats-tsk.html>. After we have added the expression, the window looks like the following screenshot:

**Create Cache Policy**

Name\*

Action\*  Store in Group

Undefined-Result Action

Expression

Invalidate all objects in the following groups

Available Content Group Name	Configured Content Group Name
calloutContentGroup	
loginstaticobjects	
TCVPNloginstaticobjects	
OCVPNloginstaticobjects	

Invalidate selected objects in the following parameterized groups

Content Group Name	Inval Selector Name	Available Selectors:	Configured
		<input type="text" value="Configured"/>	

Lastly, click on **Create**. Now that we have created the policy, we have to bind it to the vServer for which we want caching enabled. Go to the **Integrated Caching** menu and open **Cache Policy Manager**. Go to **LB Virtual Server** and choose the load-balanced vServer we wish to bind it to. Then, click on **Insert Policy** and choose the newly created policy. Click on **Apply Changes** and close the window.

We have now created a caching policy and attached it to a vServer, which has a PDF document available. When the first client connects to the vServer and tries to get the content, NetScaler will place it in the cache.

This can be viewed under **Integrated Caching | Statistics**, as shown in the following screenshot:

Integrated Cache		
	Rate (/s)	Total
Hits	0	0
Misses	0	1
Requests	0	1
Hit ratio(%)	-	0
Origin bandwidth saved(%)	-	0
Cached objects	-	1
Marker objects	-	1

As it is the first connection, we will see a miss and one cached object. If we go to **Integrated Caching | View Cache Objects**, we can see what data is stored in the cache. We can also see that the PDF file is stored in the content group we defined earlier, as shown in the following screenshot:

Cache objects				
Content Group Name	-All-	HTTP Status Code		<input type="checkbox"/> Ignore Marker Objects <input type="checkbox"/> Include <a href="#">Go</a>
Locator	Content Group Name	HTTP Request Method	Host	URL
0x0000000544730000002f	PDF-group	GET	192.168.88.16:80	/sf.pdf

As we can see from the statistics, when a new client connects, it will be served from the cache directly.

Integrated Cache			Graphical View   Details   Default Group
	Rate (/s)	Total	Integrated Cache Requests
Hits	0	1	
Misses	0	1	
Requests	0	2	
Hit ratio(%)	-	50	
Origin bandwidth saved(%)	-	45	
Cached objects	-	1	

A new client will be allowed to get data from the cache as long as the content it is trying to access has not expired. If we open an object from within **View Cache Objects**, we can see how much time an object has before it gets expired.

After we have created a caching policy, there are some settings we can do in order to fine-tune the configuration and improve the performance of this feature.

## Fine-tuning caching

One problem with caching is when multiple users try to access the same data simultaneously and NetScaler is not finished with downloading the object into the cache. This leaves the cache in a state where multiple users are directed to the backend servers instead of the cache.

There are some changes we need to make to ensure that clients do not bypass the cache. They are explained as follows:

- **Prefetch:** This allows NetScaler to refresh the content before the object expires.
- **Flash Cache:** When a large number of users try to access the same content, NetScaler sends only one connection to the backend server, and all subsequent requests are queued up. A single response is used to respond to all the users.

Both of these settings can be configured in the content group, which stores the cached objects. We can do this by opening a content group, and going to **Others | Flash Crowd | Prefetch**. Here, we can define both of these values within the group. Under **Prefetch**, we have to define how often NetScaler should evaluate the objects in the cache and check the expiration, and define how often it should refresh the cache. In the same pane, we can also see the active flash cache.

There are of course other configurations that we can set to improve the caching mechanism. By default, when we set up a cache policy, it will cache everything that is picked up by the rule.

If we have multiple load-balanced services and we have caching enabled, the cache space will fill up quickly. There are of course some web services that are more frequently accessed than others, so we do not want the less-accessed resources to fill up the cache space. We can attach a rule that defines the minimum number of requests to a server before the caching rule gets enabled. In this way we can make sure that the most active web services get better usage of the cache. This configuration can be set under **Content Group | Memory | Do Not Cache**, if the hits are less. The number here is again dependent on the amount of traffic going towards the different web services. If you are unsure about the amount of traffic going to the different vServers and services, you can use CLI commands.

For services, the following CLI command is used:

```
Stat service servicename
```

For vServers, the following CLI command is used:

```
Stat lb vservice vservicename
```



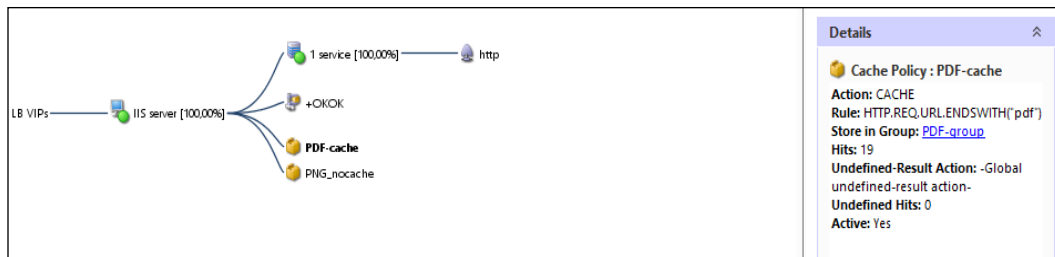
This gives you a general idea of where to set the hit rate of objects that have been fetched from the cache.



If you are unsure of the prefix on a CLI command, you can use the *Tab* key. For example, if you type `stat` and press the *Tab* key, you will get a list of the available options.

It is important to note that you can use the compression feature with the caching feature. This enables NetScaler to send compressed data from the cache to the clients that support compression.

If you want to see a graphical overview of the policies and their statistics for a vServer, you can check the visualizer feature. This can be done by right-clicking on a vServer in the NetScaler GUI and choosing **Visualizer**. This can be seen in the following screenshot:



Here, we can choose filters based upon what we want to display in the GUI. If we choose **Caching** and **Compression** from the top pane, we can see what policies are attached to the vServer. Also, we can view statistics for a policy by marking it. This is a good way to see what policies are actually attached to a vServer and what its statistics are.

## Summary

We have now configured compression and caching for NetScaler. Both of these features provide a drastic improvement of web services. They might use more CPU and memory in NetScaler. So, you need to plan properly.

In the next chapter, we will go through how we can configure high availability, application firewall, and traffic analysis.

# 5

## High Availability and Traffic Analysis

The purpose of many NetScaler instances is to provide load balancing and high availability for services. This should be the case for NetScaler itself as well, so configuring it as a high-availability pair should always be considered. NetScaler often sits in front of large web services processing large amount of data; thus troubleshooting might be cumbersome. So the ability to gather trace data is crucial; the same goes for the ability to protect the numerous services that are available. The following are some of the subjects that we will go through in this chapter:

- Different scenarios for high availability
- Setting up high availability
- AppFlow and integration with NetScaler Insight
- Traffic analysis and Wireshark
- Protecting services using AppFirewall

### Setting up high availability

Consider the scenario where we have NetScaler sitting in front of our numerous services, and thousands of different websites that are load balanced and monitored by NetScaler.

So what will happen when NetScaler goes down, and the services stop and are no longer available for the end users? That's why we should always consider setting up a high availability NetScaler solution for our services.

By having a high availability NetScaler solution, we can ensure that if one of the appliances go down we still have another one or more that are available to serve the requests and load balance the different services.

NetScaler has many solutions that we can use to ensure high availability. The most commonly used deployment is an active/passive pair. This means that we have two appliances that cooperate so that one of the nodes is active (primary) and responds to requests and maintains the connectivity to the servers in the backend and the other node sits passively (secondary) waiting for the active node to go down.

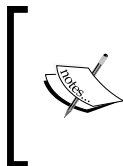
By default, this feature uses GARP to broadcast its MAC address via layer two since both of the nodes use their own MAC address. When a device failover occurs, the secondary node sends out GARP packets to update the MAC table of nearby nodes (switches, routers, and firewalls) so that the new requests will be sent to the new node.

It is important to note that some firewalls do not support GARP or where GARP is blocked and therefore we need to configure VMAC for the deployment. When using VMAC, the MAC address is shared between the two nodes and therefore it is not required to use GARP to update the MAC table on nearby nodes. I'll come back to this later in the chapter and see how we can configure VMAC.

So if the primary node should go down or if it stops responding to requests, the secondary node is going to take over. The nodes monitor each other using heartbeats that are sent between the NSIP address of each of the nodes.

By default, there are some ports that need to be open in the firewall to allow for communication between the nodes in the high availability setup, as follows:

- The UDP port 3003 is used to exchange heartbeats for communicating UP or DOWN status
- The TCP port 3008 is used for secure high availability configuration synchronization
- The TCP port 3009 is used for secure command propagation and for the **Metric Exchange Protocol (MEP)**
- The TCP port 3010 is used for high availability configuration synchronization
- The TCP port 3011 is used for command propagation and for the MEP



High availability is included in every edition of NetScaler and supports a maximum of two nodes. It is important to note that this feature requires us to have two of the same models and the same main release build version. Running an HA pair with, for example, a MPX 5550 and a VPX 1000 is not supported by Citrix.

In order to set up a high availability pair from one of the nodes, we need to know the following information about the other node: its IP address and the default system username and password. It is also required that they have the same RPC passwords; by default, this is the same across NetScaler. To set up a high availability pair, go to **System | High Availability | Nodes** and click on **Add** from one of the nodes. Here we are presented with the following options:

- **Remote IP address** (The NSIP of the other node)
- **Configure remote system to participate in high availability setup**
- **Turn off HA monitor on interfaces that are down**
- **Turn on INC** (Independent Network Configuration) on self node
- **Remote System credentials**

All we need to do is enter the IP address, configure the remote system to participate, turn off HA monitors on interfaces that are down, and enter a different username and password if it differs from the node we are configuring it on.

Turning off HA monitors on interfaces that are down means that NetScaler will not try to send HA probes from one node to another on interfaces that are not in use.

The last option is that INC is needed if the appliances are on different subnets and therefore require independent network configurations, since the regular HA option sets them up using the same network configuration.

After we have entered the information and clicked on **OK**, the primary node will start to propagate its information and configuration with the secondary node and set up a high availability pair, as shown in the following screenshot:

NetScaler > System > High Availability > Nodes						
<div>Nodes      Route Monitors      Failover Interface Set</div> <div> Add...    Open...    Remove    Action ▼ </div>						
ID	IP Address ↑	Host Name	Master State	Node State	INC	Synchronization State
0	192.168.88.3	NS1	Primary	● Up	DISABLED	ENABLED
1	192.168.88.2		Secondary	● Up	DISABLED	SUCCESS

It will also start to synchronize files such as SSL certificates and application firewall XML files; you can view the different files that are part of the synchronization process at <http://support.citrix.com/article/CTX138748>.

It is important to note that there are a few items that will not be synchronized, and those are licenses and `rc.conf` files. There might be issues with syncing SSL certificates; you can verify that they have been synced by using the CLI command:

```
sync ha files ssl
```

Since it is set up in an HA pair, changes that are made to the primary node will be propagated to the secondary node. We cannot make changes from the secondary node; this is shown in the CLI and GUI.

We can view the configuration from the GUI or using the following command:

```
Show ha node
```

In CLI, this shows us which is the primary (active) node and which interfaces are active.

We can also use failover interface sets; if we have multiple network interfaces on NetScaler attached on different switches, we can use them to failover to another interface.

For instance, if we have NetScaler with two interfaces, where interface 1 is attached to switch 1 and interface 2 is attached to switch 2, we can use failover interface sets to failover from NIC 1 to NIC 2 if, for example, switch 1 goes down. In a large environment, there are often large trunk tunnels that span multiple switches that handle this automatically.

In the GUI, we can right-click on each of the nodes and configure actions such as force sync and force failover. Force failover allows us to manually failover if, for example, we need to upgrade.

If we double-click on a node (the one we are logged in to), we get a configuration screen where we can set how the HA pair should function.

By default, when the primary node goes down, the secondary node would take over and promote itself to primary; when the main primary node comes back online, it promotes itself again to the primary node. If we, for example, have a small upgrade process and the primary node goes down, the secondary would take over. If we do not want the secondary to promote itself during the upgrade process, we need to set the secondary node as **STAYSECONDARY**. This stops the process and the primary mode will remain as primary after the reboot.

In this menu, we can also define how often the nodes should send probes to monitor if the nodes are responding.

If you are having some issues with the HA feature, we can use the `nsconmsg` feature by using the CLI. By running the command `nsconmsg -d event`, we can get a live view of events that are happening directly in the console.



By default, NetScaler uses **Gratuitous ARP (GARP)** to advertise its MAC address after a failover has occurred. Some older firewalls from vendors such as Cisco and Juniper do not accept the type of GARP request packets that NetScaler sends out. So if failover is not working in your environment, there is a slight change that needs to be made in order for GARP to function, by logging in to the CLI of NetScaler and running the `set network L2param -garpReply enabled` command.

This command needs to be set on both of the nodes in a high availability setup.

If our firewalls or routers do not support GARP, we can configure NetScaler to use VMAC. VMAC allows NetScaler to have a floating MAC address between them, therefore bypassing the ARP problems with GARP. This can be configured by navigating to **System | Network | VMAC**.

Here we have to define a virtual router ID, for example, 100 and bind it to an interface where the VIP requests come from. The virtual router ID is just used as an identifier within the VMAC. After this is done, the HA nodes will replicate addresses and if we now go to **Network | Interfaces** we can see a VMAC pane that shows the virtual MAC address, which both of the appliances use.

## Differences between clustering, HA, and GSLB

Now, besides the regular high availability feature that requires the use of two nodes, NetScaler also has some other high availability features such as clustering.

We are not going to do a deep dive into clustering but we will be going through a basic overview. If you wish to read more about clustering, you should read the clustering guide from Citrix, which can be found at <http://bit.ly/1gLB1WO>.

Regular high availability operates with a two-node instance where one is active and the other is passive. With clustering, we can scale from 2 up to 32 nodes, which are operating in an Active/Active state. This allows for a large amount of traffic throughput. In order to use this, we require an additional license. Also, there are some other requirements, as follows:

- To be on the same subnet
- To be of the same hardware type (for example, physical appliances)
- To be of the same platform type
- To have the same licenses (Standard, Enterprise, and Platinum)
- To be of the same software version and build
- To have their own feature license to be able to use clustering

Here, we can configure all NetScaler nodes from a single IP called the cluster IP, which in essence replaces the NSIP for management, so they act as one logical entity. The cluster IP is owned by the configuration coordinator, which is in essence the cluster coordinator role, which floats between the different nodes within a cluster.

Every VIP address that we create will automatically be available on all the nodes (called striped IP address) in a cluster; every **Subnet IP (SNIP)** address can also be distributed to all the nodes (striped IP address) or be available from just one node (spotted IP address). Citrix recommends using striped IP addresses for SNIPs.

Clustering can be set up either using **Equal Cost Multiple Path (ECMP)** or using cluster link aggregation.

ECMP is a routing protocol; it defines that a route has multiple paths to the destination with the same cost. This means that if I have multiple roads that I can travel to get to a destination and the distance is equally far, I just choose one of the paths; this way we can distribute traffic between the paths.

Different network vendors have different mechanisms to handle ECMP traffic; for example, Juniper uses a hash algorithm to determine if a packet should travel one path or another. Citrix has written an article on how to configure this using a Cisco Nexus 7000, which is available at <http://support.citrix.com/proddocs/topic/ns-system-10-map/ns-cluster-traf-dist-ecmp-tsk.html>.

Cluster link aggregation is an extension of link aggregation, meaning that we have one interface connected from all the nodes to a switch to create a virtual interface. So instead of a regular link aggregation where we have multiple interfaces from the same appliance, we have one interface from many appliances.



It is important to note that not all features running on a clustered environment are supported, for example, NetScaler Gateway. A list of supported services running on a clustered deployment can be viewed at <http://support.citrix.com/proddocs/topic/ns-system-10-map/ns-cluster-feat-sup-ref.html>.

Now we also have **Global Server Load Balancing (GSLB)**. This is not a high availability feature per se; however, as the name describes, it is a load balancing feature. We are not going to configure GSLB but just see how we can use it and how it works.

GSLB works with the help of DNS. It allows us to deliver a service from different data centers spread across different geographical locations; this helps us in case of data center failures and disaster recovery.

It can also help to spread the load across different locations with its proximity feature and allows users to be sent to the closest data center. This type of feature is often used with, for example, Facebook and Google.

So let us take a closer look at how GSLB works. First off, it is important to know how DNS functions, as this is the fundamental component in GSLB.

When a user connects to a service, for example, `www.myservice.company.com`, the client will send a DNS request to its DNS server. The DNS server of the client will send a recursive request to the authoritative DNS server of that record. The authoritative DNS server will respond with an A-record to the recursive DNS server, which will in turn respond to the client.

Now the client and the recursive DNS server will store the A-record in their cache based upon the **time-to-live (TTL)** entry of the record. When the TTL expires, the client has to once again query the DNS server. This is the component that allows GSLB to work, as GSLB in essence is load balancing and DNS. For example, with GSLB configured and the client again querying for the service `www.myservice.company.com`, the DNS server has a list of different A-records for that particular service that might represent a list of vServers on an HA pair NetScaler on a site. This can also depend on the setup that might return with an A-record for the closest vServer that is able to handle the request.



Now, there are different ways to configure DNS with GSLB here, which are as follows:

- **Authoritative DNS configuration:** This allows NetScaler to act as an authoritative DNS server for a domain, in this case `company.com`. This means that NetScaler will respond to all DNS queries from a recursive DNS server, and based upon where the client is located, respond with an A-record that is closest to the client.
- **Authoritative sub-domain DNS:** This allows NetScaler to act as an authoritative DNS server for a subdomain, for example, `myservice.company.com`.
- **Proxy DNS vServer:** This allows NetScaler to proxy DNS requests to an authoritative DNS server running inside the corporate network. NetScaler has a vServer DNS service running, which load balances DNS queries externally to an internal or another external DNS server.

After we have decided how we want the DNS setup to be, we need to configure GSLB. You can read more about how to configure GSLB at [http://support.citrix.com/servlet/KbServlet/download/22506-102-671576/gslb-primer\\_FINAL\\_1019.pdf](http://support.citrix.com/servlet/KbServlet/download/22506-102-671576/gslb-primer_FINAL_1019.pdf).

It is important to note that GSLB can be configured in different ways as follows:

- Active/Standby
- Active/Active
- Proximity
- Weighted round robin
- Data center persistence

How to configure GSLB for these different scenarios is explained at <http://support.citrix.com/proddocs/topic/netScaler-traffic-management-10-map/ns-gslb-config-comn-dplmnt-scenro-con.html>.

Now that we have explored some of the different high availability features that NetScaler offers, it is also important to keep in mind that running NetScaler in a virtual environment requires planning where to place it. It is important to have some sort of availability feature in place for the virtualization hosts as well, as having a high availability feature on NetScaler is not enough if something happens to the virtualization hosts. We can consider having features such as the following:

- **Failover cluster:** This allows live migration in case of hardware failure.
- **NIC teaming:** This ensures continuous availability in case of NIC failure.
- **Raid on local disks:** This ensures that the system continues to run in case of hard drive failures.

- **SAN redundancy:** This ensures that the storage on which NetScaler resides is redundant.

And, of course, these features should always be used in combination with redundancy in NetScaler as well, as in regular high availability.

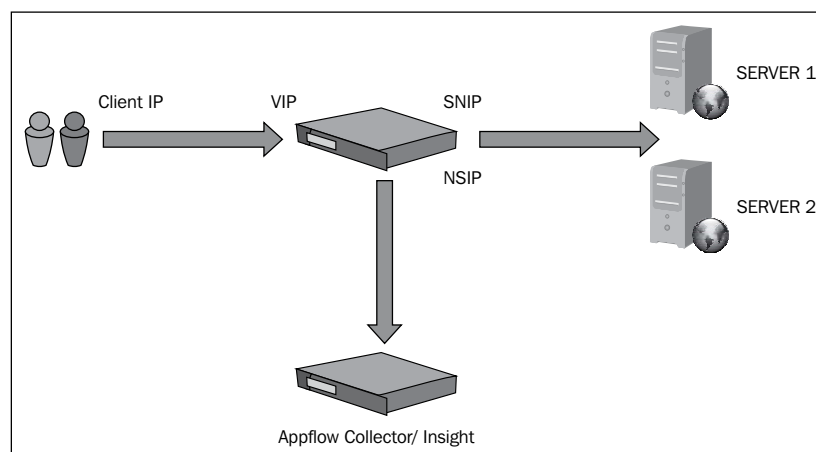
## Using AppFlow<sup>®</sup> to monitor traffic with NetScaler Insight Center<sup>™</sup>

Now, in most cases, NetScaler is used as a central component to deliver high availability services to users, both internally and externally. This means that NetScaler, in most cases, handles a large amount of traffic.

What happens if a user complains about slow performance of an application, or that something is running sluggishly? Or if we want to get an overview of the number of users accessing our services? This is where AppFlow comes in.

**AppFlow** is a feature in NetScaler, which is used to collect web performance data and also database information. It can also be used to gather performance from ICA sessions. It is built upon the IPFIX format, which is an open standard defined in RFC 5101.

As an example, in the following screenshot, when a client opens a connection to the VIP of NetScaler, it will perform a new connection to the backend server and then the traffic is returned from the backend server back to NetScaler and then to the client. The AppFlow feature will send data to a collector with information about the client that connected, which is shown in the following screenshot; the information includes which port and service it accessed and what backend server it got connected to. So we have the complete overview of all the conversations that a client has with a service.



By default, NetScaler uses its NSIP to deliver data to an AppFlow collector; it is important to note that we can use net profiles to define AppFlow to use another IP address, for instance, a SNIP.

Viewing the AppFlow data requires that we have a collector that is capable of analyzing the data.

We can, for example, use other third-party AppFlow connectors such as SolarWinds or Splunk that have the capabilities to analyze AppFlow data. Citrix also has a solution that is called NetScaler Insight, which acts as an AppFlow collector. NetScaler Insight is a virtual appliance that runs either on XenServer or VMware. This appliance can be used as a collector for AppFlow, which allows us to get an overview of the Web and ICA traffic.

 NetScaler Insight is not yet supported on Hyper-V and there is no current ETA for when Citrix is coming with a release for Hyper-V. NetScaler Insight is available for download on mycitrix.com at <http://www.citrix.com/downloads/netscaler-adc/components/netscaler-insight-center-101.html>. This requires a valid mycitrix.com account.

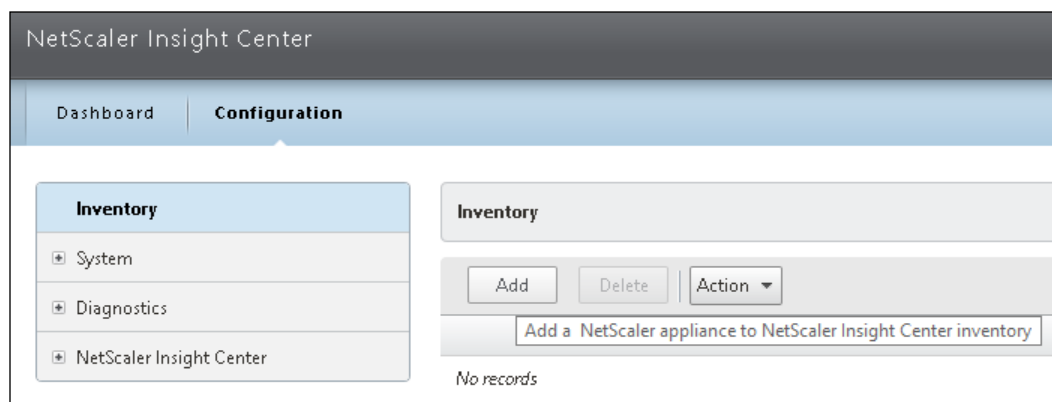
When setting up NetScaler Insight either on XenServer or VMware, just import the OVF file and it will automatically create a VM with the required configuration. The Insight appliance runs also on FreeBSD. FreeBSD is an open source operating system that is built upon UNIX.

Remember to put the Insight appliance on a network where it can reach the NSIP of NetScaler.

The start configuration of Insight is required to be done from CLI. We need to enter an IP address and a subnet mask to allow us to communicate with it using the GUI.

After we have entered the required IP configuration, we can access it by opening a web browser against the IP address. The username and password is the same as the NetScaler appliance, that is, `nsroot` and `nsroot`.

The first time we log in, we are presented with a dashboard with two main panes: **Dashboard** and **Configuration**. In order to get AppFlow traffic, we need to add a NetScaler instance.



Go to **Configuration** | **Inventory** | **Add**. Here, we need to enter a NSIP address and the nsroot username and password of our NetScaler appliance.

After we have added an appliance, it will show us a list of vServers that we can configure AppFlow for.

For example, if we have a load balanced vServer we wish to be able to see AppFlow data for, we can right-click on an LB server and choose **Enable AppFlow**.

Now, we are presented with a policy window, where we need to enter an expression. If we wish to get AppFlow data for all traffic that goes to the vServer, we can use the following expression:

```
HTTP.REQ.LB_VSERVER.NAME.EQ("nameofvserver")
```

This will create an AppFlow policy and bind it to the vServer on the NetScaler appliance. We can also enable this for a Content Switching vServer and a NetScaler Gateway vServer.

In order to enable AppFlow for a NetScaler Gateway vServer, right-click on **vServer**, choose **Enable AppFlow**, and insert `true` under expression. This will allow NetScaler to generate AppFlow data for Gateway vServers as well.

We have now configured NetScaler Insight against NetScaler. When new clients now connect to a vServer that has an AppFlow policy bound to it, data will appear in NetScaler Insight.

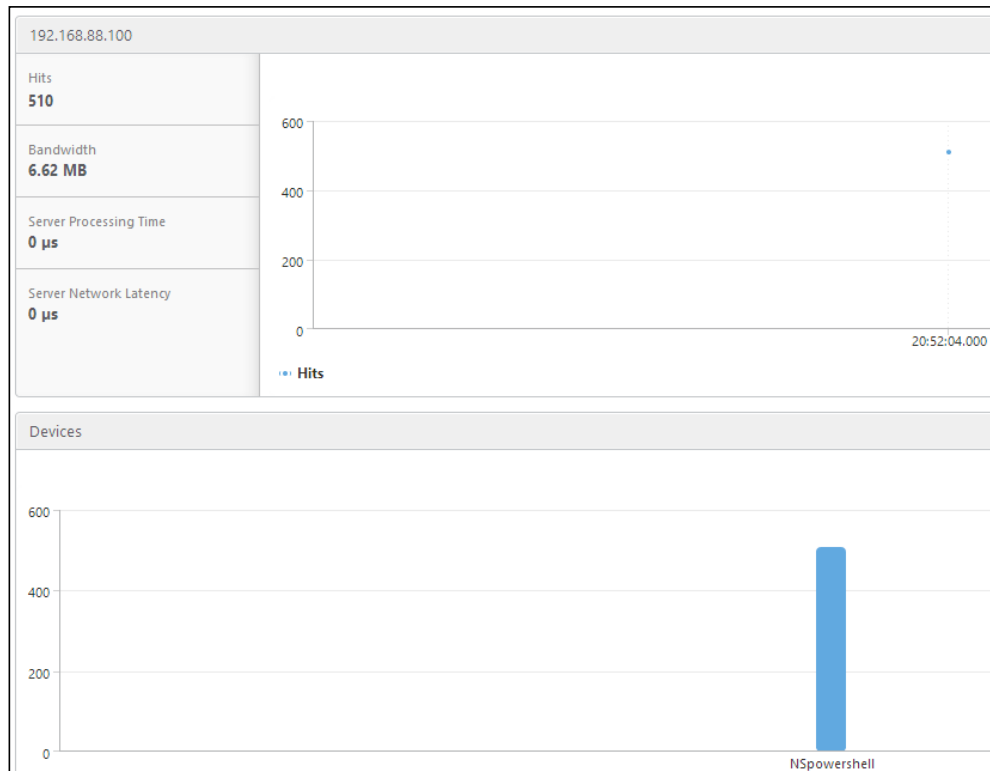
When we go into the dashboard, we are presented with two options, **Web Insight** and **HDX Insight**.



It is important to note that the amount of data that the Insight appliance stores depends on what kind of license the NetScaler appliances are running. If we have the NetScaler Standard license and we wish to use it with Insight, we can only use the Web Insight functionality. If we have NetScaler Enterprise, we can use Web Insight but the HDX insight data will only show traffic for the last month. If we have NetScaler Platinum, we can use Web Insight and HDX Insight and will be able to show traffic for the last year.

Web Insight shows us AppFlow data that is generated from load balanced vServers and Content Switching vServers, and shows us information regarding the Web traffic. HDX Insight shows us the data generated from the NetScaler Gateway vServer.

For example, if we go into Web Insight, we can browse to different categories that show us which clients have access to a server. This is shown in the following screenshot:

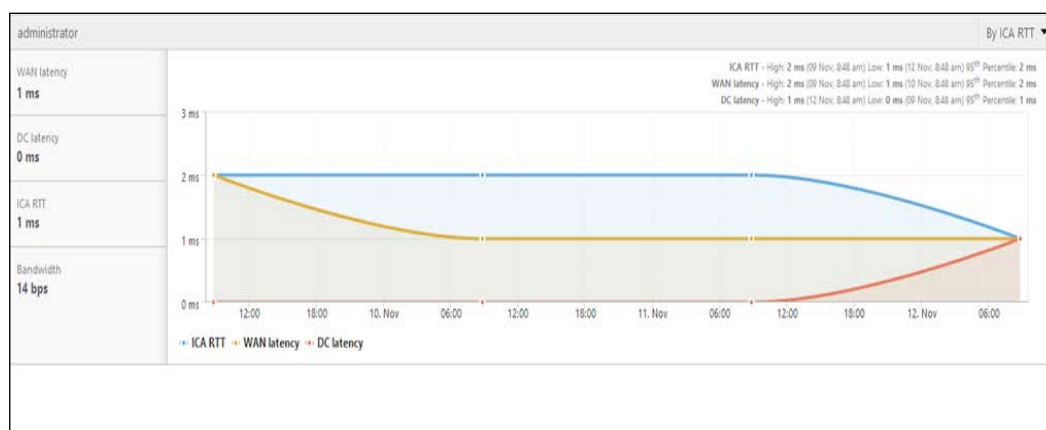


And if we go into HDX Insight, we can get an overview of how many users are accessing our gateway and what applications they are accessing.

We will also get other key information, such as:

- **WAN latency:** This is the average latency caused by the client-side network.
- **DC latency:** This is the average latency caused by the server network.
- **ICA RTT:** This is the average screen lag that the user experiences while interacting with an application or desktop hosted on XenApp or XenDesktop.
- **Bandwidth:** This is the rate at which data is transferred over the ICA session.

This can be seen in the following screenshot:



We have now successfully set up and configured AppFlow integration with NetScaler Insight; if we have a XenDesktop environment, it is also possible to integrate Insight with Director to get a live view from the traffic usage here.

Now, this gives us a look at the network flow of services. Using AppFlow with Insight is a good feature to use to get an overview of how many users are actually using the services and what content they are accessing. But AppFlow does not give us the information we need in case we want to troubleshoot something, for example, if a user has issues connecting, the network is getting sluggish, or a service is marked as down and we need to dig a bit deeper to find the issue.

## Traffic analysis with NetScaler® tools and Wireshark

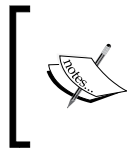
There are times when you need to get your hands dirty to troubleshoot something in a bit more detailed sense to find out what is actually wrong, for example, why we are having trouble with our NetScaler, why a client is having trouble connecting, or why the network is not working as it should.

NetScaler has a number of built-in tools that we can use to gather information and for basic troubleshooting. For example, we have the regular tools such as `ping` and `tracert` to verify network connectivity, but we also have other tools such as:

- `nstrace`
- `nstcpdump`

Both of these tools are accessible from the GUI by navigating to **System | Diagnostics | Start new trace**, and using the CLI.

The difference between these tools is that `nstrace` dumps packets in the native NetScaler format; this has an advantage since it has information regarding clients connecting to a VIP and connections from SNIP to a backend server.



It is important to note that a trace file generated with `nstrace` can also be analyzed with Wireshark, but this requires a modified version of Wireshark that has NetScaler filters installed. The current development build 1.11.2 supports this.

`Nstcpdump` does not give the native NetScaler trace format; it is more resource intensive but gives an ordinary `tcpdump` trace file that we can analyze in, for example, Wireshark.

So by using `nstrace` we can, for example, get trace files for one of our vServers; this requires the use of specific filters. If we do not use filters here, we will get all of the network traffic in the trace file, which, again, can be analyzed using Wireshark filters.

For instance, to get a trace file for our vServer called `vp1` using `nstrace` from CLI, we could use the following filter:

```
nstrace.sh -filter "vsrvname == vp1"
```

Or if we wanted a trace file for our backend service called `IIS`, we could use the following filter:

```
nstrace.sh -filter "svcname == IIS"
```

We could also filter based upon IP addresses or port numbers, as follows:

```
nstrace.sh -filter "sourceport >= 80 && destip != 88.88.88.88"
```

You can press `Ctrl + C` to terminate the trace using the CLI.

These are just to give some examples. To get a list of all the different filter abilities, you can use the `help nstrace` command. We can also use `nstcpdump` for more low-level troubleshooting; in order to use it, we need to enter the shell mode under CLI. This can be done by typing `shell` when in the CLI. With `nstcpdump`, we can also use filters, but not specific to any NetScaler resources.

For example, we can use `nstcpdump` to create a trace file for all traffic going via port 80 as follows:

```
nstcpdump.sh port 80
```

Or we can use it to filter, for example, all traffic from a specific host using TCP as follows:

```
nstcpdump.sh dst host 10.0.0.20 and tcp
```

We also need to specify a location for the trace files, else they would all appear in the console. This can be done by appending the `-w` option and defining a path as follows:

```
Nstcpdump.sh dst host 10.0.0.2 -w /var/nstrace/filename.cap
```

When using GUI, we can also append filters directly to add a trace without knowing the parameters that we need to use. Using the **Start new trace** option, we have the option to start and stop a trace and then download the trace files directly to our computer. By default, all trace files are stored under `/var/nstrace/date`. It is important to note that debugging a trace requires some sort of protocol analyzer to sort the data; this is where Wireshark comes in.

Wireshark is an open source network protocol analyzer. We can use it to monitor live time on a network interface or use it to analyze trace files. It is very powerful and the most commonly used one for troubleshooting network issues where we have a trace file.

So let us go through a basic troubleshooting scenario just to show how you would have troubleshoot network issues in real life. This scenario will cover how we can start a trace file using `nstcpdump`, download it, and then import it to Wireshark.



We can also use a bunch of different filters to sort the data, which I will cover a bit. By default, Wireshark just lists all the packets sorted by package ID.

Wireshark can be downloaded from <http://www.wireshark.org/> and runs on the most common operating systems. Another tool that is also useful in this type of scenario is Microsoft Message Analyzer. The advantage of using Microsoft Message Analyzer will be evident when you need to troubleshoot Microsoft specific services, such as SMB traffic. You can download Message Analyzer from <http://www.microsoft.com/en-us/download/details.aspx?id=40308>.

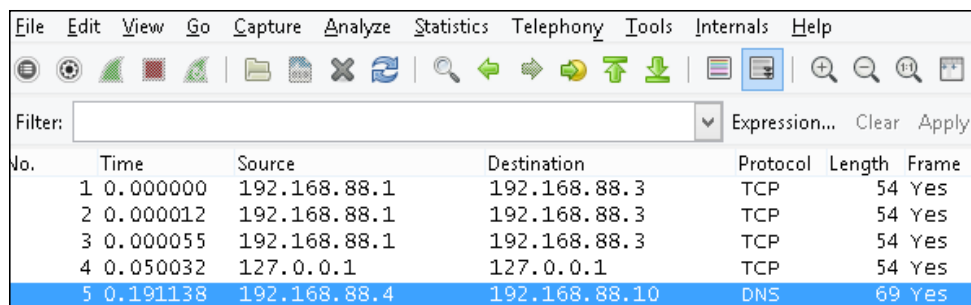
Let us go ahead with an example scenario, just to show how we can use Wireshark to analyze and debug traffic. In this scenario, a user has recently tried to access a vServer called IIS, which is a basic load-balanced web server running Windows Server 2012 R2 with the IP address of 192.168.88.100. The user tries to connect with a web browser but the connection times out. The client is located on 192.168.88.1. Let us see if we can find out why.

We can start the trace using `nstcpdump` in the GUI. Remember to put the packet size to 1514 if we want the entire packet, which in some cases makes it easier to see the entire conversation, although it puts more strain on NetScaler and creates a larger trace file. Also, remember that if we are using HTTP compression on that particular vServer, create a policy for that particular IP and exclude it from compression. Since compression *encrypts* the information using compression algorithms, it makes it a bit harder to troubleshoot.

After this is done, we ask the user to try connecting again. We gather some packets and then we can stop the trace and then download it from the GUI.

After we have downloaded the trace file to our computer, we can open Wireshark and from there go to **File | Open** and then choose the trace file.

Immediately, this will list all the packets from the trace file, sorted by packet ID, as shown in the following screenshot:



No.	Time	Source	Destination	Protocol	Length	Frame
1	0.000000	192.168.88.1	192.168.88.3	TCP	54	Yes
2	0.000012	192.168.88.1	192.168.88.3	TCP	54	Yes
3	0.000055	192.168.88.1	192.168.88.3	TCP	54	Yes
4	0.050032	127.0.0.1	127.0.0.1	TCP	54	Yes
5	0.191138	192.168.88.4	192.168.88.10	DNS	69	Yes

So all packets are listed with a source, destination, protocol, length of the frame, and more information on what is inside the packet. If it is a regular HTTP protocol packet, we can see the information that has gone back and forth in clear text.

We also have a filter box where we can enter different filters to the trace file. A good overview of the use of different filters in Wireshark can be found at <http://wiki.wireshark.org/DisplayFilters>.

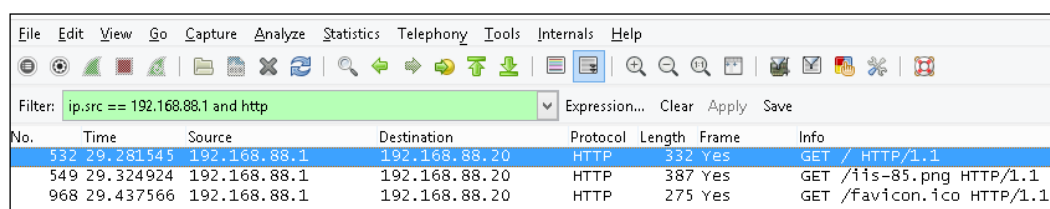
In our example, we can use the following line in the filter box and then choose **Apply**, since we know that the client is connecting from that IP address:

```
Ip.src == 192.168.88.1
```

This will now update the window with all packets coming from that IP address, and it still contains all of the different protocols. To narrow it down to HTTP, we can add the following line of code:

```
Ip.src == 192.168.88.1 and http
```

This leaves us with three packets left to analyze. From here, we can see that the client did an HTTP GET method against the main URL with a PNG and an ICO file on the website. **GET** is the **HTTP** method that is used to request content from a server, as shown in the following screenshot:



No.	Time	Source	Destination	Protocol	Length	Frame	Info
532	29.281545	192.168.88.1	192.168.88.20	HTTP	332	Yes	GET / HTTP/1.1
549	29.324924	192.168.88.1	192.168.88.20	HTTP	387	Yes	GET /iis-85.png HTTP/1.1
968	29.437566	192.168.88.1	192.168.88.20	HTTP	275	Yes	GET /favicon.ico HTTP/1.1

So we can see that the client did request the content; what we can do next is look at the TCP packets that are linked to this request. By right-clicking on one of the packets, we have an option called **Follow TCP stream**. This is useful since it shows us the entire conversation that the client and our web server had and we are presented with a stream content. This shows what was requested and what was returned from the Web server in clear text.

This is useful because it allows us to see what HTTP data is being sent and what is being returned to the client. It is also useful when we are working with configuring caching and compression.



So from the TCP stream, we can see all the information that was presented to the client. Not just HTTP data, but all the other data that is attached to the resource. For example, from the trace file we can get Wireshark to export objects inside it.

By going to **File | Export Objects | HTML**, we will get a list of objects that are referenced in the HTML traffic. For example, we can export the image files that are part of the trace file.

A screenshot of the 'Follow TCP Stream' window in Wireshark. The window title is 'Follow TCP Stream'. The 'Stream Content' pane shows the following text:

```
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,*/*
Accept-Language: nb,en-US;q=0.8,en-GB;q=0.5,en;q=0.3
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 192.168.88.20
Connection: Keep-Alive

HTTP/1.1 200 OK
Age: 1
Date: Mon, 10 Feb 2014 07:54:43 GMT
Connection: Keep-Alive
Via: NS-CACHE-10.0: 2
ETag: "63311bc03a1bcf1:0"
Content-Type: text/html
Last-Modified: Mon, 27 Jan 2014 08:35:36 GMT
Accept-Ranges: bytes
Server: Microsoft-IIS/8.5
Content-Length: 701
```

## Analyzing encrypted content with Wireshark

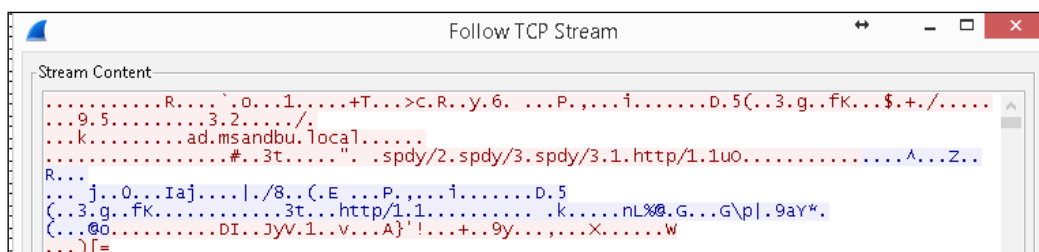
When we look at the TCP stream, we see the conversation the client had with the vServer and that the vServer responded correctly with the content, as shown in the preceding screenshot. So it is only safe to assume that the client had received the content and there most likely is something wrong with the client setup.

Since the trace file is in pure HTTP traffic, it means that it is in clear text, which makes it easy for us to make the trace file and see the entire conversation between the client and the server.

But in most cases, when we have a load-balanced web server it is running HTTPS so the entire conversation is encrypted.

So if we generate a trace file for the HTTPS service, we are unable to read what is specifically happening at the HTTP layer. All we can see is below the HTTP layer, for example, that an IP address has connected to another IP address using HTTPS and port nr.

The following is a screenshot of the TCP stream from a conversation between a client and a server using HTTPS.



## **Maintaining security using NetScaler AppFirewall™**

With the rapid change and development going on with Web 2 applications, security has always been a big concern. The same goes with network infrastructure, which is becoming more and more complex; and with the focus on software defined networking, more of the control plane is moving towards the software.

With this, there is also more that needs to be secured. For example, since websites are not just plain HTML anymore, they contain a large mix of dynamic content working with an active database in the back and adapting based upon who the users are and what kind of device the site is being accessed from.

So where does NetScaler fit in? Since it is usually between the users and the services, it might be, in many cases, the first line of defense against malicious hackers.

NetScaler is equipped with many security features that can fend off attacks, such as the following:

- SYN DoS Protection
- Sure Connect
- Surge Protection
- Rate Limiting
- ACL (Access Control Lists)
- HTTP DoS

These are just some of the features that can be used; last but not least, we have the application firewall feature.


The application firewall is used to secure services running behind NetScaler. It consists of policies and profiles. Here we use the policy to identify patterns in the traffic, and profiles are used to specify what we are going to do with the traffic, like most features in NetScaler.

There are two ways that the application firewall delivers protection; the first is signature based. This means that NetScaler recognizes a pattern based upon a signature and, depending on the action, might drop the connection. So this is useful if it is a known vulnerability that a hacker is trying to exploit.

The second way it can protect is with the deep protection features, which are used for unknown attacks. Here, we can configure it with a learning feature. This learning feature can be a useful way to see, for example, which websites the users are allowed to access internally and which they are not. This allows NetScaler to adapt a standard behavior.

Here we can also define filters. For example, users are allowed to access my website but not a URL starting with /mysite or /sales and we can define an action when someone tries to access a URL with /mysite.

So deep protections are useful for URL masking, SQL injections, buffer overflows, managing forms, and so on.

 It is important to note that Application Firewall is only part of NetScaler Platinum and can be purchased as an add-on Enterprise. It is also recommended that you update the signature file when a new version is released. If your NetScaler is allowed to communicate externally using NSIP, it is also possible to configure autoupdate. This is done by navigating to **Signatures | Action | Auto Update Settings** and enabling **Signatures Auto Update**; after this is enabled, we can go to **Action** and trigger the update version.

Now that we have talked a little bit about how Application Firewall works, let us go in and show how it operates.

We cannot go through every feature within the application firewall, since it contains so much advanced functionality. We are only going to go through the basic features and some of the deep protection features. If you are interested in knowing more about the features not covered in this book, I suggest you head on over to the article in eDocs available at <http://support.citrix.com/proddocs/topic/ns-security-10-1-map/appfw-checks-con.html>; it contains information about all the different deep protection features.

Before we start using the application firewall feature, we have to enable it. This can be done by using the GUI under **System | Settings | basic features | application firewall** or by using the following CLI command:

```
Enable ns feature appfw
```

After that we can go to **Security | Application Firewall** and start configuring the features. To show how we can enable the different protection features, we can go through **Application Firewall Wizard**.

This will start a wizard where we first need to define the name of a web application. This is not linked to vServers but is used just for descriptive purposes. We also need to define a web application type that defines what choices we get in protection features. So as an example, let us say that we want to secure a regular IIS site running on Windows Server 2012, which is basic HTML. Then we can enter the name **IIS** and choose regular web application **HTML** under type.

Next, under **Rule**, we need to define a rule that allows NetScaler to identify which traffic it should look at. Here we can use many of the same expressions we use on the other features; we can also type in `true`. Then, this rule will apply for every connection made to NetScaler.

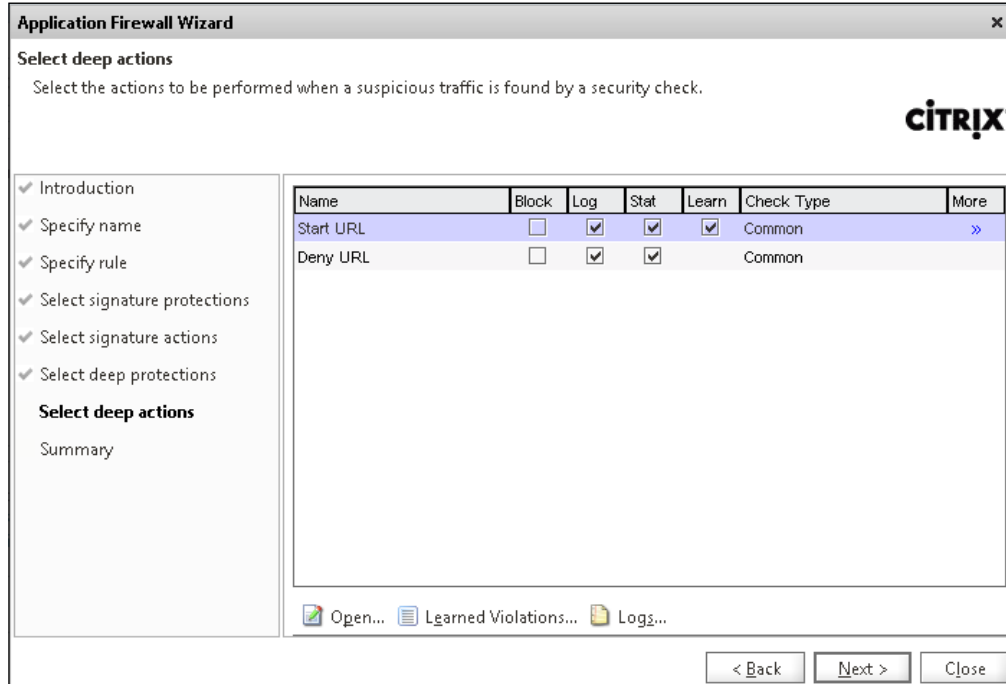
Next, we need to choose what signatures we want to enable; since this is an IIS server, we can enable Microsoft IIS signatures. When we click on **Next** here, we get some options as to what we would like NetScaler to do with the traffic that is picked up by the signatures. The default is **Log** and **Stat**.

If we click on **More...**, we can see all of the different signatures that are going to be used as part of the IIS signature set.

We can leave it at default for now just to see what happens, so click on **Next**. Now we need to configure the deep protection features.

Remember that these features are for an unknown attack, so we can use the learning feature to define a baseline or go in and change the default behavior. This feature applies for XSS, SQL injections, and more, but for now let us just enable **Start URL** and **Deny URL** under the **URL Protection** features.

When we click on **Next**, we are presented with the two features we enabled. This can be seen in the following screenshot:



The **Start URL** action defines which URL a client is allowed to use to start a connection against a service. This means that the first connection from our client can go directly to `http://mycompany.com/index.html` but not directly to `http://mycompany.com/employee/me`.

This function is used to prevent forceful browsing, which means preventing repeated attempts at random URLs. So URLs entered here cannot be used with, for example, bookmarks since they are direct URLs.

By default, the feature is set with the following values:

- **Log:** The NetScaler will log the violations made.
- **Stat:** This will maintain the statistics for the rule.
- **Learn:** This will learn what the default start URLs made to a website are, which makes it easier to deploy rules based upon what is the most accessed start URL.

We can also enable **Block**; then, the feature will start blocking requests made if a rule is violated.

If we click on **Open**, we are presented with **Checks**; these define which start URLs are allowed. We also have a **Learning** button down below; if we have enabled the learning function for this feature, it will show all the entry URLs that have been accessed.

We can also enter our own custom entry start URLs; if we want all our users to start on `index.html`, we can add this by choosing **Add** and entering the URL.

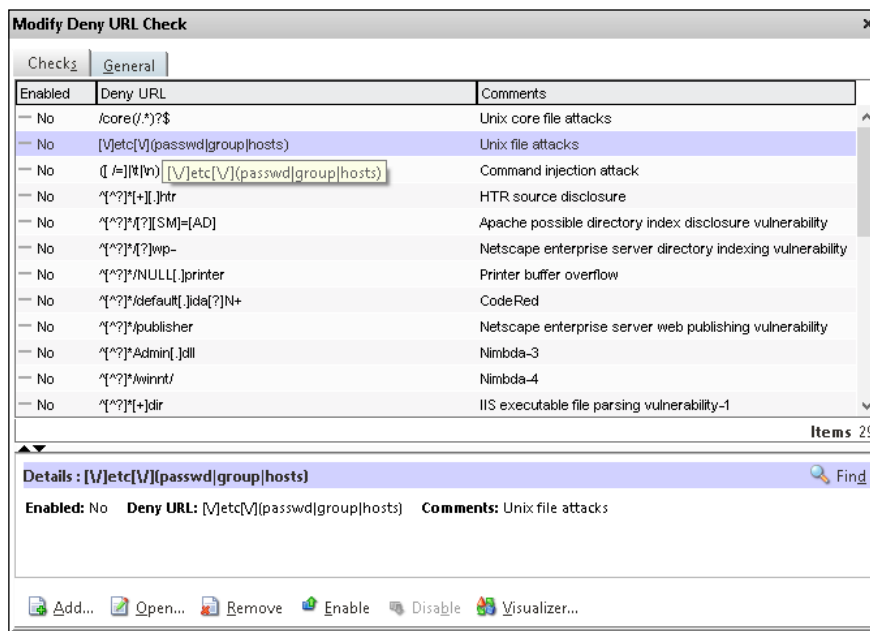


These features use regex to search through the URLs to see if they violate a rule. regex uses a sequence of characters to form a search pattern. Using these features within NetScaler requires a bit of knowledge about them. Fortunately, NetScaler includes a regex tester that allows us to test our expressions. This can be accessed within a feature, for example, by going to **Profile | Open Start URL | Add | Regex Editor**. If you are unsure how an expression should look, you can find a good list of examples at <http://support.citrix.com/proddocs/topic/ns-security-10-map/appfw-checks-url-starturl-con.html>.

But we can enable the blocking action and leave the rest at its default and make sure that the learn action is enabled.



If we go back to the deep actions, we also have the deny URL action. This is basically an allow/deny action. If we click open, we get a list of URLs where we can define the users that are not allowed to browse, as shown in the following screenshot:



Now, by default, none of these rules will do anything. We have to enable them and then define a general action for the rules. For example, if we want to block connections that are tied to HTR source disclosure, we would need to perform the following actions:

- Enable the rule
- Enable an action under the general pane

We can also add our own URLs by choosing the **Add** button and then choosing enabled. So, after we have reviewed the settings, click on **Next** and then on **Finish**.

Now, by default, this policy will be bound at a global level, meaning that it will apply to every service on NetScaler. If we want to bind it to a specific service, we should unbind it at a global level and bind it to a specific service.



It is important to note that using application firewall on a global level will put a lot of strain on NetScaler, since it has to analyze every HTTP packet.

Now, if we wish to change the bindings, go to **Application Firewall | Policy manager | Default Global**; we can see that the application we created is now bound there. Choose **Unbind** and then find the web service we wish to bind it to and choose **Bind**.

If we now start to do some random requests against the Web service, we can see that the log starts to fill up, and if we try to open some URLs that we have defined as not allowed, we can also see the violation rules.

These can be viewed by navigating to **Application Firewall | Policies** and then selecting the name of the policy we just created.

Open it, go to **Security Checks**, and choose **Logs**. We can see from the following screenshot that, after accessing some random URLs on our web service, the **Start URL** rule kicks in and starts blocking connections.

**Syslogs for security check - Start URL**

System Time: Thu Jan 01 00:00:00 1970 GMT

Module: APPFW Event Type: Severity: Find Now C X

Deploy	Timestamp	Seve...	Event Type	Event ID	Client IP	Transaction Id	Session Id	Action	Message
ma, 10 mar 2014 0...	INFO	APPFW_STARTU...	1089	192.168.88.1	1370-PPE0	-	blocked	Mar 10 02:30:24 <local0.info> 19...	
ma, 10 mar 2014 0...	INFO	APPFW_STARTU...	1090	192.168.88.1	1371-PPE0	-	blocked	Mar 10 02:30:24 <local0.info> 19...	
ma, 10 mar 2014 0...	INFO	APPFW_STARTU...	1087	192.168.88.1	1368-PPE0	-	blocked	Mar 10 02:30:21 <local0.info> 19...	
ma, 10 mar 2014 0...	INFO	APPFW_STARTU...	1088	192.168.88.1	1369-PPE0	-	blocked	Mar 10 02:30:21 <local0.info> 19...	
ma, 10 mar 2014 0...	INFO	APPFW_STARTU...	1086	192.168.88.1	1367-PPE0	-	blocked	Mar 10 02:30:19 <local0.info> 19...	

**Message**

Mar 10 02:30:24 <local0.info> 192.168.88.3 03/09/2014:22:00:24 GMT NSpowershell Q-PPE-0: APPFW APPFW\_STARTURL 1089 0: 192.168.88.1 1370-PPE0 - okok Disallow Illegal URL: http://192.168.88.5/asp.lolll <blocked>

**Choose Log**

Log Directory: /var/log Browse...

Log files:

- Current... [ns.log]
- ma, 10 feb 2014 08:30:01 [ns.log.19.gz]
- on, 19 feb 2014 22:30:02 [ns.log.9.gz]

Download Refresh

**Filter Messages**

Search in: Message

Search String: \*. APPFW (APPFW\_STARTURL).\* okok.\*

☐ Case Sensitive ☒ Regular Expression

Clear Go

Every user who tries to access a random URL on the service that triggers a violation rule will automatically get redirected to the main page. We can define a redirect URL for those who try to access random URLs.

This can be done by navigating to **Profile | Settings | Redirect URL**.

We have now seen a small portion of what Application Firewall has to offer; it offers a wide range of different features that can be used to prevent Cross Side Scripting, SQL injections, and such. As mentioned earlier, if you wish to learn more about the different features within Application Firewall, head over to the following eDocs article: <http://support.citrix.com/proddocs/topic/ns-security-10-map/appfw-config-con.html>.

## Summary

We have now gone through some of the different ways that we can configure high availability on NetScaler, and how we can analyze traffic using Wireshark and nstrace.

Also, we went through AppFlow with NetScaler Insight to get a glimpse of how much traffic is entering our network and lastly how we can protect our service using application firewall. It has been a long chapter with lots of different subjects and is unfortunately the end of this book.

Throughout this book, we have just scratched the surface of what features NetScaler has to offer. If you are interested in learning more about the different features NetScaler has to offer and news about them, I would encourage you to visit the following websites and links:

- NetScaler Knowledgebase: <http://www.NetScalerkb.com/>
- Citrix Blogs: <http://blogs.citrix.com/>
- My blog: <http://msandbu.wordpress.com>
- Kees Baggerman's blog: <http://blog.myvirtualvision.com/>
- Neil Spellings's blog: <http://neil.spellings.net/>

# Index

## A

- ACL (Access Control Lists)** 110
- ActiveSync** 70
- AppFlow®**
  - about 99
  - used, for monitoring traffic with NetScaler Insight Center 99-103
  - using 99
- Application Delivery Controller (ADC)** 6

## B

- Bridge BDPU**s 20

## C

- caching**
  - about 85
  - caching policy, creating 86-88
  - content group, creating 86
  - dynamic data 85
  - enabling 85
  - fine-tuning caching 89, 90
  - static data 85
- call ID hash method** 59
- Certificate Signing Request (CSR)** 30
- Citrix® eDocs**
  - reference link 7
- Client Keep-Alive** 19
- clientless access**
  - about 43
  - deploying 43, 44
- cluster link aggregation** 96
- compression**
  - about 77, 78
  - custom compression policies, creating 82, 83

- global compression settings, defining 81, 82
- policies, implementing 79
- policies, testing 84, 85

### **compression policies**

- implementing 79
- ns\_cmp\_content\_type 80, 81
- ns\_cmp\_msapp 80
- ns\_cmp\_mscss 80
- ns\_nocmp\_mozilla\_47 80
- ns\_nocmp\_xml\_ie 80

### **Configuration pane** 15

### **connection persistence methods**

- cookie insert 60
- Custom server ID 60
- destination IP 60
- RTSP session ID 60
- rule 60
- SIP call ID 60
- Source and destination IPs 60
- Source IP 60
- SSL session 60
- URL passive 60

### **custom compression policies**

- creating 82

### **custom load method** 60

## D

### **Dashboard pane** 14

### **Desktop Delivery Controller**

- load balancing 66

### **destination IP hash method** 59

### **Direct Route Advertisement** 20

### **Direct Server Return (DSR)** 66

### **DNS, configuration with GSLB**

- authoritative DNS configuration 98

- authoritative sub-domain DNS 98
- proxy DNS vServer 98
- domain hash method 59**

## E

- ECMP 96**
- Edge Configuration 19**
- encrypted content**
  - analyzing, Wireshark used 108, 109
- endpoint analysis 26**
- Enterprise edition, NetScaler® 9**
- Equal Cost Multiple Path. *See* ECMP**
- Exchange 2013**
  - load balancing 70, 71

## F

- failover cluster 98**
- Fast Ramp mode 18**
- Feature license, NetScaler® 10**
- Flash Cache 89**
- FreeBSD 100**

## G

- generic web application**
  - Backup vServer 62
  - connection persistence methods 60
  - failover, handling 62
  - IP addresses, adding to server 56
  - load balancing 55-60
  - Redirect URL 62
- global compression settings, compression**
  - defining 81
  - parameters 81, 82
- Global Server Load Balancing. *See* GSLB**
- Gratuitous ARP (GARP) 95**
- GSLB**
  - about 9, 97
  - configuring ways 98

## H

- high availability**
  - setting up 91-95
- HTTP DoS 110**

## I

- ICA Proxy**
  - about 26
  - deploying 29-37
  - StoreFront integration 38
- IMAP 71**
- IMAP4 70**
- Internet Information Servers (IIS) 55**
- Intranet Route Advertisement 20**
- IP addresses, NetScaler® networking**
  - about 20
  - MIP 21
  - NSIP 20
  - SNIP 21
- IPv6 Direct Route Advertisement 20**
- IPv6 Static Route Advertisement 20**

## L

- Layer 2 mode 19**
- Layer 3 mode 19**
- least connection method 59**
- least packets method 59**
- least response time method 59**
- licenses, NetScaler®**
  - allocating 10
  - downloading 10
  - Feature license 10
  - Platform license 10
  - Universal license 10
- load-balanced service**
  - about 53
  - load-balancing method 55
  - monitor, adding 56
  - Protocol and Port 55
  - servers 54
  - service 54
  - Services or Service Groups 55
  - VIP address 54
  - virtual server 54
  - vServer name 54
  - weights, assigning 61
- load balancing**
  - Desktop Delivery Controller 66
  - enabling 55
  - Exchange 2013 70, 71
  - generic web application 55-58

- MSSQL 72-75
- SharePoint 2013 67-69
- StoreFront 63, 64
- TFTP 66, 67
- Web Interface 65
- XML Broker 65, 66

**load balancing method**

- call ID hash 59
- custom loads 60
- destination IP hash 59
- domain hash 59
- least bandwidth 59
- least connection 59
- least packets 59
- least response time 59
- LRTM 60
- round robin 59
- source destination IP hash 59
- source IP hash 59
- source IP source port hash 59
- token 60
- URL hash 59

**LRTM method 60**

## M

**MAC based forwarding 19**

**MDX 26**

**Metric Exchange Protocol (MEP) 92**

**MIP address 21**

**modes, NetScaler®. *See* NetScaler® modes**

**monitors**

- HTTP 57
- HTTPS 57
- PING 57
- TCP 57
- TCPS 57

**MPX 7**

**MSSQL**

- load balancing, setting up 72-75

## N

**NetScaler®**

- about 5, 6
- caching 85
- clustering 96
- compression 78

- Enterprise edition 9

- features 7, 18

- Global Server Load Balancing (GSLB) 97

- high availability 98

- high availability, setting up 91-95

- licensing 10

- load-balanced service 53

- modes 18

- MPX 7

- networking 20

- Platinum edition 9

- scenarios, setting up 11

- SDX 8

- Standard edition 9

- types of appliances 7

- VPX 8

**NetScaler AppFirewall™**

- used, for maintaining security 110-115

**NetScaler Gateway™**

- about 25

- clientless access, deploying 43

- deployment setup, tuning 48

- enabling 29

- endpoint analysis 26

- example gateway 27, 28

- features 26

- features, implementing 44-47

- history 25

- ICA Proxy 26

- ICA Proxy, deploying 29

- license 26, 27

- MDX 26

- SmartAccess 26

- SSL VPN 26

- testing 51

- VPN 26

- VPN, deploying 41

**NetScaler® modes**

- Bridge BDPUs 20

- Client Keep-Alive 19

- Direct Route Advertisement 20

- Edge Configuration 19

- Fast Ramp 18

- Intranet Route Advertisement 20

- IPv6 Direct Route Advertisement 20

- IPv6 Static Route Advertisement 20

- Layer 2 mode 19

- Layer 3 mode 19
- MAC based forwarding 19
- Path MTU Discovery 19
- Static Route Advertisement 20
- TCP buffering 19
- Use Source IP 19
- Use Subnet IP 19
- NetScaler® tools**
  - nstcpdump 104
  - nstrace 104
  - used, for traffic analysis 104, 105
- NetScaler VPX™**
  - Configuration pane 15
  - Dashboard pane 14
  - DNS feature 15
  - initial setup 12, 13
  - initial setup, restarting 13
  - NTP feature 15
  - Reporting pane 15
  - SNMP feature 15
  - Syslog feature 15
- networking, NetScaler®**
  - about 20
  - IP addresses 20
- NIC teaming 98**
- NSIP address 20**
- nstcpdump 104**
- nstrace 104**

## O

- Outlook Anywhere 70**
- Outlook Web Access (OWA) 70**

## P

- Path MTU Discovery 19**
- Platform license, NetScaler® 10**
- Platinum edition, NetScaler® 9**
- Prefetch 89**

## R

- Raid on local disks 98**
- Rate Limiting 110**
- Redirect URL 62**
- Reporting pane 15**
- round robin method 59**

## S

- SAN redundancy 99**
- scenarios**
  - setting up 11
- SDX 8**
- security**
  - maintaining, NetScaler AppFirewall™
    - used 110-115
- SharePoint 2013**
  - load balancing 67, 69
- SmartAccess 26**
- SNIP address 21-23**
- source destination IP hash method 59**
- source IP hash method 59**
- source IP source port hash method 59**
- SSL VPN 26**
- Standard edition, NetScaler® 9**
- Static Route Advertisement 20**
- StoreFront**
  - load balancing 63, 64
- StoreFront integration, ICA Proxy 38-40**
- Subnet IP (SNIP) address 96**
- Sure Connect 110**
- Surge Protection 110**
- SYN DoS Protection 110**

## T

- TCP buffering 19**
- TFTP**
  - load balancing 66
- time-to-live (TTL) 97**
- token method 60**
- traffic analysis**
  - NetScaler® tools, used 104
  - Wireshark, used 105
- tuning, NetScaler Gateway™ deployment**
  - setup
    - profiles 50
    - redirection 48, 50

## U

- Universal license, NetScaler® 10**
- URL hash method 59**
- Use Source IP 19**
- Use Subnet IP 19**

## **V**

### **VPN**

- about 26
- deploying 41, 42

### **VPX** 8

## **W**

### **Web Interface**

- load balancing 65

### **Wireshark**

- about 105
- downloading 106
- URL 106
- used, for analyzing encrypted content 108, 109
- used, for traffic analysis 105-107

## **X**

### **XML Broker**

- load balancing 65, 66







## **Thank you for buying Implementing NetScaler VPX™**

### **About Packt Publishing**

Packt, pronounced 'packed', published its first book "Mastering phpMyAdmin for Effective MySQL Management" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: [www.packtpub.com](http://www.packtpub.com).

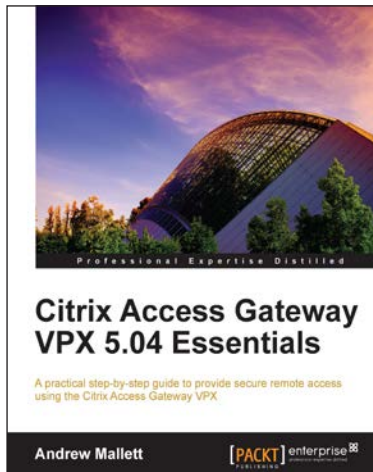
### **About Packt Enterprise**

In 2010, Packt launched two new brands, Packt Enterprise and Packt Open Source, in order to continue its focus on specialization. This book is part of the Packt Enterprise brand, home to books published on enterprise software – software created by major vendors, including (but not limited to) IBM, Microsoft and Oracle, often for use in other corporations. Its titles will offer information relevant to a range of users of this software, including administrators, developers, architects, and end users.

### **Writing for Packt**

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to [author@packtpub.com](mailto:author@packtpub.com). If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



## Citrix Access Gateway VPX 5.04 Essentials

ISBN: 978-1-84968-822-2

Paperback: 234 pages

A practical step-by-step guide to provide secure remote access using the Citrix Access Gateway VPX

1. A complete administration companion guiding you through the complexity of providing secure remote access using the Citrix Access Gateway 5 virtual appliance.
2. Establish secure access using ICA-Proxy to your Citrix XenApp and XenDesktop hosted environments.
3. Use SmartAccess technology to evaluate end users' devices before they connect to your protected network.



## Citrix® XenDesktop® 7 Cookbook

ISBN: 978-1-78217-746-3

Paperback: 410 pages

Over 35 recipes to help you implement a fully featured XenDesktop® 7 architecture with a rich and powerful VDI experience

1. Implement the XenDesktop 7 architecture and its satellite components.
2. Learn how to publish desktops and applications to the end-user devices, optimizing their performance and increasing the general security.
3. Designed in a manner which will allow you to progress gradually from one chapter to another or to implement a single component only referring to the specific topic.

Please check [www.PacktPub.com](http://www.PacktPub.com) for information on our titles



## Citrix® XenMobile™ Mobile Device Management

ISBN: 978-1-78217-214-7

Paperback: 112 pages

Gain an insight into the industry's best and most secure Enterprise Mobility Management solution

1. Deploy and manage the complete XenMobile solution.
2. Learn how to customize and troubleshoot your XenMobile apps.
3. Step-by-step instructions with relevant screenshots for better understanding.



## Citrix® XenApp® 6.5 Expert Cookbook

ISBN: 978-1-84968-522-1

Paperback: 420 pages

Over 125 recipes that enable you to configure, administer, and troubleshoot a XenApp® infrastructure for effective application virtualization

1. Create installation scripts for Citrix XenApp, License Servers, Web Interface, and StoreFront.
2. Use PowerShell scripts to configure and administer the XenApp's infrastructure components.
3. Discover Citrix and community written tools to maintain a Citrix XenApp infrastructure.

Please check [www.PacktPub.com](http://www.PacktPub.com) for information on our titles