

Optimizing Citrix NetScaler and services

MARIUS SANDBU

MSANDBU.WORDPRESS.COM | [@MSANDBU](https://twitter.com/MSANDBU)

About:

Revision:

Version 1.001 (11/02/2016)

Reviewers:

Carl Behrent https://twitter.com/cb_24

Dave Brett <https://twitter.com/dbretty> (<http://bretty.me.uk>)

Carl Stalhood <https://twitter.com/cstalhood> (<http://carlstalhood.com>)

About the author:

Marius Sandbu works as a Senior Systems Engineer at Exclusive Networks in Norway, where he focuses on software-defined datacenter, end-user computing and cloud technologies. He is a Microsoft Azure MVP and the author of, *Implementing NetScaler VPX* and *Mastering NetScaler VPX*.

He can be contacted on Twitter @msandbu or on his email msandbu@gmail.com

Marius's blogs at <http://msandbu.wordpress.com>

Information about this eBook:

This short eBook covers some basic configuration of a NetScaler VPX, both in an on premise environment and in Microsoft Azure. This eBook is aimed at system & Citrix administrators, which are already familiar with Citrix NetScaler and wish to be able to tune/tweak NetScaler and know more about using the different networking settings.

Any feedback can be directed to my email msandbu@gmail.com

Note: that the information presented in this eBook is based on NetScaler version 11.0, expect the part containing Microsoft Azure deployment, which is still running 10.5

Contents

About:.....	1
Background.....	3
Tuning NetScaler in a virtual environment	4
Licensing.....	4
CPU Sizing.....	5
Memory Sizing.....	7
Firmware upgrades.....	7
NIC Teaming and LACP	7
VLAN tagging	8
Jumbo Frames	9
NetScaler deployment in Azure.....	9
NetScaler packet flow.....	14
TCP Profiles.....	14
SSL Profiles	20
VPX SSL limitations	22
Mobilestream	22
Compression.....	23
Caching.....	25
Front-end optimization	27
HTTP/2 & SPDY	29
Tuning for ICA Traffic.....	30

Background

After working with NetScaler for a few years now, I've seen that so many have yet to grasp the full feature set that NetScaler actually offers. So many features and simple tuning that can make a big difference in terms of performance.

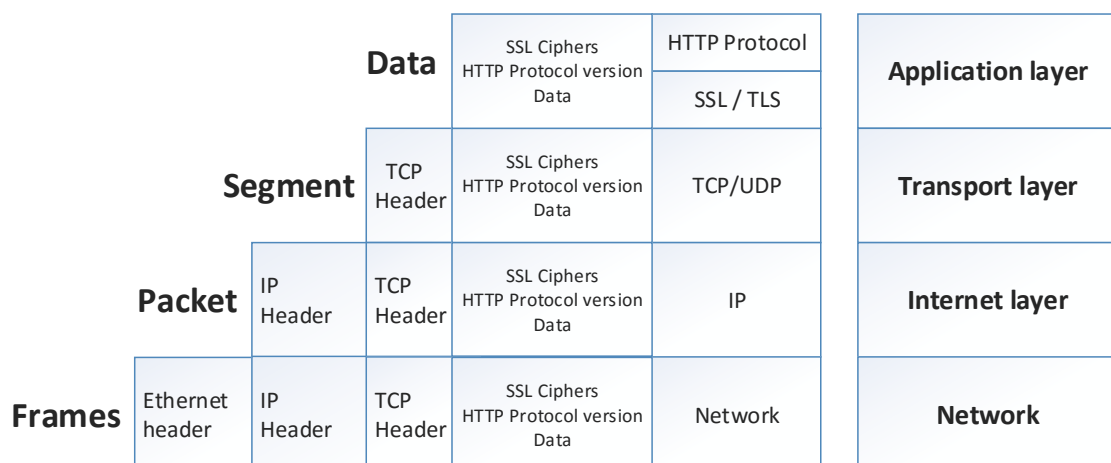
NetScaler is a key component in many environments, like for instance huge e-commerce websites which depend on having a low response time, high throughput connections to ensure that their end-users are able to buy the products they need.

Also when deploying in virtual environments, more factors apply on how well the NetScaler is going to perform.

As an example when setting up a NetScaler VPX in a simple virtual environment using all the default settings from Citrix.

- *It is not using the recommended SSL settings*
- *It is using a default TCP profile which is not optimized for performance*
- *The virtual appliance is not scaled depending on license you require*
- *HTTP profiles are not adjusted accordingly*

So this eBook is to give a bit more overview and on some features a bit more deep-dive on what you should take a closer look at when tuning your NetScaler environment. Also you need to remember that the network stack that you have, NetScaler has many features that can adjust multiple layers on the networking stack



For instance, we have SSL Profiles and http profiles which can adjust performance on the application layer, and we also have TCP profiles which can adjust how the TCP connection should behave. We also have settings like Jumbo frames and LACP which work on the lower layers of the stack.

Tuning NetScaler in a virtual environment

If we are to deploy a NetScaler in a virtual environment, it is crucial that we configure the virtual appliance properly. For instance, it is important that we properly allocate vCPU, since NetScaler uses the CPU to handle all network traffic such as SSL encryption, Application Firewall, content switching, compression, and so on. While physical NetScaler appliances perform SSL processing in a dedicated ASIC, NetScaler VPX is not so fortunate and instead performs SSL processing in main CPU. Compression can also consume significant CPU.

NetScaler VPX supports the following hypervisors: Citrix XenServer 6.2 and 6.5; VMWare ESX, Microsoft Hyper-V Server 2012 and 2012 R2, KVM Linux – (Fedora Core 20, Ubuntu 14.10) NetScaler VPX is also supported on Azure and Amazon AWS. In addition, a NetScaler 1000V virtual appliance runs on Cisco Nexus 1100.

By default, the appliance template downloaded from Citrix is setup to use the minimum system requirements of 20 GB of disk, 2 vCPU, 2 GB of memory and two vNICs.

When NetScaler is running in a virtual environment it is important to remember the performance numbers that is been validated and supported by Citrix on the VPX appliance.

- SSL transaction/sec (2K key certificates) up to 750
- SSL throughput, Gbps up to 1.0
- Compression throughput, Gbps up to 0.75
- SSL VPN/ICA proxy concurrent users up to 1500

Note: these numbers have been taken from the VPX datasheet and might change or adjusted over time. The original datasheet can be found here →

https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/citrix-netscaler-vpx-data-sheet.pdf

Licensing

The system throughput is based upon the type of license we purchase. NetScaler VPX is available in five different editions: VPX-10, VPX-200, VPX-1000, VPX-3000 and VPX-8000 where the number behind VPX- stands for the maximum licensed Mbps throughput.

All NetScaler VPX licenses are available in the same editions as the physical appliances. This includes: Standard Edition (Load Balancing), Enterprise Edition (web application acceleration, monitoring), and Platinum Edition (caching). There's also a NetScaler

Gateway VPX appliance that only does NetScaler Gateway (no Load Balancing) but is licensed for 50 Mbps.

The system throughput is based upon the type of license we choose, NetScaler VPX is available in five different editions. VPX-10, VPX-200, VPX-1000, VPX-3000 and VPX-8000 where the number behind VPX- stands for the about of throughput in terms of Mbps non-SSL, HTTP traffic) it can handle.

Citrix also has a couple free NetScaler VPX licenses:

- NetScaler VPX Express – includes NetScaler Standard Edition features, limited to 5 Mbps, and must be renewed annually.
- NetScaler VPX Developer Edition – includes NetScaler Platinum Edition features and is limited to 1 Mbps. This “free” license is available to all current NetScaler customers.

Note: NetScaler VPX license files are allocated to the MAC address of the virtual machine. If the MAC address changes then the license file is invalidated. This behavior is typically seen on Hyper-V so make sure a static MAC address is allocated to the virtual machine

The throughput is enforced only for traffic inbound to the NetScaler only, regardless of whether this is request traffic or response traffic. So for instance, a VPX-1000 can process both 1 Gbps of inbound traffic and 1 Gbps of outbound traffic at the same time. It is however important to remember that traffic going back to a NetScaler is also considered inbound traffic.

In regards to CPU, it is important that we configure the amount of vCPU based upon the VPX model we have. NetScaler uses something called packet engines, which are each assigned their own vCPU, which does the network processing. In a basic setup with two vCPUs, the first CPU is used for management tasks and the second vCPU is used to process all network traffic and handling all the different features.

CPU Sizing

it is important that we configure the number of vCPUs based upon the VPX model we have. NetScaler uses something called packet engines. In a basic setup with two vCPUs, the first CPU is used for management tasks and the second vCPU is used to process all network traffic and handling all the different features.

VPX 10 and 200 only support only one packet engine CPU, meaning a total of two vCPUs, while for instance VPX 1000 supports having two or three packet engine vCPUs. This allows NetScaler to distribute traffic between the different vCPUs and will allow for better performance and distribution of the traffic between the vCPUs.

The following chart shows the different editions and support for multiple packet engines.

Memory	2 GB	4 GB	6 GB	8 GB	10 GB	12 GB
License						
VPX-10	1	1	1	1	1	1
VPX-200	1	1	1	1	1	1
VPX-1000	1	2	3	3	3	3
VPX-3000	1	2	3	3	3	3
VPX-8000	1	2	3	4	5	5

The number of vCPUs = the number of PEs+1. For example, if you have installed a VPX-3000 license and 6 GB memory is available, to add three PEs you must allocate four vCPUs. If we are running an older version of Hyper-V, this feature might not be supported and we should in that case upgrade to the latest version of Hyper-V to get that support.

If you allocate more than the licensed number of vCPUs, then the extra vCPUs will be ignored and not used as packet engines.

On NetScaler VPX, all SSL operations are performed on main CPU. Higher single-core CPU clock speeds will increase SSL throughput. To increase the number of vCPUs available for SSL operations you'll need VPX-1000 or higher license. Note: Physical NetScaler MPX and NetScaler SDX appliances have dedicated SSL ASICs so they have much higher SSL throughput.

For maximum throughput on NetScaler VPX, use your hypervisor management tools to reserve 100% of the CPU allocated to NetScaler. Also when doing the initial setup do not enable all features, only enable the features you need. Enabling all features will impact the performance of the NetScaler appliance.

CPU Usage on the management CPU and packet CPUs can be seen on the NetScaler using the CLI command

stat system

Other Hypervisor Notes:

- NetScaler VPX does not support hypervisor features like SRV-IO or PCI device pass-through.
- Any hypervisor integration tools on NetScaler VPX should not be updated manually, since Citrix will update them automatically in newer firmware upgrades.
- NetScaler VPX should not be migrated between hypervisor hosts unless absolutely necessary. For example, disable automatic VMware vMotion of NetScaler VPX appliances.

Memory Sizing

NetScaler VPX defaults to 2 GB of RAM, which is sufficient for most packet engine operations. However, the following NetScaler features require more RAM:

- Integrated Caching
- Application Firewall
- Web Interface on NetScaler
- WebFront

Firmware upgrades

Citrix often releases new firmware upgrades, which can be downloaded from their website. It is important to properly read the release notes before during the upgrade process to ensure that there are no major changes that might affect your existing NetScaler solution and to check if there are any particular bug fixes or security vulnerabilities fixes.

Firmware upgrades can also include upgrades to the virtual guest tools which are used by the hypervisor as well, so in case might include support for a newer version of the hypervisor.

The following knowledge article has information about the different release versions and date → <http://support.citrix.com/article/CTX121840>

You should also pay close attention to the support page, which has knowledge articles and security bulletin information on NetScaler. The page can be found here → <http://support.citrix.com/search?prod=NetScaler>

NIC Teaming and LACP

By default, NetScaler VPX comes with two vNICs. If both vNICs are connected to the same VLAN then one of them should be disabled. If both are enabled, then NetScaler will use both interfaces in a round-robin fashion meaning that the MAC address of the SNIP will constantly change depending on which interface is being used for that particular packet.

NetScaler VPX supports up to 10 vNICs, depending on hypervisor capabilities.

If you need Link Aggregation or NIC teaming, it is best to do that at the hypervisor level. We need to understand how the various types of NIC teaming perform traffic processing. Most vendors have good documentation on their NIC teaming features. For example, Microsoft has documented all the different options here:

<http://www.microsoft.com/en-us/download/details.aspx?id=30160>.

For instance, Microsoft Hyper-V has a form of NIC teaming called switch independent mode that allows us to connect a physical host to different switches and does not require us to do any form of configuration on the switches. In this type of NIC teaming, the NetScaler VPX vNIC is bound to only one uplink and its throughput is limited to that one uplink. Important to remember for Hyper-V that we choose HypervPort distribution algorithm.

To go beyond a single uplink, you need a packet hashing algorithm and the switch must be configured to support the hashing. This is typically configured using LACP, which allows for aggregation of bandwidth (incoming/outgoing) and redundancy of NICs, and with setting up MLAG between switches, we can also ensure switch redundancy.

Note: If deploying NetScaler VPX on a Hyper-V environment, make sure that the host NIC drivers are running the latest version, in many cases there have been known issues with VMQ (Virtual Machine Queue) with Broadcom chips which affects the performance badly on virtual machines.

VLAN tagging

For VLAN tagging, you can either do it at the hypervisor level or you can do it inside the NetScaler VPX. At the hypervisor level, you create a VM network (e.g. VMware Port Group) and assign the VLAN tag to that VM network. The hypervisor adds the VLAN tag to all packets egressed from the NetScaler VPX and there's no need to configure any VLAN tagging inside the NetScaler VPX. However, this limits the NetScaler to only one VLAN per vNIC.

Alternatively, you can configure the hypervisor VM network to not touch VLAN tags and let the virtual machine and physical switch handle it. This is sometimes called VLAN trunking. In this case, the NetScaler VPX is responsible for adding all necessary VLAN tags.

If the NetScaler VPX is connected to multiple VLANs, make sure you configure VLANs inside the NetScaler whether the NetScaler is tagging them or not. NetScaler needs VLAN configuration so it knows which subnets go on which network interfaces. Typically, you create a SNIP for each VLAN, create a VLAN object for each VLAN, and bind the VLAN object to the SNIP and Network Interface. The VLAN object can be either tagged or untagged. If the hypervisor is doing the tagging, then leave the VLAN object as untagged.

If the hypervisor VM network is in VLAN trunking mode, then the NetScaler VLAN objects probably need VLAN tagging enabled.

Note: VLAN tagging inside the virtual appliance is only supported on ESX and XenServer.

Jumbo Frames

NetScaler VPX supports the use of Jumbo frames, which allows the appliance to process Ethernet frames with up to 9000 bytes of payload, which allows it to transfer larger files more efficiently than it is possible with the standard MTU size of 1500 bytes. Jumbo frames have the potential to reduce overhead and CPU cycles on the appliance.

Jumbo Frames requires that the infrastructure it is connected to has been configured to support Jumbo Frames as well. Internet connection does not support Jumbo frames since most routers only support the standard MTU, so Jumbo frames are restricted to run only within the datacenter.

Note: Jumbo frames are only supported in NetScaler when it is running on ESX or KVM, and it requires host NIC configuration so that the host NIC MTU is aligned with the appliance

The appliance can operate with jumbo frames in the following scenarios:

- Jumbo to Jumbo. The appliance receives data as jumbo frames and sends it as jumbo frames.
- Non-Jumbo to Jumbo. The appliance receives data as regular frames and sends it as jumbo frames.
- Jumbo to Non-Jumbo. The appliance receives data as jumbo frames and sends it as regular frames.

Jumbo Frames (Maximum Transmission Unit Size) is configured at the network interface and in TCP profiles. To change the MTU for an interface, go to System > Network > Interfaces. Edit an interface and set the MTU.

For TCP based traffic, the default TCP profile will override the interface value we define for Jumbo frames. The default profile *nstcp_default_profile* is bound to all TCP based load-balancing services and in this case, we should change the MSS value within the TCP profile if we want to leverage Jumbo frames. You can change the default TCP settings at System > Settings > Change TCP Parameters. Or go to System > Profiles and edit the *nstcp_default_profile*.

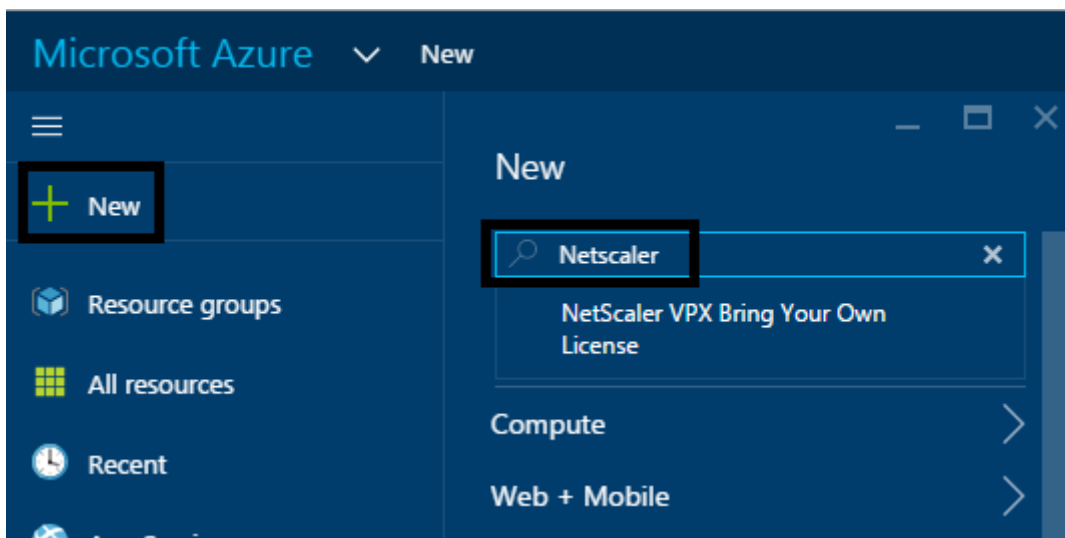
NetScaler deployment in Azure

When deploying Citrix NetScaler from Microsoft Azure its needs to be created using the Azure Marketplace, which consists of a custom firmware from Citrix. Since the networking

capabilities in Azure are still a bit restricted, the Marketplace appliance will have the following limitations.

- Restricted to a single IP address (shared between the NSIP, SNIP and VIP)
- Following ports, which cannot be used for services (Ports 21, 22, 80, 443, 8080, 67, 161, 179, 500, 520, 3003, 3008, 3009, 3010, 3011, 4001, 5061, 9000, and 7000.)
- Following services and features cannot be used (IPv6, Gratuitous ARP (GARP) ,L2 Mode, Tagged VLAN, Dynamic Routing, Virtual MAC (VMAC), USIP, GSLB and Cloudbridge Connector)
- Only VPX 10, VPX 200 and VPX 1000 are supported.

After logging into the Azure Portal, click New. Type NetScaler in the search window

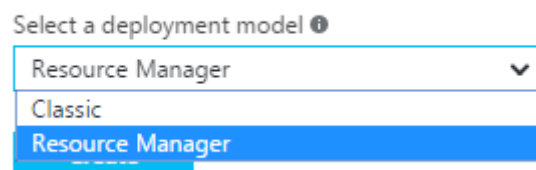


Then choose «*Netscaler VPX Bring your Own License*»

Note: You need an active subscription on Azure in order to be able to deploy an instance.

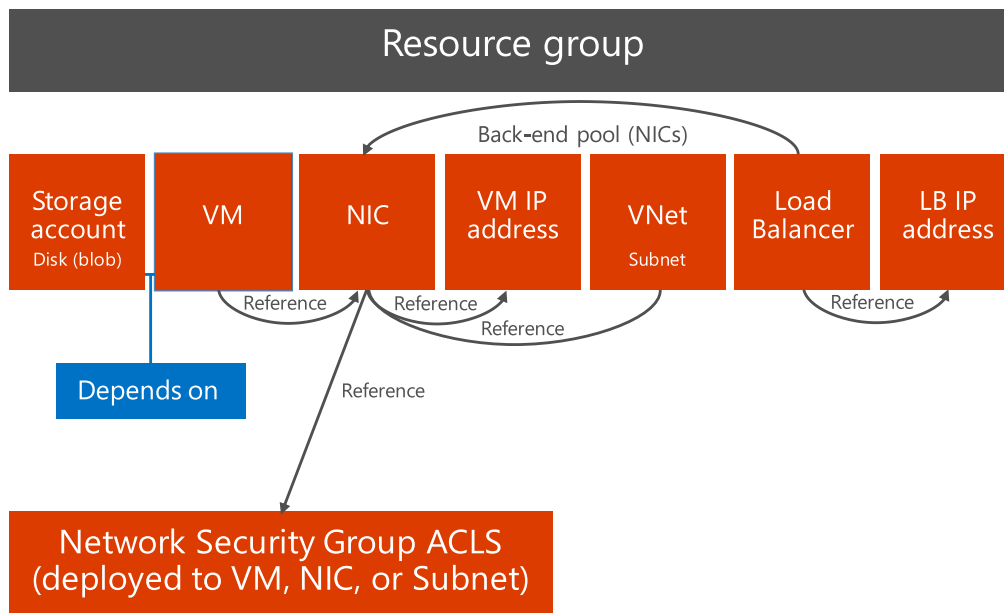
Then choose the object and select deploy.

By default NetScaler will be deployed using a provisioning engine called Azure Resource manager.



We can select if we want to use the older deployment option called Classic that is also known as Azure Service Management. If you have an existing Azure infrastructure

deployed which was provisioned using the old Azure portal, then this is most likely using the Service Management APIs, if you want the VPX appliance to be able to communicate with this environment you need to choose Classic mode before clicking Create. This is because the resources created in Azure Resource Manager cannot directly communicate/integrate with resources provisioned using Service Management. The rest of the book is going to focus on use of Resource manager as the deployment method.



Before we can finish the deployment we need to configure some specific parameters:
Name: Name of the virtual appliance

User name: Specify a user name which will be used to login. Do not use `nsroot/admin/root` as a username here.

Authentication type: Either password or SSH Public Key. If we specify Password, we need to specify a password afterwards.

Subscription: Specify the Azure subscription, which this resource should use.

Resource Group: Enter the name of a new resource group to create a new one, or choose select existing

Location: Specify which region you want this resource + resource group created. If you want to setup the appliance in an existing resource group, you must first choose the location where the resource group is located before it can be selected from the list.

After you are done click OK.

Next we are required to choose a size of the virtual appliance. This specifies what kind of resource that are available to the VPX. By default, the A2 size is sufficient for VPX 10 and

200. If you are planning to use VPX 1000 in Azure, you should consider moving it up to A3 because of the use of multiple packet engines. After you have selected a size click on the SELECT button.

Lastly, we need to define storage options and network options. When creating a new resource group all of these parameters will be set up automatically and it will create new groups.

Virtual Network is a logical network within Azure. Here we can define the IP-range that we want to use. We can add many virtual machines to the same virtual network. Within the virtual network, we can create a subnet. A virtual network can have multiple subnets.

We also have a public IP address, which by default is randomly assigned from a pool of public IP addresses that Microsoft has assigned. We should change this to a static address in case we can to restart/shutdown the virtual appliance, since Microsoft can assign another public IP is the resource is deallocated. We also get a Network Security Group, which is a list of ACLs, which span across the public IP address, virtual machines and virtual networks. By default, only port 22 is open to the appliance, which will available using a public IP address after the provisioning is done. We could also define a new inbound rule under the network security group to allow for inbound connections against port 80 to allow access to the web management portal to do the initial configuration as well.

The screenshot displays the configuration options for a new virtual machine in Azure. It is organized into three columns:

- Storage:**
 - Disk type: Standard (selected), Premium (SSD)
 - * Storage account: (new) nsdemo9142
- Network:**
 - * Virtual network: (new) NSDEMO
 - * Subnet: default (10.6.0.0/24)
 - * Public IP address: (new) NSVPX
 - * Network security group: (new) NSVPX (highlighted)
- Monitoring:**
 - Diagnostics: Disabled, Enabled (selected)

In the center, there are options to 'Create new' resources or select existing ones like 'maskda wqeqwqwe'.

On the right, the 'Network security group' configuration is shown:

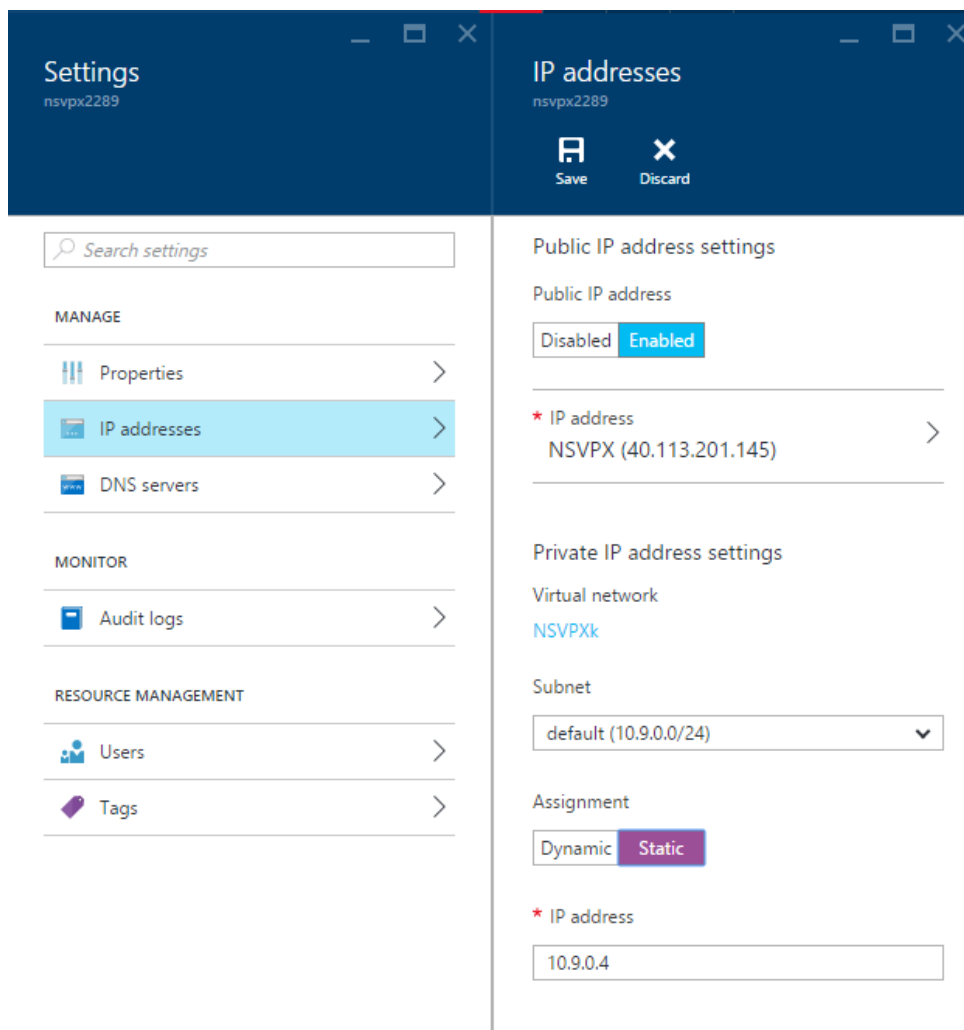
- * Name: NSVPX
- Inbound rules:
 - 1000: default-allow-ssh
 - Any SSH (TCP/22) (checked with a green checkmark)
 - + Add an inbound rule
- Outbound rules:
 - No results.
 - + Add an outbound rule

NOTE: The default username to log on to the NetScaler virtual machine is nsroot. However, the default password is set to the "deployment ID" that is generated after the

virtual machine is provisioned. You can change the password after you log on to the instance.

The password for the user that is created during the virtual machine provisioning remains the same as that you had provided.

Another important change to make before configuring the appliance, is to change from dynamic to static IP address on the VPX.



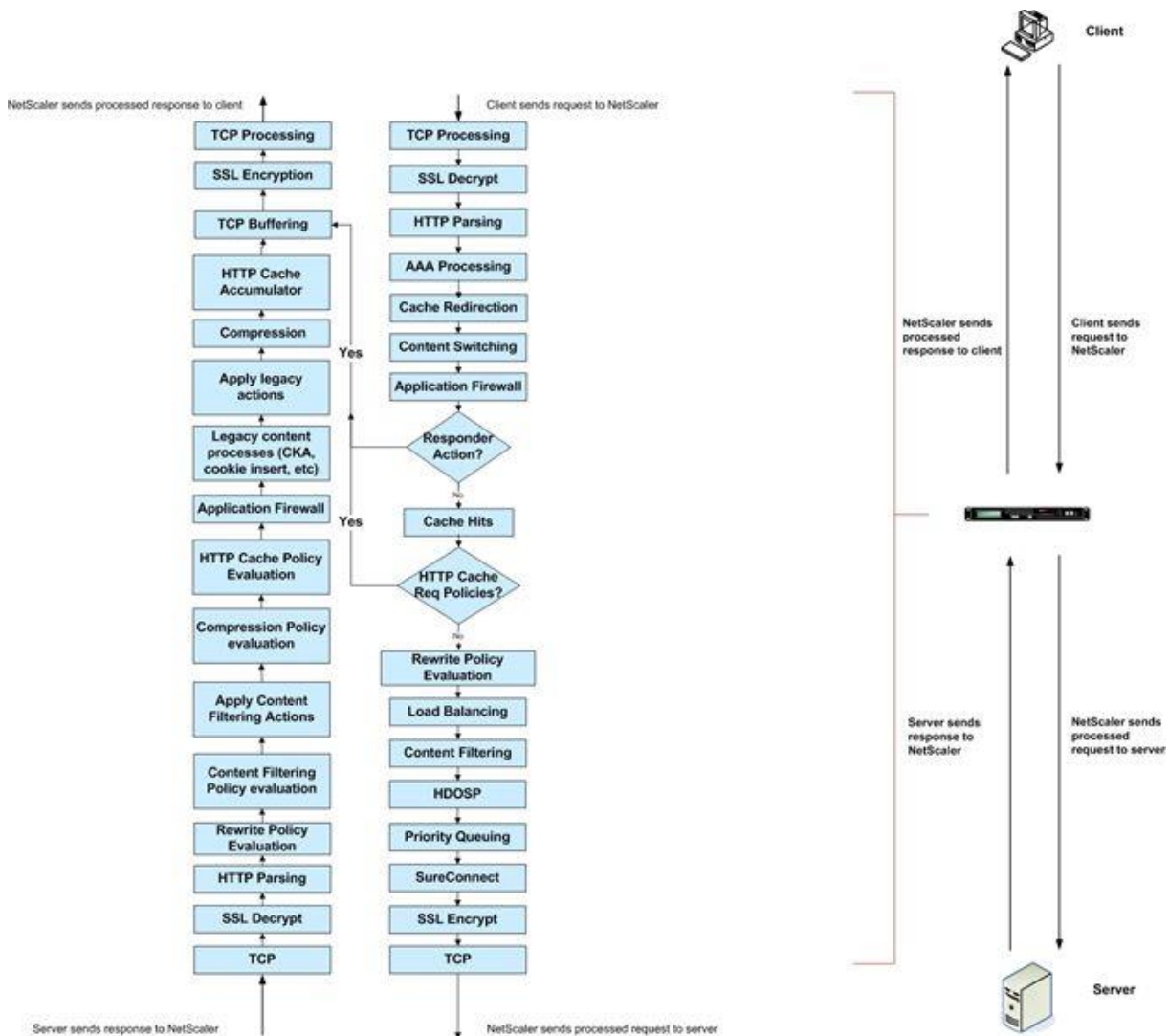
By default, this is set to dynamic, so in case of maintenance or unplanned events the appliance might restart and get another IP address from the subnet range that it is placed into. Also change any DNS settings

After you are done with the deployment, and logged into the management portal the first run wizard will appear. Remember that the wizard will ask for you to enter a SNIP address, but this is not supported in Azure so you can skip that part of the wizard.

Note: Do not try to update the firmware to a newer version, Citrix does not support this and the only way to update is to wait until Citrix releases a newer version of the image in the marketplace and a new custom firmware update.

NetScaler packet flow

In order to optimize performance, we need to have an understanding on the way NetScaler processes packets:



Source: <http://support.citrix.com/article/CTX135254>

TCP Profiles

Most of the traffic that is going through the NetScaler uses TCP protocol. TCP is a reliable protocol, which first consists of a three-way handshake to establish a connection and then consists of using **ACK** and **SEQ** to maintain packet flow and a reliable connection.

In default on a NetScaler, all TCP traffic (TCP services) follows a set of rules that are defined within the TCP Profile called *nstcp_default_profile*.

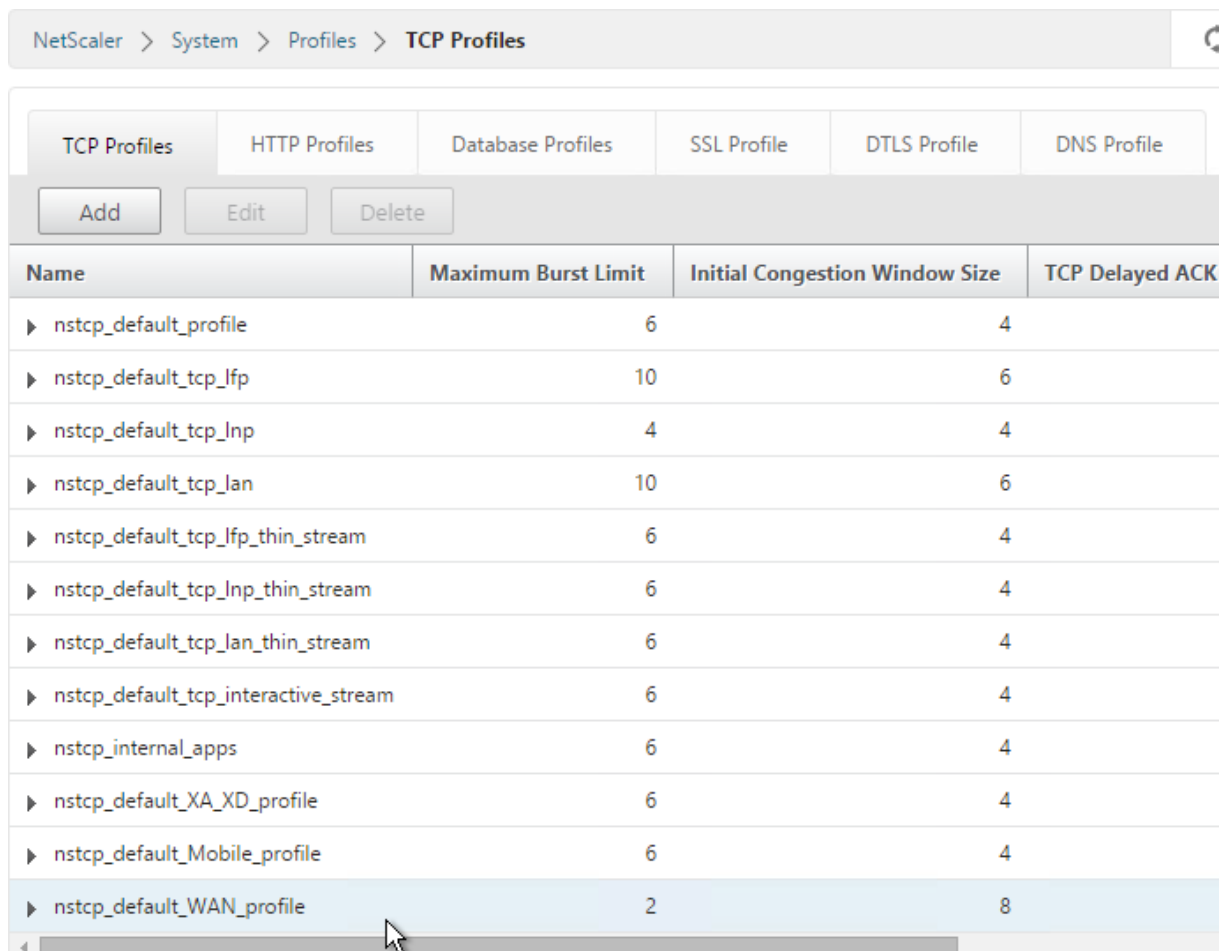
TCP has many different parameters that we can define or change, but not in the default TCP Profile. This is because NetScaler is by default configured to be able to fit into most environments and most networks. Many of the TCP options might give a performance boost, or they might also degrade performance if not properly configured or if you are unsure about the different options.

All other TCP settings should be configured in TCP Profiles so you can apply different TCP settings to different virtual servers and different services. For instance, TCP traffic should behave differently for a service containing a website for mobile devices, then for a backend service located on a fast Ethernet LAN connection.

A couple settings should be enabled in `nstcp_default_profile` on every NetScaler. These settings are disabled by default.

- Window Scaling (leave default scaling factor of 4)
- Selective Acknowledgement

Details about these parameters is described on a later page.



Name	Maximum Burst Limit	Initial Congestion Window Size	TCP Delayed ACK
▶ nstcp_default_profile	6	4	
▶ nstcp_default_tcp_lfp	10	6	
▶ nstcp_default_tcp_lnp	4	4	
▶ nstcp_default_tcp_lan	10	6	
▶ nstcp_default_tcp_lfp_thin_stream	6	4	
▶ nstcp_default_tcp_lnp_thin_stream	6	4	
▶ nstcp_default_tcp_lan_thin_stream	6	4	
▶ nstcp_default_tcp_interactive_stream	6	4	
▶ nstcp_internal_apps	6	4	
▶ nstcp_default_XA_XD_profile	6	4	
▶ nstcp_default_Mobile_profile	6	4	
▶ nstcp_default_WAN_profile	2	8	

By default, NetScaler contains the following built in TCP profiles:

- **nstcp_default_tcp_lfp**: This profile is useful for long fat pipe networks (WAN) on the client side. Long fat pipe networks have long delay, high bandwidth lines with minimal packet drops.
- **nstcp_default_tcp_lnp**: This profile is useful for long narrow pipe networks (WAN) on the client side. Long narrow pipe networks have considerable packet loss occasionally.
- **nstcp_default_tcp_lan**: This profile is useful for backend server connections, where these servers reside on the same LAN as a NetScaler appliance.
- **nstcp_default_tcp_lfp_thin_stream**: This profile is similar to the `nstcp_default_tcp_lfp` profile. However, the settings are tuned for small packet flows.
- **nstcp_default_tcp_lnp_thin_stream**: This profile is similar to the `nstcp_default_tcp_lnp` profile. However, the settings are tuned for small packet flows.
- **nstcp_default_tcp_lan_thin_stream**: This profile is similar to the `nstcp_default_tcp_lan` profile. However, the settings are tuned to small packet flows.
- **nstcp_default_tcp_interactive_stream**: This profile is similar to the `nstcp_default_tcp_lan` profile. However, it has a reduced delayed ACK timer and ACK on PUSH packet settings.
- **nstcp_internal_apps**: This profile is useful for internal applications on a NetScaler appliance. This contains tuned window scaling and SACK options for the required applications. This profile should not be bound to services other than internal services.
- **nstcp_default_XA_XD_profile**: This profile is aimed specifically for ICA connections and should only be used on the NetScaler Gateway virtual server.
- **nstcp_default_mobile** : This profile is aimed at mobile service and uses another congestion algorithm which is specifically aimed at mobile connections using 4G and such.

Now the principle behind the built-in TCP profiles is that we do not need a deep understanding of TCP and the different parameters, and that we can just assign one of the built-in profiles to a service based on what kind of network we have.

On the other hand, if we want to customize our own TCP profile we need to know the different attributes within the profile. Within a profile, we have the following attributes:

- **Window Scaling** is a TCP option that allows the receiving point to accept more data than allowed in the TCP RFC for window size before getting an acknowledgement. By default, the window size is set to accept 65,536 bytes. With Window Scaling enabled, it bitwise-shifts the window size so the window size is increased. This is an option that needs to be enabled on both endpoints in order to be used, and will only be sent in the initial three-way handshake.
- **Selective Acknowledgement (SACK)** is a TCP option that allows for better handling of TCP retransmission. In the scenario where two hosts communicate with SACK not enabled and suddenly the receiver does not receive some out of order packets, then the receiver acknowledges the last in-order packet it received and the sender has to resend all later packets, even if they've already been received (out of order). With SACK enabled, the receiver will notify the sender of only the specific packets it is missing. This allows for faster communication recovery since the sender does not need to resend all the packets.
- **Forward Acknowledgement (FACK)** is a TCP option that works in conjunction with SACK and helps avoid TCP congestion by measuring the total number of data bytes outstanding in the network. Using the information from SACK it can more precisely calculate how much data it can retransmit.
- **Nagle's algorithm** is a TCP feature that tries to cope with small packet problems. Applications such as Telnet often send each keystroke within its own packet, creating multiple small packets containing only 1 byte of data, which results in a 41-byte packet for one keystroke. The algorithm works by combining a number of small outgoing messages into the same message, thus avoiding overhead. ICA is a protocol that operates by sending many small packets, which might create congestion on the network; this is why Nagle is enabled in the *nstcp_default_XA_XD_profile*. In addition, since many might be connecting using 3G or Wi-Fi, which might in some cases, be unreliable when it comes to changing channel, we need options that require the clients to be able to reestablish a connection quickly and that allow the use of SACK and FACK.

NOTE: Nagle might have negative performance on applications that have their own buffering mechanism and operate inside the LAN. Since ICA-proxy mostly uses TCP, except for Framenhawk traffic using DTLS, this should be enabled for NetScaler Gateway vServers. If we take a look at another profile such as *nstcp_default_lan*, we can see that

FAACK is disabled; this is because the resources needed to calculate the amount of outstanding data in a high-speed network might be too much for the CPU to handle.

- **Maximum Burst Limit:** This setting controls the burst of TCP segments on the wire in a single attempt. A higher limit here ensures faster delivery of data in a congestion-free network. Limiting bursts of packets helps to avoid congestion.
- **Initial Congestion Window size:** Initial congestion window defines the number of bytes that can be outstanding in the beginning of a transaction. The default size is 4 (that is $4 \times \text{MSS}$). • **TCP Delayed ACK Time-out (msec):** To minimize the number of ACK packets on the wire, this NetScaler feature by default sends ACK only to a sending node if the NetScaler receives two data packets consecutively or the timer expires; the default timeout is 200 ms.
- **Maximum ooo packet queue size:** This feature allows out-of-order packets in TCP streams to be cached in system memory and reassembled before NetScaler processes them. By default, the value is 64; we can define a value of 0 that means no limit but this will put a lot of strain on the NetScaler memory usage.
- **MSS and Maximum Packets per MSS:** With this setting, we can specify the maximum segment size to be used for TCP transactions. It is important to note that jumbo frames will override this setting, since MSS is TCP sizes and MTU is IP packets, which is lower in the network layer. Therefore, if we specify a higher MTU size on the interface by enabling jumbo frames the MSS size will increase as well.
- **Maximum Packets Per Retransmission:** This setting controls how many packets are retransmitted in a single attempt. This is used with TCP Reno that used a partial ACK value to notify NetScaler that it has received some of the packets but not all.
- **TCP Buffer Size (bytes):** The buffer size is the receiver buffer size on the NetScaler. The buffer size is advertised to clients and servers from NetScaler and it controls their ability to send data to NetScaler. The default size is 8K and in most cases, it will be beneficial to increment this when communicating with internal server farms.

NOTE: There is a good example of tuning of TCP settings from AOL located here on Slideshare → <http://www.slideshare.net/masonke/net-scaler-tcpperformancetuningintheaolnetwork>. Also see Citrix Knowledgebase article CTX121149 *Recommended Settings and Best Practices for Generic Implementation of a NetScaler Appliance* at <http://support.citrix.com/article/CTX121149>

Another important configuration piece in the TCP profile is the congestion algorithm, which specifies how the traffic flow should react when congestion occurs.

The following congestion algorithms are available in NetScaler:

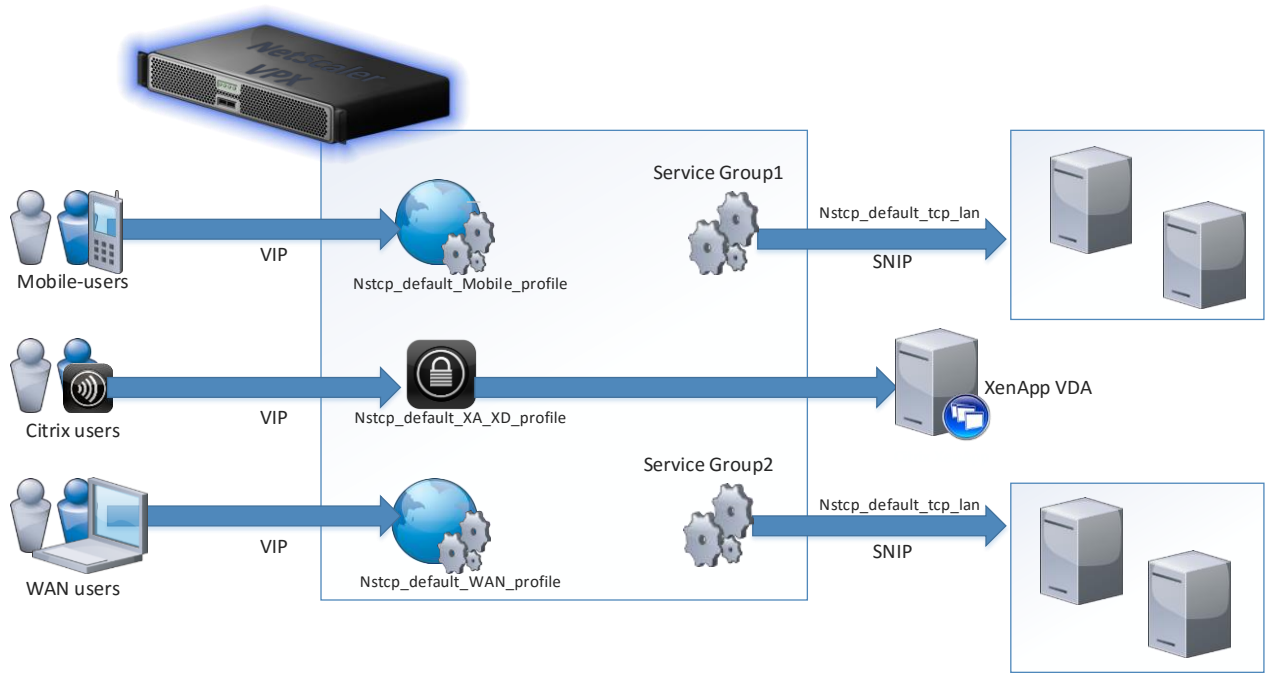
- Default (based upon TCP Reno)
- Westwood (based upon TCP Westwood+)
- BIC
- CUBIC
- Nile (based upon TCP-Illinois)

Westwood is aimed at 3G/4G connections, or other slow wireless connections. BIC is aimed at high-bandwidth connections with high latency, such as WAN connections. CUBIC is almost like BIC but not as aggressive when it comes to fast-ramp and retransmissions. Note that CUBIC is the default TCP algorithm in Linux kernels from 2.6.19 to 3.1. Nile is a new algorithm created by Citrix and was introduced in NetScaler 11, which is based upon TCP-Illinois. This algorithm is targeted at high-speed, long-distance networks. It achieves higher throughput than standard TCP and is also compatible with standard TCP.

Therefore, now we can choose that congestion algorithm which is better suited for particular services. For instance, if we have a service that serves content to mobile devices, we could use the `nstcp_default_mobile_profile` TCP profile, which uses the Westwood congestion algorithm. Westwood avoids working on the fixed parameters where we cut the congestion window by half, rather it follows the heuristic model to track the estimated bandwidth on client side and accordingly updates the congestion window

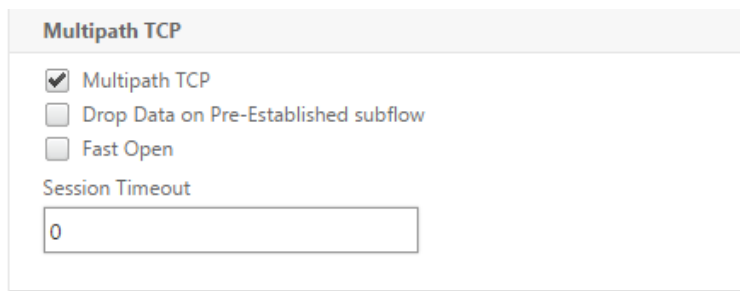
TCP Profiles, as mentioned earlier can be attached to a virtual server (incoming traffic), and to a service or service group (backend traffic) except for a NetScaler Gateway virtual server, where we can only specify a TCP Profile to the virtual servers, since there is no service attached to a Gateway virtual server.

The picture below, shows some examples of how we can use TCP profiles for different services



There are also some other interesting parameters in the TCP profile. One of these parameters is Multipath TCP. This feature permits endpoints that have multiple paths to a service, typically a mobile device that has WLAN and 3G capabilities, allowing the device to communicate with a Virtual Server on a NetScaler using both channels at the same time. This requires that the device support communication on both methods and that the service or application on the device support Multipath TCP (MPTCP).

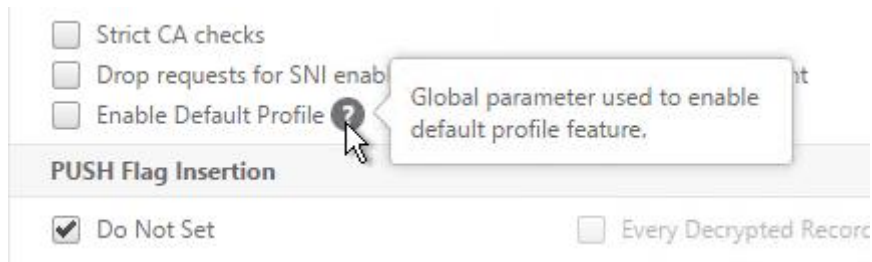
Enabling MPTCP is done in the TCP Profile



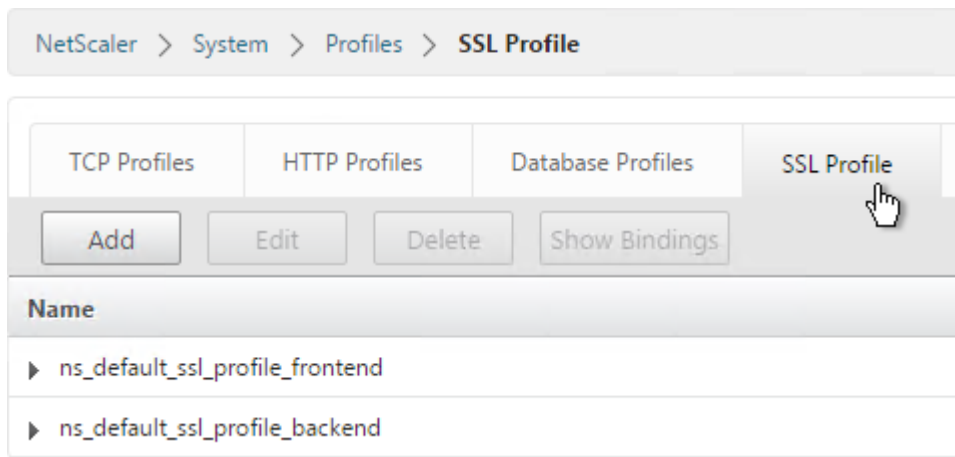
SSL Profiles

SSL Profiles work similar to TCP profiles in that you can bind different SSL profiles to different SSL Virtual Servers (including NetScaler Gateway) and different SSL services.

NetScaler 11.0 build 64 and newer provides significant enhancements for SSL Profiles. Prior to NetScaler 11.0 build 64 it was not possible to change SSL ciphers globally. Now you can now enable the default SSL Profile and change SSL settings globally by editing the default front-end or default back-end SSL profiles. To enable the default SSL Profiles, go to Traffic Management > SSL, and on the right is Change Advanced SSL Settings. Scroll down and check the box next to **Enable Default Profile**.



To create or edit SSL profiles, go to System > Profiles. On the right, switch to the SSL Profile tab. You will see the built-in default profiles. In 11.0 build 64 and newer, edit the default profiles to apply SSL settings globally. Or you can create new SSL profiles and bind them to specific SSL Virtual Servers and SSL services.



The Basic Settings section of an SSL Profile lets you disable SSLv3 and maybe disable TLSv1.

If you configured a Virtual Server for SSL Offload (SSL on front-end, HTTP on back-end), you might have to go to the Basic Settings section and check the box for SSL Redirect. With this option enabled, any 301 or 302 redirect sent from the web server will be rewritten with an https Location field instead of an http Location field. Example web applications that require this setting are Citrix StoreFront and Citrix Web Interface.

The SSL Ciphers section lets you bind a more secure Cipher Group instead of the Default Cipher Group, which contains RC4 ciphers. In addition, ECDHE ciphers will not work without ECC curves. See Citrix Blog Post *Scoring an A+ at SSLlabs.com with Citrix NetScaler (the sequel)* at <https://www.citrix.com/blogs/2015/05/22/scoring-an-a-at-ssllabs-com-with-citrix-netscaler-the-sequel/> for some recommended SSL ciphers.

There are also some more advanced parameters within the SSL profile that we can adjust depending on what kind of service we are delivering via NetScaler.

- Quantum size
- PUSH Encryption Trigger
- Enable DH PARAM

The Quantum size defines how much data in Kilobytes should be processed before it is encrypted. By default, this value is 8,192 KB; if we are hosting a service that provides media download, for instance, or large files in general, we might benefit if we change this to a larger quantum size.

PUSH Encryption Trigger value is used to tell NetScaler how long it should wait before consolidating the data and encrypting it. For instance, ICA-proxy sessions have the PSH flag set, which means that NetScaler should always forward encrypted traffic immediately; if we set this value to 5ms, NetScaler will gather data for 5ms before sending it out back to the client. By default, this value is set to 1ms; in an environment with many ICA-proxy sessions the network might be congested because of the high amount of small packets that need to be encapsulated.

Enable DH PARAM enables NetScaler to regenerate a new Diffie-Hellman private/public pair after a number of transactions. This feature needs to be enabled and defined to, for instance, a value of 1000; we also need to bind it with a DH key. This will make NetScaler regenerate a new key pair after 1,000 sessions to ensure that a malicious attacker cannot use a compromised DH key to get access to information.

It is also important to remember that fact that depending on the ciphers we choose, this will also affect performance. For instance, RSA is faster than DH, but it's less secure. RC4 encryption is faster than AES 256 but is less secure.

VPX SSL limitations

Unfortunately, NetScaler VPX today does not possess the same SSL cipher and protocol capabilities as physical NetScaler MPX and NetScaler SDX appliances. For example, there is no support for GCM/SHA2 cipher suites on NetScaler VPX. In addition, there is no support for TLS 1.1/1.2 to back-end SSL services. See Citrix Knowledgebase article CTX201710 *Cipher/Protocol Support Matrix of NetScaler Appliances* at <http://support.citrix.com/article/CTX201710> for more information.

Mobilestream

Mobilestream is a set of different features on NetScaler that does HTTP compression, caching and frontend optimization.

- **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 quicker fetching of data.

- **Frontend optimization:** This allows NetScaler to perform smart data changes to web content such as CSS, PNG, JS, HTML, and such. For instance, it can resize and convert image based upon endpoint connecting.

All these features allow for the endpoint to be able to reach the content faster, and process it more quickly on the endpoint.

Note: Compression and Frontend Optimization require NetScaler Enterprise Edition or Platinum Edition. Caching requires NetScaler Platinum Edition or there's an add-on for NetScaler Enterprise Edition.

Compression

The compression feature enables a NetScaler virtual server to compress HTTP data that is going to or from the client. Another benefit of this feature is that the HTTP compression algorithm scrambles the data going from the client to the server which does not make it completely readable in a MiTM scenario, but should not be a replacement for SSL/TLS. HTTP Compression on NetScaler is based upon the GZIP and DEFLATE algorithms. HTTP compression feature of NetScaler will compress data within 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.

By default, there are five built-in global policies, each of which has an action attached to it. The policies are explained as follows:

- **ns_nocmp_xml_ie:** This policy does not compress when a request is sent from Internet Explorer. The content type is either text or XML.
- **ns_nocmp_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.

Compression has some global settings we should take a closer look at before we configure custom policies. [Optimization](#) | [HTTP Compression](#) | [Settings](#) | [Change](#)

Compression Settings. Here we have the following settings

- **Quantum Size:** The amount of data that has to go through before NetScaler starts to compress data. The default value here is 57,344 or 57 KB.
- **Compression Level:** What level NetScaler should compress data. Best performance equals less compression and less CPU usage. Best compression equals more CPU usage, but allows for much less bandwidth usage. This is by default set to optimal which corresponds to a GZIP level of 5-7.
- **Minimum HTTP Response Size:** 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:** 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 enabled, allows the backend web servers to enable compression. If disabled, 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 and NetScaler will instead perform the compression. By offloading compression to the NetScaler, the web server is freed up to perform more tasks that are important.
- **Compress Push Packet:** If enabled, when NetScaler receives a TCP packet with the PUSH flag enabled, enables NetScaler to compress what it has already received without waiting until it reaches the full quantum size.

Most of these settings to be left at default, but this is of course depending on what type of service you want to deliver using NetScaler. For instance, if we have a pretty idle NetScaler

delivering a lot of web content we might consider changing the compression level and what size the content should be in order to even optimize further the content.

When creating a compression policy, the setup is pretty simple. Create a Compression policy, specify a valid expression and then apply a compression action. Here we have the following options GZIP, Deflate, No Compress, Compress (The last value compresses data depending on the request, but it prefers GZIP if nothing else is specified)

After we have created a policy, we can then bind it to a Virtual Server. If we want to verify that the policy is taking effect, the simplest way is to use Wireshark and check the HTTP stream



```

Follow TCP Stream (tcp.stream eq 1)
Stream Content
Request
GET
Host:
Connection: keep-alive
Accept: */*
User-Agent: Mozilla/5.0 (windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.103 Safari/537.36
Referer:
Accept-Encoding: gzip, deflate, sdch
Accept-Language: nb-NO, nb; q=0.8, no; q=0.6, nn; q=0.4, en-US; q=0.2, en; q=0.2

Response
Transfer-Encoding: chunked
Connection: keep-alive
Keep-Alive: timeout=15
Last-Modified: Tue, 19 Jan 2016 11:33:03 GMT
Content-Encoding: gzip
  
```

Caching

NetScaler Platinum Edition supports static and dynamic caching for HTTP, MySQL and MSSQL content. When caching is enabled, it automatically caches all static content with no additional configuration. Dynamic content must be configured manually.

When you enable caching you must specify a maximum amount of memory. The more memory allocated to the NetScaler VPX, the more memory you can assign to the caching feature. This is one reason for giving the NetScaler VPX appliance more than 2 GB of RAM.

Cache Global Settings

Memory Usage Limit (MB)
0

Active Memory Usage Limit (MB)
16

Maximum value of Memory Usage Limit (MB)
786

NOTE: the default global memory limit is set to zero. Therefore, even if Integrated Caching is enabled, the NetScaler appliance does not cache any dynamic objects. You must explicitly set the global memory limit when integrated caching is enabled.

If you use the built-in wizards to configure NetScaler Gateway, then the appliance will attempt to cache static content. This sometimes interferes with administrator attempts to change the appearance of the NetScaler Gateway webpages since the cached content group needs to be invalidated before the modified NetScaler Gateway webpages can be downloaded.

It is also important that the NetScaler and the backend services be time-synced properly with an NTP source, since this ensures that content and caching works better so it does not store expired data in the cache.

Configuring caching consists of setting up the following:

- A policy
- An action
- A content group

So for instance within a policy we define what kind of rules that need to be evaluated and the action decided if the content should be cached or not, and if the content is to be cached it will be placed in a content group.

NOTE: If we have a NetScaler with multiple packet engines, all the PEs have access to the cache and are aware of the content that is stored there via a shared memory table.

An example policy to cache PDF from HTTP virtual servers in a content group

Configure Cache Policy

Name

Actions*

Store in Group

Undefined-Result Action*

Expression*

`HTTP.REQ.URL.ENDSWITH("pdf")`

Front-end optimization

Front-end optimization is a feature which allows the NetScaler to optimize web content before sending it back to the requesting endpoint.

Front-end optimization has the following features:

JavaScript

- ° Make inline
- ° Minify

Image

- ° Shrink to attributes
- ° Convert GIF to PNG
- ° Lazy load
- ° Make inline
- ° Optimize
- ° Convert to WEBP
- ° Convert to the JXR format

CSS

- ° Make inline
- ° Combine

- Convert imports to links
- Minify
- Image inline

HTML

- Remove comments from HTML

Miscellaneous optimization

- Domain sharding
- Extent Page Cache

Most of these settings are self-explanatory but some require a bit more understanding for instance, image lazy loading.

When loading a website on a mobile device, it is common that the mobile device browser loads the entire website into buffer before the website is displayed for the user. For a website containing multiple images, it might take some time, and the user might not even be interested in looking at all the images on the page.

Image lazy load is a JavaScript feature that allows the images to be loaded to the client when the user is scrolling down the webpage.

Domain sharding is a feature that allows us to overcome the limitations of the HTTP 1.1 protocol. By default, most web browsers can only open a certain number of connections to the same domain—domain sharding tries to overcome this limitation by splitting the connections on multiple domains.

It is also important that we have the caching feature enabled and working if we want to use front-end optimization. When we have a frontend policy applied, NetScaler will start parsing the content of a request, create an entry in the cache, apply the frontend policy, and then store the optimized content in the cache entry.

To setup frontend optimization we must create a policy that contains an expression, and an action which defines what kind of features that we want to trigger. NetScaler, by default has five built-in policies, BASIC, MODERATE, AGGRESSIVE, IMG_OPTIMIZE and NONE. All these policies have the different actions set that was described earlier. We can also specify our own actions which can then be used against a policy.

There are also some global FE settings that should be taken a closer look at before creating policies and settings. These can be found under Optimization...Front End Optimization...Change Front End Optimization Settings.

Front End Optimization Settings

Cache Maximum Expiry Period (Days)
30

JPEG Quality Percentage
75

Inline CSS Threshold Size (Bytes)
2040

Inline JavaScript Threshold Size (Bytes)
2040

Inline Image Threshold Size (Bytes)
2040

OK Close

If you are serving a lot of JPEG pictures you have the option to reduce the JPEG quality for instance, which would reduce the file size of all pictures being delivered to an endpoint.

HTTP/2 & SPDY

More and more of the different services on the internet are moving towards replacing the traditional HTTP protocol, and moving towards the use of HTTP/2 (SPDY v4) You can read the RFC on HTTP/2 protocol here → <https://tools.ietf.org/html/rfc7540>

HTTP2 and SPDY are both binary protocols, unlike HTTP/1, which is a textual protocol. HTTP/2 Protocol includes, many enhancing changes improves the performance, especially for clients connecting over a mobile connection.

To enable HTTP2 or SPDY, define it in a HTTP profile, and then attach the HTTP profile to a virtual server. If we enable both HTTP2 and SPDY, then NetScaler will prioritize the protocol selection as shown below:

- HTTP/2 (if enabled in the HTTP profile)
- SPDY (if enabled in the HTTP profile)
- HTTP/1.1

Note: The HTTP/2 functionality is supported on the NetScaler MPX and SDX models. For NetScaler VPX, the HTTP/2 functionality is not supported yet, hence the option is greyed out, however SPDY is fully supported.

To enable SPDY in a HTTP profile, go to System | Profiles | HTTP Profiles and either create new profile or edit an existing one. Choose ENABLED under the SPDY parameter to enable SPDY v2 and v3 and allow the client to negotiate between the two.

The screenshot shows the configuration interface for SPDY and HTTP/2 settings. The SPDY section is expanded, showing the following configuration:

- SPDY*: **ENABLED**
- Maximum Header Length: **24820**
- Client IP Header Expression: **Expression Editor** (with dropdowns for Operators, Saved Policy Expressions, and Frequently Used Expressions, and a Clear button)
- Press Control+Space to start the expression and then type '.' to get the next set of options
- Evaluate button

The HTTP/2 section is collapsed, showing the following configuration:

- HTTP/2
- HTTP/2 Header Table Size: **4096**
- HTTP/2 Initial Window Size: **65535**
- HTTP/2 Maximum Concurrent Streams: **100**
- HTTP/2 Maximum Frame Size: **16384**
- HTTP/2 Maximum Header List Size: **24576**

Then we can attach the newly created HTTP profile to a virtual server.

Tuning for ICA Traffic

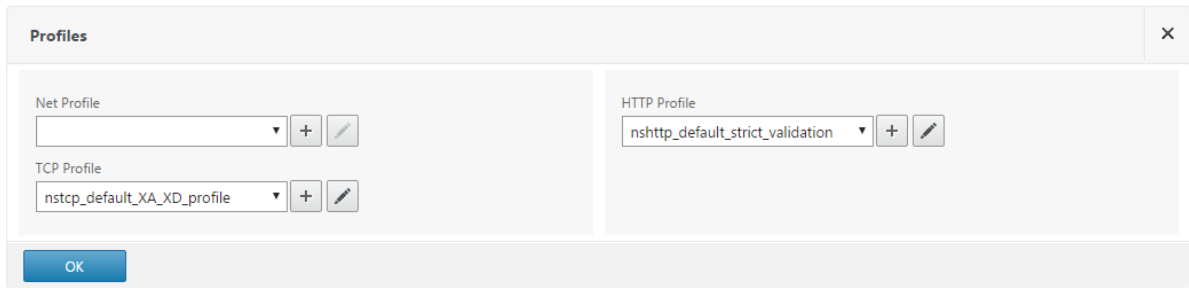
There are two connections from any Citrix Receiver client: HTTP (TLS encrypted) and ICA. HTTP is used for authentication, icon enumeration, and icon launching. All traffic to Citrix StoreFront is HTTP protocol.

Once a published application or desktop is launched using Citrix Connection Manager, a second connection will be established using the ICA protocol

For users that are on the internal network, ICA traffic goes directly to the Citrix Virtual Delivery Agent's private IP address and bypasses the NetScaler.

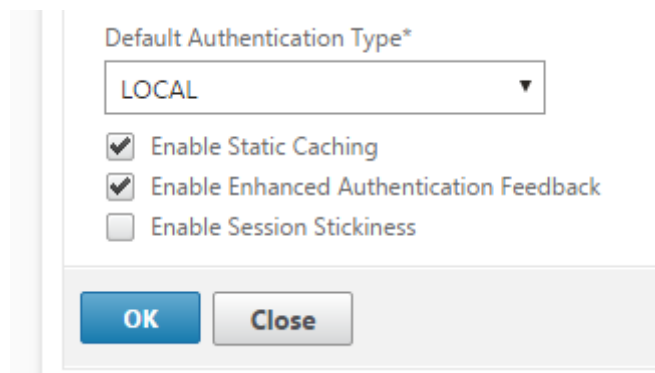
For users that are external to the network, ICA traffic must be SSL-proxied through NetScaler Gateway. This is because it is not typically possible for external users to connect to the private IP address of the Virtual Delivery Agent. Instead, all ICA traffic is proxied through a NetScaler Gateway Virtual Server in the DMZ and only this one VIP needs to be NAT'd and exposed to the Internet.

To optimize ICA traffic flowing through a NetScaler Gateway, simply bind the **nstcp_default_XA_XD_profile** TCP profile to the NetScaler Gateway Virtual Server. This TCP profile contains TCP settings recommended for the small packet nature of the ICA protocol.



If we for instance have a remote location which suffers from high-latency, we should consider creating a new XA_XD TCP profile using BIC congestion algorithm since it is much more efficient in high-latency scenarios.

Another thing to think about authentication piece, by default if a user's tries to login with wrong user or password, the user will get a default generic message stating "Authentication Failed" it is possible to enable a feature called Enhanced Authentication Feedback which gives an error code which then can be translated into a specific error. The feature is configured under Global AAA settings under NetScaler Gateway



The specific error codes can be found here → <http://bit.ly/1RoIH7N>

NetScaler Gateway SSL encrypts and decrypts all ICA traffic. On NetScaler VPX, this SSL encryption is performed on main CPU. Citrix estimates around 700 concurrent sessions on a single CPU core (VPX-200). Alternatively, more users are supported with higher VPX licenses that can use more CPU cores. Note: the smallest physical NetScaler can handle 5,000 concurrent sessions due to its dedicated SSL ASICs. More information on the physical NetScaler models can be found here → https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/netscaler-data-sheet.pdf

The SSL settings on the NetScaler Gateway should be configured to achieve an A+ at SSLLABS.com. These settings include:

- Disable SSLv3.
- Use only SHA2 certificates.

- Make sure the Gateway certificate is linked to any intermediate Certificate Authority certificates.
- Bind modern secure ciphers, especially ECDHE GCM ciphers at the top of the list.
- Use NetScaler Rewrite policy to enable Strict Transport Security.

Using this simple Rewrite Policy, we can enable Strict Transport Security

Create a new rewrite action with the following values:

- **Name:** Namefortheaction
- **Type:** INSERT_HTTP_HEADER
- **Header Name:** Strict-Transport-Security
- **String Expression:** "\max-age=157680000\""

NetScaler VPX Enterprise Edition and Platinum Edition can also produce AppFlow (similar to NetFlow) records for ICA traffic flowing the appliance. These AppFlow records can be sent to an AppFlow collector for further analysis. Citrix provides Insight Center as an AppFlow Collector. Insight Center can display several networking characteristics of the ICA protocol including: datacenter latency, WAN latency, ICA round trip time, jitter, session bandwidth, virtual channel bandwidth, client retransmits, Receiver version, etc. This information is immensely useful when troubleshooting slow Citrix connections.

NOTE: Citrix Framehawk does not support AppFlow, hence AppFlow needs to be disabled if you want Framehawk to work.

Since ICA traffic from external users is already flowing through the NetScaler appliance, it is very easy to turn on AppFlow with Insight Center without needing to change anything else on the network. However, it might also be beneficial to get AppFlow records for internal ICA users. In that case, you need some method of sending ICA traffic through the NetScaler appliance. There are four common methods of achieving this:

- Enable SSL-ICA Proxy (NetScaler Gateway) for internal users. Note: SSL encryption increases bandwidth and foils WAN optimization devices. This can be enabled using the Optimal Gateway routing feature on Storefront. (More info on that here → <http://bit.ly/1RnEO3W>)
- NetScaler 11 supports SOCKS proxy through a Cache Redirection Virtual Server. This traffic is unencrypted but does not use the normal ICA port numbers.
- Policy-based routing – configure routers to send all ICA traffic through a NetScaler SNIP. This is a form of Transparent mode.
- Use NetScaler SNIP as default gateway for VDAs. Thus, all ICA traffic is forced through a NetScaler. This is also a form of Transparent mode.

NetScaler Enterprise Edition only stores one hour of AppFlow information. NetScaler Platinum Edition can store up to 30 days of AppFlow information.

NOTE: If you do not have the correct license and wish to have longer data retention, you can look at a third party vendor called Goliath and their product IT Analytics for NetScaler. Which also serves as an AppFlow Collector.

NetScaler Gateway for external users is typically deployed in the DMZ. NetScaler Load Balancing of Citrix StoreFront is typically deployed on the internal network. It might be tempting to use a single NetScaler appliance for both operations but that would require you to connect the single NetScaler to both the DMZ and the internal network, thus essentially bypassing (or straddling) the DMZ-to-internal firewall.

NetScaler has a couple technologies to mitigate this problem but they do not currently work for NetScaler Gateway and GSLB:

- Traffic Domains – multiple routing tables on a single NetScaler appliance. Similar to Cisco VRF.
- Admin Partitions – also achieves multiple routing tables on a single NetScaler appliance but the administrator interface is also partitioned.

Since NetScaler Gateway and GSLB do not support these multi-routing-table features yet, the most secure option is to deploy different NetScaler appliances for DMZ and internal. NetScaler VPX makes this very easy since it is just another virtual machine (and another license).