Load Balancer /ADC ALB-X Test Drive Tutorial

(ADC Application Delivery Controller)

EDGE NEXUS

EDGE NEXUS

pre-sales@edgenexus.io
edgenexus.io

We hope you enjoy your experience with the ALB-X load balancer and would like you to have a realistic configuration to play with.

- As part of this test drive we have pre configured a few example services to get you up and running.
- We host this setup on Azure but don't worry you dont need to have an azure account and also its FREE.
- You are welcome to re-configure the appliance to try it on your own servers.

Here are a few words to help you navigate the user interface and get the most out of your test drive, so fasten your seat belt....

Once you have signed up to your ALB-X test drive you will be presented with your own unique URL to access the ALB-X GUI from you web browser. If possible we would recommend using the Google Chrome browser.

Note the GUI is accessed using a non standard port :27376 so that the standard HTTPS port :443 is available for allocation as a load balanced service.

Do not be concerned that you are presented with an SSL security warning – this is because the management connection is secured by default using a local certificate.

For live deployment you are free to upload your own certificate to confirm authenticity of the management connection.

When prompted enter your unique username and password for this test drive session (you have been sent this in the email).

Azure Test Drive Setup

Note the GUI is accessed using a non standard port :27376 so that the standard HTTPS port :443 is available for allocation as a load balanced service.

- The IP address automatically configured on the load balancer appliance uses an Azure private IP address.
- You can of course configure Azure to open and NAT more ports for additional services.
- Only ports 80, 443 and 27376 (for the GUI) have been opened for this demo.

ALB-X Test Drive Pre-populated Services

Because this is a custom test drive we have pre-populated the IP-Services with some services you can try straight away.

For the purposes of the test drive we have made real server content available on 2 publicly available web servers:

webserver2.loadbalancer.software

webserver3.loadbalancer.software

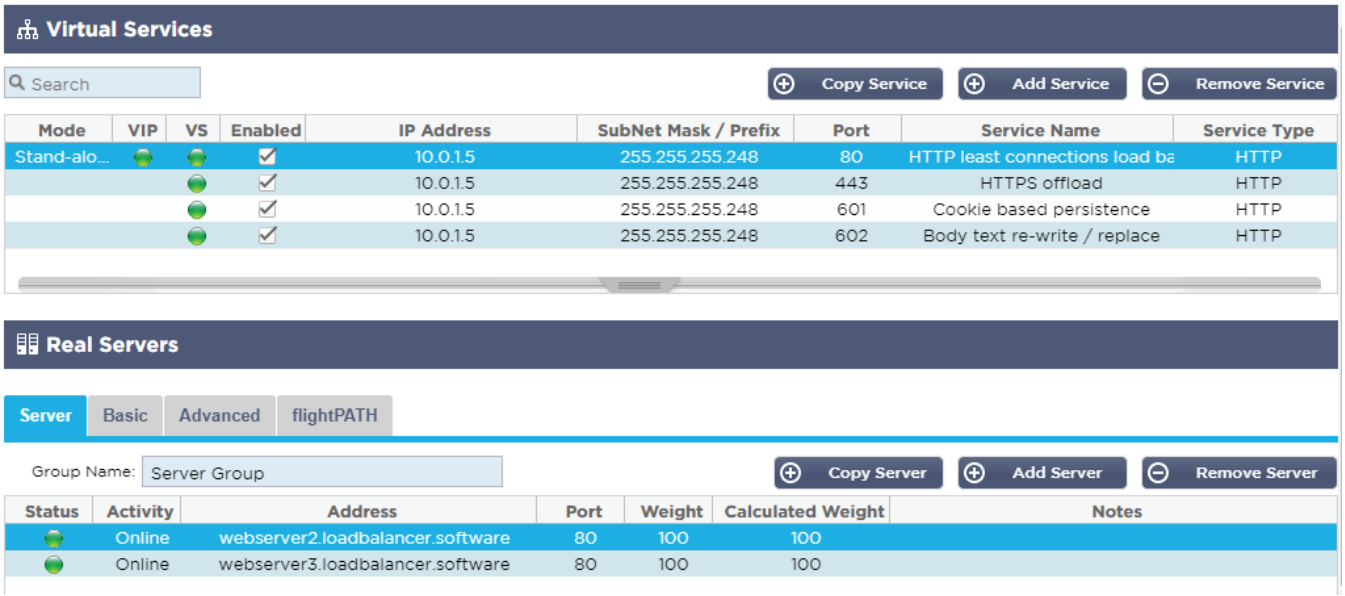
The ALB-X is able to use DNS to resolve the names to the public IP addresses. Each of these sites has text/images to show which site has served the content so you can see the load balancing process in action.

There are 4 demo services we have set up for you:

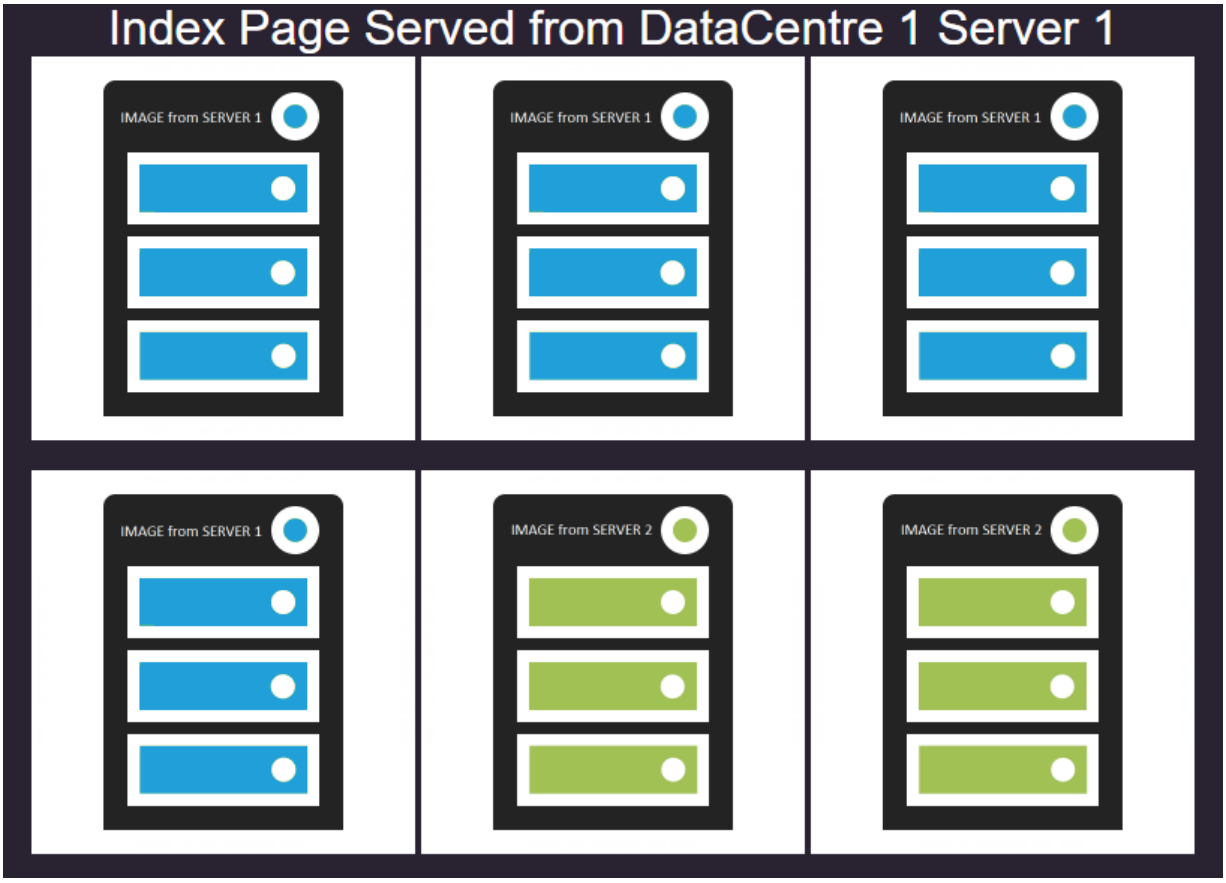
Name	Port	Accesibility
HTTP least connections load balancing	80	http://yoururl
HTTPS Offload	443	https://yoururl
Cookie based persistence	601	http://yoururl/601/
Body test re-write	602	http://yoururl/?602

Service on port 80 – HTTP least connections load balancing

The first service is a basic port 80 web server load balancer using our 'least connections' load balancing policy to 2 'real servers'.



Use your browser to open a HTTP connection to the same Public IP address as used for the management access of the ALB-X and you should get something similar to the following returned.



Service on port 443 – SSL Offloading

The second service is on port 443. In this case the load balancer is doing the encryption sometimes called SSL 'offload'.

We have used the default SSL certificate for the test drive demo so you will get the same security alert in your browser when connecting to this channel.

Please feel free to upload your own SSL certificate and apply it to the service, see instructions later. Once you have clicked past the security exception you will see the same content displayed in your browser.

HTTP to HTTPS redirection

The first thing we can take a look at that uses flightPATH is HTTP to HTTPS redirection.

This is a feature that is often used to ensure web traffic is served using a secure connection.

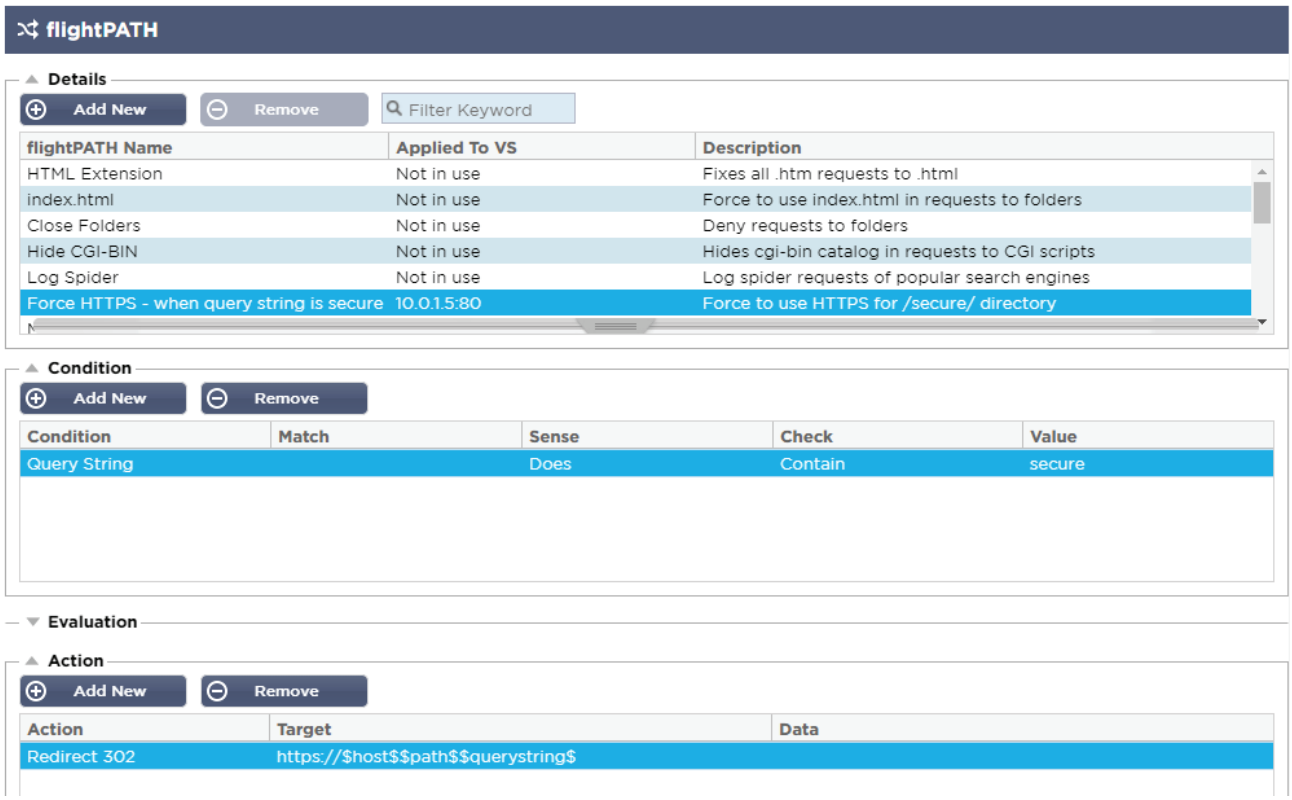
We have preconfigured a flightPATH rule and applied it to the port 80 HTTP channel to look for requests that have a query string of: /?secure

```
/?secure
```

If flightPATH sees this 'condition' it will act upon the traffic and return a 302 redirect to the browser to tell it to perform another GET request to HTTPS://your.original.request.location which in this case will be the same public IP address of the service but on the 443 channel.

You might like to take a look now at how the flightPATH rule is configured.

This you do by clicking on the Library tab on the left and selecting flightPATH.



Click on the 'Force HTTPS – when query string is secure' entry and you can see the

"Condition" is Query String Does Contain secure

OR

"Action" is Redirect 302 https://$host$$path$$querystring$ ($ are useful variables you can use in rule actions)

Cookie Persistence and Use Server based on Path

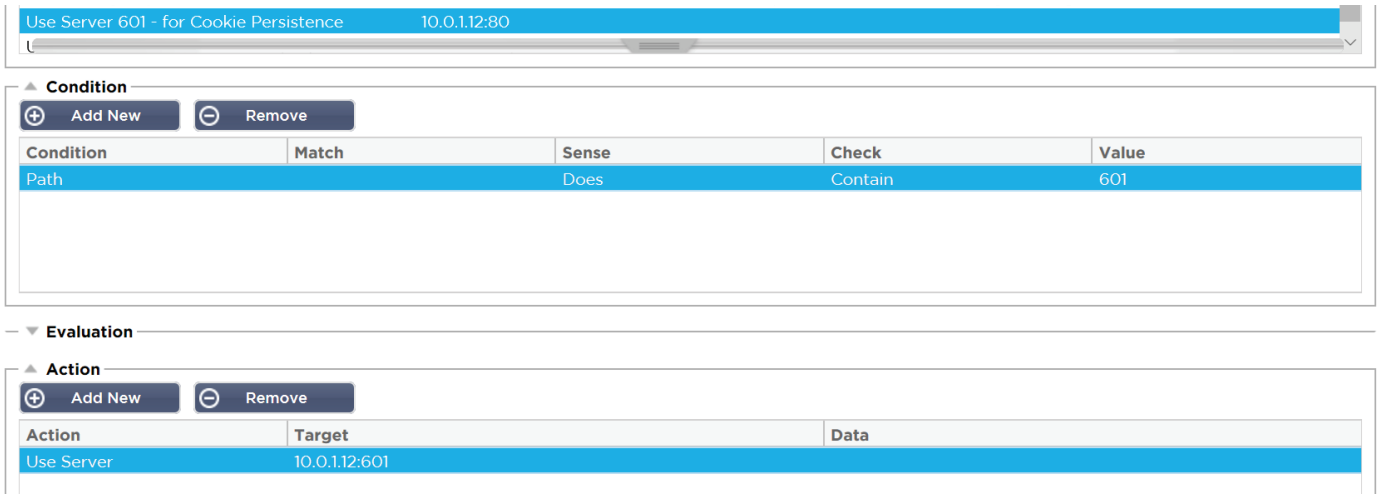
As you will have seen the connections have so far been spread across both real servers – this is why you see images returned from both server 1 and server 2. This is because the default load balancing policy is 'least connections', this will try to maintain an even number of connections to all the servers configured for a service.

REMEMBER: All objects in a web page such as images, video, JS will each have a separate GET request to the webserver. This behaviour may not be compatible with the application, the application may require 'persistence' or 'stickiness'.

For HTTP services this can be achieved by applying a special session cookie that the browser will present on all subsequent requests to that service, normally for the period of the 'session'.

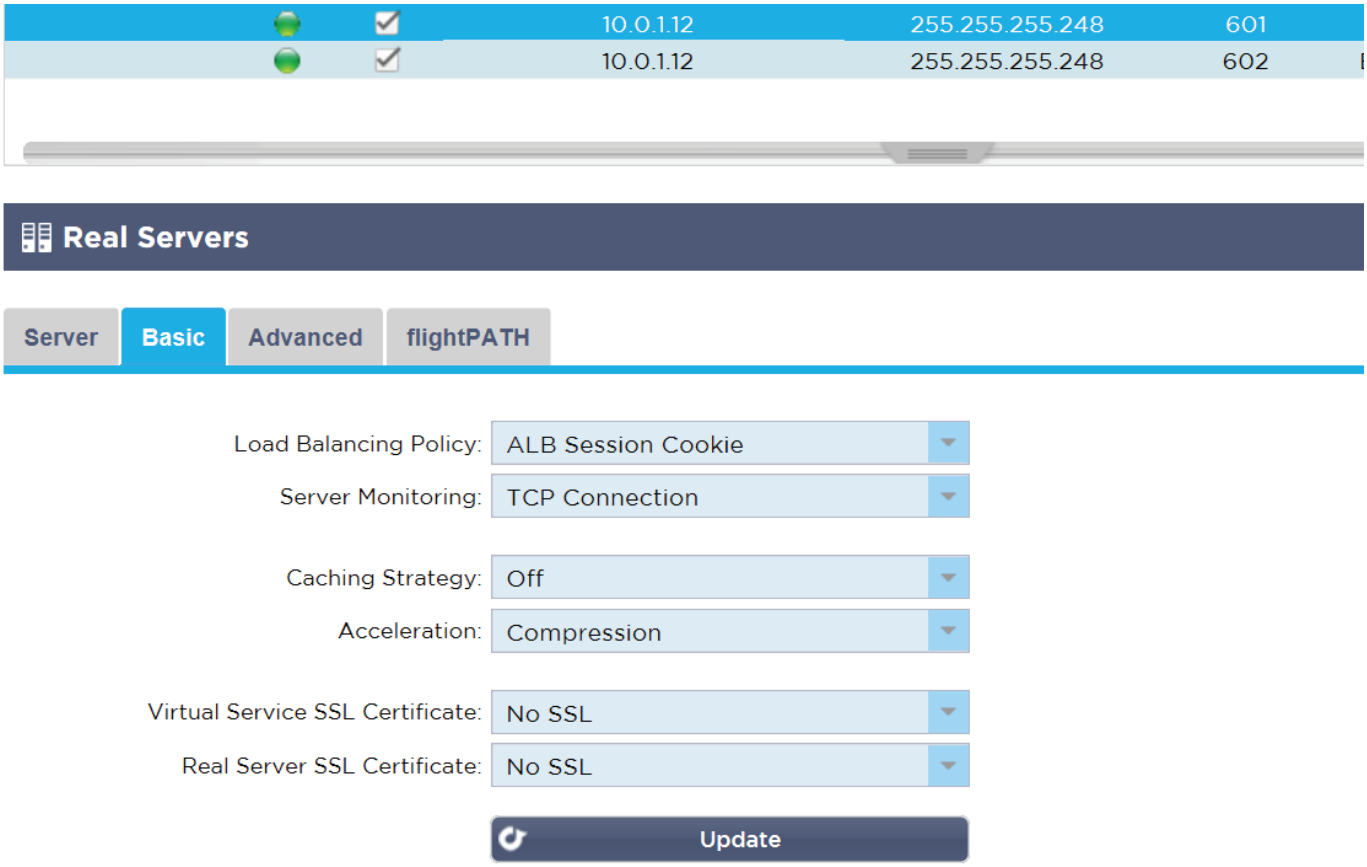
To demonstrate this we have configured the 'internal' 601 service for ALB session cookie based load balancing. As we said because the 601 service is not directly accessible from outside the Azure network we have used a flightPATH rule to direct traffic to this service.

The flightPATH rule looks at the requested path and if it is /601/ it will send the traffic to the service running on port 601. Here are the details of this flightPATH rule.



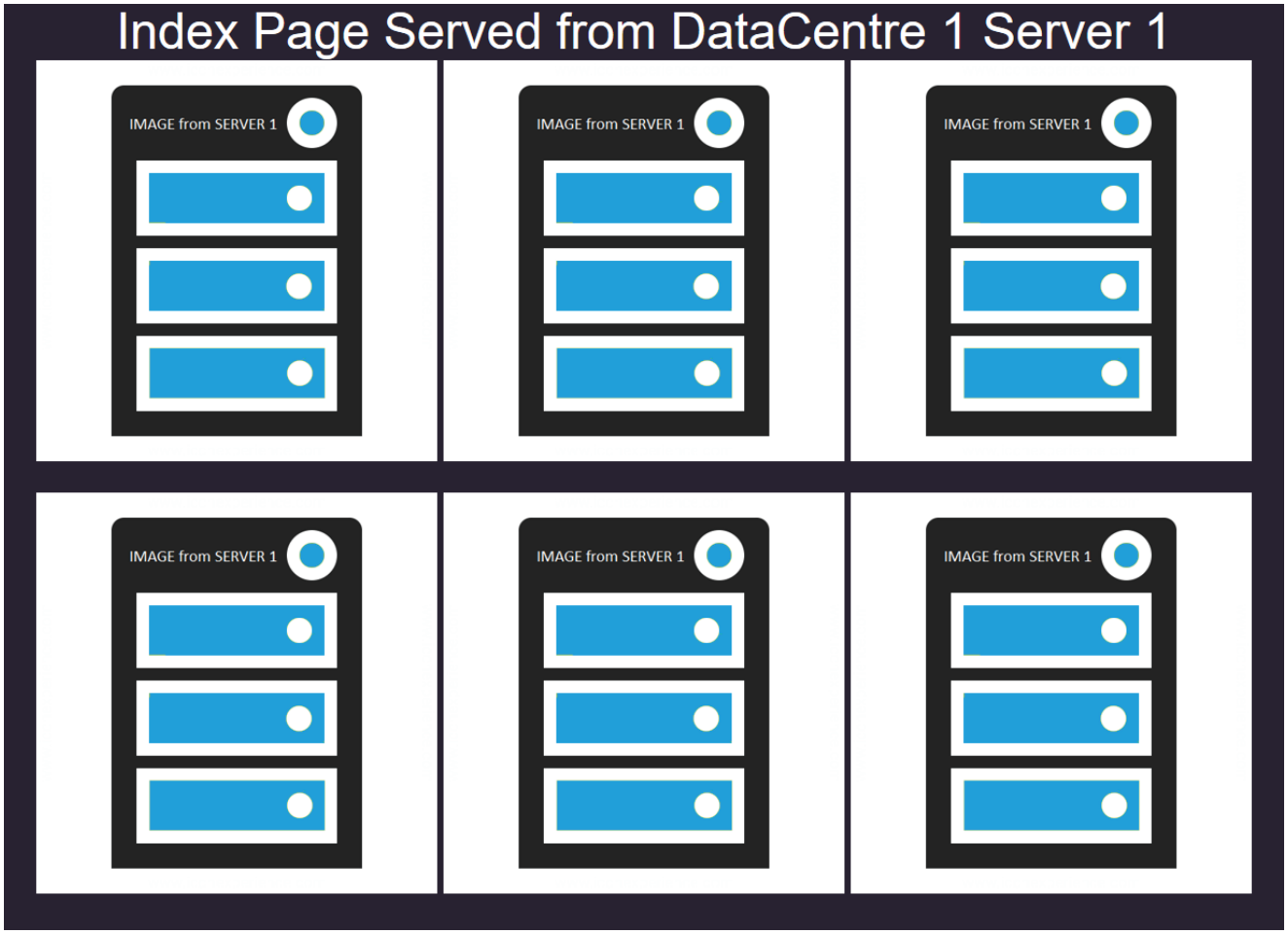
This flightPATH rule shows how traffic can be sent another service based on the path, this is a powerful tool for traffic manipulation or content steering.

We can see the Configuration of the VIP on port 601:



Now try adding the /601/ path to the public IP "http://myurl/601/", you should get a result like this:

```
http://myurl/601/
```



You can see here all the content is served from Server 1 – you can check your browser developer tools and you will see that a cookie called jnAccel= has been set.

Path Rewrite and RegEx evaluation

As the real server does not have any content under the /601/ path we needed to remove it from the request or we would get a 404 error (which we can also hide using flightPATH).

This we have done by applying another flightPATH rule called 'Remove Path 601' to the service running on port :601.

Remove Path 601 10.0.1.12:601

Condition

Add New | Remove

Condition	Match	Sense	Check	Value
Path		Does	Contain	601

Evaluation

Add New | Remove

Variable	Source	Detail	Value
$NewPath1$	Path		^/601/(.*)$

Action

Add New | Remove

Action	Target	Data
Rewrite Path	/$NewPath1$$querystring$	

Here we again look for 601 in the path as a condition to trigger this rule.

In this case we make use of the Evaluation function which uses Regular Expression to allow a new variable to be created by manipulating an existing System variable, in this case the original Path value we saw.

So we create the NewPath by just extracting the data after the leading /601/ path prefix.

The 'Action' is to Rewrite Path using the $NewPath1$ and $querystring$ if it were present.

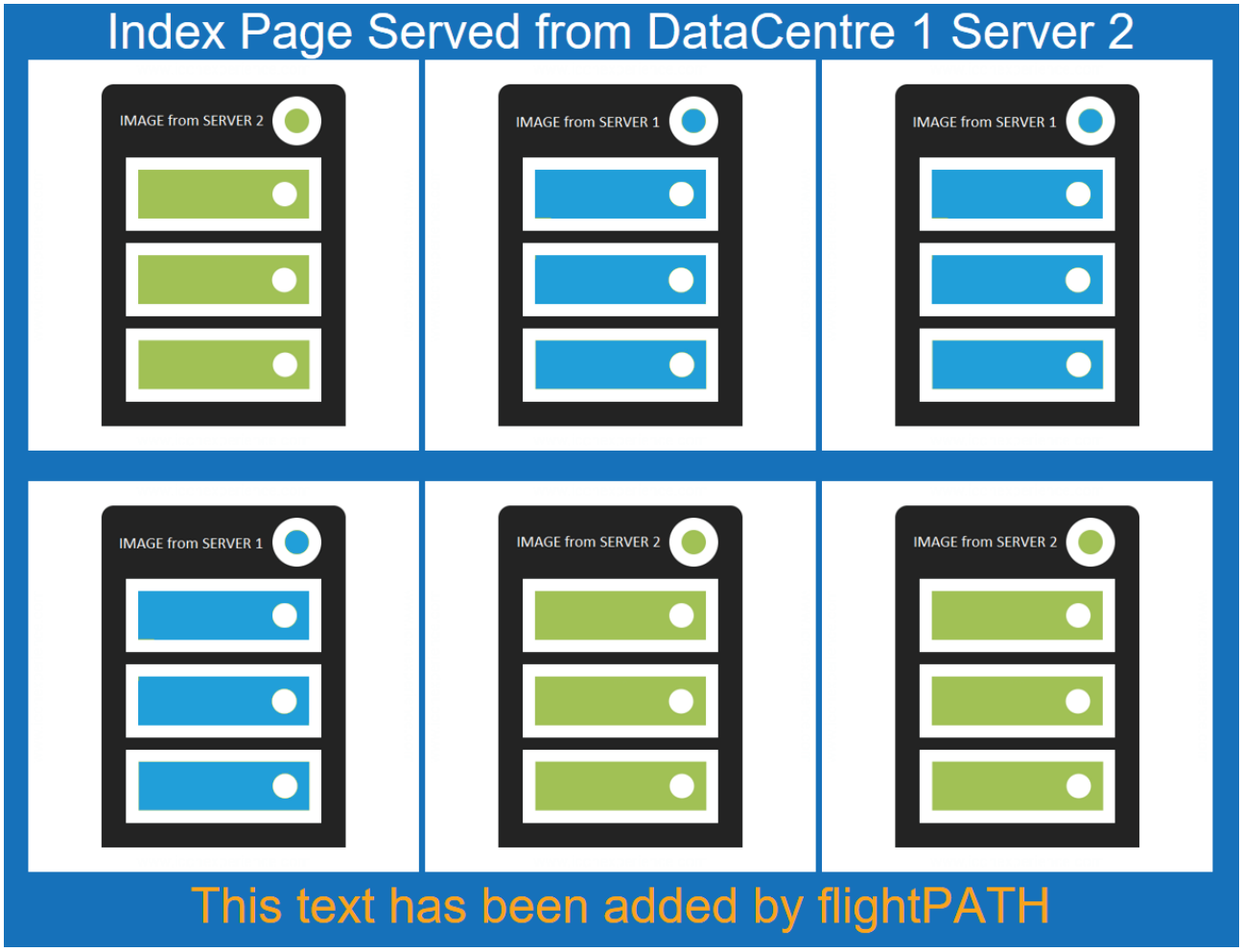
HTML Body text replace

In the other service example using port :602 we show how you can also manipulate the HTML content and not just the HTTP header.

Instead of using a path to direct traffic arriving on port 80 to use the 602 service we have used querystring /?602. - "http://myurl/?602"

```
http://myurl/?602
```

If you enter the public IP address of your test drive ALB-X in your browser and append /?602 you should get the following result returned.

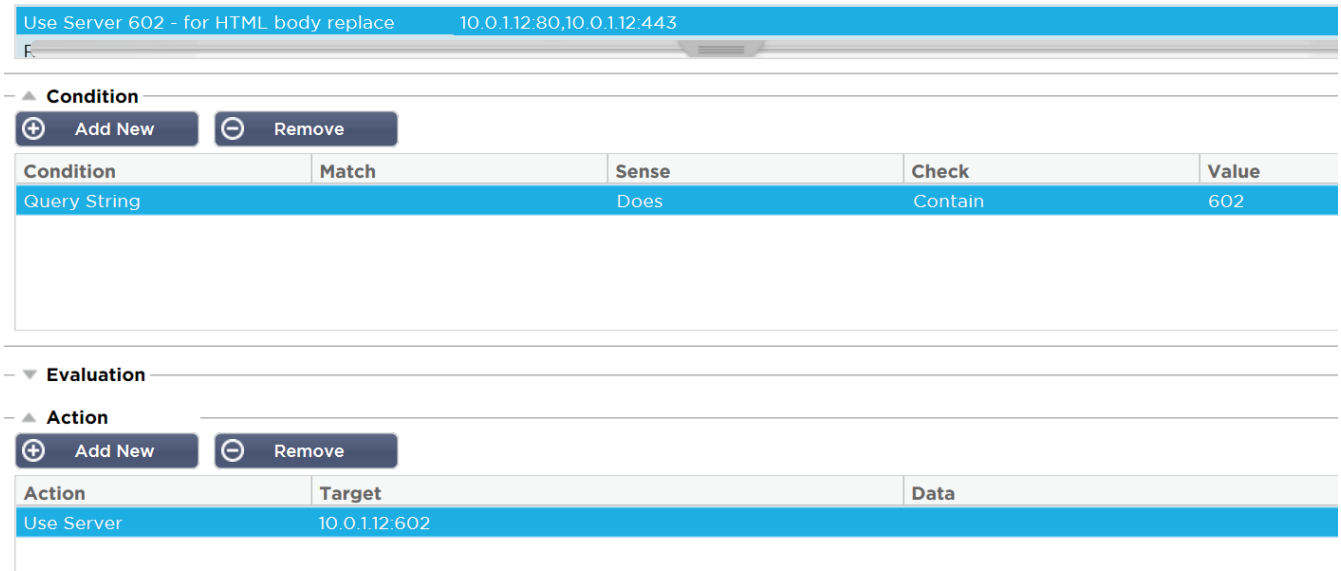


You can see an additional line of text in orange stating that -

> **"This text has been added by flightPATH"**

This is achieved by means of 2 flightPATH rules.

The following rule applied to port:80 looks for a query string value of 602 and sends all matching requests to the service running on port 602.



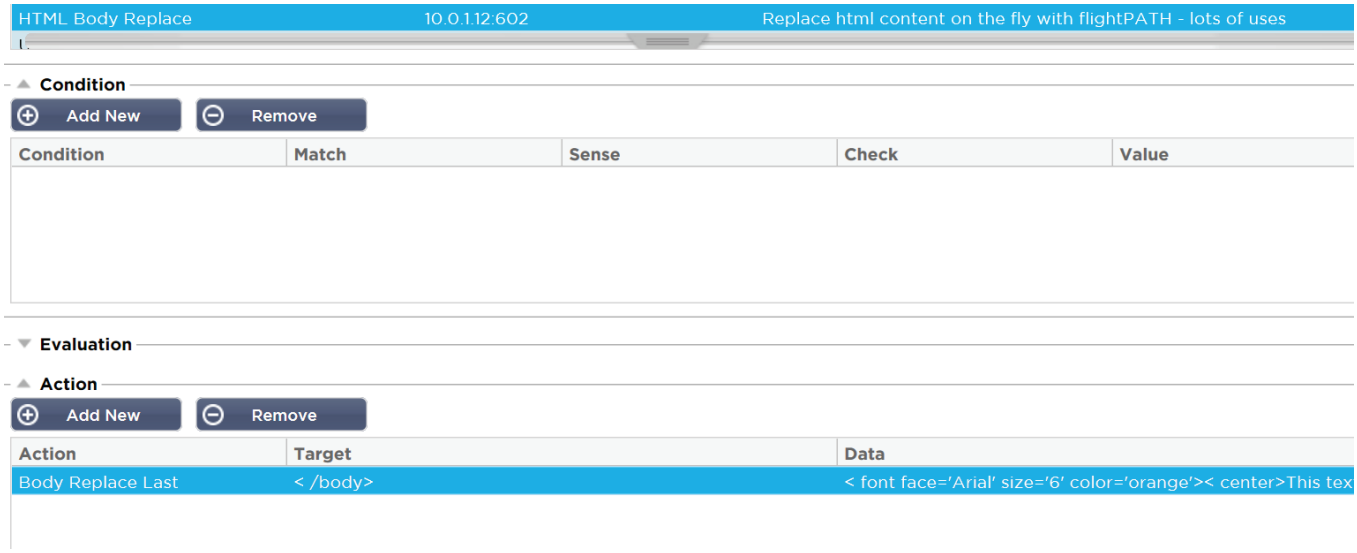
Applied to the service running on port :602 is a flightPATH rule that performs a 'Body Replace Last' function looking for the closing body tag

```
<script
src="https://www.edgenexus.io/wp-content/cache/min/1/dc61abce0
097c91f6c6588c6189ae217.js" data-minify="1"
defer></script></body>
```

and replacing it with

```
<font face='Arial' size='6'color='orange'><center>This text
has been added by flightPATH</center></font><script
src="https://www.edgenexus.io/wp-content/cache/min/1/dc61abce0
097c91f6c6588c6189ae217.js" data-minify="1"
defer></script></body>
```

There is no condition or evaluation required in this case so the function will apply to all traffic passing through the service.



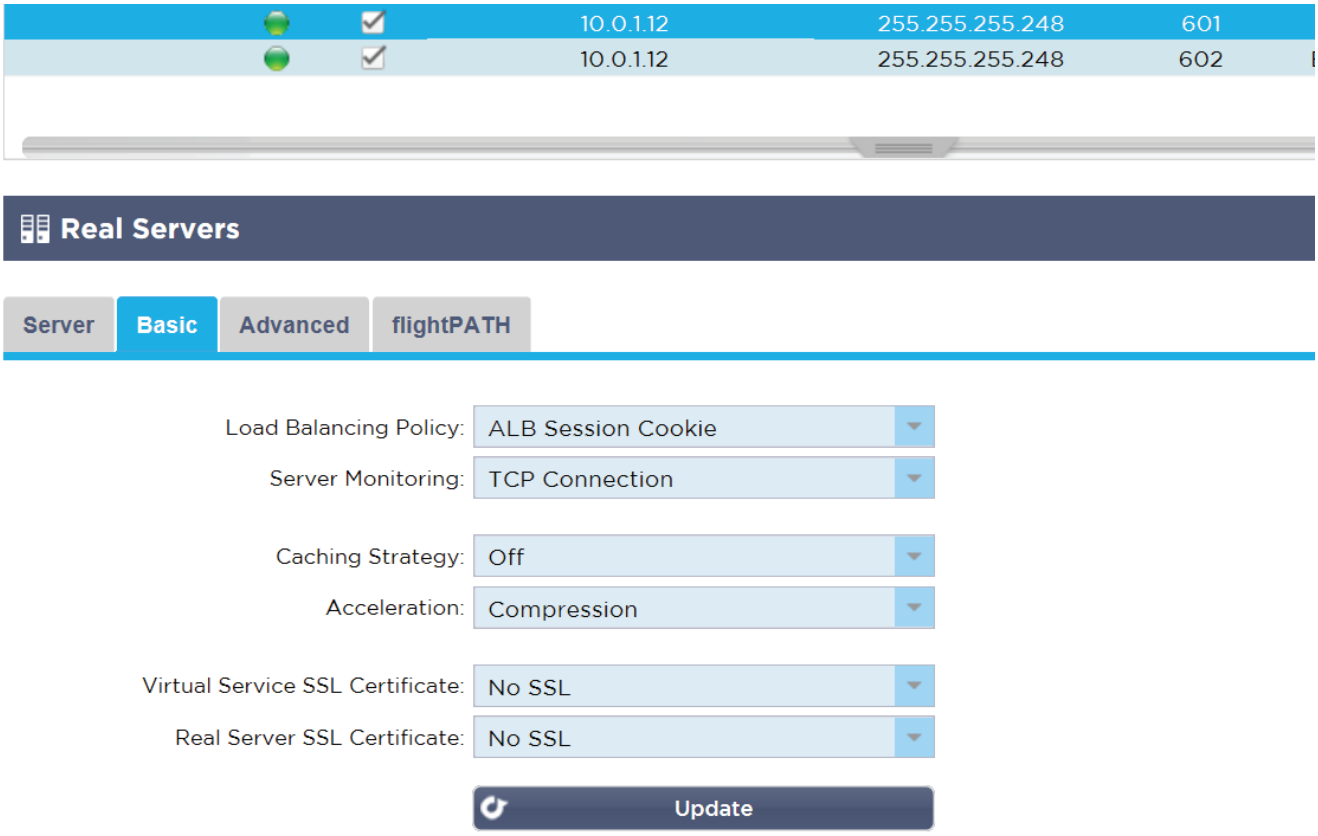
Hopefully you can see how else these facilities could be combined to perform some clever HTTP traffic manipulation and these are just the tip of the iceberg of what flightPATH can do.

Please experiment for yourself and we would be glad to hear about your specific requirements.

Real Server Health Checks

The next thing we will take a look at is real server health checking. It is vital that the right type of health check is applied to a service to enable reliable detection of the health of the back end server(s) to ensure client traffic is only sent to servers that are operational.

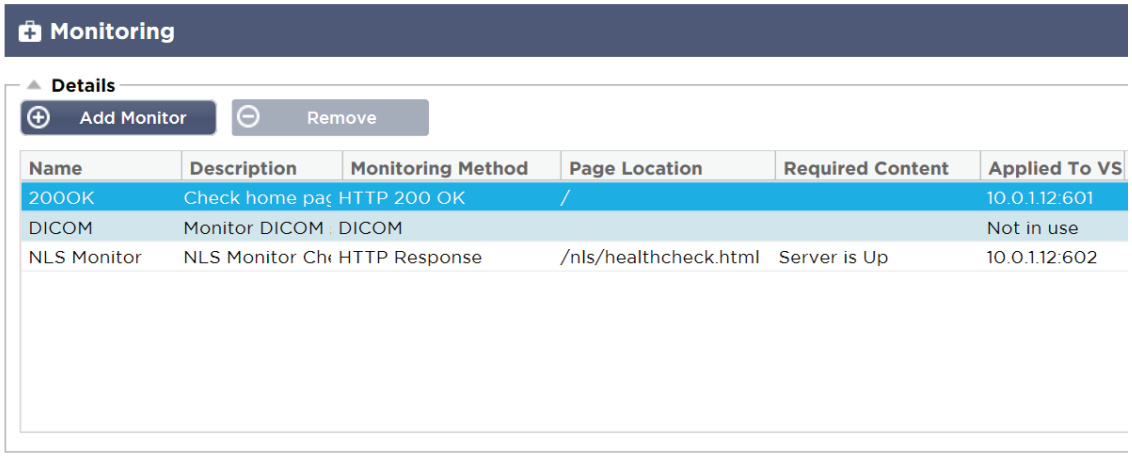
When creating a new service we have to define a default set of parameters. For the Server Monitoring we use the TCP Connection health check. The Server Monitoring option is found under the Basic tab for the service being configured.



Whilst this is a reasonable starting point it is highly recommended that a more reliable health check be configured and applied to a service, and we have made it super easy to create custom HTTP health checks.

Health check can be found under "Library / Real Server Monitors".

In the Test Drive we have added a third Real Server Monitor to show an example of a HTTP response health check.



HTTP 200 OK Monitoring Method

The 200OK Uses an HTTP GET request to the page location configured for the check and makes sure a 200OK status response is returned.

It doesn't look for any specific content it just checks that a web server is running on the port and is able to serve the page requested.

> **This is a better monitor than the TCP Connection as it is operating at Layer 7 checking the application is running but it does not check for a specific response of content**

We have applied this monitor as you can see above to the service running on port:601.

HTTP Response Monitoring Method

The NLS Monitor configured in the Test Drive makes use the HTTP Response check.

For this you define the specific page location (this can be the complete URL where host headers are required) and you define content (a text string) that should be present in the returned data from the server / application.

This is a far better test as the page must be present and the specific content needs to be available.

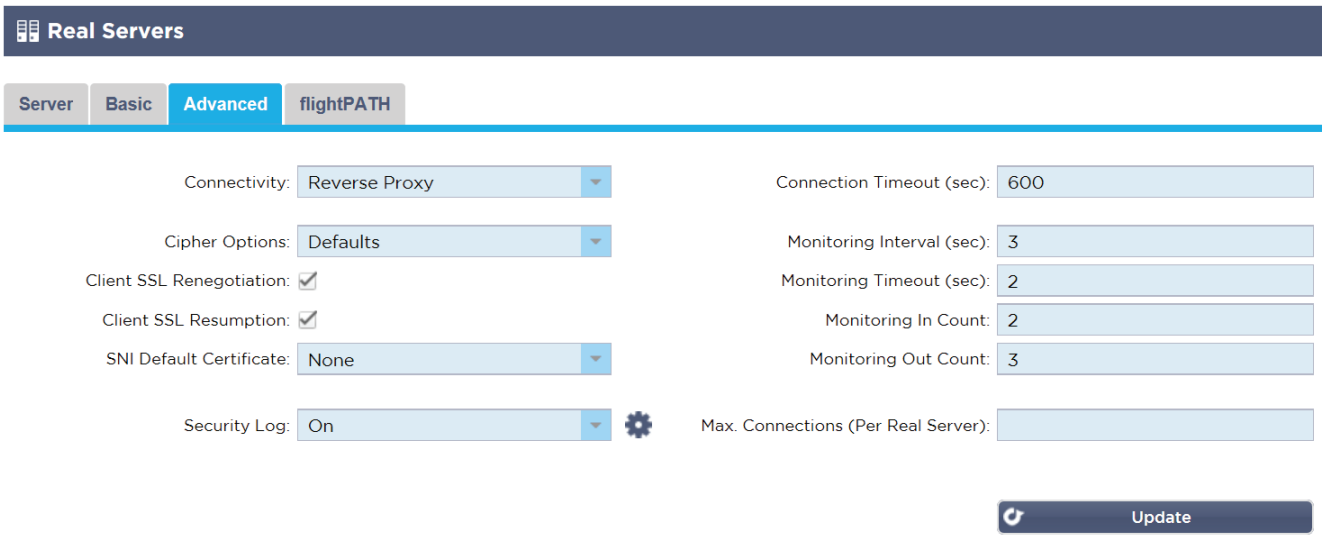
> **Where the application is fronting a back end database it is a good idea to make the status of the content retrieved on the health checked page dependant on live responses from the database rather than just static content on the web front end server.**

You can see we have applied this monitor to the service running on port:602.

You can modify the text in the Required Content field for this health check and you will see the servers go red to show they have failed the health check. Revert the required content back to the correct value and the servers will come back OK green.

Monitoring Interval

Under the advanced tab you are able to manipulate the frequency and time out etc for the health check operation on that service.



For the test drive we have set the Monitor Interval to 3 seconds with a 2 second time out.

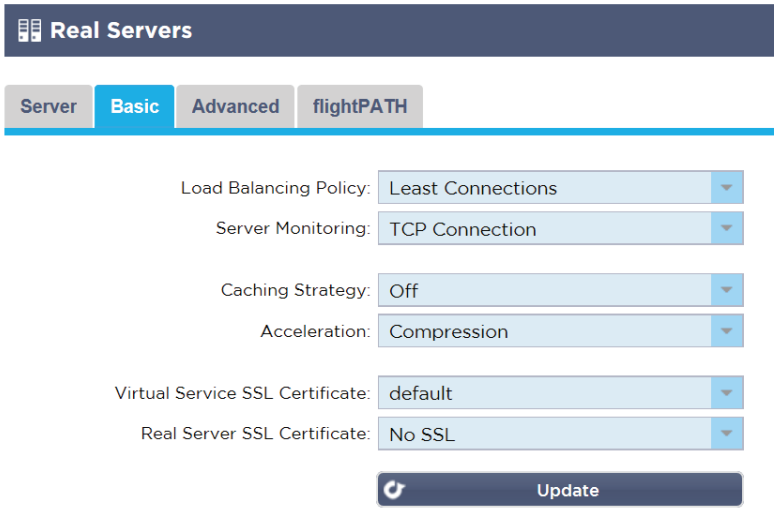
With a Monitor In Count of 2 there must be 2 consecutive successful responses before marking the server back in service. These are sensible values to use in most cases.

HTTPS and SSL Certificates

More and more websites are using HTTPS and by the beginning of 2017 the percentage had swung in favour of HTTPS.

Most enterprise applications require protection through encryption so it is a pretty safe bet that you will need to use certificates on the ALB-X and deploying a load balancer with HTTPS is a quick and easy way to secure access to non encrypted applications.

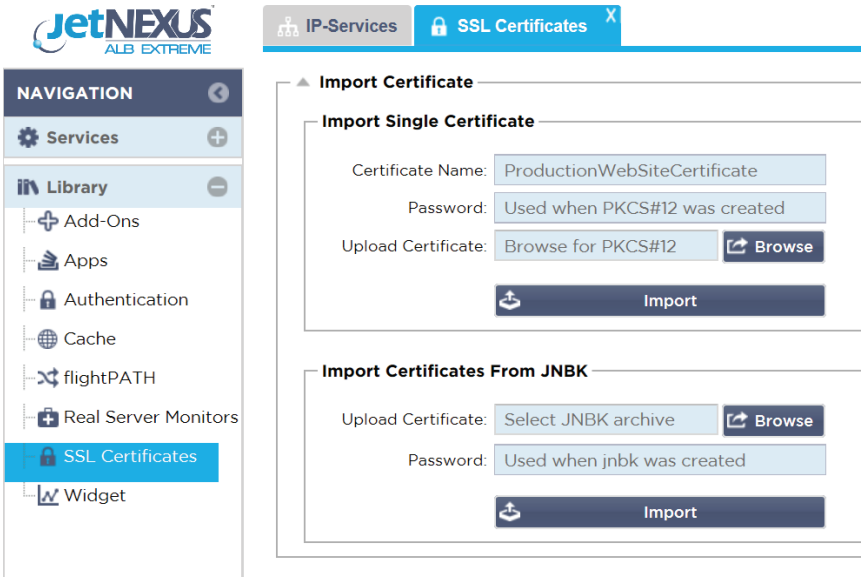
The ALB-X has one private certificate installed by default called 'default' that is used to allow HTTPS connectivity to the management GUI. We have applied this certificate to the Test Drive :443 HTTPS configured service on the Virtual Service or client side.



The service is configured for SSL offload and so the Real Server side is configured for No SSL.

Certificate upload / import

You can upload your own signed certificates to the ALB-X using the SSL Certificates menu in the Library section.



As highlighted in the text input fields the certificate needs to be the PKCS#12 format to be imported in to ALB-X.

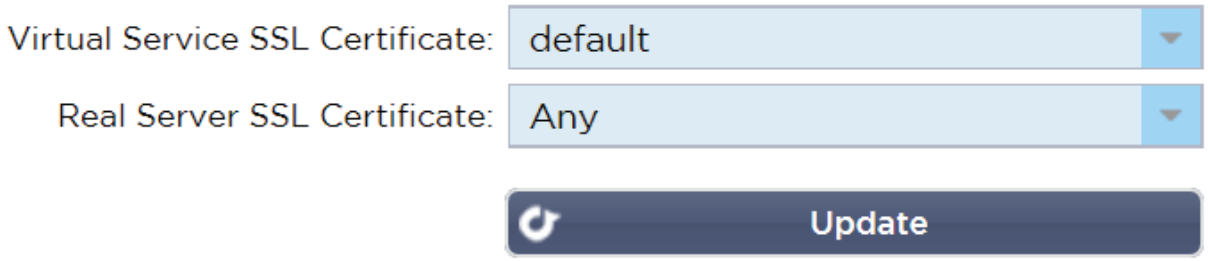
> **This type of certificate contains the private key and is secured with a password which is required at time of import.**

The name (which must not contain spaces) you give the certificate at import will be what appears in the certificate selection drop downs in the IP service Basic tab.

Real Server Re-Encryption

If the real servers require SSL/TLS re-encryption the most sensible option to select is 'Any' from the Real Server SSL Certificate drop down.

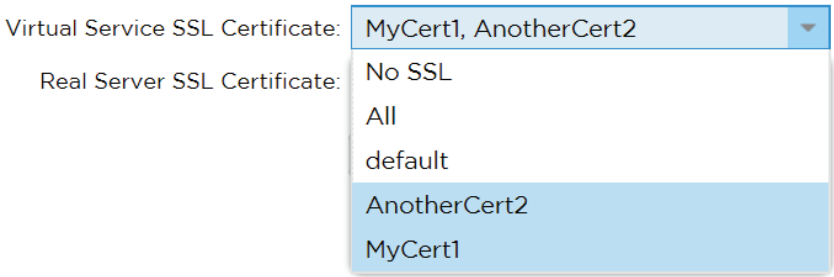
> **This means the ALB-X will accept any certificate presented by the real servers as being valid.**



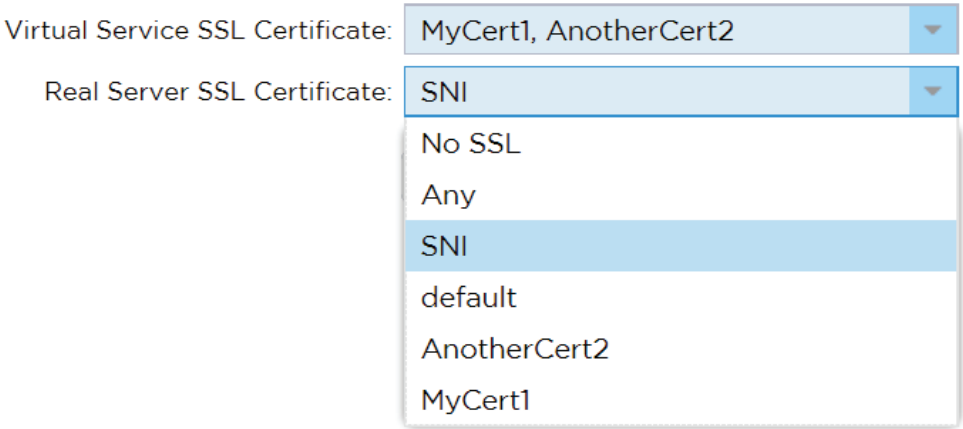
SNI (Server Name Indication)

With the scarcity of Public IP addresses and the fact that only one VIP can be configured in Azure it is useful to be able to support multiple secure domains / host URLs through one virtual service and this is possible on ALB-X because we support SNI.

To use SNI all you need to do is select all the necessary certificates from the Virtual Service SSL Certificate drop down box. Each click will either toggle select or deselect the certificate as part of the SNI list.



On the Real Server side if the services are re-encrypted and hosted on common servers they will require SNI for the correct service identification and negotiation, so SNI should be selected as the option in the Real Server SSL Certificate drop down.



Apps

ALB-X supports the deployment of additional features and functionality by purchasing and downloading Apps from our App Store for deployment within the ALB-X containerised environment.

The first 2 key add-on available are a Web Application Firewall and Global Server Load Balancing for automated Data centre redundancy and hybrid cloud failover / load balancing.

We have set up 2 separate Test Drives in Azure where you can see this functionality in operation, so please check these out if you are interested.

App Store | **Test Drive**

Authentication

ALB-X supports Pre Authentication in conjunction with a MS AD (Microsoft Active Directory) / LDAP server.

You are free to explore this functionality if you have access to a MS AD / LDAP server that is publicly accessible (use LDAPs). The online ALB-X user guide walks through configuration of this feature.

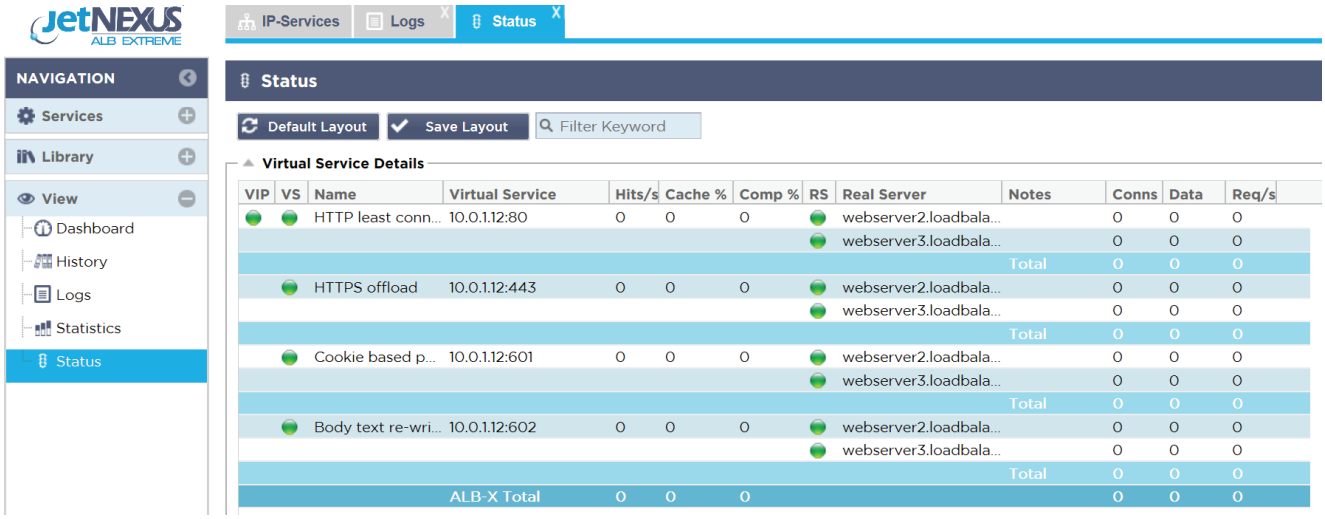
This functionality is a popular replacement for the discontinued Microsoft TMG product.

ALB-X user guide

Appliance usage and connection monitoring

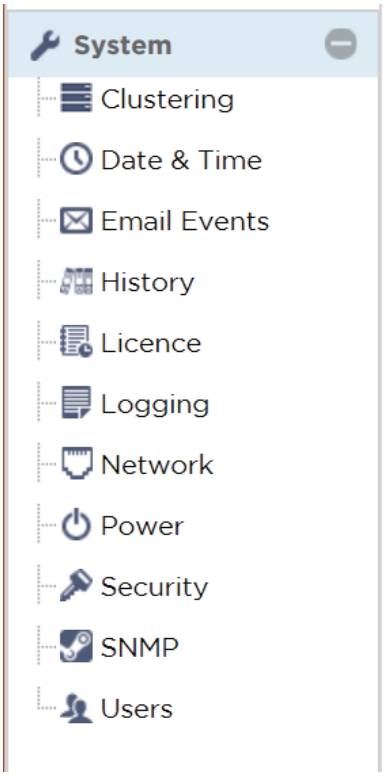
The view section of the menu provides you the means to be able see connection status and real server health either in real time (Status) or as a trend over time (History).

This is the place also where you will find W3C and system logs. You can also create your own custom widgets to display on the Dashboard. We would encourage you to take a look at the various options in this section.



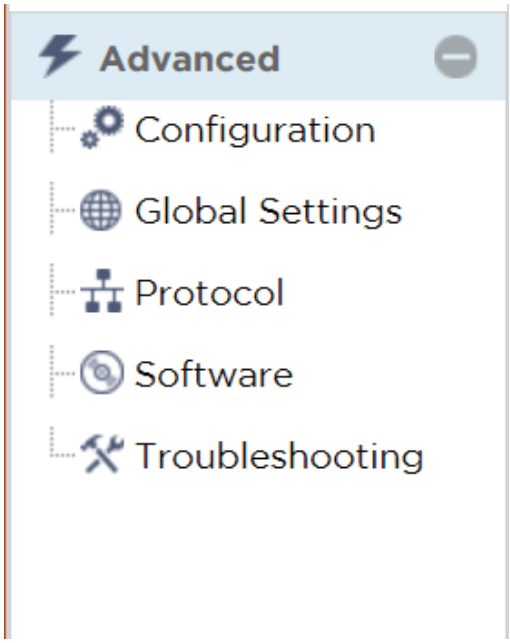
System

This is where you will find configuration options that apply to the system functions such as setting Time & Date, enabling email alerting, Licensing the product, choosing how logging is performed and where logs are sent, rebooting and restarting the appliance configuring SNMP and adding other management users etc.



Advanced

In the advanced menu you are able to backup and restore configuration and also upload jetPACK template configurations.



You can modify common HTTP protocol behaviour and you can perform software upgrade and download other software packages from the cloud library.

Lastly there is a troubleshooting section where we have made it easy to download a bundled support file package, perform network PING and perform various system traces and network packet captures.

Thank you.

We really hope you enjoy your ALB-X test drive.

We would welcome any questions you may have to

pre-sales@edgenexus.io

0808 1645876

(866) 376-0175

edgenexus.io

EDGENEXUS