Optimization Configuration Quick Start Guide

Version 8.5

Contents

1 Application Server Configuration Options 2
1.1 Excluded Application Servers 2
1.2 Transparency 2
1.3 SSL Configuration 2
1.3.1 Provide Original Application Server Certificates 2
1.3.2 Configure the Virtual Appliance as a Certificate Authority 3
1.3.3 Permitted SSL Versions 3
1.3.4 Verify Server Certificate 3
1.4 Services 3
1.4.1 Applying Services 4
2 Global Configuration Options 5
2.1 WAN Connection Pooling 5
2.2 Congestion Control Algorithm 5
2.3 Block Size Configuration 5
2.4 HTTP Settings 5
2.4.1 Cache Configuration 5
2.5 SNI Filter Configuration 6
2.5.1 All Optimizations 6
2.5.2 TCP Only 6
2.5.3 Adding to a list 6
3 Optimizing Everything 8
3.1 Location of Application Servers 8
3.1.1 Client-Side Networks 8
3.1.2 Geographically Distant Networks 8
3.2 Applications that use Certificate Pinning 8
3.3 Certificate Issues 8
3.4 Applications that Perform User Profiling 8
3.5 Cache Pollution 9

1 Application Server Configuration Options

To enable optimization, the Replify Accelerator must be configured with one or more *Application Servers*.

An application server can be a single IP address or a subnet in CIDR notation. For example, 10.0.0.0/8. When this is added to a VA, any remote peered appliances or clients will be instructed to intercept traffic destined to these IP addresses. This traffic will then be optimized according to the configuration applied on the Virtual Appliance.

1.1 Excluded Application Servers

Sometimes you may wish to optimize a subnet except for a few hosts. If this is required, the subnet can be configured as an *Application Server* and the hosts can be configured as *Excluded Application Servers*.

1.2 Transparency

Optimized traffic is intercepted and sent across the WAN to a Virtual Appliance which will then send on the original application request to the application server.

The TCP connection from the Virtual Appliance to the Application Server will have a source address that is the IP address of the LAN interface of the Virtual Appliance. If this is undesirable, the *Transparency* option can be selected on the Application Servers UI. This allows the source address and port to be spoofed to match that of the original request.

Note that if this option is turned on, routing must be in place to ensure that traffic from the application server to the remote address is routed back via the Virtual Appliance.

1.3 SSL Configuration

Replify Accelerator can optimize traffic that is secured with TLS. For example, HTTPS traffic. However, for many of Replify Accelerator's optimization algorithms to work, it needs to be able to decrypt the traffic, apply optimization, send this across the WAN and then re-encrypt before sending on to the final destination.

By default, no application layer optimization will be applied to TLS traffic unless *SSL Optimization* has been enabled for the application server. There are two ways in which this can be configured:

1.3.1 Provide Original Application Server Certificates

If the SSL certificate and private key of the application server are available, these can be added to the VA on the *Certificate Management* page of the Virtual Appliance UI. These certificates will be used to decrypt/encrypt traffic and the end user devices will be unaware that traffic has been intercepted.

This can be useful when the client's already trust the application server, as no further configuration will be required on the client when optimization is enabled.

To configure this for an *Application Server*, configure *SSL Optimization* on the application server and choose the certificate that was previously uploaded.

Note that Replify's SSL interception is designed so that the private key will never be sent out from the VA has the application server configured.

1.3.2 Configure the Virtual Appliance as a Certificate Authority

The Virtual Appliance can generate its own certificate and key for encryption/decryption of TLS traffic. This can be configured by choosing *Auto Generate* for the certificate on the *Application Servers* page.

For this to work, the end user application or OS needs to have the VA's CA Certificate (available from the *SSL Management* page) configured as a *Trusted Certification Authority*. Note that the client will configure this automatically if the Automatically Trust CA Certificate on Clients setting is enabled on the *SSL Settings* section of the *SSL Management page* on the VA's GUI.

Warning: If this is not done the end user application will typically block traffic and display a security warning for the end user.

1.3.3 Permitted SSL Versions

Replify Accelerator secures TLS connections between the end user device and across the WAN using the most secure version of TLS that is supported by both the VA and the end user application. This option allows application servers that are using less secure versions of TLS to be blocked.

It should be noted that if an application server is using an older version of TLS, the WAN Connection between VAs is always secured using the most recent version of TLS that is supported by Replify Accelerator.

1.3.4 Verify Server Certificate

If this option is selected (it is enabled by default) Replify Accelerator will block the connection unless it can verify the application server's certificate. Note that for some certificates this may require the signer's CA certificate to be added to the to the VA. This can be done on the *SSL Management* UI.

1.4 Services

Services allow granular configuration of the optimizations that can be applied to traffic destined for a specific port or set of ports on an *Application Server*.

There are several pre-defined services configured on the system that have appropriate optimizations configured.

Custom services can be configured from the *Services Configuration* UI. To define a *Service* you need to specify the following information:

Handler

Replify Accelerator contains several protocol handlers that provide specific optimizations for different protocols. If a suitable protocol handler is available this should be chosen. If not, the *Standard* protocol handler should be used.

Note that if the traffic is encrypted using TLS, a protocol handler relating to the underlying protocol should be specified. For example, if defining a service for HTTPS traffic, the HTTP protocol handler should be used.

Note that if the default handler is chosen, the service with this handler can be used as a fallback. That is, if traffic is destined for a port that no other service can process, a service with a default handler can be used to process this.

Name

This is simply the name that will be used to refer to the protocol handler on the *Services Configuration* and *Application Servers* UIs.

Ports

This is the list of TCP ports that is associated with the service.

Optimizations

This allows you to specify the individual optimizations that should be associated with the service.

1.4.1 Applying Services

Services can be associated with an Application Server on the *Application Servers* UI. Note that if overriding a predefined service, the predefined service should be unchecked, and the custom service should be checked.

The fallback behaviour can also be specified on this UI. This allows a service that is configured with the default handler to be used for any traffic that can't be associated with an enabled service for this Application Server.

2 Global Configuration Options

2.1 WAN Connection Pooling

WAN Connection Pooling can be enabled and disabled from the settings page of the Virtual Appliance UI.

When WAN Connection Pooling is enabled, the VA maintains a “pool” of TCP data connections between itself and all connected peered appliances or clients. Using a pooled connection means that no TCP handshake needs to take place across the WAN for new connections. On high latency connections this can result in significant performance improvements.

It should be noted that there is a small data overhead associated with this that means it will have very limited benefits on lower latency connections.

2.2 Congestion Control Algorithm

The TCP protocol has inbuilt functionality to handle congestion on the underlying network. The default congestion algorithm used by most operating systems is the *Cubic* congestion control algorithm which works well in a variety of different scenarios.

However, when there is high latency or significant packet loss, other algorithms can be more effective. The *Settings* page of the VA UI allows you to choose the *BBR* and *Hybla* algorithms instead of *Cubic*.

Depending on network conditions, these may provide a performance boost.

Note that the default setting chosen here can be overridden for individual peered appliances on the *Peered Appliances* configuration screen.

2.3 Block Size Configuration

Replify Accelerator’s de-duplication technology uses a cache that is maintained on each Virtual Appliance and Accelerator Client.

The cache contains blocks of data that have traversed the network. If these blocks are encountered on subsequent requests, Replify Accelerator can instruct the remote node to retrieve the block from its cache instead of transmitting it across the network.

The Accelerator Intelligent Cache Engine (ICE) can store blocks of two sizes. These are either small (4KB) or large (100KB).

Using large blocks generally results in increased throughput and offload, however a small change in content results in more retransmission of cache data when content changes.

If working with data that changes frequently, for example when making lots of small changes to a text file and saving after each change, small blocks should be used, otherwise the default setting where large blocks are used by default is fine.

2.4 HTTP Settings

2.4.1 Cache Configuration

The Replify Accelerator HTTP protocol handler can determine the HTTP content of any requests/responses.

It is often worthwhile to change caching behaviour for an HTTP response based on this. For example, it may not be desirable to cache real-time video content as this content is unlikely to be seen again.

2.5 SNI Filter Configuration

Server Name Indication filtering, or SNI filtering for short, allows the Replify Accelerator engine to selectively apply optimization based on the Server Name specified in a TLS request. This allows the layer 7, or application layer, optimizations to be de-activated to avoid wasting processor time, or enabled for a specific selection of servers.

The SNI is part of a TLS request header so that when an application makes a connection to an application server, the Replify Accelerator can inspect the request to read the hostname. If SNI filtering is inactive, optimisations will be applied if SSL has been enabled for that application server. This results in the Replify Accelerator terminating the TLS connection and presenting the configured certificate to the end-user.

SNI Filtering has two available options to choose from: *All Optimizations* or *TCP Only*.

2.5.1 All Optimizations

When this option is enabled, connections made to application servers with an SNI listed in the *All Optimizations* list will be fully optimized by Replify Accelerator's protocol optimization, caching and compression. Connections to any other application server, with an SNI which is not specified in the list, will only have TCP optimization.

For optimization to be applied to application servers with an SNI in this list, ensure that the configured certificate for the application server is trusted, as the client will be presented with this certificate.

This option cannot be enabled unless there is at least one SNI added to the *All Optimizations* list.

2.5.2 TCP Only

When this option is enabled, connections made to application servers with an SNI listed in the *TCP Only* list will only have TCP Optimization. This means that Replify Accelerator will not decrypt the traffic and apply its protocol optimization, caching and compression functionality to the connection. Connections to any other application server, with an SNI which is not specified in the list, will be optimized fully by Replify Accelerator.

In this case the client will see the certificate from the application server and so must trust that, rather than the one configured on the Replify VA for that application server.

2.5.3 Adding to a list

To add an SNI to one of the two lists, enter a SNI hostname or list of hostnames separated by a comma and select which list you wish to add it to. Once an SNI has been added it will be filtered based on the behaviour previously selected.

Adding a hostname to a list can be specific to a single hostname or it can be expanded to include the subdomains of a hostname. This can be done as follows:

`example.com` will only filter example.com and not sub.example.com;

`.example.com` will filter example.com as well as all subdomains of example.com

The * wildcard also be used, though isn't required. For example:

`*.example.com` will filter example.com as well as all subdomains of example.com

3 Optimizing Everything

Replify can be configured to optimize all traffic by specifying an Application Server of **0.0.0.0/0** and it can be tempting to use this and expect all internet traffic to be accelerated, however there are several considerations to be made.

3.1 Location of Application Servers

Replify Accelerator is intended to accelerate traffic in a scenario where the network between the VA and the application server has much higher bandwidth or lower latency than between the VA and its peer or a connected Accelerator Client.

If this is not the case, performance degradation or even the blocking of connections can take place. Considerations are:

3.1.1 Client-Side Networks

If a client wants to access something on their local LAN, traffic should **not** be routed via Replify Accelerator. Any IP addresses/ranges on the local client network should be added as *Excluded Application Servers*.

3.1.2 Geographically Distant Networks

Unless using a very high latency link such as a satellite connection, it will be the case that some application servers will be closer to the client and redirecting them via the VA will cause performance degradation.

For example, if a client is in London and a VA is in Singapore, it will likely be quicker to access a website in Paris directly rather than via a VA in Singapore.

In this scenario, application servers that will benefit from optimization should be specified, rather than using 0.0.0.0/0.

3.2 Applications that use Certificate Pinning

Some applications that use TLS for communication may have strict checks on the certificates used and can detect that Replify Accelerator is intercepting the traffic.

For applications like these, Replify Accelerator's *SSL optimization* should be turned off. This can be done by adding new Application Servers for these applications that have *SSL Optimization* disabled or by using SNI Filtering and adding the domain of these services to the *TCP Only* list.

3.3 Certificate Issues

If *Verify Server Certificate* is turned on for the application server, you may encounter issues optimizing some sites that use TLS. This may be because the CA certificates trusted by your browser differ from that which is available on the VA. The CA certificates for the affected application servers should be added to the VA using the *CA Certificates* page.

3.4 Applications that Perform User Profiling

Some websites may perform profiling of connections. For example streaming sites, online booking sites etc. They may detect that traffic originating from a Virtual Appliance based in the cloud is not end-user traffic and block this. Such sites should be excluded from SSL optimization.

3.5 Cache Pollution

If accelerating everything, it is possible that your cache will be filled up with content that isn't valuable. While the Replify Accelerator Intelligent Caching Engine protects popular or hot content, it may be worth considering whether all content needs cached. For example, video traffic may not be considered important and caching can be turned off for this in the *Cache Configuration* section of the *HTTP Settings* page.