Reverse Proxy Automated Installer

Introduction

Download the reverse-proxy installer and associated artifacts from https://software.cisco.com/download/home/283613135/type/284259728/release/12.6(2)

The content in this chapter is provided as a guidance for customers to install and configure the reverse-proxy artifacts provided by Cisco. We ship an embedded Nginx based OpenResty® reverse-proxy.

For information on the deployment details and the pre-requisites required, see the VPN-less access to Finesse desktop section.

We don't support install or configuration requests for custom reverse-proxy images and network configurations related issues. Queries that are related to this subject can be discussed on Cisco community forums.

For older format of VPN-less Finesse, see Cisco Finesse 12.6 ES07 Readme.

Prerequisites

Requirements

We recommend that you have knowledge of the following:

- Cisco Unified Contact Center Enterprise (Unified CCE) Release
- Cisco Finesse

- Linux administration
- Network administration and Linux network administration

Components Used

The information in this section is based on the following software and hardware versions:

- Cisco Finesse - 12.6 ES07 and above
- Cisco Unified Intelligence Center - 12.6 ES03 and above
- Cisco Identity Service - 12.6 ES03 and above
- Cisco Unified CCE and Packaged CCE - 12.0 and above
- Cisco Cloud Connect-12.6(2)
- ADFS 3.0 for being used as IDP in SSO deployments

Note To use VPN-Less access to Finesse desktop feature, you must upgrade Finesse, Cisco IdS, and Cisco Unified Intelligence Center to releases mentioned above.

If you are using LiveData 12.6(1), you must upgrade LiveData to releases mentioned above.

Packaged CCE and Unified CCE 2k deployments must be on 12.6 version of CCE to support the coresident deployment of Livedata (LD) and Cisco Unified Intelligence Center.

Background Information

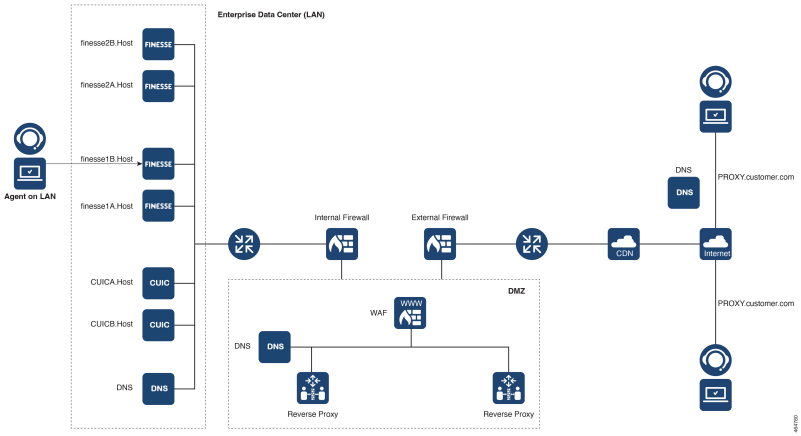
This deployment model is supported for the Unified CCE and Packaged CCE solutions.

Deployment of a reverse-proxy is supported (available from 12.6 ES07) as an option to access the Cisco Finesse desktop without connecting to a VPN. This feature provides the flexibility for agents to access the Finesse desktop from anywhere through the Internet.

To enable this feature, a reverse-proxy pair must be deployed in the Demilitarized Zone (DMZ).

Media access remains unchanged in reverse-proxy deployments. To connect to the media, agents can use Cisco Jabber over MRA solution or the Mobile Agent capability of Unified CCE with a Public Switched Telephone Network (PSTN) or mobile endpoint. The following diagram shows how the network deployment looks when you access two Finesse clusters and two Cisco Unified Intelligence Center nodes through a single HA pair of reverse-proxy nodes.

Concurrent access from agents on the Internet and agents who connect from LAN is supported as shown in the following image:



Note For more information on how to select an appropriate reverse-proxy that supports this deployment, see the section Reverse-Proxy Selection Criteria at *Security Guide for Cisco Unified ICM/Contact Center Enterprise, Release 12.6(1).*

Before you read this section, it is suggested to refer to the VPN-less Access to Finesse Desktop section. Also, see the *Security Considerations for Mobile Agent Deployments* section in *Security Guide for Cisco Unified ICM/Contact Center Enterprise, Release 12.6(1)*.

Reverse-Proxy Installer

The Reverse-Proxy Installer (referred to as Installer in this document) is an automated tool to make the reverse-proxy deployment for Cisco Unified Contact Center a simple and error free exercise.

This Installer replaces the older VPN-less Finesse configuration provided as part of the 12.6 ES 01 and ES07 releases. This required manual installation of the proxy along with editing of the provided rules for creating a VPN-less deployment.

The following are the Cisco Unified Contact Center solution components which are supported by the reverse-proxy Installer:

- Cisco Finesse
- Cisco Identity Service
- Cisco Unified Intelligence Center
- Cisco Unified Cloud Connect

Installer Components

Reverse-proxy within container

The Installer deploys the latest load-tested and qualified OpenResty® reverse-proxy binary, in a docker container and adds it within the required configurations automatically. (Follow the process in the sections below).

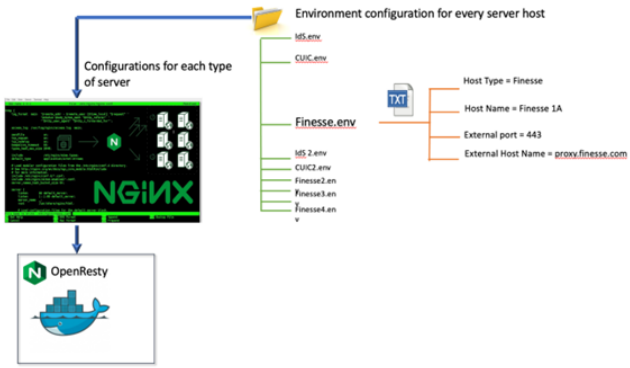
Reverse Proxy Automated Installer

This makes it very easy to run the reverse-proxy configuration that is required to support the VPN-less infrastructure. This simplifies the deployment immensely, without requiring compilation or the knowledge of NGINX configuration.You don't need to know how to compile or install the open source NGINX.

Containerized proxy instances are also more secure as it is locked down and provides an additional barrier for an intruder to overcome compared to a proxy process running on a bare metal operating system.

Proxy Configuration
Environment Files and Templates

The proxy configuration is split into environment configuration and proxy rule configurations is also known as templates.



The simple and unique environment values which differentiate each upstream component server is collated within the respective environment files, with one file for each upstream component server. These are automatically combined into the proxy rules for each unique type of upstream component server (For example: Cisco Finesse, Cisco Identity Service, and so on) by the Installer. These and are then pushed into the OpenResty® NGINX proxy which then proceeds to deploy these rules.

This allows easy instantiation of any number of supported upstream component server hosts as required by adding new environment files corresponding to new servers.

The proxy configuration rules, known as rule templates, contain the necessary NGINX rules to access the server and you don't have to understand or change them.

This also makes Installer upgrades easy as the environment files containing the configurations are rarely changed and can be persisted through multiple Installer changes without requiring much NGINX expertise.

Sample environment files

The Installer archives come with a sample environment that can be used as a starting point to create a new VPN-less Finesse deployment.

Each file the Installer contains is an environment for a unique type of upstream host natively supported by the configurations or reverse-proxy rules provided, such as Finesse, Unified Intelligence Center and Cisco IdS.

The administrator should clone this directory and create multiple copies of each environment corresponding to each upstream component host which has to be exposed via the proxy and supply this directory location to the Installer which will then proceed to instantiate each host configuration based on the supplied parameters contained in the environment file.

Reverse Proxy Installer components	Directory/ File Name	Description
Reverse proxy instance	*reverse-proxy-openresty-container/* |	Contains the docker container image that is used to create the container instance. This contains the OpenResty® NGINX proxy and other dependent libraries and modules **OpenResty® version packaged:** Latest CentOS version 7 based OpenResty® available in the docker hub.
Component Configuration Templates	*reverse-proxy-openresty-configs/*	Contains the OpenResty® configuration templates. These templates are used to generate final OpenResty® configurations from provided deployment configuration data.
Host OS configurations:	*reverse-proxy-os-configs/*	These templates are used to generate the final OpenResty® configurations from the deployment configuration data provided. Contains the OS configurations for hardening the host. CentOS version 7 is the only supported OS. You must manually install the OS configurations using the `install_os_settings.sh` that is available in this directory. **Note** The OS configurations are tested with OpenResty® version 1.19. These configurations are expected to work with other distributions. You may need to make some minor updates as required.
Installer configuration	*installer.env*	Contains the configuration data for the reverse-proxy Installer.

Reverse Proxy Installer components	Directory/ File Name	Description
Proxy configuration	*sample_envs/*	Contains the sample `env` configuration data for reference. Use this sample `env` when you are preparing the configuration data for your deployment.
Launcher scripts	*proxy_launcher.sh*	Launcher script to perform the **start|stop|reload|clear_cache** operations for a given Installer instance.
version.txt:		Contains Installer versions. Reverse-proxy Installer creates all the configurations afresh on every restart and no configuration is retained. Any additional changes made to the existing proxy configurations are lost after the restart.

Configure TCP rate limit

As an administrator, you can limit the incoming TCP requests to secure the application.

Upgrade notes for 12.6(1) ES01-7 based reverse proxy configurations

The configuration formats have been modified with the new Installer-based configuration and can't be reused as is. The information contained, however, can be easily extracted and plugged into the new Installer configuration, and the format will not be changed further.

The CLI configurations and proxy-map data need not be altered. However, as previously described, the manner in which the upstream component server hosts and their associated configurations are provided to the reverse-proxy instance has now changed. For more information, see the *Environment Files and Templates* section in Reverse-Proxy Installer, on page 3

The following are some important points to consider when you upgrade your reverse-proxy instance using the automated Installer:

- The data required in the component host environment will match the individual values replaced in the template configurations using the `## Must-Change` notations from the older configurations. This can be used as a reference to fill the data if required.
- *Tmpfs* is not used in the new Installer, and earlier configurations that were run with "*tmpfs*" can be ignored. However, disk subsystem slowness can impact the proxy performance and needs to be efficient.

Install and Operations

Setup reverse-proxy

To setup the reverse-proxy server, refer to the following sections:

Related Topics

Proxy Hardware requirements, on page 7
Prepare Host, on page 8
Install the reverse proxy Installer package, on page 8
Configure Host OS, on page 8
Configure proxy hardware resources and other critical runtime options, on page 9
Configure SSL certificates, on page 10
Configure the Mapping File
Configure deployment environment configurations, on page 13

Proxy Hardware requirements

The following are the hardware requirements to set up a reverse-proxy server for a 2K cluster. This includes Cisco Finesse, Cisco Identity Server (IdS), Cisco Unified Intelligence Center, Live Data, Enterprise Chat and Email, and Cisco Cloud Connect:

- CPU: 2vCPU for 2000 agents and 4vCPU for 4000 agents Deployment.
- Memory: 8 GB
- Disk: 80 GB
  - Cache disk space requirements for specific components:
    - **Finesse: 3 GB for one upstream**
    - **CUIC: 200 MB for one upstream**
    - **IdS: There's nothing cached.**
- Ethernet interfaces must be gigabit speed and connected to gigabit ethernet switches. 10/100 ethernet isn't supported.

Note Disk slowness can hamper the proxy performance. Please monitor the solution to ensure that the disk has adequate IO throughput.

Running the linux command **dd if=/dev/zero of=/root/junk bs=2k count=1000 oflag=dsync** should show a minimum of 5 MB per second of throughput and completion time of less than 0.3 seconds to write the data out.

Prepare Host

Steps to prepare the host are as follows:

Step 1 Install the latest build of CentOS Linux 7.x (7.9 or later).

Step 2 To Install the **envsubst utility**, run the command **yum install gettext**.

Step 3 Install the Docker. For instructions, see the Docker documentation at https://docs.docker.com/get-docker/

Note Uninstall podman on CentOS or RHEL. If already installed the podman conflicts with the Docker installation. Run the **yum erase podman buildah** command to uninstall the podman.

Step 4 Perform the post-installation steps to manage the Docker. For instructions, see the Post-installation steps for Linux section at https://docs.docker.com/engine/install/linux-postinstall.

Step 5 Run the **yum install logrotate -y command** command to install **logrotate** on the host.

Step 6 Run the following commands to uninstall or stop the firewall daemon service on CentOS:

- **sudo systemctl stop firewalld**
- **sudo systemctl disable firewalld**
- **sudo systemctl mask --now firewalld**

Step 7 Run the following commands to install the iptables service on CentOS:

- **sudo yum install iptables-services -y**
- **sudo systemctl start iptables**
- **sudo systemctl enable iptables**

Install the reverse proxy Installer package

To install the package:

Step 1 Download or copy the Installer zip on the host.

Step 2 Extract the archive (.zip) to the location where you need the Installer to be running from.

Configure Host OS

The following are the OS hardening configurations for the reverse-proxy host that are included in the `Installer/reverse-proxy-os-configs/` folder:

- Kerner hardening configurations that is **sysctl** configurations
- Logrotate config
- CentOS version 7

The Installer script is provided to install the required configurations automatically. Various options can be provided in the script to control the installation and configuration.

Run the **`install_os_settings.sh`** script with the required options. The different options are as follows:

```
USAGE: ./install_os_settings.sh [OPTIONAL_ARGS]
OPTIONAL_ARGS: -k -l -i -p <source-ip1> -p <source-ip2> ... -s <interface1> -s <interface2>
 ... -r <interface1:source-ip1> -r <interface1:source-ip2> -r <interface2:source-ip1> ...
-k: configure kernel hardening
-l: configure logrotate for given log directory
-i: configure iptables
-p: allowed source ip for ICMP ping messages. By default ICMP ping is blocked for all hosts.
 This option is ignored if -i or iptables configuration option is not given
-s: network interface name to allow SSH access to. By default ssh access is blocked for all
 hosts. This option is ignored if -i or iptables configuration option is not given
-r: disable rate limits for a source-ip on an interface. Provide value as INTERFACE:SOURCE_IP.
 By default rate limits applies for all. This option is ignored if -i or iptables
configuration option is not given

Example usage: ./install_os_settings.sh -k -l ~/reverse_proxy/proxy25.autobot.cvp/logs -i
-p allowed.host.for.ping.1 -p allowed.host.for.ping.2 -s ssh_interface1 -s ssh_interface2
-s -s ssh_interface3 -r interface1:host1 -r interface1:host2 -r interface2:host1
```

Note If you're installing the reverse-proxy to proxy a specific component, for example Cloud Connect, you must not install the iptable rules for other components. As an administrator, you must select the component rules and the port values to be installed and configured for the iptables. The port value may change depending on the deployment. The script doesn't control the customization of iptable rules.

Configure proxy hardware resources and other critical runtime options

The Installer script `installer/proxy_launcher.sh` that is used to deploy the reverse-proxy takes the following input arguments:

- `installer.env`: Path to `installer.env` file containing Installer configuration data.
- `proxy_env_dir/`: Path to `proxy_env_dir/` file containing proxy configuration data.

Note The `installer.env` file contains properties to configure Installer options. The sample file is provided in the Installer package. Use it as a reference to prepare the actual configuration.

The steps to configure the proxy hardware resources and other runtime options are as follows:

Step 1 Copy the sample file `installer.env` to any other directory, and rename it. If there are multiple proxy instances running on the same host, use the proxy name or the customer name that maps to a particular proxy instance.

Step 2 Modify the installer options as required. Options are included in the configuration file with their intended purpose.

Configure SSL certificates

The environment configuration file for each component includes an **SSL CONFIG** section that has configurations to set up the SSL connector for the component. In addition, the configurations are used to configure the following:

- Either custom certificates that you have generated manually or certificates that the Installer has generated can be used for reverse-proxy.
  - If you choose to use the custom certificate that is either CA-signed or self-signed, which you've generated, place the certificate inside the ssl directory mentioned in the option *HOST_SSL_VOL(defaults to ${HOST_WORKING_DIR}/ssl)*.
  - You can also allow the Installer to generate the self-signed certificate. When starting the *Installer/proxy_launcher.sh* script, set the **CREATE_SELF_SIGNED_SSL_CERT** option to true. For more information, see Configure proxy hardware resources and other critical runtime options. If the required certificate and key names aren't present in the ssl directory, the Installer generates the certificate and includes it in the ssl directory mentioned in the option *HOST_SSL_VOL(defaults to ${HOST_WORKING_DIR}/ssl)*. The Installer doesn't overwrite the existing files.

  These certificates are used to configure the SSL connector for individual component configurations.
- Supported TLS protocol versions
- Supported TLS ciphers
- SSL session cache size and timeout
- SSL stapling configurations
- Mutual TLS authentication for upstream connections: By default, this option is disabled. To enable this option, modify the following configurations:
  - Set the **NGX_PRXY_SSL_VERIFY** option to "**on**"
  - **NGX_PRXY_SSL_TRUST_CRT** → Trust file containing certificate of upstream being proxied. Certificate from this file will be verified by NGINX against what is provided by upstream during the TLS handshake.

Note Self-signed certificates are to be used only for testing and development purposes and CA-signed certificates are mandatory for production deployments. If the certificate received from the CA isn't a certificate chain containing all the respective certificates, compose all the relevant certificates into a single certificate chain file.

Create Custom Diffie-Hellman Parameter

1. Create a custom Diffie-Hellman parameter by using the following commands:

```
openssl dhparam -out /usr/local/openresty/nginx/ssl/dhparam.pem 2048
chmod 400 /usr/local/openresty/nginx/ssl/dhparam.pem
```

2. Modify the server configuration to use the new parameters in the file `/usr/local/openresty/nginx/conf/conf.d/ssl/ssl.conf` by using the following command:

```
ssl_dhparam /usr/local/openresty/nginx/ssl/dhparam.pem;
```

Enable OCSP Stapling

Note To enable the Online Certificate Status Protocol (OCSP) stapling, the server should be using a CA-signed certificate. Also, the server should have access to the CA which signed the certificate.

Use the following parameters to configure stapling **NGX_SSL_STAPLING & NGX_SSL_STAPLING_VERIFY** on the respective component's `env` files. By default, they are set to "**off**".

Configure mutual TLS authentication between reverse-proxy and components

The **mutual TLS (mTLS)** is a standard security requirement for connections established from DMZ into the data center. For more information, see Nginx CIS benchmarks-https://www.cisecurity.org/benchmark/nginx

For mTLS, both the server and client must be pre-configured with mutual information about each other. Also, the mutual certificates must be properly verified. So the term **"mutual TLS (mTLS)"**. A properly configured proxy server will be able to circumvent TCP rate limits and provide the client IP to the server for logging purposes. As a result, it's critical to verify the proxy identity before connecting as a reverse-proxy. For security reasons, by default this feature is turned on.

This requires the upstream component certificates to be made available to the proxy and vice-versa. By default, reverse-proxy establishes verified TLS connections to the upstream server and it's the proxy verification at the client which is optional. So, this must be enabled at the upstream client server.

Enabling mutual TLS

The mTLS must be enabled at the upstream component servers using the provided CLI.

Use the **utils system reverse-proxy client-auth enable** CLI to enable proxy certificate verification at the upstream component server.

After running the CLI, upload the proxy SSL certificate corresponding to the reverse-proxy hostname that is used to connect to the same server. This can be used to verify TLS connections when the reverse-proxy attempts to establish an upstream connection.

Configure the Mapping File

Refer to Host-Mapping file for network translation.

Use reverse-proxy as the Mapping File server

The following steps are required only if the reverse-proxy is also used as the proxy mapping file host:

1. Configure the reverse-proxy hostname in the domain controller used by the Finesse, Cisco Unified Intelligence Center, and IdS hosts such that its IP address can be resolved.
2. Upload the generated OpenResty® Nginx signed certificates on both the nodes under tomcat-trust of `cmplatform` and restart the server.

3. Update the **Must-change** values in `<NGINX_HOME>/html/proxymap.txt`.
4. Reload OpenResty® Nginx configurations with the `nginx -s reload` command.
5. Use the `curl` command to validate if the configuration file is accessible from another network host.

CentOS 8 Kernel Hardening

If the operating system is CentOS 8 and the installations use a dedicated server for hosting the proxy, harden the kernel by using the following `sysctl` configurations:

```
## Configurations for kernel hardening - CentOS8. The file path is /etc/sysctl.conf
## Note that the commented configurations denote that CentOS 8's default value matches
## the recommended/tested value, and are not security related configurations.

# Avoid a smurf attack
net.ipv4.icmp_echo_ignore_broadcasts = 1
# Turn on protection for bad icmp error messages
net.ipv4.icmp_ignore_bogus_error_responses = 1

# Turn on syncookies for SYN flood attack protection
net.ipv4.tcp_syncookies = 1

# Turn on and log spoofed, source routed, and redirect packets
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1

# Turn off routing
net.ipv4.ip_forward = 0
net.ipv4.conf.all.forwarding = 0
net.ipv6.conf.all.forwarding = 0

net.ipv4.conf.all.mc_forwarding = 0
net.ipv6.conf.all.mc_forwarding = 0

# Block routed packets
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0

# Block ICMP redirects
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0

# Filter routing packets with inward-outward path mismatch(reverse path filtering)
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

# Router solicitations & advertisements related.
net.ipv6.conf.default.router_solicitations = 0
net.ipv6.conf.default.accept_ra_rtr_pref = 0
net.ipv6.conf.default.accept_ra_pinfo = 0
net.ipv6.conf.default.accept_ra_defrtr = 0
net.ipv6.conf.default.autoconf = 0
net.ipv6.conf.default.dad_transmits = 0
net.ipv6.conf.default.max_addresses = 1
```

```
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.default.accept_ra = 0

# Backlog - increased from default 1000 to 5000.
net.core.netdev_max_backlog = 5000

# Setting syn/syn-ack retries to zero, so that they don't stay in the queue.
net.ipv4.tcp_syn_retries = 0
net.ipv4.tcp_synack_retries = 0

# Max tcp listen backlog. Setting it to 511 to match nginx config
net.core.somaxconn = 511

# Reduce the duration of connections held in TIME_WAIT(seconds)
net.ipv4.tcp_fin_timeout = 6

# Maximum resources allotted
# fs.file-max = 2019273
# kernel.pid_max = 4194304
# net.ipv4.ip_local_port_range = 32768 60999

# TCP window size tuning
# net.ipv4.tcp_window_scaling = 1
# net.core.rmem_default = 212992
# net.core.rmem_max = 212992
# net.ipv4.tcp_rmem = 4096 87380 6291456
# net.ipv4.udp_rmem_min = 4096
# net.core.wmem_default = 212992
# net.core.wmem_max = 212992
# net.ipv4.tcp_wmem = 4096 16384 4194304
# net.ipv4.udp_wmem_min = 4096
# vm.lowmem_reserve_ratio = 256 256 32 0 0
# net.ipv4.tcp_mem = 236373 315167 472746

# Randomize virtual address space
kernel.randomize_va_space = 2

# Congestion control
# net.core.default_qdisc = fq_codel
# net.ipv4.tcp_congestion_control = cubic

# Disable SysReq
kernel.sysrq = 0

# Controls the maximum size of a message, in bytes
kernel.msgmnb = 65536

# Controls the default maximum size of a message queue
kernel.msgmax = 65536

# Controls the eagerness of the kernel to swap.
vm.swappiness = 1
```

Reboot the server after you make the recommended changes.

Configure deployment environment configurations

The environment configuration data is the main input that the Installer needs to generate the actual proxy configurations from the templates. There is a sample environment data, `Installer/sample-envs/` that is provided within the installer.zip. The sample environment data contains up-to-date reference **envs** for all supported components. The following are the contents of the sample **env** directory:

```
installer/sample_envs/
 |- core.env
 |- dirs.env
 |- finesse.env
 |- ids.env
 |- cuic.env
 |- livedata.env
 |- chat.env
 |- cloudconnect.env
```

These property files are divided into three categories:

- The `core.env` : **Mandatory**: File containing OpenResty® NGINX core configurations data. This is required to configure OpenResty® NGINX core configurations.

  This environment configuration file contains data for *reverse-proxy core configuration template files*. Core configuration files include details specific to the running NGINX instance and is applied generally to all the components until or unless it is overridden at the component level.

  The core configuration template file includes:

  - `cache.conf`: Template file containing cache configurations
  - `common.conf`: Template file containing common configurations
  - `logging.conf`: Template file containing logging configurations
  - `maps.conf`: Template file containing constants and other variable configurations
  - `rate_limit.conf`: Template file containing rate limit configurations
  - `static.conf`: Template file containing static configurations
  - `ssl_config.conf`: Template file containing ssl connector configurations for common server blocks such as status endpoint and static files endpoint

  Values provided in the `core.env` file is used to substitute all the place holders in the above files.

- `dirs.env` : **Mandatory**: File containing various OpenResty® NGINX directory paths as per OpenResty® installation. This is required to configure directory paths in the configuration templates.

  This environment data contains information regarding the OpenResty® installation directory structure. The default values are included as per the default OpenResty® installation.

  ```
  • # Directory location for various openresty folders required to
  • # configure configurations accordingly.
  •
  • # Home directory for openresty nginx installation
  • NGX_HOME="/usr/local/openresty/nginx"
  • # Openresty directory containing static resources
  • NGX_HTML_DIR="${NGX_HOME}/html"
  • # Openresty directory containing lua resources
  • NGX_LUA_DIR="${NGX_HOME}/lua"
  • # Cache directory where various resources for components will be cached
  • NGX_CACHE_DIR="${NGX_HOME}/cache"
  • # Openresty directory containing SSL resources like certs, keys etc.
  • NGX_SSL_DIR="${NGX_HOME}/ssl"
  • # Openresty directory where openresty logs will be put
  • NGX_LOG_DIR="${NGX_HOME}/logs"
  • # Openresty directory containing NGINX configurations - core configs, components
  configs etc.
  NGX_CONF_DIR="${NGX_HOME}/conf"
  ```

- `component envs` : **Optional**: Files containing configuration data for proxied solution components such as Finesse, Cisco IdS, Unified Intelligence Center, Live Data, Cisco IM&P, and so on. One environment configuration file must be created per upstream solution component that is being proxied.

  Some properties are mandatory in component environment config files, without which the configurations are be generated for those components. The properties are as follows:

  - `TEMPLATE_TYPE`: Defines which type of upstream component is being configured, so that the correct templates can be referred to generate the actual configurations. The value can be `finesse`, `ids`, `cuic`, `livedata`, or `chat`
  - `NGX_COMP_DIR_NAME` : Defines the output directory where configuration files for the component will be generated. The final output location for the files will be `./configs_out/conf/components/<NGX_COMP_DIR_NAME>/`. Also, this directory is used to form the file including the paths in various configuration files of the component.

Note Ensure that each environment configuration file has a unique output directory name (`NGX_COMP_DIR_NAME`) and hostname (`NGX_COMP_HOSTNAME`).

Other properties are different for different components and the default values for all the components are provided in their respective `env` files.

The following are the steps to configure these options:

Step 1 Copy the **installer/sample-envs/** directory and the **installer/installer.env** file to a separate directory and modify it.

Step 2 After the files are copied to a new directory at **~/proxy_config/proxy_instance_name**, rename the files such that they can be mapped to a running proxy instance.

Step 3 Modify the **core.env** file for OpenResty® configuration.

For most of the options the default values can be used. However, for some of the options you must change the values as per the deployment.

Step 4 Validate all the property values given in the **core.env** file.

Note Do Not Rename this file.

Step 5 Provision is available in the **dirs.env** file to deploy the configurations on a custom NGINX installation. If you choose to use the Installer as is, avoid modifying the **dirs.env** file.

Note Do Not Rename this file.

Step 6 Retain the `.env` files of components that you require and delete the remaining files.

For example, for a proxied Finesse cluster running on non-sso mode with Live Data and Unified Intelligence Center reports, you must retain the **finesse.env, cuic.env,** and **livedata.env** files. You must delete the remaining files **chat.env, ids.env** and **cloudconnect.env**. The remaining `.env` files present in the directory is processed by the Installer.

Step 7 Rename the component `.env` files as per their hostnames, as it is easy to identify them. Modify the component `.env` file values as per the requirement of the deployment. Generally, you can modify only the hostname values as per the deployment and retain the default values for the other options.

Step 8 Property description in all the `.env` files should be self-explanatory and it should provide the information regarding the purpose and the usage of a given property. Also, **Do not modify any property name** or **Delete any property from the `.env` file.**

Note All the properties are essential for the Installer, and incase of any missing property, the Installer will not be able to open the proxy instance. Override or change the default values only for the required properties.

Add or Remove the Unified CCE solution component

Any number of Unified CCE solution components can be proxied through the installer.

To add or remove any of the component proxies, the corresponding component environment configuration file must be available in the **env** directory. The Installer generates the proxy configurations for all the required components from the beginning as per the contents in the **env** directory.

Configure Auth URL for components

The component configuration file has an option to redirect to Finesse nodes' authorization url (auth url) to perform the authentication at the proxy. Configure this for the component configuration files as per the deployment, to redirect them to the same cluster Finesse node which contains user data. For more information, see NGX_AUTH_URL= https://reverseproxy.host.domain:8445/finesse/api/UserAuth.

Multi-cluster deployment

The reverse-proxy installer supports Unified CCE or Packaged CCE that are larger than 2k deployments. These deployments must expose multiple Finesse nodes to the Agents over the internet and needs extra CUIC nodes.

These additional nodes are supported by multiple pairs of reverse-proxy or by configuring the extra nodes. The extra nodes work as added upstream servers on the same proxy pair using a single HA pair of the reverse-proxy.

Adding more upstream servers is as simple as creating a new environment (env) file. The **env** file corresponds to the upstream server type and modify specific details such as its hostname.

For example, a deployment containing three Finesse clusters must have three Finesse **env** files in the **env** directory as follows:

- Side A proxy **env** directory:
  - finesse1a.env
  - finesse2a.env
  - finesse3a.env
- Side B proxy **env** directory:
  - finesse1b.env
  - finesse2b.env
  - finesse3b.env

You can extended the same for multiple clusters of other components as required.

Consider multi-cluster deployments for the port and the hostname management. The prerequisite for the installer to communicate through proxy is that the hostname and the port pair are unique for a component across all other components.

To plan the hosts and ports used in the individual component **env** files, see the Port Management section.

Starting the reverse-proxy

To start the proxy instance from the Installer, we must open the script with the required `installer.env` file from the `proxy_env_dir/` path as input arguments. Check the following steps:

```
USAGE: ./proxy_launcher.sh [options...] (start|stop|reload|clear_cache)
Options: -e <ENV-DIR> -i <INSTALLER-ENV-FILE1> -i <INSTALLER-ENV-FILE2> ...
INSTALLER-ENV-FILE: Mandatory : Installer env files ... Multiple files can be provided to
override base env
ENV-DIR: Mandatory for start, Optional for other actions : Reverseproxy environment config
 data directory
Example usage: ./proxy_launcher.sh -e /path/to/env/dir -i /installer/env/1 -i
/installer/env/override/1 -i /installer/env/override/2 start
```

When the command **start** is initiated, the Installer performs the following:

Step 1 Validates if the input arguments are correct, directories and the mandatory files are available.

Step 2 Creates the required working directory, and volume mounts on the host as per the **Installer.env** file entries.

Step 3 Generates the required OpenResty® configurations and runs the command `run.sh` inside the `reverse-proxy-openresty-configs/` directory.

Step 4 Modifies the generated configurations to their respective directories inside the working directory.

Step 5 Creates the self-signed SSL certificate for the reverse-proxy to use it if necessary. Configures it in the `installer.env` file.

The SSL certificate is generated only if there's no other file with the same filename in the directory or no other file is overwritten.

Note Load docker image is provided as part of the Installer. This can be overridden from the `installer.env` file. You can also choose to load a different image.

Step 6 It runs the container with the required arguments as per the Installer configuration data.

Serviceability

Bootstrap checks or validations

The Installer validates the configurations that are provided through the *.env* files and stops the deployment if it identifies certain common errors. This is done to prevent lengthy debugging on the configurations provided, which can be easily identified in the validation phase.

The following are the errors which are currently identified and reported during the validation phase.

Scenario	Sample Error Message
An unknown template type is mentioned on the *.env* file which isn't known to the Installer.	[ERROR]: Unknown TEMPLATE_TYPE cuic_1230 found in file cuic.env. Exiting.
The *.env* file doesn't contain the property TEMPLATE_TYPE which identifies the type of upstream component.	[ERROR]: TEMPLATE_TYPE variable missing in file cuic.env. Exiting.
A particular variable isn't present in the primary *.env* file for the template type. However, it's available in a particular *.env* file that is being processed in the custom `env` directory.	[ERROR]: Below unused variable found in *./sample_envs/*. Exiting. **NGX_FIN_TEST_HOSTNAME**
The **NGX_LOAD_BALANCER_IPS** contains values which can't be parsed as a valid IP.	[ERROR]: **NGX_LOAD_BALANCER_IPS** should contain only IP addresses. Exiting.
The **NGX_LOAD_BALANCER_REAL_IP_HEADER** is configured but the **NGX_LOAD_BALANCER_IPS** isn't configured.	[ERROR]: **NGX_LOAD_BALANCER_REAL_IP_HEADER** should be configured only when **NGX_LOAD_BALANCER_IPS** is configured. Exiting.
The **NGX_LOAD_BALANCER_REAL_IP_HEADER** is empty but the **NGX_LOAD_BALANCER_IPS** is configured.	[ERROR]: **NGX_LOAD_BALANCER_REAL_IP_HEADER** is empty. It should contain header details when **NGX_LOAD_BALANCER_IPS** is configured, Exiting.
One of the mandatory variables isn't configured. (Currently, limited to host and port of the upstream).	[ERROR]: **NGX_PRXY_CHAT_HOSTNAME**'s value is not configured. Exiting.
Same variable is encountered more than once in the *.env* file that is being processed.	[ERROR]: **NGX_PRXY_CLOUDCONNECT_HOSTNAME**'s value is configured in multiple places. Exiting.
Mandatory variable is configured more than once.	[ERROR]: **NGX_FIN_HOSTNAME**'s configured more than one time. Exiting.
Duplicate environment variable.	[ERROR]: Following variables were found to be duplicate in file `sample_env/finesse.env`. Exiting.
More than one version for Unified Intelligence Center or LiveData is configured.	[ERROR]: Multiple versions of env files detected for Unified Intelligence Center, retain one type and retry. Exiting.
More than one Cisco IdS instance is configured. (Each side of the proxy should have only a single instance of IdS configured).	[ERROR]: Number of Cisco IdS instance should not be more than 1. Exiting.
The **.env** file is not readable.	[ERROR]: File sample_env/core.env does not exist or does not have appropriate permissions. Exiting.

Scenario	Sample Error Message
The primary template is altered. This is just a warning, it won't exit the installation.	[!!! WARNING !!!] Primary templates have been altered. Note: Some of the pre-install checks that are based on the templates configurations will be skipped.
The primary **env** file is altered. This is just a warning. It won't exit the installation.	[!!! WARNING !!!] Primary master_env have been altered. Note: Some of the pre-install checks that are based on the templates configurations will be skipped.
The custom **env** directory which is passed as a run time option to the Installer is missing.	[ERROR]: Directory sample_env/core doesn't exist. Exiting.
Certificate-based authentication is enabled for a particular upstream server (Using NGX_PRXY_SSL_VERIFY="on"), without defining the certificate path.	[ERROR]: Mutual Transport Layer Security validation is enabled for finesse, but the upstream server certificate path in `NGX_PRXY_SSL_TRUST_CRT` is empty. Exiting.
Certificate-based authentication is enabled for a particular upstream server. (Using NGX_PRXY_SSL_VERIFY="on"). However, the certificate isn't present, nonreadable, or empty.	[ERROR]: Mutual TLS validation is enabled for Finesse, but the upstream server certificate /root/reverse_proxy/contactcenter-reverseproxy/ssl/upstream_finesse_trust.crt is not present, not readable or invalid. Exiting.

Launcher logs

Proxy instance launcher logs can be located at *${HOST_WORKING_DIR}/logs/openresty_launcher.log*. During the NGINX startup, check the logs to see if there are any error information inside the container instance.

Openresty pid file is also located in the same folder at ***${HOST_WORKING_DIR}/logs/openresty.pid.***

Access and error logs

You can locate the Nginx access and error logs for a given proxy instance at the `logs` directory inside the proxy working directory as `${HOST_WORKING_DIR}/logs/access.log` and `${HOST_WORKING_DIR}/logs/error.log`. Check these log files for any debugging information about the OpenResty® startup.

To identify the Digital Routing task requests, the reverse-proxy server generates access logs with the `trackingId` field. The following is the snippet of the `access.log` with the `trackingId` field:

```
[09/Feb/2023:07:24:25 +0000] conn_stats:"7 : 1" client:"35.168.152.254"
host:"pccedrdmzproxy-cc.cisco.com" host_addr:"173.39.15.27"
host_to_upstream:"pccedrdmzproxy-cc.cisco.com->10.10.10.95:8445"
user:"-" server_block:"173.39.15.27:443" request:"POST /drapi/v1/tasks HTTP/1.1" requestid:"-"
 server_cache_bypass:"-" cookie:"-" user_agent:"Apache-HttpClient/4.5.2 (Java/1.8.0_242)"
referer:"-" cache_status:"-" rsp_status:"201(201)" body_bytes_sent:"56"
time_taken:"0.021(0.022)" up_connect_time:"0.002" up_header_time:"0.022" up_bytes_sent:"1297"
 up_bytes_rcvd:"852" trackingId:"WebexConnect_ea54eac0-1d2a-4e09-9fa2-cb212dad13df"
```

If there are failures in the Digital Routing task requests, the reverse-proxy server generates error logs with the `trackingId` field only when you set the trace level to debug.

To enable the debug trace level for the reverse-proxy server:

1. In the `"<reverse_proxy_installed_dir>/conf"` directory, locate and open the **nginx.conf** file.
2. In the **nginx.conf** file, find the statement `[error_log ${NGX_LOG_DIR}/error.log info;]`.

3. Change the trace level from `info` to `debug` as follows: `[error_log ${NGX_LOG_DIR}/error.log debug;]`.
4. Reload the reverse-proxy server for the change to take effect.

The following is the snippet of the `error.log` with the `trackingId` field:

```
2023/02/14 08:01:59 [debug] 206#206: *5 [lua] log_dr_requests.lua:4:  conn_stats:5:1
client:172.16.102.61 host:173.39.15.27 host_addr:173.39.15.27
host_to_upstream:173.39.15.27->10.10.10.95:8445
user:nil server_block:pccedrdmzproxy-cc.cisco.com:443 request:GET /drapi/v1/tasks?from=0
HTTP/1.1 requestid:nil server_cache_bypass:nil cookie:nil user_agent:PostmanRuntime/7.29.2
 referer:nil
cache_status:nil rsp_status:200(200) body_bytes_sent:46 time_taken:0.004(0.005)
up_connect_time:0.002 up_header_time:0.005 up_bytes_sent:3411 up_bytes_rcvd:733
trackingId:WebexConnect_ea54eac0-1d2a-4e09-9fa2-cb212dad13df
```

IP blocking logs

A separate log file is maintained to track the IPs that block the running proxy instance at `${HOST_WORKING_DIR}/logs/blocking.log`. This file can be supplied to the tools such as *fail2ban* to automate the blocking of IP addresses at IP table level.

Client IPs are blocked if a client makes several failed authentication requests in a given time interval.

Syslogs

Syslogs are released by the reverse-proxy. By default, syslogs are pushed to the local endpoint. However, proxies can be configured to push this to the remote endpoint.

Syslogs are released when the client IP is blocked by the reverse proxy.

Reloading configuration and clearing cache

Static file hosting

Reverse-proxy provides provision to host the static files as required at `${HOST_WORKING_DIR}/html`. You can add any of the static files that must be accessed through proxy such as *proxymap.txt*. These files are accessible through a static file access endpoint provided by the proxy. The endpoint hostname and the port are configurable through the *core.env* file.

By default, you can access the static files deployed on the reverse-proxy at the URL *https://[ip-of-proxy-host]:10000/staticfile*.

To configure access from a different port, use the *NGX_PRXY_STATIC_FILES_PORT* option provided in the **core.env** file.

The static file port isn't opened by default in the IP tables. If necessary, it must be explicitly opened by the administrator. The same must be opened in the DMZ firewall to access from the internet.

Note While enabling access to this port over the internet, you must be cautious as this port isn't covered under DOS preventive measures.

Reverse-proxy caching

Each and every proxy instance caches the files as specified by different components inside the `${HOST_WORKING_DIR}/cache` directory. Inside the cache directory, every upstream has a separate directory where the cache files for that upstream is present. The sample on how the cache is maintained is as follows:

```
${HOST_WORKING_DIR}/cache
|- client_temp
|- proxy_temp
|- finesse125.autobot.cvp
|- desktop
|- layout
|- openfire
|- rest
|- shindig
|- cuic126.autobot.cvp
|- cuic
|- cuicdoc
```

To get the latest upstream resources, the cache has to be cleared. Administrator can either do this manually by clearing all the files inside each and every directory as required or can run the script provided inside the container to clear the cache automatically.

```
docker exec <PROXY_INSTANCE_NAME>
        /usr/local/openresty/nginx/sbin/openresty_launcher.sh clear_cache
```

Caching behaviors such as cache expiration, cache sizes, and so on, can be configured from the individual component **env** files. The configuration options for different components' **env** files are as follows:

- Finesse
    - NGX_FIN_DESKTOP_CACHE_SIZE
    - NGX_FIN_DESKTOP_CACHE_MAX_SIZE
    - NGX_FIN_DESKTOP_CACHE_INACTIVE_DURATION
    - NGX_FIN_SHINDIG_CACHE_SIZE
    - NGX_FIN_SHINDIG_CACHE_MAX_SIZE
    - NGX_FIN_SHINDIG_CACHE_INACTIVE_DURATION
    - NGX_FIN_OPENFIRE_CACHE_SIZE
    - NGX_FIN_OPENFIRE_CACHE_MAX_SIZE
    - NGX_FIN_OPENFIRE_CACHE_INACTIVE_DURATION
    - NGX_FIN_REST_CACHE_SIZE
    - NGX_FIN_REST_CACHE_MAX_SIZE
    - NGX_FIN_REST_CACHE_INACTIVE_DURATION
    - NGX_FIN_LAYOUT_CACHE_SIZE
    - NGX_FIN_LAYOUT_CACHE_MAX_SIZE
    - NGX_FIN_LAYOUT_CACHE_INACTIVE_DURATION
- CUIC

- NGX_CUIC_CACHE_SIZE
- NGX_CUIC_CACHE_MAX_SIZE
- NGX_CUIC_CACHE_INACTIVE_DURATION
- NGX_CUICDOC_CACHE_SIZE
- NGX_CUICDOC_CACHE_MAX_SIZE
- NGX_CUICDOC_CACHE_INACTIVE_DURATION

Use configurations with custom NGINX installation

The proxy Installer package can be deployed as a standalone. However, you can use the following steps to deploy only the generated configuration with the third-party NGINX installations:

Step 1 Navigate to the directory **`reverse-proxy-openresty-configs/`** inside the proxy Installer.

Step 2 For third-party NGINX installations, ensure to change the **dirs.env** as per the NGINX installation directory structure.

Step 3 Generate the configurations by running the command **./run.sh <ENV-DIR>** where the **ENV-DIR** is the path of the directory containing the environment configuration data files.

Step 4 Copy the **conf, html, lua** folders from the **~/configs-out** directory to the **NGX_HOME** directory.

Note This requires NGINX installation with Lua support.

Upstream component configuration specifics

Verifying Reverse-Proxy Configuration

Finesse

Step 1 From the DMZ, open *https://<reverseproxy:port>/finesse/api/SystemInfo* and check if it's reachable.

Step 2 Check if the **<host>** values in both **<primaryNode>** and **<secondaryNode>** are valid in the reverse-proxy hostnames. It shouldn't be the Finesse hostnames.

Note
- If CORS status is **enabled**, you must explicitly add the reverse-proxy domain name to the list of CORS trusted domain names.
- Reverse-proxy supports a maximum of 8000 folders (including sub-directories) in the `finesse/3rdpartygadget` folder.

Cisco Unified Intelligence Center and LiveData

Step 1 If you find the Finesse hostnames in the response instead of reverse-proxy hostnames, validate the proxy-mapping configurations. Also, check if the allowed hosts are properly added in Finesse servers as described in the *Populate Network Translation Data* section.

Step 2 If the LiveData gadgets load properly in the Finesse Desktop, the CUIC and LiveData proxy configurations are correct.

Step 3 To validate the Cisco Unified Intelligence Center and LiveData configurations, make the HTTP requests from the DMZ to the following URLs and check if they are reachable:

- https://<reverseproxy:cuic_port>/cuic/rest/about
- https://<reverseproxy:ldweb_port>/livedata/security
- https://<reverseproxy:ldsocketio_port>/security

Cisco Identity Service

To validate the Cisco IdS configuration, perform the following steps:

Step 1 Log in to the Cisco IdS Admin interface at **https://<ids_LAN_host:ids_port>:8553/idsadmin** from the LAN because the admin interface isn't exposed over reverse-proxy.

Step 2 Choose **Settings** > **IdS Trust**.

Step 3 Verify that the proxy cluster publisher node is listed on the Download SP metadata page, and click **Next**.

Step 4 Verify that the IDP proxy is correctly displayed (if configured on the Upload IDP metadata page) and click **Next**.

Step 5 Initiate test SSO through all proxy cluster nodes from the Test SSO page and validate that all are successful. This requires client system connectivity to reverse-proxy nodes.

Security

Authentication

Note Authentication isn't enabled for Digital Channel requests accepted by the proxy.

Proxy supports the authentication at the Edge. Authentication is supported for Single Sign-On (SSO) and Non-SSO deployments. Authentication is enforced for all the requests and the protocols that are accepted by the proxy before they are forwarded to the upstream component servers.

The authentication is enforced by the component servers locally. All authentication uses the common Finesse sign-in credentials to authenticate the requests. Persistent connections, such as websockets which rely on application protocols such as Extensible Messaging and Presence Protocol (XMPP) for authentication, the

connections are authenticated at the proxy by validating the IP address. Connections from an IP address are allowed only if there's a successful application authentication made from the IP address, before initiating the websocket connection.

Non-SSO authentication

Non-SSO authentication doesn't require any extra configurations. It works without the NGINX configuration scripts after the required script replacements are made. Authentication relies on the username and password used to sign in to Finesse.

Access to all the endpoints are validated with Finesse authentication services. The list of valid users is cached at the proxy locally (updates the cache every 15 minutes), which is used to validate the user in a request. User credentials are validated by forwarding the request to the configured Finesse URI and thereafter the credential hash is cached locally (cached 15 minutes) to authenticate new requests locally. If there's any change to the username or password, it takes effect only after 15 minutes.

SSO authentication

SSO authentication for Cisco IdS 12.6(1) (latest ES) requires that the administrator configure the Cisco IdS token encryption key at the NGINX server within the configuration file. You can obtain the Cisco IdS token encryption key from the Cisco IdS server with the **show ids secret** CLI command. They key has to be configured as part of the **core.env (NGX_JWT_SECRET option)** file that the administrator has to perform in the scripts before the SSO authentication can work.

For Cisco IdS in 12.6.(2) and above this need not be configured, as the proxy automatically add this information from the backend. For more information on Single Sign-On, see Cisco Unified Contact Center Enterprise Features Guide.

The SSO user guide for the Cisco IdS SAML configurations to be performed for the proxy resolution to work for Cisco IdS. After SSO authentication is configured, a valid pair of tokens can be used to access any of the endpoints in the system. The proxy configuration validates the credentials by intercepting the token retrieval requests made to Cisco IdS or by decrypting valid tokens and thereafter caching them locally for further validations.

Authentication for Websocket connections

Websocket connections can't be authenticated with the standard authorization header, as custom headers aren't supported by original websocket implementations in the browser. Application-level authentication protocols, where the authentication information contained in the payload doesn't prevent websocket connection establishment. So, malicious entities can render DOS or DDOS attacks just by creating innumerable connections to overwhelm the system.

To mitigate this possibility, the NGINX reverse-proxy configurations provided have specific checks to allow the websocket connections to be accepted ONLY from those IP addresses which have successfully made an authenticated REST request before establishing the websocket connection. It implies that the clients which attempt to create websocket connections before a REST request is issued, gets an authorization failed error and isn't the supported usage scenario.

Validating unauthenticated static resources

All valid endpoints that can be accessed without any authentication are actively tracked in the ES04 scripts . If invalid URIs are requested to these unauthenticated paths, they are rejected without sending the requests to the components' servers.

Brute Force attack prevention

The proxy authentication scripts actively prevent brute force attacks which can be used to guess the user password. It does this by blocking the IP address which is used to access the service. After some failed attempts in a short time, these requests are rejected with the HTTP error 418. You can access the details of the blocked IP addresses from the **${HOST_WORKING_DIR}/logs/blocking.log** and **${HOST_WORKING_DIR}/logs/error.log** files.

You can configure the threshold for failed requests, the time interval for the threshold, and the blocking duration. The configurations are present in the **core.env** file. The following are the options:

- **NGX_CLIENT_LOCK_THRESHOLD**: Request authorization failure threshold for a source IP
- **NGX_CLIENT_LOCK_DURAION**: Request authorization failure threshold over a given interval for a source IP
- **NGX_CLIENT_BLOCK_DURAION**: Sets the duration (in seconds) of blocking a client to avoid brute force attack

Attack Detection Parameters

Configurations are present in the **<nginx-install-directory>/conf/conf.d/maps.conf** file.

```
## These two constants indicate five auth failures from a client can be allowed in thirty
seconds.
## if the threshold is crossed,client ip will be blocked.
map $host $auth_failure_threshold_for_lock {
## Must-change Replace below two parameters as per requirement
default 5 ;
}
map $host $auth_failure_counting_window_secs {
## Must-change Replace below two parameters as per requirement
default 30;
}
## This indicates duration of blocking a client to avoid brute force attack
map $host $ip_blocking_duration {
## Must-change Replace below parameter as per requirement
default 1800;
}
```

Logging

You can find the IP addresses that are blocked.

To find the IP addresses that are blocked, run the following commands from the directory ***{HOST_WORKING_DIR}/logs/***.

```
grep "will be blocked for" blocking.log
grep "IP is already blocked." error.log
2021/10/29 17:30:59 [emerg] 1181750#1181750: *19 [lua] block_unauthorized_users.lua:153:
_redirectAndSendError(): 10.68.218.190 will be blocked for 30 minutes for exceeding retry limit.,
client: 10.68.218.190, server: saproxy.cisco.com, request: "GET
/finesse/api/SystemInfo?nocache=1636456574482 HTTP/2.0", host: "saproxy.cisco.com:8445", referrer:
"https://saproxy.cisco.com:8445/desktop/container/?locale=en_US&"

2021/10/29 19:21:00 [error] 943068#943068: *43 [lua] block_unauthorized_users.lua:53: 10.70.235.30
:: IP is already blocked..., client: 10.70.235.30, server: saproxy.cisco.com, request: "GET
```

```
/finesse/api/SystemInfo?nocache=1635591686497 HTTP/2.0", host: "saproxy.cisco.com:8445", referrer:
"https://saproxy.cisco.com:8445/desktop/container/?locale=en_US"
```

Note It's recommended that the customers integrate with Fail2ban or similar to add the ban to the IP table or firewall rules.

Caching CORS headers

When the first option request is successful, then the following response headers are cached at the proxy for five minutes. These headers are cached for each respective upstream server.

- access-control allow-headers
- access-control-allow-origin
- access-control-allow-methods
- access-control-expose-headers and
- access-control-allow-credentials

Install and configure Fail2ban

Fail2ban scans log files and bans IPs that show the malicious signs such as too many password failures, seeking for exploits, and so on. Generally, Fail2Ban is used to update the firewall rules to reject the IP addresses for a specified amount of time. It can also be configured for any arbitrary actions such as sending an email. For more information, see https://www.fail2ban.org/.

Fail2ban can be configured to monitor the blocking log to identify the IP addresses that are blocked by NGINX on detecting brute force attacks, and ban them for a configurable duration.

The following are the steps to install and configure Fail2ban on a CentOS reverse-proxy:

Step 1 Install the Fail2ban using **yum**.

```
yum update && yum install epel-release
yum install fail2ban
```

Step 2 Create a local jail.

Jail configurations allow the administrator to configure various properties such as the ports that are to be banned from being accessed by any blocked IP address. The duration for which the IP address stays blocked, the filter configuration used for identifying the blocked IP address from the log file monitored, and so on.

Use the following steps to add a custom configuration for banning the IP addresses that are blocked from accessing the upstream servers:

a) Navigate to the Fail2ban installation directory (in this example `/etc/fail2ban`) `cd /etc/fail2ban`.
b) Create a copy of **jail.conf** into **jail.local** to keep the local changes isolated in **cp jail.conf jail.local**.
c) Add the following jail configurations to the end of the **jail.local** file. Substitute the ports in the template with the actual ones. Update the ban time configurations as required.

```
# Jail configurations for HTTP connections.
[finesse-http-auth]
enabled = true
# The ports to be blocked. Add any additional ports.
port = http,https,<finesse-ports>,<cuic-ports>,<any-other-ports-to-be-blocked>
# Path to nginx blocking logs.
logpath = ${HOST_WORKING_DIR}/logs/blocking.log
# The filter configuration.
filter = finesseban
# Block the IP from accessing the port, once the IP is blocked by lua.
maxretry= 1
# Duration for retry set to 3 mins. Doesn't count as the maxretry is 1
findtime= 180
# Lock time is set to 3 mins. Change as per requirements.
bantime = 180
```

Step 3 Configure a filter. A filter tells Fail2ban what to look for in the logs to identify the host to be banned. The steps to create a filter are as follows:

a) Create **filter.d/finesseban.conf**. touch **filter.d/finesseban.conf**

b) Add the following lines into the file `filter.d/finesseban.conf` [Definition] # The regex match that would cause blocking of the host. failregex = <HOST> will be blocked for

Step 4 Start Fail2ban. Run the **fail2ban-client start** command to start Fail2ban.

Open the Fail2ban log files and verify that there are no errors. By default, logs for Fail2ban go into the `/var/log/fail2ban.log` file.

Step 5 Validate static resource URLs. All valid endpoints which can be accessed without authentication are actively tracked in the proxy scripts.

Requests to these unauthenticated paths are actively rejected, if an invalid URI is requested, without sending these requests to the upstream server.

Frequently Asked Questions

Why does the proxy launcher fail to restart the Reverse Proxy?

The environment settings are incorrect. Correct any errors in the environment data and retry.The log file is stored at `${HOST_WORKING_DIR}/logs/openresty_launcher.log`. Using the command **docker ps -a**, see if the container is up and running.

How can I solve the OpenResty® launch error?

Some error during OpenResty® start. Fix any of the errors listed in the error log file available at `${HOST_WORKING_DIR}/logs/error.log` and try to restart.

Why is the content not refreshed to the end user?

Cache is not updated with latest contents. Run the following command to clear the cache:

docker exec <PROXY_HOSTNAME> `/usr/local/openresty/nginx/sbin/openresty_launcher.sh clear_cache`. The error log file available at `${HOST_WORKING_DIR}/logs/access.log`

Why is configuration generation from templates unsuccessful?

Failed to validate while generating the configuration. Correct any problems or failures reported on the console or in the error file. The error file as follows "Configuration generation from templates fails".

How can I fix problems or failures reported on the console or in the error file?

Reverse proxy is not included in the authorized list. Use this list of CLI Reverse Proxy authorized hosts and confirm if the list of Reverse Proxy authorized host names configured on Cisco IdS and Finesse boxes. This must contain the Reverse Proxy hostname and the allowed IP address.

What causes intermittent failures of Finesse REST API?

Because of the NGINX proxy rate limit issue, gadgets are not loading in the Finesse desktop. This results in intermittent Finesse REST API failures.

How do I determine which OpenResty® version is being used in the Installer?

Run the following command in the proxy instance to check the OpenResty® version on the Installer:

docker inspect <proxy_instance_name> | grep resty_rpm_version | cut -d ":" -f2

Why does proxy send HTTP error code 4xx ?

Refer to the HTTP HTTP Return codes returned by the reverse-proxy section.

Environment Files

The Reverse Proxy Installer behavior is driven using user-editable configuration files called environment files (.env). The environment file contains configuration data in the form of **key=value** pairs, which are referred to as properties. Each upstream component has custom environment files and properties specific to the respective component. The Installer also has its own specific environment files, used to customize its behavior. Reverse proxy installation requires the administrator to modify the properties to match the deployment. The following tables list and describe these properties, with their default values and guidance about changing them:

Note Reverse Proxy Automated Installer, on page 1 is a per-requisite reading for this chapter.

Installer env properties

The installer runs the container (which is in a docker), that contains the proxy. The properties determine the configuration of the container like the resources made available to it and the network configurations and such. By default, the properties are set to 2000 users deployment. Deployments which are bigger or smaller than 2000 users must verify these values and modify them appropriately.

Property Name, Description, and Default	Change Recommended?	When to Change?
CONTAINER_NAME Specifies the name of the reverse-proxy container—generally the reverse-proxy hostname. **Default:** proxy25.autobot.cvp	Yes	When you change the name of the container.

Property Name, Description, and Default	Change Recommended?	When to Change?
CONTAINER_NETWORK_MODE Specifies the network mode of the container. **Default:** host	Yes, If required.	If you use the host network mode for a container, the network stack for that container isn't isolated from the Docker host. [1] The other value is **bridge**. A bridge network creates a separate network for containers to communicate with each other, even if it is isolated from other networks on the host. This is useful when you want to deploy multiple containers on a single host and communicate with each other, but not with the outside world.
CONTAINER_DNS_RESOLVER Specifies a list of DNS servers separated by the | symbol. **Default:** 1.1.1.1|8.8.8.8	Yes	If an IP address changes, update the list.
CONTAINER_DNS_SEARCH_DOMAIN Specifies a DNS search domain to use when resolving hostnames inside the container. This property takes one or more domain names as arguments, separated by commas. In this example, the DNS search domain is `example.com`. Inside the container, the DNS resolver appends the search domain to the hostname and attempts to resolve it. If you ping the webserver inside the container, the DNS resolver tries to resolve `webserver.example.com`; if that fails, it tries to resolve `webserver`. **Default:** search.domain.1|search.domain.2	Yes	—
CREATE_SELF_SIGNED_SSL_CERT Specifies whether to create a self-singed certificate during the reverse-proxy installation. **Default:** TRUE	Yes, If required.	If the CA-signed certificates are present, you don't need to install self-signed certificates during the installation. In this case, change to FALSE.

Property Name, Description, and Default	Change Recommended?	When to Change?
CERTIFICATE_COMMON_NAME Specifies the common name for the certificate. This value is required to create self-signed certificates. Used on the next property. **Default:** `*.cisco.com`	Yes, If required.	Required only for creating self-singed certificates.
CERTIFICATE_SUBJECT Specifies the subject line to be used on the self-signed certificate. **Default:** /C=IN/ST=KA/L=BLR/O=Cisco/OU=CCBU/CN=${CERTIFICATE_COMMON_NAME}	Yes, If required.	Required only for creating self-singed certificates.
SSL_CERT_NAME Specifies the name of the certificate file to be auto-generated. **Default:** reverseproxy.crt	Yes, If required.	Required only for creating self-singed certificates.
SSL_KEY_NAME Specifies the name of the key file to be auto-generated. **Default:** reverseproxy.key	Yes, If required.	Required only for creating self-singed certificates.
SSL_CERT_KEY_LENGTH Specifies the certificate key length to create the self-signed certificate. **Default:** 2048	Yes, If required.	Required only for creating self-singed certificates.
SSL_CERT_EXPIRY_IN_DAYS Certificate expiry in days, to be specified in the self-signed certificate. **Default:** 1095	Yes, If required.	Required only for creating self-singed certificates.
AUTO_RESTART_CONTAINER Toggles auto-restart of the reverse-proxy container when the host system reboots. **Default:** 0	Yes	Enable this property only when the reverse-proxy is in working condition. [2]
NOFILE_LIMIT Specifies the initial and maximum number of open file descriptors that a container can have. Option in Docker is used to set system resource limits on a container. **Default:** nofile=102400:102400	Yes, If required.	nofile=204800:204800 for a 4000 deployment.
CPU_LIMIT Specifies the number of CPUs that a container can use. **Default:** 2	Yes, If required.	4 for 4000 deployment

Property Name, Description, and Default	Change Recommended?	When to Change?
MEM_LIMIT Specifies the maximum amount of memory that a container can use, in bytes or using a human-readable format. **Default:** 4G	Yes, If required.	8G for 4000 deployment
MEM_SWAP_LIMIT Specifies the maximum amount of memory and swap for a container—in bytes or using a human-readable format such as 1G for 1 gigabyte. **Default:** 8G	Yes, If required.	
MEM_RES Sets a soft limit on the minimum amount of memory to be available for the container. **Default:** 2G	Yes, If required.	4G for 4000 deployment

[1] The container shares the host networking namespace, and the container doesn't allocate its own IP address. For example, if you run a container which binds to port 80 and you use host networking, the container application is available on port 80 on the host IP address.

[2] If enabled and the container stops because of miss-configuration, setting the value with 1 keeps trying to restart the container. Also, the reverse-proxy container keeps running, until it is explicitly stopped.

Note Ensure that the host has adequate resources to run the container with the modified resource constraints.

Installer env properties that aren't recommended to be altered

Note These properties are provided for reference and they are available in the configuration to provide flexibility, to adjust the behavior if necessary, and in exceptional situations. It isn't recommended to change casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
CONTAINER_IMAGE A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform. **Default:** reverse-proxy-openresty-container:12.6(2)	No	Never
HOST_WORKING_DIR Specifies the working directory of the container **Default:** ~/reverse_proxy/${CONTAINER_NAME}	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_HOME Specifies the home directory of the NGINX server inside the container. **Default:** /usr/local/openresty/NGINX	No	
HOST_CACHE_VOL Specifies the host system directory used to mount on the container.[3] Mapped with the following container directory: NGX_CACHE_DIR. **Default:** ${HOST_WORKING_DIR}/cache	No	
HOST_SSL_VOL Specifies the host system directory used to mount on the container. Mapped to the following container directory: NGX_SSL_DIR **Default:** ${HOST_WORKING_DIR}/ssl	No	
HOST_LOGS_VOL Specifies the host system directory used to mount on the container. Mapped to the following container directory: NGX_LOG_DIR **Default:** ${HOST_WORKING_DIR}/logs	No	
HOST_CONF_VOL Specifies the host system directory used to mount on the container. Mapped with the container directory mentioned here: NGX_CONF_DIR **Default:** ${HOST_WORKING_DIR}/conf	No	
HOST_HTML_VOL Specifies the host system directory used to mount on the container. Mapped to the following container directory: NGX_HTML_DIR **Default:** ${HOST_WORKING_DIR}/html	No	
HOST_LUA_VOL Specifies the host system directory used to mount on the container. Mapped to the following container directory: NGX_LUA_DIR **Default:** ${HOST_WORKING_DIR}/lua	No	
NGX_CACHE_DIR Specifies the container directory location mapped with the corresponding host system directory specified in the HOST_CACHE_VOL property. **Default:** ${NGX_HOME}/cache	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_SSL_DIR Specifies the container directory location mapped with the corresponding host system directory mentioned in the HOST_LOGS_VOL property. **Default:** ${NGX_HOME}/ssl	No	
NGX_LOG_DIR Specifies the container directory location mapped with the corresponding host system directory mentioned in the HOST_LOGS_VOL property. **Default:** ${NGX_HOME}/logs	No	
NGX_CONF_DIR Specifies the container directory location mapped with the corresponding host system's directory mentioned in the HOST_CONF_VOL property. **Default:** ${NGX_HOME}/conf	No	
NGX_HTML_DIR Specifies the container directory location mapped with the corresponding host system directory mentioned in the HOST_HTML_VOL property. **Default:** ${NGX_HOME}/html	No	
NGX_LUA_DIR Specifies the container's directory location mapped with the corresponding host system's directory mentioned on this property HOST_LUA_VOL. **Default:** ${NGX_HOME}/lua	No	
MEM_SWAPPINESS Controls how aggressively the kernel should swap memory pages of the container to disk when the container exceeds its memory limit. **Default:** 1	No	
LOAD_CONTAINER_IMAGE_FROM_TAR This property is commented out by default. The default value (when it's commented) is true. **Default:** This property is commented by default.	No	You can change the value to load the container image from a different location.

Property Name, Description, and Default	Change Recommended?	When to Change?
REVERSE_PROXY_CONTAINER_IMAGE_TAR Specifies the location of the container image tar file. This property is commented out by default. ${SCRIPTPATH} is the location of the proxy_launcher.sh script. **Default:** ${SCRIPTPATH}//reverse-proxy-openresty-container/reverse-proxy-openresty-container.tar.gz	No	
RESTART_COND	No	Not used.

[3] Volume mounting is a concept used in computer systems to make a directory or file from one file system available to another file system. It's a method for sharing data between containers in a Docker environment or between a container and the host system.

Core properties

These are the basic properties that determine the behavior of the included OpenResty® Nginx proxy and control various aspects of its runtime behavior. It also contains request rates and various cache sizes setting for Nginx.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_DNS_RSLVR Specify the name servers used to resolve the names of upstream servers into addresses. Use spaces to separate multiple DNS server IP addresses. **Default:** 192.168.1.3 192.168.1.4 192.168.1.5	Yes	Update the entries with the IP addresses of the DNS servers.
NGX_JWT_SECRET OpenResty® Constants(defined in maps.conf) configuration. JWT secret pulled from IdS host using CLI "show ids secret" This secret is used to verify and validate tokens at proxy for authentication in SSO mode This secret is applicable only for IdS < 12.6(2). **Default:** TWSFbB9J6fBnu/D/hrHiQl2O0WEgrVj69ZiHJCtwahI=	Yes, if IdS is running in < 12.6(2) version	Update it with the output of this command from IdS: "show ids secret"
NGX_SYSLOG_SVR_IP Specifies the syslog server IP to which NGINX pushes some specific notification logs when the access for an IP is blocked. **Default:** 127.0.0.1	Yes, if necessary.	The current syslog server is the current reverse-proxy. This can be changed to the IP for any syslog server, based on the configuration.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_VALID_REFERRERS Specifies the “Referrer” request header field values for which the request is allowed. Request is blocked for all other referrers. The value is case-sensitive. Include all reverse-proxy hostnames, IdS hostnames and ADFS hostname in this list. They are required for reverse-proxy and other functionality. **Default:** proxy_pub.host.domain|proxy_sub.host.domain| ids_pub.host.domain|ids_sub.host.domain|adfs.host.domain	Yes	If not updated, the pages return with 417 HTTP error code. Make sure there are no typos in the hostnames.
NGX_LOCALHOST_IPS Specifies the list of IPs assigned to the reverse-proxy host across all NICs. Include all public and private IPs for reverse-proxy in this list. Include the alternate side reverse-proxy's IP addresses as well. **Default:** 192.168.1.69|192.168.1.169	Yes	Update all the reverse-proxy IPs here.
NGX_RATELIMIT_DISABLE_IPS Specifies a list of IP addresses for which rate limits aren't applied. **Default:** 192.168.1.69|192.168.1.169|127.0.0.1	Yes	All the IP address that should be allowed to exclude on rate-limiting. Update the list with all the public and private IPs of both the primary and secondary reverse-proxy. It can also include any other load balancer or proxy which are forwarding requests to reverse-proxy.
NGX_LOAD_BALANCER_IPS Hostnames aren’t supported as a permissible value in NGX_LOAD_BALANCER_IPS The list of entries should be | separated # Example: "192.162.1.0/24|10.78.95.76" Alternatively, if the internet client connection is stopped at the reverse-proxy directly, these variables MUST be empty.	Yes, If required.	If the load balancer forwards requests to the reverse-proxy, populate with the load balancer IP addresses.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_LOAD_BALANCER_REAL_IP_HEADER Devices must also send the end client IP alone, in a custom header. Add the name of the custom header used for this purpose to the NGX_LOAD_BALANCER_REAL_IP_HEADER variable. For example, "**X-Real-IP**". If you use the **X-Forwarded-For** as the field used to detect the client IP, include all trusted devices that can appear in this list in the NGX_LOAD_BALANCER_IPS variable. The first untrusted IP encountered is used as the client IP. We don't recommend using this field (**X-Forwarded-For**) for detecting the client IP.	Yes, If required.	

Core properties that are not recommended to be altered

Note These properties are provided for reference and they are available in the configuration, to provide flexibility and adjust the behavior if necessary, in exceptional situations, and aren't recommended to be changed casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_NUM_WKR_PRC OpenResty® NGINX core configurations. Specifies the number of worker processes. The value "auto" uses the number of available CPU cores. **Default:** auto	No	
NGX_PID_FILE Defines a file that stores the process ID of the main process. **Default:** openresty.pid	No	
NGX_WKR_CPU_AFFINITY Binds the worker processes to the sets of CPUs. The value "auto" binds worker processes automatically to the available CPUs. **Default:** auto	No	
NGX_WKR_PRIORITY Defines the scheduling priority for worker processes like it's done by the nice command. A negative number means higher priority. The allowed range varies from -20 to 20. **Default:** 0	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_NUM_RLIMIT Changes the limit on the maximum number of open files (RLIMIT_NOFILE) for worker processes. Used to increase the limit without restarting the main process. **Default:** 102400	No	
NGX_MULTI_ACCEPT If multi_accept is disabled, a worker process accepts one new connection at a time. Otherwise, a worker process accepts all new connections at a time. **Default:** on	No	
NGX_NUM_WKR_CONN Specifies the maximum number of simultaneous connections that can be opened by a worker process. **Default:** 10240	No	
NGX_SEND_FILE Enables or disables the use of **sendfile**.	No	No
NGX_TCP_NOPUSH Enables or disables the use of the TCP_NOPUSH socket option on FreeBSD or the TCP_CORK socket option on Linux. The options are enabled only when the **sendfile** is used. **Default:** on	No	
NGX_MAP_HASH_BUCKET_SIZE Specifies the bucket size for the map variables hash tables. **Default:** 128	No	
NGX_SERVERNAMES_HASH_BUCKET_SIZE Specifies the bucket size for the server names hash tables. **Default:** 512	No	
NGX_JWT_EXPIRY Specifies the JWT token expiry in seconds as configured in the IdS host. Token cache expiry time in reverse-proxy. Reverse-proxy keeps the cached token for 2 hours for the default configuration of 1-hour access token expiry time configured in IdS. **Default:** 7200	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_IDS_PUBLIC_KEY_POLL_INTERVAL Specifies the IdS public key poll frequency in seconds. The frequency at which reverse-proxy polls the ids to get the public key value. The default is once in 5 minutes. **Default:** 300	No	
NGX_CLIENT_LOCK_THRESHOLD If the threshold to detect DoS attacks is crossed in the specified interval, the client IP is blocked for the specified duration. **Default:** 5	No	
NGX_CLIENT_LOCK_DURATION Specifies the request authorization failure threshold over a given interval for a source IP. **Default:** 30	No	
NGX_CLIENT_BLOCK_DURATION Specifies the duration of blocking (in seconds) for clients to avoid brute force attacks. The block duration for the client IP is 30 minutes. **Default:** 1800	No	
NGX_SYSLOG_SVR_PORT Specifies the port for the syslog server. **Default:** 514	No	Usually the syslog server listens on 514, if the syslog server is configured to listen on some other port then this can be changed.
NGX_LOG_FILE Specifies the OpenResty® logging file. **Default:** access.log	No	
NGX_LOG_FORMAT Specifies the OpenResty® NGINX access log format name as specified in logging.conf. **Default:** info	No	Not recommended to change on a production system. You can change it to the debug format in LAB setup for more detailed logging.
NGX_LOG_BUFFER Specifies the OpenResty® NGINX access log buffer size. When this buffer is full or the flush interval is reached, the system writes the logs to the disk. **Default:** 16k	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_LOG_FLUSH_INTERVAL Specifies the OpenResty® NGINX access log flush interval. Logs are written to the disk after this interval is reached or the log buffer is full. **Default:** 30s	No	Not recommended changing on production servers. For a LAB system, you can reduce this value to 1 to 5s so you can check the access.log file updates immediately.
NGX_PROXY_CACHE_LOCK Only one request at a time can populate a new cache element identified according to the proxy_cache_key directive by passing a request to the server, which is enabled with reverse-proxy. Other requests of the same cache element either wait for a response to appear in the cache or the cache lock for this element to be released, up to the time set by the NGX_PROXY_CACHE_LOCK_TIMEOUT value. **Default:** on	No	
NGX_PROXY_CACHE_LOCK_TIMEOUT Specifies the timeout for NGX_PROXY_CACHE_LOCK. When the time expires, the request is passed to the server, which is enabled with reverse-proxy; however, the response isn't cached. **Default:** 30s	No	
NGX_PROXY_CACHE_LOCK_AGE If the last request passed to the server, which is enabled with reverse-proxy, for populating a new cache element hasn’t completed for the specified time, one more request passes to the server, which is enabled with reverse-proxy. **Default:** 5s	No	
NGX_PROXY_CACHE_BACKGROUND_UPDATE Allows starting a background sub request to update an expired cache item, while a stale cached response is returned to the client. **Default:** on	No	
NGX_PROXY_CACHE_REVALIDATE Enables revalidation of expired cache items using conditional requests with the “If-Modified-Since” and “If-None-Match” header fields. **Default:** on	No	
NGX_PROXY_CACHE_VALID Specifies the caching time for 200, 301, and 302 responses. **Default:** 24h	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_VARIABLES_HASH_BUCKET_SIZE Specifies the bucket size for the variables hash table. **Default:** 128	No	
NGX_KEEPALIVE_TIMEOUT Specifies a timeout during which a keep-alive client connection stays open on the server side. The zero value disables keep-alive client connections. **Default:** 20s	No	
NGX_SEND_TIMEOUT Specifies a timeout for transmitting a response to the client. The timeout is set only between two successive write operations, not for the transmission of the whole response. **Default:** 10s	No	
NGX_CLIENT_HEADER_TIMEOUT Specifies the timeout for reading the client request header. **Default:** 10s	No	
NGX_CLIENT_BODY_TIMEOUT Specifies a timeout for the reading the client request body. The timeout is set only for a period between two successive read operations, not for the transmission of the whole request body. **Default:** 10s	No	
NGX_RESET_TIMEDOUT_CONNECTION Enables or disables resetting timed out connections and connections closed with the non-standard code 444. **Default:** on	No	
NGX_CLIENT_HEADER_BUFFER_SIZE Specifies the buffer size for reading the client request header. **Default:** 4K	No	
NGX_CLIENT_BODY_BUFFER_SIZE Specifies the buffer size for reading the client request body. **Default:** 2k	No	
NGX_CLIENT_MAX_BODY_SIZE Specifies the maximum allowed size of the client request body. **Default:** 15m	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_LARGE_CLIENT_HEADER_BUFFER_NUM Specifies the maximum number of buffers used for reading a large client request header. Buffers are allocated only on demand. **Default:** 2	No	
NGX_LARGE_CLIENT_HEADER_BUFFER_SIZE Specifies the maximum size of buffers used for reading a large client request header. A request line can’t exceed the size of one buffer. Buffers are allocated only on demand. **Default:** 8K	No	
NGX_UNDERSCORES_IN_HEADERS Enables or disables the use of underscores in client request header fields. **Default:** on	No	
NGX_KEEPALIVE_REQUESTS Specifies the maximum number of requests that are served through one keep-alive connection. After the maximum number of requests are made, the connection is closed. **Default:** 500	No	
NGX_HTTP2_MAX_CONCURRENT_STREAMS Specifies the maximum number of concurrent HTTP/2 streams in a connection. **Default:** 150	No	
NGX_SERVER_TOKENS Enables or disables emitting NGINX version on error pages and in the “Server” response header field. **Default:** off	No	
NGX_LIMIT_CONN_DRY_RUN Enables the dry-run mode for limiting HTTP connections. In this mode, the number of connections isn’t limited. However, in the shared memory zone, the number of excessive connections is considered as usual. **Default:** off	No	On a production system, this should be always "off". If the system is running in lab mode, you can toggle this "on" to avoid rate limiting.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_LIMIT_REQ_DRY_RUN Enables the dry-run mode for limiting HTTP requests. In this mode, the number of connections isn't limited, however, in the shared memory zone, the number of excessive connections is considered as usual. **Default:** off	No	On a production setup, this should be always "off". If the system is running in lab mode, you can toggle this "on" to avoid rate limiting.
NGX_LIMIT_CONN_LOG_LEVEL Specifies the desired logging level for cases when the server limits the number of connections. **Default:** error	No	
NGX_LIMIT_REQ_LOG_LEVEL Specifies the desired logging level for cases when the server refuses to process requests due to rate exceeding, or delays request processing. **Default:** error	No	
NGX_LIMIT_REQ_STATUS Specifies the status code to return in response to rejected requests due to HTTP request rate limits. This is the standard HTTP error code for rate-limiting errors. **Default:** 429	No	
NGX_LIMIT_CONN_STATUS Specifies the status code to return in response to rejected requests due to HTTP connection rate limits. **Default:** 503	No	Error code returned when the connection limits are reached.
NGX_CHAT_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for chat access. **Default:** 30r/s	No	
NGX_IDS_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for IdS access. **Default:** 5r/s	No	
NGX_FIN_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for Finesse access. **Default:** 45r/s	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_FIN_CLIENT_LOG_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for Finesse client log requests. **Default:** 5r/s	No	
NGX_FIN_SSO_VALVE_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for Finesse SSO valve requests. **Default:** 5r/s	No	
NGX_CUIC_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for CUIC access. **Default:** 50r/s	No	
NGX_CUIC_HISTORICAL_REPORT_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for CUIC historical report requests. **Default:** 16r/s	No	
NGX_CUIC_REALTIME_REPORT_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for CUIC realtime report requests. **Default:** 48r/s	No	
NGX_CUIC_REPORT_EXECUTION_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for CUIC report execution requests. **Default:** 12r/s	No	
NGX_LIVEDATA_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for livedata access. **Default:** 25r/s	No	
NGX_CLOUDCONNECT_DR_TASK_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for DR API task request access. **Default:** 100r/s	No	
NGX_CLOUDCONNECT_USER_SYNC_CALLBACK_REQUEST_RATE_LIMIT Specifies the HTTP request rate limit for user sync callback request access. **Default:** 5r/m	No	
NGX_PRXY_STATIC_FILES_PORT Specifies the OpenResty® static content configuration. The reverse-proxy port is used to serve static files under the HTML directory. **Default:** 10000	No	This location serves the proxy-map information. You can change the port number if necessary.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_STATUS_IP Specifies the reverse-proxy IP used to access OpenResty® NGINX stats over the "/reverseproxy_status" endpoint Internal request is accessible from only the host system. **Default:** 127.0.0.1	No	
NGX_PRXY_STATUS_PORT Specifies the reverse-proxy port used to access OpenResty® NGINX stats over the "/reverseproxy_status" endpoint. **Default:** 10001	No	
NGX_USERTIMERTHREAD_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100k	No	
NGX_USERLIST_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 50m	No	
NGX_CREDENTIALSSTORE_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100m	No	
NGX_USERCOUNT_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100k	No	
NGX_CLIENTSTORAGE_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100m	No	
NGX_BLOCKINGRESOURCES_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100m	No	
NGX_TOKENCACHE_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 10m	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_IPSTORE_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 10m	No	
NGX_DESKTOPURLLIST_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 10m	No	
NGX_DESKTOPURLCOUNT_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100k	No	
NGX_THIRDPARTYGADGETURLLIST_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100m	No	
NGX_THIRDPARTYGADGETURLCOUNT_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100k	No	
NGX_CORSHEADERSSTORE_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100k	No	
NGX_TIMERTHREADSSTORE_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100k	No	
NGX_ALTERNATE_HOSTS_SHRD_DICT_SIZE Specifies the LUA shared dictionary sizes used by reverse-proxy internally. **Default:** 100k	No	

Directory (DIR) properties

The following table lists the directory properties and the default values for various OpenResty® folders.

Note These properties are provided for reference and they are available in the configuration. They provide flexibility to adjust the behavior if necessary, in exceptional situations, and aren't recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_CACHE_DIR Specifies the cache directory where various resources for components are cached. **Default:** ${NGX_HOME}/cache	No	
NGX_CONF_DIR Specifies the OpenResty® directory containing NGINX configurations, such as core and component configurations. **Default:** ${NGX_HOME}/conf	No	
NGX_HOME Specifies the home directory for OpenResty® nginx installation. **Default:** /usr/local/openresty/nginx	No	
NGX_HTML_DIR Specifies the OpenResty® directory **Default:** ${NGX_HOME}/html Directory containing static resources.	No	
NGX_LOG_DIR Specifies the OpenResty® directory where OpenResty® logs are stored. **Default:** ${NGX_HOME}/logs	No	
NGX_LUA_DIR Specifies the OpenResty® directory containing lua resources. **Default:** ${NGX_HOME}/lua	No	
NGX_SSL_DIR Specifies the OpenResty® directory containing SSL resources like certs and keys. **Default:** ${NGX_HOME}/ssl	No	

Common Properties

Common SSL-Related Properties

The following table lists SSL-related properties that are common across components.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_SSL_TRUST_CRT Specifies a file with trusted CA certificates in the PEM format used to verify the certificate of the Finesse HTTPS server, which is enabled for reverse-proxy. **Default:** ${NGX_SSL_DIR}/upstreams_finesse_trust.crt	Yes, if necessary.	Point to the exact location of upstream Finesse's certificate for mutual trust establishment.
NGX_SSL_CRT SSL connector configuration for the component access. Specifies the location of a file with the certificate, for the given component, in the PEM format. **Default:** ${NGX_SSL_DIR}/reverseproxy.crt	Yes, if necessary.	Update the location of the Finesse reverse-proxy certificate.
NGX_SSL_KEY Specifies the location of a file with the secret key, for the given component, in the PEM format. **Default:** ${NGX_SSL_DIR}/reverseproxy.key	Yes, if necessary.	If the location of the file changes.

Common SSL-Related properties that are not recommended to be altered

Note These properties are provided for reference and they are available in the configuration. They provide flexibility to adjust the behavior if necessary, in exceptional situations, and aren't recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
#NGX_SSL_DHPARAM Specifies a file with DH parameters for DHE ciphers. **Default:** ${NGX_SSL_DIR}/dhparam.pem This property is disabled by default. Uncomment the parameter to use it.	No. Can be changed if necessary.	Enable for more security, to prevent affecting by an attack exploiting the Logjam vulnerability. For more information, see *Understanding Logjam and Future-Proofing Your Infrastructure* at: https://cisco.blogs.com
NGX_PRXY_SSL_VERIFY Enables or disables verification of the HTTPS server certificate, which is enabled for reverse-proxy. **Default:** on	No	Changing this to off disables SSL verification of the requests.
NGX_PRXY_SSL_VERIFY_DEPTH Specifies the verification depth in the HTTPS server certificates chain, which is enabled for reverse-proxy. This is for DoS prevention. Building the chain might be an exponential algorithm with backoff, so with the openssl default of 100 a malicious backend might cause denial of service in nginx. **Default:** 10; less than 4 is easy to break.	No	
NGX_SSL_CACHE_SIZE Specifies the SSL session cache size for session parameters storage for client connections. This cache is shared between all worker processes. The cache size is specified in bytes; one megabyte can store about 4000 sessions. Each shared cache should have an arbitrary name. **Default:** 10m	No	
NGX_SSL_CIPHERS Specifies the OpenSSL format for enabled ciphers. **Default:** EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH	No	This list contains all the strong ciphers available. You can update the list if ciphers are found to be vulnerable or to add new supported ciphers.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_SSL_PRFR_SRVR_CIPHERS Prefer server ciphers over client ciphers. **Default:** on	No	
NGX_SSL_PROTO Specifies the TLS versions enabled for the connections. To specify multiple values, use space delimiters. # Example: NGX_SSL_PROTO="TLSv1 TLSv1.1 TLSv1.2" Generally TLS version 1.2 is supported for the webapp requests. TLS v1 and TLS v1.1 aren't recommended. **Default:** TLSv1.2	No	
NGX_SSL_SESSION_TICKETS Enables or disables session resumption through TLS session tickets. **Default:** off	No	Enable to resume sessions and avoid keeping a per-client session state. The TLS server encapsulates the session state into a ticket and forwards it to the client. The client can later resume a session using the obtained ticket.
NGX_SSL_SSN_TIMEOUT Specifies a time during which a client may reuse the session parameters—how long each session lives in reverse-proxy. **Default:** 30m	No	
NGX_SSL_STAPLING Enables or disables stapling of OCSP responses by the server. SSL stapling means that revocation information about the servers certificate (that is, the OCSP response) are included in the TLS handshake together with the server certificate. **Default:** off	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_SSL_STAPLING_VERIFY Enables or disables the verification of OCSP responses by the server. Allows the presenter of a certificate to bear the resource cost involved in providing OCSP responses by appending ("stapling") a time-stamped OCSP response signed by the CA to the initial TLS handshake, eliminating the need for clients to contact the CA". **Default:** off	No	

Cisco Finesse Properties

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_FIN_HOSTNAME Reverse-proxy hostname through which this Finesse host is accessed. **Default:** reverseproxy.host.domain	Yes	New installation or if the hostname of the reverse-proxy must be accessed from the internet changes.
NGX_PRXY_FIN_PORT Specifies the reverse-proxy port over which this Finesse host are accessed. **Default:** 8445	Yes, if necessary.	New installation or if the port number of the reverse-proxy must be accessed from the internet changes.
NGX_FIN_HOSTNAME Specifies the upstream Finesse hostname. **Default:** finesse.host.domain	Yes	New installation or if the hostname of the Finesse box changes.
NGX_AUTH_URL Specifies the Finesse URL to fetch the users list to perform authentication at proxy. **Default:** https://reverseproxy.host.domain:8445/finesse/api/UserAuth	Yes	Replace "reverseproxy.host.domain" with the FQDN of the reverse-proxy.

Cisco Finesse properties that are not recommended to be altered

Note These properties are provided for reference and they exist in the configuration to provide flexibility to adjust the behavior if necessary, in exceptional situations and aren't recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_FIN_UPSTREAM_KEEPALIVE Specifies the proxy backend configurations for Finesse, and activates the cache for connections to an upstream IdS server. The value sets the maximum number of idle keep-alive connections to upstream servers that are preserved in the cache of each worker process. When this number is exceeded, the least recently used connections are closed. **Default:** 128	No	This default is the optimal value for keeping alive the upstream connection per component.
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, Finesse, ids, livedata, cuic_12.6(1), livedata_12.6(1), and idp-adfs3. **Default:** Finesse	No	Never
NGX_FIN_DESKTOP_CACHE_SIZE Specifies the proxy cache configuration for the Finesse desktop and the initial size of the Finesse desktop endpoints cache. **Default:** 10m	No	The default is the optimal value.
NGX_FIN_DESKTOP_CACHE_MAX_SIZE Specifies the maximum size for the Finesse desktop endpoints cache. When the size exceeds or there isn't enough free space, it removes the least recently used data. **Default:** 50m	No	The default is the optimal value.
NGX_FIN_DESKTOP_CACHE_INACTIVE_DURATION Specifies the content inactive duration for the Finesse desktop endpoints cache. Cached data that is not accessed during the time specified gets removed from the cache regardless of their duration. **Default:** 3y	No	The default is the optimal value. After this time, the cache contents expire.
NGX_FIN_SHINDIG_CACHE_SIZE Specifies the proxy cache configuration for Finesse Shindig and the initial size of the Finesse Shindig endpoints cache. **Default:** 10m	No	
NGX_FIN_SHINDIG_CACHE_MAX_SIZE Max size for Finesse shindig endpoints cache. When the size exceeds or there isn't enough free space, it removes the least recently used data. **Default:** 500m	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_FIN_SHINDIG_CACHE_INACTIVE_DURATION Specifies the content inactive duration for the Finesse desktop endpoints cache. Cached data that aren't accessed during the time specified get removed from the cache regardless of their freshness. **Default:** 3y	No	
NGX_FIN_OPENFIRE_CACHE_SIZE Specifies the initial size of the Finesse openfire endpoints cache. **Default:** 10m	No	
NGX_FIN_OPENFIRE_CACHE_MAX_SIZE Specifies the maximum size for the Finesse openfire endpoints cache. When the size exceeds or there isn't enough free space, it removes the least recently used data. **Default:** 10m	No	
NGX_FIN_OPENFIRE_CACHE_INACTIVE_DURATION Specifies the content inactive duration for the Finesse desktop endpoints cache. Cached data that is not accessed during the time specified is removed from the cache regardless of their duration. **Default:** 3y	No	
NGX_FIN_REST_CACHE_SIZE Specifies the initial size of the Finesse REST endpoints cache **Default:** 10m	No	
NGX_FIN_REST_CACHE_MAX_SIZE Specifies the maximum size for the Finesse REST endpoints cache. When the size exceeds or there isn't enough free space, it removes the least recently used data. **Default:** 1500m	No	
NGX_FIN_REST_CACHE_INACTIVE_DURATION Specifies the content inactive duration for the Finesse desktop endpoints cache. Cached data that is not accessed during the time specified, get removed from the cache regardless of their duration. **Default:** 40m	No	
NGX_FIN_LAYOUT_CACHE_SIZE Specifies the initial size of the Finesse layout endpoints cache. **Default:** 150m	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_FIN_LAYOUT_CACHE_MAX_SIZE Specifies the maximum size for the Finesse layout endpoints cache. When the size exceeds or there isn't enough free space, the system removes the least recently used data. **Default:** 300m	No	
NGX_FIN_LAYOUT_CACHE_INACTIVE_DURATION Specifies the inactive duration for content stored in the Finesse desktop endpoints cache. Cached data that aren't accessed during the time specified get removed from the cache regardless of their duration. **Default:** 40m	No	
NGX_FIN_HTTP1_CONN_LIMIT Specifies the number of concurrent HTTP/1.1 connections allowed per source IP **Default:** 12	No	
NGX_FIN_HTTP2_CONN_LIMIT Specifies the number of concurrent HTTP/2 streams allowed per source IP. **Default:** 150	No	
NGX_FIN_DESKTOP_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for desktop endpoints **Default:** 50	No	
NGX_FIN_SHINDIG_CORE_RPC_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for shindig rpc endpoints **Default:** 40	No	
NGX_FIN_SHINDIG_IFR_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for shindig ifr endpoints **Default:** 30	No	
NGX_FIN_SSOVALVE_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for SSO valve endpoints **Default:** 5	No	
NGX_FIN_LIMIT_REQ_STATUS Specifies the HTTP response code to return when the HTTP request rate reaches the limit **Default:** 429	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_FIN_PROXY_BUFFER_SIZE Specifies the size of the buffer used for reading the first part of the response received from the server, which is enabled for reverse-proxy. Configurations are overridden from the common config for Finesse. **Default:** 8k	No	
NGX_FIN_MAX_TEMP_FILE_SIZE Applies when buffering of responses from the server (which is enabled for reverse-proxy) is enabled. If the whole response doesn't fit into the buffers set by the NGX_FIN_PROXY_BUFFER_SIZE, the system saves part of the response in a temporary file. **Default:** 100m	No	

Chat properties

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_CHAT_PORT Specifies the reverse-proxy port over which this chat server is accessed. **Default:** 5280	Yes, if necessary.	New installation or if not changed, 5280 is the port for the chat server.
NGX_PRXY_CHAT_HOSTNAME Specifies the reverse-proxy hostname over which this chat server is accessed. **Default:** reverseproxy.host.domain	Yes	New installation or if the chat hostname of the reverse-proxy is accessed from the the internet changes.
NGX_CHAT_HOSTNAME Specifies the hostname of the chat server. **Default:** chat1.host.domain	Yes	New installation or if the host name of the upstream chat server changes.
NGX_CHAT_HOST[1-8]_PROXY Required for substituting user home and backup node information in the log-in response. (Similar keys configurations exist for 4 HA clusters of chat servers.) **Default:** reverseproxy.host.domain:5280/reverseproxy-sub.host.domain:15280	Yes	New installation or if the chat server proxy hostname FQDN changes.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_CHAT_HOST[1-8] Specifies the proxy access information for all chat nodes in the deployment. Required for substituting user home and backup node information in the log-in response. (Similar keys configurations exist for 4 HA clusters of chat servers.) **Default:** chat[1-4].host.domain/chat[1-4]-sub.host.domain	Yes	New installation or if the chat server upstream hostname FQDN changes.
NGX_CHAT_BIND_PATH Specifies the binding URL for the chat server. **Default:** httpbinding	Yes	If the binding URL for the chat server changes.
NGX_AUTH_URL Specifies the Finesse URL to fetch the users list to perform authentication at proxy. **Default:** https://proxy.host.domain:8445/finesse/api/UserAuth	Yes	New installation or if proxy.host.domain is the hostname of the the reverse-proxy.

Chat properties that are not recommended to be altered

Note These properties are provided for reference and they are available in the configuration. They provide flexibility to adjust the behavior if necessary, in exceptional situations, and aren't recommended for changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to change?
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, finesse, ids, livedata. **Default:** chat	No	
NGX_CHAT_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit. **Default:** 10	No	
NGX_CHAT_PORT Specifies the port for the chat server. **Default:** 5280	No	New installation or if the port number for the upstream chat server is different.

Property Name, Description, and Default	Change Recommended?	When to change?
NGX_CHAT_HTTP2_CONN_LIMIT Specifies the number of concurrent HTTP/2 streams allowed per source IP. **Default:** 30	No	
NGX_CHAT_HTTP1_CONN_LIMIT Specifies the rate limit for desktop chat, the number of concurrent HTTP/1.1 connections allowed per source IP. **Default:** 6	No	

Cloud Connect properties

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_CLOUDCONNECT_PORT Specifies the reverse-proxy port over which this Cloud Connect server is accessed. **Default:** 443	Yes, if necessary.	If there is a change in the port number.
NGX_PRXY_CLOUDCONNECT_HOSTNAME Specifies the reverse-proxy hostname over which this Cloud Connect server is accessed. **Default:** reverseproxy.host.domain	Yes	If there is a change in the chat hostname of the reverse-proxy that has must accessed from the internet.
NGX_CLOUDCONNECT_HOSTNAME Specifies the hostname for the Cloud Connect server and the hostname of the Publisher or Primary Cloud Connect node. (Conversely on the alternate side) **Default:** cloudconnect.host.domain	Yes	If there is a change in the FQDN for the upstream Cloud Connect server.
NGX_CLOUDCONNECT_FAILOVER_HOSTNAME Specifies the hostname of the Cloud Connect failover node and the hostname of the Subscriber or Secondary Cloud Connect node. (Conversely on the alternate side) **Default:** cloudconnct.failover.hostname	Yes	If there is change in the FQDN of the alternate Cloud Connect server.
NGX_CLOUDCONNECT_CLIENT_IPS Creates a list of known IP addresses for clients connecting to Cloud Connect services like DigitalRouting and UserSync. **Default:** 35.161.238.252|35.166.68.236|34.240.73.178|3.9.155.97|54.206.189.15|52.62.185.51| 13.210.45.137|52.40.46.90|52.214.81.91|3.9.151.19|3.105.22.233|52.17.23.194	Yes	Validate and update the list when there is a change in the IP addresses.

Cloud Connect properties that are not recommended to be altered

Note These properties are provided for reference and they exist in the configuration to provide flexibility to adjust the behavior if necessary, in exceptional situations and aren't recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, finesse, ids, livedata, cuic_12.6(1), livedata_12.6(1), and idp-adfs3. **Default:** cloudconnect	No	
NGX_CLOUDCONNECT_USER_SYNC_CALLBACK_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for user sync bulk requests. **Default:** 10	No	
NGX_CLOUDCONNECT_UPSTREAM_KEEPALIVE Activates the cache for connections to the upstream Cloud Connect server. Specifies the maximum number of idle keepalive connections to upstream servers that are preserved in the cache of each worker process. When the count exceeds this number, the least recently used connections close. **Default:** 150	No	
NGX_CLOUDCONNECT_LIMIT_REQ_STATUS Specifies the HTTP response code to return when the HTTP request rate reaches the limit. **Default:** 429	No	
NGX_CLOUDCONNECT_HTTP2_CONN_LIMIT Specifies the number of concurrent HTTP/2 streams allowed per source IP. **Default:** 250	No	
NGX_CLOUDCONNECT_HTTP1_CONN_LIMIT Specifies the number of concurrent HTTP/1.1 connections allowed per source IP—the rate limit for Digital Routing requests. **Default:** 50	No	
NGX_CLOUDCONNECT_HEALTHCHECK_TIMEOUT Specifies the timeout for the health check API in milliseconds, if there is a failure. **Default:** 2000	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_CLOUDCONNECT_HEALTHCHECK_SSL_VERIFY Specifies whether to verify the SSL certificate for health checks. **Default:** FALSE	No	
NGX_CLOUDCONNECT_HEALTHCHECK_RISE Specifies the number of successive health check successes before turning up a peer. **Default:** 2	No	
NGX_CLOUDCONNECT_HEALTHCHECK_RESPONSE_CODES Generates a comma-separated list of valid HTTP status codes for the health check API. **Default:** {200}	No	
NGX_CLOUDCONNECT_HEALTHCHECK_INTERVAL Specifies the interval between health checks in milliseconds. **Default:** 2000	No	
NGX_CLOUDCONNECT_HEALTHCHECK_FALL Specifies the number of successive health check failures before turning down a peer. **Default:** 2	No	
NGX_CLOUDCONNECT_HEALTHCHECK_CONCURRENCY Specifies the number of concurrent health checks. **Default:** 2	No	
NGX_CLOUDCONNECT_HEALTHCHECK_API Specifies the active health check configurations for the DR API. The system uses this URL to check the health of the Cloud Connect. **Default:** /drapi/v1/ping	No	
NGX_CLOUDCONNECT_DR_TASK_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for digital routing tasks. **Default:** 200	No	

Cisco IdS properties

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_IDS_HOSTNAME Specifies the reverse-proxy hostname over which this IdS host is accessed. **Default:** reverseproxy.host.domain	Yes	New installation or if the reverse-proxy IdS hostname changes.
NGX_PRXY_IDS_PORT Specifies the reverse-proxy port over which this IdS host is accessed. **Default:** 8553	Yes, If required.	New installation or if the reverse-proxy port changes.
NGX_IDS_HOSTNAME IdS server actual hostname **Default:** ids.host.domain	Yes	New installation or if the FQDN host name of the IdS server changes.

Cisco IdS properties that are not recommended to be altered

Note These properties are provided for reference and they are available in the configuration. They provide flexibility to adjust the behavior if necessary, in exceptional situations, and aren't recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, finesse, ids, livedata, cuic_12.6(1), livedata_12.6(1), and idp-adfs3. **Default:** ids	No	
NGX_IDS_UPSTREAM_KEEPALIVE Proxy backend configurations for IDS. Activates the cache for connections to the upstream IDP server. Specifies the maximum number of idle keepalive connections to upstream servers that are preserved in the cache of each worker process. When the count exceeds this number, the least recently used connections close. **Default:** 128	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_IDS_HTTP1_CONN_LIMIT Rate limits configuration for IdS. Specifies the number of concurrent HTTP/1.1 connections allowed per source IP. **Default:** 4	No	
NGX_IDS_HTTP2_CONN_LIMIT Specifies the number of concurrent HTTP/2 streams allowed per source IP. **Default:** 4	No	
NGX_IDS_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit. **Default:** 4	No	

IdP Properties (ADFS 3.0)

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_IDP_HOSTNAME Specifies the reverse-proxy hostname over which this IdP host will be accessed. **Default:** adfs-reverseproxy.host.domain	Yes	New installation or if the FQDN of the reverse-proxy IdP host changes.
NGX_PRXY_IDP_PORT Specifies the reverse-proxy port over which this IdP host will be accessed. **Default:** 443	Yes, If required.	New installation or if unchanged, reverse-proxy uses the 443 port for CUIC.
NGX_IDP_HOSTNAME Specifies the CUIC server hostname. **Default:** idp.host.domain	Yes	New installation or if the FQDN of the IdP host changes.

IdP Properties (ADFS 3.0) that are not recommended to be altered

Note These properties are provided for reference and they are available in the configuration. They provide flexibility to adjust the behavior if necessary, in exceptional situations, and are not recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, finesse, ids, livedata, cuic_12.6(1), livedata_12.6(1), and idp-adfs3. **Default:** idp-adfs3	No	
NGX_IDP_UPSTREAM_KEEPALIVE Proxy backend configurations for IDP. Activates the cache for connections to upstream IDP server Specifies the maximum number of idle keepalive connections to upstream servers that are preserved in the cache of each worker process. When the count exceeds this number, the least recently used connections close. **Default:** 128	No	

Livedata Properties

LiveData 12.6(1) Properties

Note Use livedata_12.6(1).env when the upstream Livedata is still on Release12.6(1). Otherwise, use livedata.env.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_LD_HOSTNAME Specifies the reverse-proxy hostname over which this LD host is accessed. **Default:** reverseproxy.host.domain	Yes	If the reverse-proxy livedata host FQDN changes.
NGX_PRXY_LD_PORT Specifies the reverse-proxy port over which this livedata host is accessed. **Default:** 12005	Yes, If required.	If there is a change in the reverse-proxy port for livedata.
NGX_PRXY_LD_SCKT_IO_PORT Specifies the reverse-proxy port over which the socket IO endpoint of this livedata host will be accessed. **Default:** 12008	Yes, If required.	If there is a change in the reverse-proxy port for socket IO connections.
NGX_LD_HOSTNAME Specifies the Livedata server hostname. **Default:** livedata.host.domain	Yes	If there is a change in the upstream livedata host FQDN.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_LD_SIO_UPSTREAM_KEEPALIVE Activates the cache for connections to the upstream livedata server socketio endpoint. **Default:** 128	No	—
NGX_LD_BACKEND_FAILOVER Livedata failover configuration. This is used to translate the location header during failover. For SideA LiveData, point to SideB LiveData and for SideB Livedata, point to SideA Livedata.	Yes	If there is a change in the FQDN or port.
NGX_LD_BACKEND_PROXY_FAILOVER Livedata proxy failover configuration. Each side points to the other side of the reverse-proxy node.	Yes	If there is a change in the FQDN or port.

LiveData 12.6(1) properties that are not recommended to be altered

Note These properties are provided for reference and they are available in the configuration. They provide flexibility to adjust the behavior if necessary, in exceptional situations, and aren't recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, finesse, ids, livedata, cuic_12.6(1), livedata_12.6(1), and idp-adfs3. **Default:** livedata_12.6(1)	No	—
NGX_LD_WEB_UPSTREAM_KEEPALIVE Proxy backend configurations for livedata. Activates the cache for connections to the upstream livedata server. Specifies the maximum number of idle keep alive connections to upstream servers that are preserved in the cache of each worker process. When the count exceeds this number, the least recently used connections close. **Default:** 128	No	—

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_LD_SIO_UPSTREAM_KEEPALIVE Activates the cache for connections to the upstream livedata server socketio endpoint. **Default:** 128	No	—
NGX_LD_HTTP1_CONN_LIMIT Specifies the number of concurrent HTTP/1.1 connections allowed per source IP. **Default:** 12	No	—
NGX_LD_HTTP2_CONN_LIMIT Specifies the number of concurrent HTTP/2 streams allowed per source IP. **Default:** 150	No	—
NGX_LD_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit. **Default:** 25	No	—
NGX_LD_PROXY_BUFFER_SIZE **Default:** 8k	No	—
NGX_LD_MAX_TEMP_FILE_SIZE **Default:** 100m	No	—

LiveData 12.6(2) Properties

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_LD_HOSTNAME Specifies the reverse-proxy hostname over which this LD host is accessed. **Default:** reverseproxy.host.domain	Yes	If the reverse-proxy livedata host FQDN changes.
NGX_PRXY_LD_PORT Specifies the reverse-proxy port over which this livedata host is accessed. **Default:** 443	Yes, If required.	If the reverse-proxy port for livedata changes.
NGX_LD_HOSTNAME Specifies the Livedata server hostname. **Default:** livedata.host.domain	Yes	If the upstream livedata host FQDN changes.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_LD_BACKEND_FAILOVER Specifies the livedata failover configurations. This is used to translate the location header during failover. **Default:** hostname:port	Yes	For the SideA upstream LiveData, point to the upstream SideB LiveData. For the SideB upstream, point to the upstream SideA.
NGX_LD_BACKEND_PROXY_FAILOVER Specifies the livedata failover node for this livedata host. This is used to translate the location header during failover. **Default:** hostname:port	Yes	Point to the other side of the reverse-proxy node.

LiveData 12.6(2) properties that are not recommended to be altered

Note These properties are provided for reference and they exist in the configuration to provide flexibility to adjust the behavior if necessary, in exceptional situations and aren't recommended to be changed casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, finesse, ids, livedata, cuic_12.6(1), livedata_12.6(1), and idp-adfs3. **Default:** livedata	No	
NGX_LD_WEB_UPSTREAM_KEEPALIVE Proxy backend configurations for livedata. Activates the cache for connections to the upstream livedata server. Specifies the maximum number of idle keepalive connections to upstream servers that are preserved in the cache of each worker process. When the count exceeds this number, the least recently used connections are closed. **Default:** 128	No	
NGX_LD_SIO_UPSTREAM_KEEPALIVE Activates the cache for connections to the upstream livedata server socketIO endpoint. **Default:** 128	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_LD_HTTP1_CONN_LIMIT Specifies the number of concurrent HTTP/1.1 connections allowed per source IP. **Default:** 12	No	
NGX_LD_HTTP2_CONN_LIMIT Specifies the number of concurrent HTTP/2 streams allowed per source IP. **Default:** 150	No	
NGX_LD_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit. **Default:** 25	No	
NGX_LD_PROXY_BUFFER_SIZE Sets the size of the buffer used for reading the first part of the response received from the server, which is enabled with reverse-proxy. **Default:** 8k	No	
NGX_LD_MAX_TEMP_FILE_SIZE When buffering of responses from the server (which is enabled with reverse-proxy) is enabled, and the whole response doesn't fit into the NGX_LD_PROXY_BUFFER_SIZE, a part of the response can be saved to a temporary file. **Default:** 100m	No	

CUIC Properties

Unified Intelligence Center 12.6(1) properties

Note Use this env file when the upstream Unified Intelligence Center is still on 12.6(1) release. Otherwise use cuic.env.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_CUIC_HOSTNAME Specifies the reverse-proxy hostname over which this CUIC host is accessed. **Default:** reverseproxy.host.domain	Yes	If the reverse-proxy CUIC host FQDN changes.

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_CUIC_PORT Specifies the reverse-proxy port over which this cuic host is accessed. **Default:** 8444	Yes, If required.	New installation or if the port number changes.
NGX_PRXY_CUIC_DOC_PORT Specifies the reverse-proxy port over which this CUIC host doc endpoint is accessed. **Default:** 8447	Yes, If required.	New installation or if the port number changes.
NGX_CUIC_HOSTNAME Specifies the CUIC server hostname. **Default:** cuic.host.domain	Yes	If the upstream CUIC host FQDN changes.

Unified Intelligence Center 12.6(1) properties that are not recommended to be altered

Note These properties are provided for reference and they are available in the configuration. They provide flexibility to adjust the behavior if necessary, in exceptional situations, and aren't recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, finesse, ids, livedata, cuic_12.6(1), livedata_12.6(1), idp-adfs3. **Default:** cuic_12.6(1)	No	Never
NGX_CUIC_UPSTREAM_KEEPALIVE Activates the cache for connections to the upstream CUIC server. Specifies the maximum number of idle keep alive connections to upstream servers that are preserved in the cache of each worker process. When this number exceededs, the least recently used connections are closed. **Default:** 128	No	—
NGX_CUICDOC_UPSTREAM_KEEPALIVE Activates the cache for connections to the upstream CUIC server doc endpoint. **Default:** 128	No	—

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_CUIC_CACHE_SIZE Specifies the initial size of the CUIC proxy cache. **Default:** 15m	No	—
NGX_CUIC_CACHE_MAX_SIZE Specifies the maximum size for the CUIC cache. When the size exceeds or there's not enough free space, the system removes the least recently used data. **Default:** 50m	No	—
NGX_CUIC_CACHE_INACTIVE_DURATION Specifies the content inactive duration for the CUIC cache. Cached data that is not accessed during the time specified get removed from the cache regardless of their freshness. **Default:** 3y	No	—
NGX_CUICDOC_CACHE_SIZE Specifies the initial size of the CUIC doc cache. **Default:** 15m	No	—
NGX_CUICDOC_CACHE_MAX_SIZE Specifies the maximum size for the CUIC doc cache. When the size exceeds or there isn’t enough free space, the system removes the least recently used data. **Default:** 50m	No	—
NGX_CUICDOC_CACHE_INACTIVE_DURATION Specifies the content inactive duration for CUIC doc cache. Cached data that aren’t accessed during the time specified get removed from the cache regardless of their freshness. **Default:** 3y	No	—
NGX_CUIC_HTTP1_CONN_LIMIT Specifies the number of concurrent HTTP/1.1 connections allowed per source IP. **Default:** 12	No	—
NGX_CUIC_HTTP2_CONN_LIMIT Specifies the number of concurrent HTTP/2 streams allowed per source IP. **Default:** 150	No	—

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_CUIC_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit. **Default:** 100	No	—
NGX_CUIC_HISTORICAL_REPORT_CONN_LIMIT Specifies the number of concurrent connections allowed, per source IP, for CUIC historical reports endpoints. **Default:** 4	No	—
NGX_CUIC_HISTORICAL_REPORT_NEW_CONN_LIMIT Specifies the number of concurrent connections allowed, per source IP, for CUIC historical reports newRest endpoints. **Default:** 4	No	—
NGX_CUIC_HISTORICAL_REPORT_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for CUIC historical reports endpoints. **Default:** 4	No	—
NGX_CUIC_REALTIME_REPORT_CONN_LIMIT Specifies the number of concurrent connections allowed, per source IP, for CUIC realtime reports endpoints. **Default:** 4	No	—
NGX_CUIC_REALTIME_REPORT_NEW_CONN_LIMIT Specifies the number of concurrent connections allowed, per source IP, for CUIC realtime reports newRest endpoints. **Default:** 4	No	—
NGX_CUIC_REALTIME_REPORT_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for CUIC realtime reports endpoints. **Default:** 4	No	—
NGX_CUIC_REPORT_EXECUTION_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for CUIC reports execution endpoints. **Default:** 4	No	—

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_CUIC_PROXY_BUFFER_SIZE Specifies the size of the buffer used for reading the first part of the response received from the server, which is enabled for reverse-proxy. **Default:** 8k	No	—
NGX_CUIC_MAX_TEMP_FILE_SIZE When buffering of responses from the server (which is enabled for reverse-proxy) is enabled, and the whole response doesn't fit into the buffers set by the NGX_CUIC_PROXY_BUFFER_SIZE, a part of the response is saved to a temporary file. **Default:** 100m	No	—

Unified Intelligence Center 12.6(2) properties

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_PRXY_CUIC_HOSTNAME Specifies the reverse-proxy hostname over which this cuic host is accessed. **Default:** reverseproxy.host.domain	Yes	If the reverse-proxy CUIC host FQDN changes.
NGX_PRXY_CUIC_PORT Specifies the reverse-proxy port over which this cuic host is accessed. **Default:** 443	Yes, If required.	If the reverse-proxy port changes.
NGX_CUIC_HOSTNAME Specifies the CUIC server hostname. **Default:** cuic.host.domain	Yes	If the upstream CUIC host FQDN changes.

Unified Intelligence Center 12.6(2) properties that are not recommended to be altered

Note These properties are provided for reference and they exist in the configuration to provide flexibility to adjust the behavior if necessary, in exceptional situations and aren’t recommended changing casually without extensive testing.

Property Name, Description, and Default	Change Recommended?	When to Change?
TEMPLATE_TYPE Specifies the component template type—valid values are: chat, cuic, finesse, ids, livedata, cuic_12.6(1), livedata_12.6(1), and idp-adfs3. **Default:** cuic	No	
NGX_CUIC_UPSTREAM_KEEPALIVE Activates the cache for connections to the upstream CUIC server. Specifies the maximum number of idle keepalive connections to upstream servers that are preserved in the cache of each worker process. When the count exceeds this number, the system closes the least recently used connections. **Default:** 128	No	
NGX_CUIC_CACHE_SIZE Specifies the initial size of the CUIC cache. **Default:** 30m	No	
NGX_CUIC_CACHE_MAX_SIZE Specifies the maximum size for the CUIC cache. When the size exceeds the set maximum, or there isn't enough free space, the system removes the least recently used data. **Default:** 100m	No	
NGX_CUIC_CACHE_INACTIVE_DURATION Specifies the content inactive duration for the CUIC cache. Cached data that aren't accessed during the time specified get removed from the cache regardless of their freshness. **Default:** 3y	No	
NGX_CUIC_HTTP1_CONN_LIMIT Specifies the number of concurrent HTTP/1.1 connections allowed per source IP. **Default:** 12	No	
NGX_CUIC_HTTP2_CONN_LIMIT Specifies the number of concurrent HTTP/2 streams allowed per source IP. **Default:** 150	No	
NGX_CUIC_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit. **Default:** 100	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_CUIC_HISTORICAL_REPORT_CONN_LIMIT Specifies the number of concurrent connections allowed per source IP for CUIC historical reports endpoints. **Default:** 4	No	
NGX_CUIC_HISTORICAL_REPORT_NEW_CONN_LIMIT Specifies the number of concurrent connections allowed per source IP for CUIC historical reports newRest endpoints. **Default:** 4	No	
NGX_CUIC_HISTORICAL_REPORT_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for CUIC historical reports endpoints. **Default:** 4	No	
NGX_CUIC_REALTIME_REPORT_CONN_LIMIT Specifies the number of concurrent connections allowed per source IP for CUIC realtime reports endpoints. **Default:** 4	No	
NGX_CUIC_REALTIME_REPORT_NEW_CONN_LIMIT Specifies the number of concurrent connections allowed per source IP for CUIC realtime reports new REST endpoints. **Default:** 4	No	
NGX_CUIC_REALTIME_REPORT_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for CUIC realtime reports endpoints. **Default:** 4	No	
NGX_CUIC_REPORT_EXECUTION_REQUEST_BURST_LIMIT Specifies the HTTP request burst limit for CUIC reports execution endpoints. **Default:** 4	No	
NGX_CUIC_PROXY_BUFFER_SIZE Configurations overridden from common config for CUIC. Specifies the size of the buffer used for reading the first part of the response received from the server, which is enabled for reverse-proxy. **Default:** 8k	No	

Property Name, Description, and Default	Change Recommended?	When to Change?
NGX_CUIC_MAX_TEMP_FILE_SIZE When buffering of responses from the server (which is enabled for reverse-proxy) is enabled, and the whole response doesn't fit into the buffers set by the NGX_CUIC_PROXY_BUFFER_SIZE, a part of the response is saved to a temporary file. **Default:** 100m	No	