Cloud-Native Network Functions

- *Secure*
- *Robust*
- *Intelligent*

Contents

Overview

F5's Cloud-Native Network Functions (CNFs) deliver full-proxy security and traffic management use-cases, designed for communication service provider (CSP) 5G networks. CNFs integrate F5's containerized Traffic Management Microkernel (TMM) data plane, Edge Firewall, and Custom Resource Definitions (CRDs) into the Robin Cloud Native Platform, to secure, proxy and optimize low-latency 5G workloads.

This document describes the CNFs feature set and software components.

Features

CNFs supports the following features:

- Distributed denial-of-service (DDoS) attack prevention.
- Intrusion detection (IDS) and prevention (IPS).
- Industry-standard network firewall rules.
- Carrier-grade NAT (CGNAT) with large-scale NAT (LSN).
- High performance DNS resolution with caching.
- High performance SR-IOV interface networking.
- Kubernetes IPv4/IPv6 dual-stack networking
- Redundant data storage with persistence
- Diagnostics, statistics and debugging
- Centralized logging collection
- Pod health monitoring

Components

CNFs software comprises four primary components:

BIG-IP Controller

The BIG-IP Controller watches the Kube-API for Custom Resource (CR) update events, and configures both the Edge Firewall and Traffic Management Microkernel (TMM) Proxy Pods based on the update.

Custom Resource Definitions

Custom Resource Definitions (CRDs) extend the Kubernetes API, enabling Edge Firewall and TMM to be configured using CNFs' Custom Resource (CR) objects. CRs configure Edge Firewall and TMM to inspect and protect 5G application traffic. CRs also configure TMM's DNS Caching feature and networking components such as self IP addresses and static routes.

Firewall Policy Compiler

The Firewall Policy Compiler Pod is comprised of a Packet Classification Compiler daemon (PCCD) that converts network firewall rules and CG-NAT configurations into binary large objects (BLOBs). BLOBS are optimized for fast lookup performance, and are then sent to the Traffic Management Microkernel (TMM) Proxy Pod for traffic matching and processing.

TMM Proxy Pod

The Traffic Management Microkernel (TMM) Proxy Pod provide intelligent packet inspection, network address translation, and DNS caching. Additional Proxy Pod containers may also be installed to assist TMM with dynamic routing, logging collection and debugging.

Next step

Continue to the CNFs Release Notes for recent software updates and bug information.

Supplemental

- Kubernetes API

Release Notes

F5 Cloud-Native Network Functions (CNF) - 1.0.0 Limited Availability (LA)

New Features and Improvements

- The F5BigDnsApp CR's `dns64Mode` converts IPv4 DNS responses into IPv6 format.

  Reference CNFs NAT64 and the `dns64Mode` section of the F5BigDnsApp Reference.

- CNFs CRs can now be managed using Kubernetes **shortNames**.

  *Reference the **CRD shortName** section in any of the CNFs CRs.*

- The F5BigDnsApp CR's `udp` parameter has changed to `udpSettings`.

  *Reference the **spec** section of the F5BigDnsApp Reference.*

Bug Fixes

1076469 (TMM)

The Traffic Management Microkernel (TMM) is now able to process Point-to-Point Tunneling Protocol (PPTP) traffic using the F5BigAlgPptp Custom Resource (CR).

Known Issues

1115593 (TMM)

The TMM Proxy Pod is unable to process large files (~1GB) using the F5BigAlgPptp CR when the maximum transmission unit (MTU) is greater than 1450 bytes.

Workaround: *Set the PPTP interface MTU to 1450 bytes or less.*

1112949 (Controller)

The F5BigNatPolicy configuration may appear missing after restarting the TMM Proxy Pod causing subscriber NAT connections to fail.

Workaround: *Delete the AFM Pod to spawn a new Pod, and restore communication between the BIG-IP Controller and PCCD PODs.*

```
kubectl delete pod <afm-pod-name> -n <namespace>
```

1063321 (TMM)

When multiple TMMs are running in a single Namespace, the IP addresses allocated by the F5BigNatPolicy are not reclaimed and reallocated after scaling the TMM deployment down and back up. Client connections may fail due to NAT IP address exhaustion.

Workaround: *Delete and reinstall the F5BigNatPolicy CR.*

1053293 (Controller)

TMM Proxy Pods may fail to receive a self-IP address when the F5BigNetVlan CR allocates the same number self-IPs as running TMM Proxy Pods.

Workaround: *Configure the F5BigNetVlan to allocate twice the number of self-IP addresses as running TMM Proxy Pods.*

Software upgrades

Use these steps to upgrade the CNFs software components:

Important: *Steps **2** through **4** should be performed together, and during a planned maintenance window.*

1. Review the **New Features and Improvements** section above, and integrate any updates into the existing configuration. Do not apply Custom Resource (CR) updates until after the BIG-IP Controller has been upgraded.
2. Follow **Install the CRDs** in the CNFs Software guide to upgrade the CRDs. Be aware that newly applied CRDs will replace existing CRDs of the same name.
3. Uninstall the previous version BIG-IP Controller, and follow the **Installation** procedure in the BIG-IP Controller guide to upgrade the Controller and TMM Proxy Pods. Upgrades have not yet been tested using Helm Upgrade.
4. Once the BIG-IP Controller and TMM Proxy Pods are available, apply any updated CR configurations (step 1) using the `kubectl apply -f <file>` command.
5. The dSSM Databases can be upgraded at anytime using the Upgrading dSSM guide.
6. The Fluentd Logging collector can be upgraded anytime using Helm Upgrade. Review **Extract the Images** in the [CNF Software] guide for the new Fluentd Helm chart location.

Next step

Continue to the Cluster Requirements guide to ensure the cluster has the required software components.

Cluster Requirements

Overview

Prior to integrating Cloud-Native Network Functions (CNF) into the Robin Cloud-Native Platform (CNP), review this document to ensure the required software components are installed and properly configured.

Note: *CNF supports Robin CNP release 5.3.5-207, and contains additional features not present in other Robin CNP releases.*

CPU Allocation

Multiprocessor servers divide memory and CPUs into multiple NUMA nodes, each having a non-shared system bus. The CNF Controller requires that the CPUs and SR-IOV VFs allocated to the Traffic Management Microkernel (TMM) share the same NUMA node. To ensure the NUMA node alignment is handled properly, the Robin installation must include the following parameters and values:

- The `--cpu-manager-policy` must be set to **static**.
- The `--topology-manager-policy` must be set to **single-numa-node**.

CPU Hyper-threading

To optimize 5G workload throughput and avoid container scheduling failures, F5 recommends disabling Simultaneous multithreading (SMT) hyper-threading (HT) on those worker nodes that CNF Pods will be scheduled.

SR-IOV Support

To ensure the BIG-IP Controller discovers and properly allocates Physical Functions (PF) to the TMM container, the following PF/VF driver requirements must be met:

1. Upgrade the E810 ice (SR-IOV PF) driver from version **0.7.1** to version **1.6.4 or later**.
2. Enable SR-IOV support for E810 NICs.
3. Replace **iavf** with the **vfio-pci** VF driver in **/etc/modprobe.d/vfio.conf**:

```
blacklist iavf
options vfio-pci ids=8086:1889
```

4. Enable both **IOMMU** and **hugepages** in **/etc/default/grub** file:

```
GRUB_CMDLINE_LINUX should include iommu=pt intel_iommu=on hugepagesz=2M
 ↪  hugepages=32768  default_hugepagesz=2M vfio-pci.ids=8086:1889
```

5. Generate GRUB configuration file. The commands may differ if you are using BIOS or UEFI:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

```
grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
```

6. Load the VFIO modules at boot time by adding the below entries to the **/etc/modules-load.d/vfio.conf** file:

```
vfio
vfio-pci
```

7. Configure VFs for SR-IOV capable NICs by adding the entries below to the **/etc/rc.d/rc.local** file:

   Note: *You may need to run* ***chmod +x /etc/rc.d/rc.local****.*

   ```
   echo 16 > /sys/class/net/p1p1/sriov_numvfs
   echo 16 > /sys/class/net/p1p2/sriov_numvfs
   ```

8. Reboot the worker node.
9. Confirm hugepages are enabled on the worker node:

   ```
   grep HugePages_Total /proc/meminfo
   ```

   ```
   HugePages_Total:   32768
   ```

10. Confirm the VFIO modules have loaded:

    ```
    lsmod | grep -i vfio
    ```

    ```
    vfio_pci               41993  0
    irqbypass              13503  2 kvm,vfio_pci
    vfio_iommu_type1       22440  0
    vfio                   32657  2 vfio_iommu_type1,vfio_pci
    ```

11. If the VF resources are not discovered by Kubernetes, use the `robin host probe` command to rediscover the host configuration:

    ```
    robin host probe [<hostname>|--all] --rediscover
    ```

IP Pools

Configure the Robin IP Pools parameters with the following values:

- Set the `--spoofchk-disabled` parameter.
- Set the `--trusted` parameter.
- Do not set the `--vlan` parameter, use the F5BigNetVlan CR `tag` parameter.

Example:

```
robin ip-pool add robin-pool-1095 --ranges 10.144.100.1-255 --prefix 16 --driver sriov \
--nictags name=p1p1 --vfdriver vfio-pci --trusted --spoofchk-disabled
```

Persistent storage

The optional Fluentd logging collector, dSSM database and Traffic Management Microkernel (TMM) Debug Sidecar require an available Kubernetes persistent storage to bind to during installation.

Next step

Continue to the Getting Started guide to begin integrating CNFs.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- The CNI project.

Getting started

This document describes each stage of the Cloud-Native Network Functions (CNFs) integration process, and the command line interface (CLI) tools required to complete the integration. A careful review of this document ensures a positive experience.

Note: *You can click* ***Next*** *at the bottom of each page, or scroll through the CNFs PDF to follow the integration process.*

Integration tools

Install the CLI tools listed below on your Linux based workstation:

- Helm CLI - Manages the CNFs Pod installations.
- OpenSSL toolkit - Creates SSL certificates to secure Pod communication.
- Podman - Tags and pushes images to a local registry.

Integration stages

Integrating the CNFs software images involves four *essential* stages to begin processing application traffic, and two *optional* stages to enable logging collection and session-state data persistence:

1. CNFs Software - Extract and upload the CRDs and software images to a local container registry.
2. CNFs Secrets - Secure communication between the Controller and the AFM and TMM Pods.
3. Fluentd Logging - ***Optional:*** Centralize logging data sent from each of the CNF Pods.
4. dSSM Database - ***Optional:*** Store session-state data for the AFM and TMM Pods.
5. BIG-IP Controller - Prepare the cluster to proxy and load balance application traffic.
6. CNFs CRs - Configure a Custom Resource (CR) to begin processing application traffic.

Next step

Continue to the CNFs Software guide to extract and install the CNFs software images and CRDs.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- CNF [Config File Reference]
- Kubernetes Custom Resources
- Kubernetes Ingress

CNFs Software

Overview

The Cloud-Native Network Functions (CNFs) software images and installation Helm charts are provided in a single tape archive (TAR) file. A CNFs Custom Resource Definitions (CRD) TAR file is also provided. Once validated and extracted, the CNFs software images can be integrated into the cluster.

This document describes the CNFs software, and guides you through validating, extracting and installing the CNF software components.

Software images

The table below lists and describes the software images for this software release. For a full list of software images by release, refer to the Software Releases guide.

Note: *The software image name and deployed container name may differ.*

Image	Version	Description
f5ingress	v6.0.13	The *helm_release-f5ingress* container is a custom CNF controller that watches the K8S API for CR updates, and configures either AFM or TMM based on the update.
tmm-img	v2.0.5	The *f5-tmm* container is a Traffic Management Microkernel (TMM) instance that proxies and load balances application traffic between the external and internal networks.
f5-l4p-engine	v1.3.48	The *f5-afm-pccd* container is an Application Firewall Manager (AFM) instance that converts firewall rules and NAT policies into the binary large objects (BLOBs) used by TMM.
f5-nsec-ips-daemon	v1.4.14	The *f5-ipsd* container is the intrusion detection and prevention instance, providing deep packet inspection and prevention of malignant network packets.
tmrouted-img	v0.8.21	The *f5-tmm-tmrouted* container proxies and forwards information between the *f5-tmm-routing* and *f5-tmm* containers.
f5dr-img	v0.5.7	The *f5-tmm-routing* container maintains the dynamic routing tables used by TMM.
f5-toda-tmstatsd	v1.7.7	The *f5-toda-stats* container collects application traffic processing statistics from the *f5-tmm* container, and forwards the data to the *f5-fluentbit* container.
f5-dssm-store	v1.21.0	Contains two sets of software images; The *f5-dssm-db* containers that store shared, persisted session state data, and the *f5-dssm-sentinel* containers to monitor the *f5-dssm-db* containers. For more info, refer to dSSM database.

Image	Version	Description
f5-debug-sidecar	v5.54.3	The *debug* container provides diagnostic tools for viewing TMM's configuration, traffic processing statistics and gathering TMM diagnostic data. For more info, refer to Debug Sidecar.
f5-fluentbit	v0.2.0	The *fluentbit* container collects and forwards statistics to the *f5-fluentd* container.
f5-fluentbit	v1.2.29	The *fluentbit* container collects and forwards statistics to the *f5-fluentd* container.
f5-fluentd	v1.4.9	The *f5-fluentd* container collects statistics and logging data from the Controller, TMM and dSSM Pods. For more info, refer to Fluentd Logging.
opentelemetry-collector	0.46.0	The **otel-collector** container gathers metrics and statistics from the TMM Pods. Refer to [OTEL Collector].
f5-dssm-upgrader	1.0.5	The *dssm-upgrade-hook* enables dSSM DBs upgrades without service interruption or data loss. Refer to Upgrading dSSM.

Requirements

Ensure you have:

- Obtained the CNF software tarball.
- A local container registry.
- A workstation with Podman and OpenSSL.

Procedures

Validate and extract

Use the following steps to extract the CNFs software images.

1. Create a new directory for the CNFs files:

   ```
   mkdir <directory>
   ```

   *In this example, the new directory is named **cnfinstall**:*

   ```
   mkdir cnfinstall
   ```

2. Move the CNFs files into the directory:

   ```
   mv f5-cnf-tarball* f5-cnf-1.0.0.pem cnfinstall
   ```

3. Change into the directory and list the files:

   ```
   cd cnfinstall; ls -1
   ```

   The files appear as:

```
f5-cnf-1.0.0.pem
f5-cnf-tarball-1.0.0.tgz
f5-cnf-tarball-sha512.txt-1.0.0.sha512.sig
f5-cnf-tarball.tgz-1.0.0.sha512.sig
```

4. Use the PEM signing key and each SHA signature file to validate the CNFs TAR file:

```
openssl dgst -verify <pem file>.pem -keyform PEM \
-sha512 -signature <sig file>.sig <tar file>.tgz
```

*The command output states **Verified OK** for each signature file:*

```
openssl dgst -verify f5-cnf-1.0.0.pem -keyform PEM -sha512 \
-signature f5-cnf-tarball.tgz-1.0.0.sha512.sig f5-cnf-tarball-1.0.0.tgz
```

```
Verified OK
```

```
openssl dgst -verify f5-cnf-1.0.0.pem -keyform PEM -sha512 \
-signature f5-cnf-tarball-sha512.txt-1.0.0.sha512.sig f5-cnf-tarball-1.0.0.tgz
```

```
Verified OK
```

5. Extract the CNFs images, Helm charts, and CRDs from the TAR file:

```
tar xvf f5-cnf-tarball-1.0.0.tgz
```

6. List the newly extracted files:

```
ls -1
```

The file list shows the CRD bundless and the SPK image TAR file named f5-cnf-images-1.0.0.tgz:

```
f5-cnf-1.0.0.pem
f5-cnf-crds-n6lan-0.36.7.tgz
f5-cnf-images-1.0.0.tgz
f5-cnf-tarball-1.0.0.tgz
f5-cnf-tarball-sha512.txt-1.0.0.sha512.sig
f5-cnf-tarball.tgz-1.0.0.sha512.sig
```

7. Extract the CNF Helm charts and software images:

```
tar xvf f5-cnf-images-1.0.0.tgz
```

8. List the extracted Helm charts and software images:

```
ls -1R
```

_The file list shows a new **tar** directory with the following files:

```
f5-cnf-1.0.0.pem
f5-cnf-crds-n6lan-0.36.7.tgz
f5-cnf-images-1.0.0.tgz
f5-cnf-tarball-1.0.0.tgz
f5-cnf-tarball-sha512.txt-1.0.0.sha512.sig
f5-cnf-tarball.tgz-1.0.0.sha512.sig
tar

./tar:
```

```
cnf-docker-images.tgz
f5-dssm-0.22.12.tgz
f5-toda-fluentd-1.8.30.tgz
f5ingress-6.0.13.tgz
```

Install CRDs

Use the following steps to extract and install the new CNF CRDs.

1. List the CNF CRD bundle:

   ```
   ls -1 | grep crd
   ```

   The file list shows three CRD bundles:

   ```
   f5-cnf-crds-n6lan-0.36.7.tgz
   ```

2. Extract the CRDs from the bundle:

   ```
   tar xvf f5-cnf-crds-n6lan-0.36.7.tgz
   ```

3. Install the CRDs:

   ```
   kubectl apply -f f5-cnf-crds-n6lan/crds
   ```

   *Note the command output: Newly installed CRDs will be indicated by **created**, and updated CRDs will be indicated by **configured**:*

   ```
   customresourcedefinition.apiextensions.k8s.io/f5-big-alg-ftps.k8s.f5net.com created
   customresourcedefinition.apiextensions.k8s.io/f5-big-alg-pptps.k8s.f5net.com created
   customresourcedefinition.apiextensions.k8s.io/f5-big-alg-rtsps.k8s.f5net.com created
   customresourcedefinition.apiextensions.k8s.io/f5-big-alg-tftps.k8s.f5net.com created
   customresourcedefinition.apiextensions.k8s.io/f5-big-cec-pe-
    ↪  globaloptionses.k8s.f5net.com
    ↪  created
   customresourcedefinition.apiextensions.k8s.io/f5-big-
    ↪  classificationprofiles.k8s.f5net.com
    ↪  created
   customresourcedefinition.apiextensions.k8s.io/f5-big-cne-addresslists.k8s.f5net.com
    ↪  created
   customresourcedefinition.apiextensions.k8s.io/f5-big-cne-portlists.k8s.f5net.com
    ↪  created
   customresourcedefinition.apiextensions.k8s.io/f5-big-cne-snatpools.k8s.f5net.com
    ↪  created
   customresourcedefinition.apiextensions.k8s.io/f5-big-context-secures.k8s.f5net.com
    ↪  created
   customresourcedefinition.apiextensions.k8s.io/f5-big-datagroups.k8s.f5net.com created
   customresourcedefinition.apiextensions.k8s.io/f5-big-ddos-policies.dos.k8s.f5net.com
    ↪  created
   customresourcedefinition.apiextensions.k8s.io/f5-big-dns-apps.dns.k8s.f5net.com
    ↪  created
   customresourcedefinition.apiextensions.k8s.io/f5-big-dns-caches.k8s.f5net.com created
   customresourcedefinition.apiextensions.k8s.io/f5-big-dns-zones.k8s.f5net.com created
   customresourcedefinition.apiextensions.k8s.io/f5-big-
    ↪  dynamicappscategorieses.k8s.f5net.com
    ↪  created
   ```

```
customresourcedefinition.apiextensions.k8s.io/f5-big-fastl4-settings.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-fw-policies.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-ips-policies.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-log-hslpubs.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-log-profiles.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-nat-policies.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-net-staticroutes.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-net-vlans.k8s.f5net.com created
customresourcedefinition.apiextensions.k8s.io/f5-big-pe-policies.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-pe-profiles.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-tcp-settings.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-udp-settings.k8s.f5net.com
 ↪  created
customresourcedefinition.apiextensions.k8s.io/f5-big-zeroratingirules.k8s.f5net.com
 ↪  created
```

4. List the installed CNFs CRDs:

```
kubectl get crds | grep f5-big
```

The CRD listing will contain the full list of CRDs:

```
f5-big-alg-ftps.k8s.f5net.com                  2022-06-14T18:03:26Z
f5-big-alg-pptps.k8s.f5net.com                 2022-06-14T18:03:26Z
f5-big-alg-rtsps.k8s.f5net.com                 2022-06-14T18:03:26Z
f5-big-alg-tftps.k8s.f5net.com                 2022-06-14T18:03:26Z
f5-big-cec-pe-globaloptionses.k8s.f5net.com    2022-06-14T18:03:26Z
f5-big-classificationprofiles.k8s.f5net.com    2022-06-14T18:03:26Z
f5-big-cne-addresslists.k8s.f5net.com          2022-06-14T18:03:26Z
f5-big-cne-portlists.k8s.f5net.com             2022-06-14T18:03:26Z
f5-big-cne-snatpools.k8s.f5net.com             2022-06-14T18:03:26Z
f5-big-context-secures.k8s.f5net.com           2022-06-14T18:03:26Z
f5-big-datagroups.k8s.f5net.com                2022-06-14T18:03:26Z
f5-big-ddos-policies.dos.k8s.f5net.com         2022-06-14T18:03:26Z
f5-big-dns-apps.dns.k8s.f5net.com              2022-06-14T18:03:26Z
f5-big-dns-caches.k8s.f5net.com                2022-06-14T18:03:26Z
f5-big-dns-zones.k8s.f5net.com                 2022-06-14T18:03:26Z
f5-big-dynamicappscategorieses.k8s.f5net.com   2022-06-14T18:03:26Z
f5-big-fastl4-settings.k8s.f5net.com           2022-06-14T18:03:26Z
f5-big-fw-policies.k8s.f5net.com               2022-06-14T18:03:26Z
f5-big-ips-policies.k8s.f5net.com              2022-06-14T18:03:26Z
f5-big-log-hslpubs.k8s.f5net.com               2022-06-14T18:03:26Z
f5-big-log-profiles.k8s.f5net.com              2022-06-14T18:03:26Z
f5-big-nat-policies.k8s.f5net.com              2022-06-14T18:03:26Z
f5-big-net-staticroutes.k8s.f5net.com          2022-06-14T18:03:26Z
f5-big-net-vlans.k8s.f5net.com                 2022-06-14T18:03:26Z
```

```
f5-big-pe-policies.k8s.f5net.com              2022-06-14T18:03:26Z
f5-big-pe-profiles.k8s.f5net.com              2022-06-14T18:03:26Z
f5-big-tcp-settings.k8s.f5net.com             2022-06-14T18:03:26Z
f5-big-udp-settings.k8s.f5net.com             2022-06-14T18:03:26Z
f5-big-zeroratingirules.k8s.f5net.com         2022-06-14T18:03:26Z
```

Upload the images

Use the following steps to upload the CNFs software images to a local container registry.

1. Install the CNFs images to your workstation's Podman image store:

```
podman load -i tar/cnf-docker-images.tgz
```

2. List the CNF images to be tagged and pushed to the local container registry in the next step:

```
podman images local.registry/*
```

```
REPOSITORY                              TAG         IMAGE ID
local.registry/f5ingress                v6.0.13     4cc1d240c9f8
local.registry/tmm-img                  v2.0.5      c984e4fce366
local.registry/f5-dssm-upgrader         1.0.5       09ef63c78f9a
local.registry/f5ingress                v6.0.12     9a4bd5a78e21
local.registry/tmm-img                  v2.0.4      1d6bf53236cf
local.registry/f5-l4p-engine            v1.3.48     79ac6ac14b7e
local.registry/f5-nsec-ips-daemon       v1.4.14     dbbdd0fd7adc
local.registry/f5-toda-tmstatsd         v1.7.7      3b98fcca9779
local.registry/f5-fluentd               v1.4.9      c76475173491
local.registry/f5-fluentbit             v0.2.0      c4a875a37273
local.registry/f5dr-img                 v0.5.7      c398809553fd
local.registry/f5-dssm-store            v1.21.0     5037f6eced8d
local.registry/f5dr-img-init            v0.5.7      7463a23b1459B
local.registry/tmrouted-img             v0.8.21     3c7bcc79f890
local.registry/f5-debug-sidecar         v5.54.3     a48ab5c12f96
local.registry/opentelemetry-collector  0.46.0      81b28598879eB
local.registry/f5-fluentbit             v0.1.29     9fb5608ff56cB
```

3. Tag and push each image to the local container registry. For example:

```
podman tag <local.registry/image name>:<version> <registry>/<image name>:<version>
```

```
podman push <registry_name>/<image name>:<version>
```

*In this example, the **f5ingress:v6.0.13** image is tagged and pushed to the remote registry **registry.com**:*

```
podman tag local.registry/f5ingress:v6.0.13 registry.com/f5ingress:v6.0.13
```

```
podman push registry.com/f5ingress:v6.0.13
```

4. Once all of the images have uploaded, verify the images exist in the local container registry:

```
curl -X GET https://<registry>/v2/_catalog -u <user:pass>
```

For example:

```
curl -X GET https://registry.com/v2/_catalog -u cnfadmin:cnfadmin
```

```
"repositories":["f5-debug-sidecar","f5-dssm-store","f5-fluentbit","f5-fluentd","f5-
↪  toda-tmstatsd","f5dr-img","f5ingress","tmm-img","tmrouted-img"]}
```

Next step

Continue to the CNFs Secrets guide to secure CNFs communications.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Using Podman load.

CNFs Secrets

Overview

The Cloud-Native Network Functions (CNFs) BIG-IP Controller, Edge Firewall, and Traffic Management Microkernel (TMM) Proxy Pods communicate over a secure channel using the gRPC (remote procedure call) framework. To secure the gRPC channel, SSL/TLS keys and certificates must be generated and stored as Secrets in the cluster.

Note: *The gRPC channel is established over TCP service port* ***8750****.*

This document guides you through understanding, generating and installing the CNFs gRPC Secrets.

Validity period

SSL/TLS certificates are valid for a specific period of time, and once they expire, secure connections fail when attempting to validate the certificate. When creating new SSL/TLS certificates for the gRPC channel, it is recommended that you choose a period of **one year**, or **two years** to avoid connection failures.

Example SSL Certificate validity period:

```
Validity
    Not Before: Jan 1  10:30:00 2021 GMT
    Not After : Jan 1  10:30:00 2022 GMT
```

Updating Secrets

When planning to replace previously installed gRPC Secrets, you must restart the BIG-IP Controller and Service Proxy TMM Proxy Pods to begin using the new Secrets. To replace existing Secrets, refer to the Restarting section of this guide.

Important: *Restarting the Service Proxy TMM Proxy Pods impacts traffic processing.*

Requirements

Ensure you have:

- A Robin cluster.
- A workstation with OpenSSL installed.

Procedures

Generating Secrets

Use the following steps to generate the gRPC SSL/TLS keys and certificates.

Note: *The commands used to generate the Secrets can be downloaded here.*

1. Change into the directory with the CNF files:

   ```
   cd <directory>
   ```

   In this example, the CNF files are in the ***cnfinstall*** *directory:*

```
cd cnfinstall
```

2. Create a new directory for the gRPC Secret keys and certificates, and change into the directory:

```
mkdir <directory>
```

```
cd <directory>
```

*In this example, a new directory named **grpc_secrets** is created and changed into:*

```
mkdir grpc_secrets
```

```
cd grpc_secrets
```

3. Create the gRPC Certificate Authority (CA) signing key and certificate:

***Note:** Adapt the number of **-days** the certificate will be valid, and the **-subj** information for your environment.*

```
openssl genrsa -out grpc-ca.key 4096
```

```
openssl req -x509 -new -nodes -key grpc-ca.key -sha256 -days 365 -out grpc-ca.crt \
-subj "/C=US/ST=WA/L=Seattle/O=F5/OU=Dev/CN=ca"
```

4. The following code creates a new file named **server.ext** with the required SSL/TLS attributes:

```
echo "[req_ext]" > server.ext
echo " " >> server.ext
echo "subjectAltName = @alt_names" >> server.ext
echo " " >> server.ext
echo "[alt_names]" >> server.ext
echo " " >> server.ext
echo "DNS.1 = grpc-svc" >> server.ext
echo "DNS.2 = grpc-pccd-svc" >> server.ext
echo "DNS.3 = grpc-ipsd-svc" >> server.ext
```

*The **server.ext** file should contain the following SSL/TLS attributes:*

```
[req_ext]

subjectAltName = @alt_names

[alt_names]

DNS.1 = grpc-svc
DNS.2 = grpc-pccd-svc
DNS.3 = grpc-ipsd-svc
```

5. Create the gRPC server SSL/TLS key, certificate signing request (CSR), and signed certificate:

***Note:** Adapt the number of **-days** the certificate will be valid, and the **-subj** information for your environment.*

```
openssl genrsa -out grpc-server.key 4096
```

```
openssl req -new -key grpc-server.key -out grpc-server.csr \
-subj "/C=US/ST=WA/L=Seattle/O=F5/OU=PD/CN=f5net.com"
```

```
openssl x509 -req -in grpc-server.csr -CA grpc-ca.crt -CAkey grpc-ca.key \
-CAcreateserial -out grpc-server.crt -extensions req_ext -days 365 -sha256 \
-extfile server.ext
```

6. The following code creates a new file named **client.ext** with the required SSL/TLS attributes:

```
echo "[req_ext]" > client.ext
echo " " >> client.ext
echo "subjectAltName = @alt_names" >> client.ext
echo " " >> client.ext
echo "[alt_names]" >> client.ext
echo " " >> client.ext
echo "email.1 = clientcert@f5net.com" >> client.ext
```

*The **client.ext** file should contain the following SSL/TLS attributes:*

```
[req_ext]

subjectAltName = @alt_names

[alt_names]

email.1 = clientcert@f5net.com
```

7. Create the TMM gRPC client key, CSR and signed certificate:

***Note:** Adapt the number of **-days** the certificate will be valid, and the **-subj** information for your environment.*

```
openssl genrsa -out grpc-client.key 4096
```

```
openssl req -new -key grpc-client.key -out grpc-client.csr \
-subj "/C=US/ST=WA/L=Seattle/O=F5/OU=PD/CN=f5net.com"
```

```
openssl x509 -req -in grpc-client.csr -CA grpc-ca.crt -CAkey grpc-ca.key \
-set_serial 101 -outform PEM -out grpc-client.crt -extensions req_ext -days 365 \
-sha256 -extfile client.ext
```

8. Create the Edge Firewall (pccd) gRPC client key, CSR and signed certificate:

***Note:** Adapt the number of **-days** the certificate will be valid, and the **-subj** information for your environment.*

```
openssl genrsa -out grpc-pccd-client.key 4096
```

```
openssl req -new -key grpc-pccd-client.key -out grpc-pccd-client.csr \
-subj "/C=US/ST=WA/L=Seattle/O=F5/OU=PD/CN=f5net.com"
```

```
openssl x509 -req -in grpc-pccd-client.csr -CA grpc-ca.crt -CAkey grpc-ca.key \
-set_serial 101 -outform PEM -out grpc-pccd-client.crt -extensions req_ext -days 365 \
-sha256 -extfile client.ext
```

9. Create the Intrusion Prevention (ipsd) gRPC client key, CSR and signed certificate:

***Note:** Adapt the number of **-days** the certificate will be valid, and the **-subj** information for your environment.*

```
openssl genrsa -out grpc-ipsd-client.key 4096
```

```
openssl req -new -key grpc-ipsd-client.key -out grpc-ipsd-client.csr \
-subj "/C=US/ST=WA/L=Seattle/O=F5/OU=PD/CN=f5net.com"
```

```
openssl x509 -req -in grpc-ipsd-client.csr -CA grpc-ca.crt -CAkey grpc-ca.key \
-set_serial 101 -outform PEM -out grpc-ipsd-client.crt -extensions req_ext -days 365 \
-sha256 -extfile client.ext
```

Installing Secrets

Use the following steps to encode, and store the SSL/TLS keys and certificates as Secrets in the cluster.

1. The following code performs a **Base64 encoding** of the keys and certificates:

```
openssl base64 -A -in grpc-ca.crt -out grpc-ca-encode.crt
openssl base64 -A -in grpc-server.crt -out grpc-server-encode.crt
openssl base64 -A -in grpc-client.crt -out grpc-client-encode.crt
openssl base64 -A -in grpc-pccd-client.crt -out grpc-pccd-client-encode.crt
openssl base64 -A -in grpc-ipsd-client.crt -out grpc-ipsd-client-encode.crt
openssl base64 -A -in grpc-server.key -out grpc-server-encode.key
openssl base64 -A -in grpc-ca.key -out grpc-ca-encode.key
openssl base64 -A -in grpc-client.key -out grpc-client-encode.key
openssl base64 -A -in grpc-pccd-client.key -out grpc-pccd-client-encode.key
openssl base64 -A -in grpc-ipsd-client.key -out grpc-ipsd-client-encode.key
```

2. The following code creates the K8S Secret object used to store SSL/TLS **keys**:

   Important: *The syntax in the bottom three lines;* ***grpc-svc.key***, ***priv.key***, *and* ***f5-ing-demo-f5ingress.key***, *must be set as in the example.*

```
echo "apiVersion: v1" > keys-secret.yaml
echo "kind: Secret" >> keys-secret.yaml
echo "metadata:" >> keys-secret.yaml
echo " name: keys-secret" >> keys-secret.yaml
echo "data:" >> keys-secret.yaml
echo " grpc-svc.key: `cat grpc-server-encode.key`" >> keys-secret.yaml
echo " priv.key: `cat grpc-ca-encode.key`" >> keys-secret.yaml
echo " f5-ing-demo-f5ingress.key: `cat grpc-client-encode.key`" >> keys-secret.yaml
echo " grpc-pccd-client.key: `cat grpc-pccd-client-encode.key`" >> keys-secret.yaml
echo " grpc-ipsd-client.key: `cat grpc-ipsd-client-encode.key`" >> keys-secret.yaml
```

3. The following code creates the K8S Secret object used to store the SSL/TLS **certificates**:

   Important: *The syntax in the bottom three lines;* ***grpc-svc.crt***, ***ca_root.crt***, *and* ***f5-ing-demo-f5ingress.crt***, *must be set as in the example.*

```
echo "apiVersion: v1" > certs-secret.yaml
echo "kind: Secret" >> certs-secret.yaml
echo "metadata:" >> certs-secret.yaml
echo " name: certs-secret" >> certs-secret.yaml
echo "data:" >> certs-secret.yaml
echo " grpc-svc.crt: `cat grpc-server-encode.crt`" >> certs-secret.yaml
echo " ca_root.crt: `cat grpc-ca-encode.crt`" >> certs-secret.yaml
echo " f5-ing-demo-f5ingress.crt: `cat grpc-client-encode.crt`" >> certs-secret.yaml
echo " grpc-pccd-client.crt: `cat grpc-pccd-client-encode.crt`" >> certs-secret.yaml
echo " grpc-ipsd-client.crt: `cat grpc-ipsd-client-encode.crt`" >> certs-secret.yaml
```

4. Create a new Namespace for the CNF Pods:

```
robin namespace add cnf-gateway
```

5. Install the Secret key and certificate objects:

```
kubectl apply -f keys-secret.yaml -n cnf-gateway
kubectl apply -f certs-secret.yaml -n cnf-gateway
```

*The command responses should state the Secrets have been **created**:*

```
secret/keys-secret created
secret/certs-secret created
```

6. The new Secrets will now be used to secure the gRPC channel.

Next step

Continue to one of the following guides listed by installation precedence:

- ***Optional***: Install the Fluentd Logging collector to centralize CNF container logging.
- ***Optional***: Install the dSSM Database to store session-state information.
- ***Required***: Install the BIG-IP Controller, Edge Firewall and TMM Pods.

Restarting

This procedure assumes that you have deployed the Controller, Edge Firewall and TMM Pods, and have created a new set of Secrets to replace the existing Secrets. New Secrets will not be used until the Controller and TMM Pods have been restarted.

Important: *Restarting the TMM Pod impacts traffic processing.*

1. Obtain the name and number of Controller and TMM Pods:

```
kubectl get deploy -n cnf-gateway
```

*In this example, there is **1** Controller and **3** TMM Pods:*

```
NAME                  READY   AVAILABLE
f5ingress-f5ingress   1/1     1
f5-tmm                3/3     3
```

2. Scale the number of TMM Pods to **0**:

```
kubectl scale deploy f5-tmm --replicas=0 -n cnf-gateway
```

3. Ensure **0** (none) of the **f5-tmm** Pods are **AVAILABLE**:

```
NAME                  READY   AVAILABLE
f5ingress-f5ingress   1/1     1
f5-tmm                0/0     0
```

4. Scale the TMM Pods back to the previous number:

```
kubectl scale deploy f5-tmm --replicas=<number> -n <namespace>
```

*In this example the TMM Pods are scaled back to **3**:*

```
kubectl scale deployment f5-tmm --replicas=3  -n cnf-gateway
```

5. Ensure **3** (all) of the **f5-tmm** Pods are **AVAILABLE**:

```
NAME                  READY   AVAILABLE
f5ingress-f5ingress   1/1     1
f5-tmm                3/3     3
```

6. Scale the Controller to **0**:

```
kubectl scale deployment <name> --replicas=0 -n <namespace>
```

For example:

```
kubectl scale deploy f5ingress-f5ingress --replicas=0 -n cnf-gateway
```

7. Ensure **0** (none) of the Controller Pods are **AVAILABLE**:

```
NAME                  READY   AVAILABLE
f5ingress-f5ingress   0/0     0
f5-tmm                3/3     3
```

8. Scale the Controller back to the previous number:

```
kubectl scale deployment <name> --replicas=1  -n <namespace>
```

*In this example the Controller is scaled back to **1**:*

```
kubectl scale deployment f5ingress-f5ingress --replicas=1 -n cnf-gateway
```

9. Ensure the Controller Pod is **AVAILABLE**:

```
NAME                  READY   AVAILABLE
f5ingress-f5ingress   1/1     1
f5-tmm                3/3     3
```

10. The new Secrets should now be used to secure the gRPC channel.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental Information

- The list of commands used to create the Secrets.
- Introduction to gRPC
- Kubernetes Secrets

Fluentd Logging

Overview

The Cloud-Native Network Functions (CNFs) Fluentd Pod is an open source data collector that can be configured to receive logging data from the BIG-IP Controller, Traffic Management Microkernel (TMM), Edge Firewall, Distributed Session State Managment (dSSM) Pods, and BGP updates from the TMM routing container. The Fluentd Pod must bind to a Kubernetes persistence volume in order to create the necessary log file directories.

This document guides you through understanding, configuring and deploying the **f5-fluentd** logging container.

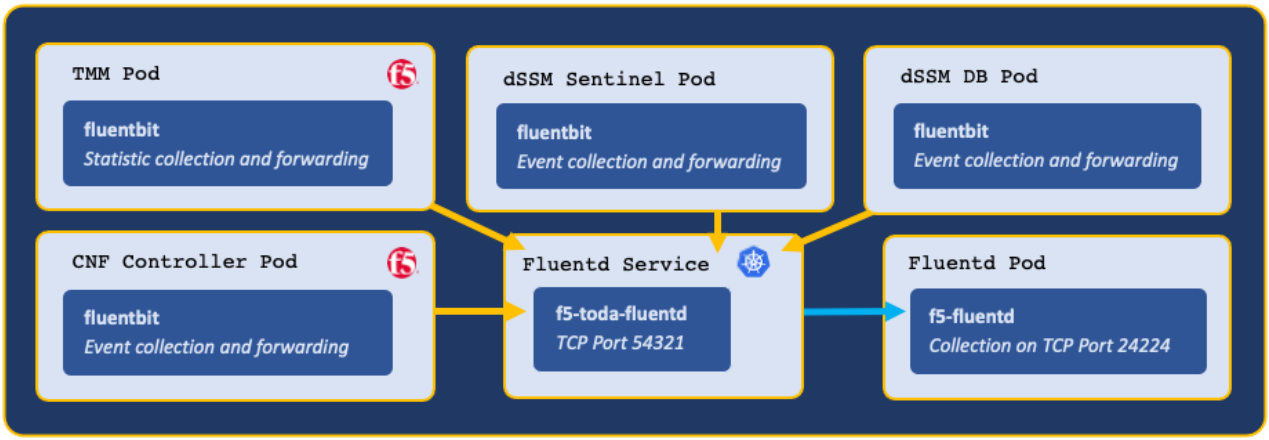
Fluentd Service

When installing Fluentd, a Service object is created to receive logging data on TCP service port **54321**, and forward the data to Fluentd on TCP service port **24224**.

Example Fluentd Service:

```
Name:              f5-toda-fluentd
Namespace:         cnf-gateway
IP:                10.109.102.215
Port:              <unset>  54321/TCP
Endpoints:         10.244.1.75:24224
```

Example Fluentd integration:



Log file locations

Fluentd collects logging data in the following log files:

Container	Log file
f5-dssm-sentinel	/var/log/f5/f5-dssm-sentinel-0/**sentinel.log**
f5-dssm-db	/var/log/f5/f5-dssm-db-0/**dssm.log**
f5ingress	/var/log/f5/helm_release-f5ingress/pod_name/**f5ingress.log**
f5-tmm	/var/log/f5/f5-tmm/pod_name/**f5-fsm-tmm.log**
f5-tmm-routing	/var/log/f5/f5-tmm/pod_name/**f5-tmm-routing.log**

Note: *To modify the TMM logging level, review the* ***tmm_cli*** *section of the Debug Sidecar overview.*

Requirements

Prior to installing Fluentd, ensure you have:

- An available persistence volume.
- Installed the CNFs software.
- A Linux based workstation with Helm installed.

Procedures

Installation

Use the following steps to the install the **f5-fluentd** container.

1. Change into local directory with the CNF files, and list the files in the **tar** directory:

   In this example, the CNF files are in the ***cnfinstall*** *directory:*

   ```
   cd cnfinstall
   ```

   ```
   ls -1 tar
   ```

   In this example, Fluentd Helm chart is named ***f5-toda-fluentd-1.8.30.tgz****:*

   ```
   cnf-docker-images.tgz
   f5-dssm-0.22.12.tgz
   f5-toda-fluentd-1.8.30.tgz
   f5ingress-6.0.13.tgz
   ```

2. Create a Helm values file named **fluentd-values.yaml**, and set the `image.repository`, `persistence.storageClass`, and `robinNetworks` parameters:

   ```
   image:
     repository: <registry>

   persistence:
     enabled: true
     storageClass: "<name>"

   robinNetworks: true
   ```

   In this example, Helm pulls the ***f5-fluentd*** *image from* ***registry.com****, and the container will bind to the storageClass named* ***managed-nfs-storage****:*

   ```
   image:
     repository: registry.com

   persistence:
     enabled: true
     storageClass: "managed-nfs-storage"

   robinNetworks: true
   ```

3. **Optional:** Add the following parameters to the values file to collect logging data from the Controller, dSSM, and PCCD Pods:

```
# Collect logging from the Ingress Controller Pod
f5ingress_logs:
  enabled: true
  stdout: true
# Collect logging from the dSSM Pods
dssm_logs:
  enabled: true
  stdout: true
# Configuration for sentinel logs
dssm_sentinel_logs:
  enabled: true
  stdout: true
pccd_logs:
  enabled: true
  stdout: true
```

4. Install the **f5-fluentd** container and save the Fluentd hostname for the Controller installation:

```
helm install f5-fluentd <helm chart> -f <values>
```

For example:

```
helm install f5-fluentd f5-toda-fluentd-1.8.30.tgz -f fluentd-values.yaml
```

***Note:** In this example, the Fluentd hostname is **f5-toda-fluentd.cnf-gateway.svc.cluster.local.**:*

```
FluentD hostname: f5-toda-fluentd.cnf-gateway.svc.cluster.local.
FluentD port: "54321"
```

5. The **f5-fluentd** container should now be successfully installed:

```
oc get pods
```

*In this example, the Fluentd Pod **STATUS** is **Running**:*

```
NAME                              READY   STATUS
f5-toda-fluentd-8cf96967b-jxckr   1/1     Running
```

6. Fluentd should also be bound to the persistent volume:

```
oc get pvc
```

*In this example, the Fluentd Pod PVC displays **STATUS** as **Bound**:*

```
NAME              STATUS   VOLUME                                     STORAGECLASS
f5-toda-fluentd   Bound    pvc-7d36b530-b718-466c-9b6e-895e8f1079a2
 ↪  managed-nfs-storage
```

Viewing logs

After installing the BIG-IP Controller and dSSM Pods, you can use the following steps to view the logs in the `f5-fluentd` container:

1. Log in to the fluentd container:

```
kubectl exec -it deploy/f5-toda-fluentd -n <project> -- bash
```

*In this example, the container is in the **cnf-gateway** Project:*

```
kubectl exec -it deploy/f5-toda-fluentd -n cnf-gateway -- bash
```

2. Change to the main logging directory, and list the subdirectories:

```
cd /var/log/f5; ls
```

*In this example, logging directories are present for the **f5ingress**, **f5-tmm**, **f5-dssm-db**, and **f5-dssm-sentinel** Pods:*

```
f5-dssm-db-0  f5-dssm-db-1  f5-dssm-db-2  f5-dssm-sentinel-0
f5-dssm-sentinel-1  f5-dssm-sentinel-2  f5-ingress-f5ingress  f5-tmm
```

3. Change into one of the subdirectories, for example **f5-dssm-db-0**:

```
cd f5-dssm-db-0
```

4. View the logs using the **more** command:

```
more -d dssm.log
```

Next step

Continue to one of the following steps listed by installation precedence:

- ***Optional***: Install the dSSM Database to store session-state information.
- ***Required***: Install the BIG-IP Controller, TMM and AFM Pods.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Fluentbit
- Fluentd
- [CNF Event Logs]

dSSM Database

Overview

The Cloud-Native Network Functions (CNF) distributed Session State Management (dSSM) Pods provide centralized and persistent storage for the Edge Firewall and Traffic Management Microkernel (TMM) Proxy Pods. The dSSM Pods are Redis data structure stores that maintain application traffic data such as NAT translation mappings. The dSSM Pods bind to Kubernetes persistence volumes to persist data in the event of a container restart.

This document describes the dSSM Pods, and guides you through configuring and installing the **f5-dssm-sentinel** and **f5-dssm-db** containers.

Sentinels and DBs

The dSSM Pods integrate as a StatefulSet, containing three dSSM Sentinel Pods and three dSSM DB Pods to maintain high availability. The Sentinel Pods elect and monitor a primary dSSM DB Pod, and if the primary dSSM DB Pod fails, a secondary DB will assume the primary role.

Additional high availability

The dSSM Pods also use the standard Kubernetes node `affinity` and `PodDisruptionBudget` features to maintain additional levels of high availability.

Affinity

Each dSSM Sentinel and DB Pod schedules onto a unique cluster node by default. The dSSM scheduling behavior can be modified using the dSSM Helm `affinity_type` parameter:

Setting	Description
required	Ensures the target cluster node does not currently host a Pod with the `app=f5-dssm-db` annotation (**default**).
preferred	Attempt to schedule Pods onto unique nodes, but two dSSM Pods may schedule onto a single node when no schedulable nodes exists.
custom	Scheduling behavior may be tuned specifically to the cluster admins requirements using the dSSM `values.yaml` file.

Helm parameter examples:

```
sentinel:
  affinity_type: "required"

db:
  affinity_type: "required"
```

Kubernetes Assigning Pods overview.

PodDisruptionBudget

A minimum of 2 dSSM Pods remain available at all times based on the dSSM Helm `pod_disruption_budget` parameter. This parameter blocks **voluntary** interruptions to the dSSM Pod's **Running** status. For example, if three

schedulable nodes are available, and the admin runs `oc adm drain` on two of nodes in quick succession, the second action will be blocked until another schedulable node is added to the cluster.

Helm parameter examples:

```
sentinel:
  pod_disruption_budget:
    min_available: 2

db:
  pod_disruption_budget:
    min_available: 2
```

Kubernetes Disruptions overview.

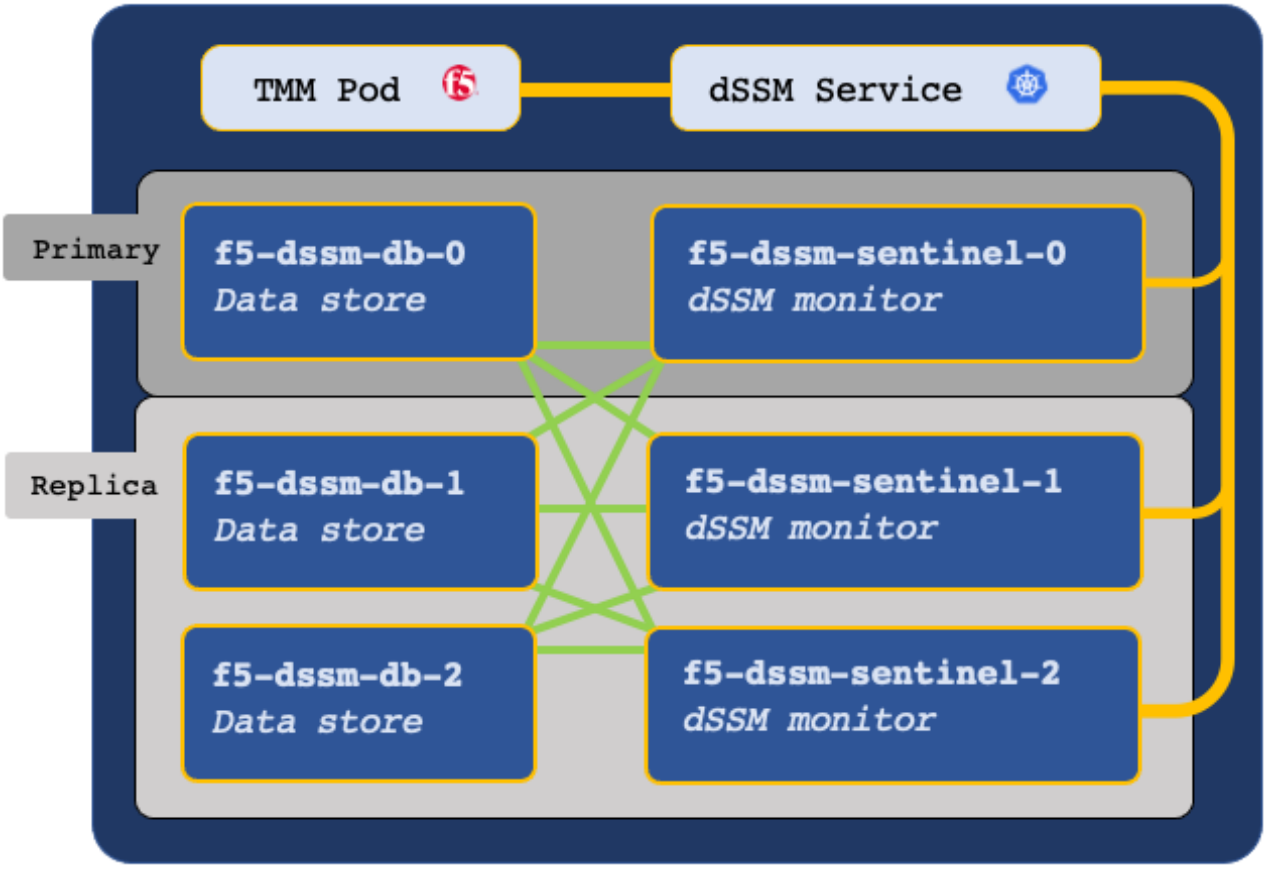
Sentinel Service

The dSSM Sentinel Service receives data from TMM on TCP service port **26379**, and forwards to the dSSM DB Pods using the same service port number.

Example dSSM Service:

```
Name:              f5-dssm-sentinel
Namespace:         cnf-gateway
IP:                10.106.99.127
Port:              sentinel  26379/TCP
Endpoints:         10.244.1.15:26379,10.244.1.20:26379,10.244.4.3:26379
```

Example dSSM deployment:



Secure communication

The TMM, dSSM Sentinel and dSSM DB Pods communicate over a mesh of secure channels. These channels are secured using SSL/TLS keys and certificates stored as Secrets in the cluster. When deploying dSSM, the first step involves creating the SSL/TLS keys and certificates, and installing them as Secrets. Ensure you understand the key points in the following subsections:

Certificate Validity

SSL/TLS certificates are valid for a specific period of time, and once they expire, secure connections fail when validating the certificate. When creating new SSL/TLS certificates for the secure dSSM channels, choose a period of **one year**, or **two years** to avoid connection failures.

Example Certificate Validity:

```
Validity
    Not Before: Jan 1  10:30:00 2021 GMT
    Not After : Jan 1  10:30:00 2022 GMT
```

Updating Secrets

If you plan to replace a current set of Secrets with a new set, you must restart both the dSSM and TMM Pods to begin using the new Secrets. It is important to understand that restarting the TMM Pods causes a brief interruption to traffic processing, and should be performed during a planned maintenance window. To restart dSMM and the TMM Pods, refer to the Restarting procedure.

Requirements

Ensure you have:

- A Robin cluster.
- A workstation with Helm and OpenSSL installed.

Procedures

Install the Secrets

Use the following steps to create the required SSL/TLS keys and certificates, and install them as Secrets in both the TMM and dSSM Namespaces:

Note: *The commands used to generate the Secrets can be downloaded here.*

1. Change into the local directory with the CNF files:

   ```
   cd <directory>
   ```

   *In this example, the CNF files are in the **cnfinstall** directory:*

   ```
   cd cnfinstall
   ```

2. Create a new directory for the dSSM Secret keys and certificates, and change into the directory:

```
mkdir <directory>
```

```
cd <directory>
```

*In this example, a new directory named **dssm_secrets** is created and changed into:*

```
mkdir dssm_secrets
```

```
cd dssm_secrets
```

3. Create the dSSM Certificate Authority (CA) key and certificate:

   In this example, the CA signing certificate is valid for one year.

```
openssl genrsa -out dssm-ca.key 4096
```

```
openssl req -x509 -new -nodes -sha384 \
    -key dssm-ca.key -days 365 \
    -subj '/O=Redis Test/CN=Certificate Authority' \
    -out dssm-ca.crt
```

4. Create the dSSM client key and certificate:

   In this example, the dSSM client certificate is valid for one year.

```
openssl genrsa -out dssm-key.key 4096
```

```
openssl req -new -sha384 -key dssm-key.key \
    -subj '/O=Redis Test/CN=Server' | \
    openssl x509 -req -sha384 -CA dssm-ca.crt \
        -CAkey dssm-ca.key -CAserial dssm-ca.txt \
        -CAcreateserial  -days 365 \
        -out dssm-cert.crt
```

5. Create the mTLS certificate for the TMM and dSSM communication channels:

   ***Note:** The mTLS certificate can take up to a minute to generate.*

```
openssl dhparam -out dhparam2048.pem 2048
```

6. Encode the keys and certificates:

```
cat dssm-ca.crt | base64 -w 0  > dssm-ca-encode.crt
cat dssm-cert.crt | base64 -w 0 > dssm-cert-encode.crt
cat dhparam2048.pem | base64 -w 0 > dhparam2048-encode.pem
cat dssm-key.key | base64 -w 0 > dssm-key-encode.key
```

7. Create the Secret certificate object file:

```
echo "apiVersion: v1" > certs-secret.yaml
echo "kind: Secret" >> certs-secret.yaml
echo "metadata:" >> certs-secret.yaml
echo " name: dssm-certs-secret" >> certs-secret.yaml
echo "data:" >> certs-secret.yaml
echo " dssm-ca.crt: `cat dssm-ca-encode.crt`"  >> certs-secret.yaml
echo " dssm-cert.crt: `cat dssm-cert-encode.crt`" >> certs-secret.yaml
echo " dhparam2048.pem: `cat dhparam2048-encode.pem`" >> certs-secret.yaml
```

8. Create the Secret key object file:

```
echo "apiVersion: v1" > keys-secret.yaml
echo "kind: Secret" >> keys-secret.yaml
echo "metadata:" >> keys-secret.yaml
echo " name: dssm-keys-secret" >> keys-secret.yaml
echo "data:" >> keys-secret.yaml
echo " dssm-key.key: `cat dssm-key-encode.key`" >> keys-secret.yaml
```

9. Install the Secret key and certificate files:

```
kubectl apply -f keys-secret.yaml -n <namespace>
kubectl apply -f certs-secret.yaml -n <namespace>
```

*In this example, the Secrets install to the **cnf-gateway** Namespace:*

```
kubectl apply -f keys-secret.yaml -n cnf-gateway
kubectl apply -f certs-secret.yaml -n cnf-gateway
```

*The command response should state the Secrets have been **created**:*

```
secret/dssm-keys-secret created
secret/dssm-certs-secret created
```

Install the Pods

Use the following steps to deploy the dSSM Pods with persistence.

1. Change into local directory with the CNF TAR files, and ensure the Helm charts have been extracted:

   *In this example, the CNF files are in the **cnfinstall** directory:*

   ```
   cd cnfinstall
   ```

   ```
   ls -1 tar
   ```

   *In this example, the dSSM Helm chart is named **f5-dssm-0.22.12.tgz**:*

   ```
   cnf-docker-images.tgz
   f5-dssm-0.22.12.tgz
   f5-toda-fluentd-1.8.30.tgz
   f5ingress-6.0.13.tgz
   ```

2. Create a Helm values file named **dssm-values**, and set the `image.repository` parameters:

   ```
   image:
     repository: <registry>

   sentinel:
     fluentbit_sidecar:
       image:
         repository: <registry>

   db:
     fluentbit_sidecar:
       image:
         repository: <registry>
   ```

   *In this example, Helm pulls the **f5-dssm-store** images from **registry.com**:*

```
image:
  repository: registry.com

sentinel:
  fluentbit_sidecar:
    image:
      repository: registry.com

db:
  fluentbit_sidecar:
    image:
      repository: registry.com
```

3. **Optional:** If you installed the Fluentd Logging Pod, you can send logging data to the **f5-fluentd** container by adding the following parameters to the **dssm-values.yaml** values file:

```
sentinel:
  fluentbit_sidecar:
    fluentd:
      host: '<fluentd hostname>'

db:
  fluentbit_sidecar:
    fluentd:
      host: '<fluentd hostname>'
```

*In this example, the Fluentd container is in the **cnf-gateway** Namespace:*

```
sentinel:
  fluentbit_sidecar:
    fluentd:
      host: 'f5-toda-fluentd.cnf-gateway.svc.cluster.local.'

db:
  fluentbit_sidecar:
    fluentd:
      host: 'f5-toda-fluentd.cnf-gateway.svc.cluster.local.'
```

4. Install the dSSM Pods:

***Important:** The string **f5-dssm** is the Helm release name. If a different release name is used, ensure the name is added to the privileged SCC.*

```
helm install f5-dssm <helm chart> -f <values>.yaml -n <namespace>
```

For example:

```
helm install f5-dssm tar/f5-dssm-0.22.12.tgz -f dssm-values.yaml -n cnf-gateway
```

5. All dSSM Pods will be available after the election process, which can take up to a minute.

***Important:** DB entries may fail to be created during the election process if TMM installs prior to completion. TMM will connect after the process completes.*

```
kubectl get pods -n cnf-gateway
```

*In this example, the dSSM Pods in the **cnf-gateway** Namespace have completed the election process, and the Pod **STATUS** is **Running**:*

```
NAME                              READY   STATUS
f5-dssm-db-0                      1/1     Running
f5-dssm-db-1                      1/1     Running
f5-dssm-db-2                      1/1     Running
f5-dssm-sentinel-0                1/1     Running
f5-dssm-sentinel-1                1/1     Running
f5-dssm-sentinel-2                1/1     Running
```

6. The dSSM DB Pods should be bound to the persistent volumes:

```
kubectl get pvc -n cnf-gateway
```

*In this example, the dSSM Pod's PVC **STATUS** is **Bound**:*

```
NAME                 STATUS   VOLUME
data-f5-dssm-db-0    Bound    pvc-c7060354-64d2-456b-9328-aa38f19b44b5
data-f5-dssm-db-1    Bound    pvc-8358b993-bf21-4fd7-a0fa-ee84ec420aac
data-f5-dssm-db-2    Bound    pvc-de65ed0f-f616-4021-a158-e0e78ed4539e
```

Next step

Continue to the BIG-IP Controller installation guide. To securely connect the TMM and dSSM Pods, add the following parameters to the Controller's Helm values file:

Important: *Set the `SESSIONDB_EXTERNAL_SERVICE` parameter to the Namespace of the dSSM Pod.*

```
tmm:

  sessiondb:
    useExternalStorage: "true"

  customEnvVars:
  - name: REDIS_CA_FILE
    value: "/etc/ssl/certs/dssm-ca.crt"
  - name: REDIS_AUTH_CERT
    value: "/etc/ssl/certs/dssm-cert.crt"
  - name: REDIS_AUTH_KEY
    value: "/etc/ssl/private/dssm-key.key"
  - name: SESSIONDB_EXTERNAL_STORAGE
    value: "true"
  - name: SESSIONDB_DISCOVERY_SENTINEL
    value: "true"
  - name: SESSIONDB_EXTERNAL_SERVICE
    value: "f5-dssm-sentinel.cnf-gateway"
```

Restarting

This procedure assumes that you have deployed the dSSM Pods, and have created a new set of Secrets to replace the existing Secrets. The new Secrets will not be used until the dSSM and TMM Pods have been restarted.

Important: *Restarting the TMM Pods impacts traffic processing.*

1. Obtain the name and number of TMM Pods:

   *In this example, the CNF Pods are in the **cnf-gateway** Namespace:*

```
kubectl get deploy -n cnf-gateway | grep tmm
```

*In this example, there are **3** TMM Pods in Namespace:*

```
kubectl get deploy -n cnf-gateway | grep f5-tmm
```

```
f5-tmm                3/3     3       3
```

2. Scale the number of TMM Pods to **0**:

```
kubectl scale deploy/f5-tmm --replicas=0 -n cnf-gateway
```

3. Wait 15 or 20 seconds for the TMM Pods to terminate, and scale the TMM Pods back to the previous number:

   *In this example the TMM Pods are scaled back to **3**:*

```
kubectl scale deploy/f5-tmm --replicas=3 -n cnf-gateway
```

4. Restart the dSSM Sentinel and DB Pods:

   The dSSM Sentinel and DB Pods run as StatefulSets, and will be restarted automatically.

```
kubectl delete pods -l 'app in (f5-dssm-db, f5-dssm-sentinel)' -n cnf-gateway
```

```
pod "f5-dssm-db-0" deleted
pod "f5-dssm-db-1" deleted
pod "f5-dssm-db-2" deleted
pod "f5-dssm-sentinel-0" deleted
pod "f5-dssm-sentinel-1" deleted
pod "f5-dssm-sentinel-2" deleted
```

5. Verify the dSSM Pods **STATUS** is **Running**:

```
kubectl get pods -n cnf-gateway
```

```
NAME                              READY   STATUS
f5-dssm-db-0                      2/2     Running
f5-dssm-db-1                      2/2     Running
f5-dssm-db-2                      2/2     Running
f5-dssm-sentinel-0                2/2     Running
f5-dssm-sentinel-1                2/2     Running
f5-dssm-sentinel-2                2/2     Running
```

6. The new Secrets should now be used to secure the dSSM channels.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- The list of commands used to create the Secrets.
- Redis
- Redis Sentinels
- StatefulSet Basics

BIG-IP Controller

Overview

The Cloud-Native Network Functions (CNFs) BIG-IP Controller, Edge Firewall, and Traffic Management Microkernel (TMM) Proxy Pods are the primary CNFs software components, and install together using Helm. Once integrated, Edge Firewall and the TMM Proxy Pods can be configured to process and protect high-performance 5G workloads using CNFs CRs.

This document guides you through creating the CNFs installation Helm values file, installing the Pods, and creating TMM's clientside (upstream) and serverside (downstream) F5BigNetVlan interfaces.

Requirements

Ensure you have:

- Installed the CNFs Software.
- Installed the CNFs Secrets.
- A Linux based workstation with Helm installed.

Procedures

Helm values

The CNFs Helm values file requires a number of custom parameter values to successfully integrate the CNFs software. Use the steps below to obtain important cluster configuration data, and configure the CNFs parameter values for a successful installation.

1. CNFs relies on Kubernetes Topology Manager to dynamically allocate and properly align TMM's CPU cores. Create a new Helm values file named **ingress-values.yaml** and add the `tmm.topologyManager` parameter:

   ```
   tmm:
     topologyManager: "true"
   ```

2. Robin ip-pools provide information required to discover and order TMM's SR-IOV network interface list. The interface numbers will be required later when configuring and installing the F5BigNetVlan Custom Resource (CR). Use the steps below to obtain the Robin ip-pool information:

   Note: *The BIG-IP Controller Namespace;* ***cnf-gateway****, was created during the CNFs Secrets installation.*

   A. To configure the `tmm.cniNetworks` parameter, obtain the **Names** of the clientside (upstream) and serverside (downstream) ip-pools:

   ```
   robin ip-pool list
   ```

   In this example, the clientside ip-pool Name is ***e801-180*** *and the serverside ip-pool Name is* ***e810-181****:*

   ```
   Name       | Driver | Network           | VLAN
   -----------+--------+-------------------+-------
   e810-180   | sriov  | 192.168.10.100/24 | -
   e810-181   | sriov  | 10.10.10.100/24   | -
   ```

   B. To configure the `tmm.customEnvVars` parameters, obtain the **NIC Tags** value for the clientside ip-pool:

```
robin ip-pool info e810-180 | grep NIC
```

*In this example, the **NIC Tags** value is **p1p1**:*

```
NIC Tags: [{'name': 'p1p1'}]
```

C. Obtain the **NIC Tags** value for the serverside ip-pool:

```
robin ip-pool info e810-181 | grep NIC
```

*In this example, the **NIC Tags** value is **p1p2**:*

```
NIC Tags: [{'name': 'p1p2'}]
```

D. Use the ip-pool **Name** and **NIC Tags** values to create TMM's interface list, using the BIG-IP Controller's Helm values. The interface maximum transmission unit (MTU) size can also be set here:

*In this example, TMM's clientside interface is **1.1**, and the serverside interface is **1.2**:*

```
cniNetworks: '[{"ippool": "e810-180", "mtu": 9000}, {"ippool": "e810-181", "mtu":
↪  9000}]'
  robinNetworks: "true"
  customEnvVars:
   - name: ROBIN_VFIO_RESOURCE_1
     value: "P1P1_VFIOPCI"
   - name: ROBIN_VFIO_RESOURCE_2
     value: "P1P2_VFIOPCI"
```

3. To use the Calico CNI, set the `TMM_CALICO_ROUTER` parameter. If the CNI relies on a router to perform proxy ARP, set the `TMM_IGNORE_GATEWAYS` parameter to ensure TMM does not configure a default gateway:

   ***Important:** Enabling `TMM_IGNORE_GATEWAYS` may cause cluster (Pod-to-Pod) traffic to fail. To set routes for specific cluster IPs, review **Cluster Traffic** in the F5BigNetStaticroute CR guide.*

```
tmm:
  customEnvVars:
   - name: TMM_CALICO_ROUTER
   value: "default"
   - name: TMM_IGNORE_GATEWAYS
   value: "TRUE"
```

4. To advertise routing information between networks, or to scale TMM beyond a single instance, the **f5-tmm-routing** container must be enabled, and a Border Gateway Protocol (BGP) session must be established with an external neighbor. The parameters below configure an external BGP peering session:

   ***Note:** For additional BGP configuration parameters, refer to the BGP Overview guide.*

```
dynamicRouting:
  enabled: true
  exportZebosLogs: true
  tmmRouting:
    image:
      repository: "local.registry.com"
    config:
      bgp:
        asn: 123
        neighbors:
        - ip: "192.168.10.200"
```

```
          asn: 456
          acceptsIPv4: true

  tmrouted:
    image:
      repository: "local.registry.com"
```

5. The Fluentd Logging collector is enabled by default, and requires setting the `f5-toda-logging.fluentd.host` parameter. If you installed Fluentd, ensure the `host` parameter targets the BIG-IP Controller Namespace, and replace the **afm**, **controller** and **f5-toda-logging** parameters as follows:

   Note: *In this example, the BIG-IP Controller, Edge Firewall and TMM Proxy Pods are installing to the* ***cnf-gateway*** *Namespace:*

```
afm:
  fluentbit_sidecar:
    fluentd:
      host: 'f5-toda-fluentd.cnf-gateway.svc.cluster.local.'
    image:
      repository: "local.registry.com"

controller:
  fluentbit_sidecar:
    fluentd:
      host: 'f5-toda-fluentd.cnf-gateway.svc.cluster.local.'
    image:
      repository: "local.registry.com"

f5-toda-logging:
  fluentd:
    host: "f5-toda-fluentd.cnf-gateway.svc.cluster.local."
  sidecar:
    image:
      repository: "local.registry.com"
```

6. By default, the Edge Firewall's default firewall mode **accepts** all network packets not matching an F5BigFwPolicy firewall rule. You can modify this behavior using the `defaultFirewallRule.action` parameter. For addition details about the default firewall mode and logging parameters, refer to the **Firewall mode** section of the F5BigFwPolicy overview:

```
afm:
  defaultFirewallRule:
    action: accept
    log: true
```

7. Be default, the TMM container uses the **default** Kubernetes `serviceAccount`. Use the parameter below to modify the `serviceAccount` used by TMM:

```
tmm:
  serviceAccount:
    name: tmm_sa
```

8. The completed Helm values file should appear similar to the following:

   Note: *Set the `image.repository` parameter for each container to your local container registry.*

```
tmm:

  image:
    repository: "local.registry.com"

  hugepages:
    enabled: true

  sessiondb:
    useExternalStorage: "true"

  topologyManager: true

  cniNetworks: '[{"ippool": "e810-180", "mtu": 9000}, {"ippool": "e810-181", "mtu":
   ↪  9000}]'
  robinNetworks: "true"
  customEnvVars:
   - name: ROBIN_VFIO_RESOURCE_1
     value: "P1P1_VFIOPCI"
   - name: ROBIN_VFIO_RESOURCE_2
     value: "P1P2_VFIOPCI"
   - name: TMM_IGNORE_GATEWAYS
     value: "TRUE"
   - name: TMM_CALICO_ROUTER
     value: "default"
   - name: REDIS_CA_FILE
     value: "/etc/ssl/certs/dssm-ca.crt"
   - name: REDIS_AUTH_CERT
     value: "/etc/ssl/certs/dssm-cert.crt"
   - name: REDIS_AUTH_KEY
     value: "/etc/ssl/private/dssm-key.key"
   - name: SESSIONDB_EXTERNAL_STORAGE
     value: "true"
   - name: SESSIONDB_DISCOVERY_SENTINEL
     value: "true"
   - name: SESSIONDB_EXTERNAL_SERVICE
     value: "f5-dssm-sentinel.cnf-gateway"

  dynamicRouting:
    enabled: true
    exportZebosLogs: true
    tmmRouting:
      image:
        repository: "local.registry.com"
      config:
        bgp:
          asn: 123
          neighbors:
          - ip: "192.168.10.200"
            asn: 456
            acceptsIPv4: true

    tmrouted:
      image:
        repository: "local.registry.com"
```

```
afm:
  enabled: true
  defaultFirewallRule:
    action: reject
    log: true
  pccd:
    enabled: true
    image:
      repository: "local.registry.com"
  fluentbit_sidecar:
    enabled: true
    image:
      repository: "local.registry.com"
    fluentd:
      host: 'f5-toda-fluentd.cnf-gateway.svc.cluster.local.'

ipsd:
  enabled: true
  image:
    repository: "local.registry.com"

controller:
  image:
    repository: "local.registry.com"
  fluentbit_sidecar:
    enabled: true
    image:
      repository: "local.registry.com"
    fluentd:
      host: 'f5-toda-fluentd.cnf-gateway.svc.cluster.local.'

f5-toda-logging:
  fluentd:
    host: "f5-toda-fluentd.cnf-gateway.svc.cluster.local."
  sidecar:
    image:
      repository: "local.registry.com"
  tmstats:
    config:
      image:
        repository: "local.registry.com"

debug:
  image:
    repository: "local.registry.com"
```

Installation

1. Change into the local directory with the SPK files, and list the files in the **tar** directory:

   *In this example, the CNF files are in the **cnfinstall** directory:*

```
cd cnfinstall
```

```
ls -1 tar
```

*In this example, the Helm chart for the BIG-IP Controller, Service Proxy TMM and Edge Firewall is named **f5ingress-6.0.13.tgz**:*

```
cnf-docker-images.tgz
f5-dssm-0.22.12.tgz
f5-toda-fluentd-1.8.30.tgz
f5ingress-6.0.13.tgz
```

2. Install the BIG-IP Controller, TMM Proxy and Edge Firewall Pods, referencing the Helm values file created in the previous procedure:

```
helm install f5ingress <helm chart> -f <values file> -n <namespace>
```

For example:

```
helm install f5ingress tar/f5ingress-6.0.13.tgz -f ingress-values.yaml -n cnf-gateway
```

3. Verify the Pods have installed successfully, and all containers are **Running**:

```
kubectl get pods -n cnf-gateway
```

*In this example, all containers have a **STATUS** of **Running** as expected:*

```
NAME                                    READY   STATUS
f5-afm-5bb5cd989b-b8qpf                 2/2     Running
f5-dssm-db-0                            1/1     Running
f5-dssm-db-1                            1/1     Running
f5-dssm-db-2                            1/1     Running
f5-dssm-sentinel-0                      1/1     Running
f5-dssm-sentinel-1                      1/1     Running
f5-dssm-sentinel-2                      1/1     Running
f5-ingress-f5ingress-7965947785-b8f5c   1/1     Running
f5-ipsd-74f5754b5d-kjzft                1/1     Running
f5-tmm-576df78f88-mpzbq                 4/4     Running
```

Interfaces

The F5BigNetVlan Custom Resource (CR) applies TMM's interface configuration; IP addresses, VLAN tags, MTU, etc. Use the steps below to configure and install clientside and serverside F5BigNetVlan CRs:

1. You can place both of the example CRs into a single YAML file:

   ***Important:** Set the `cmp_hash` parameter values to **SRC_ADDR** on the clientside (upstream) VLAN, and **DST_ADDR** on the serverside (downstream) VLAN.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetVlan
metadata:
  name: "subscriber-vlan"
  namespace: "cnf-gateway"
spec:
  name: clientside
```

```
  interfaces:
    - "1.1"
  selfip_v4s:
    - 10.10.10.100
    - 10.10.10.101
  prefixlen_v4: 24
  selfip_v6s:
    - 2002::10:10:10:100
    - 2002::10:10:10:101
  prefixlen_v6: 116
  mtu: 9000
  cmp_hash: DST_ADDR
---
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetVlan
metadata:
  name: "application-vlan"
  namespace: "cnf-gateway"
spec:
  name: serverside
  interfaces:
    - "1.2"
  selfip_v4s:
    - 192.168.10.100
    - 192.168.10.101
  prefixlen_v4: 24
  selfip_v6s:
    - 2002::192:168:10:100
    - 2002::192:168:10:101
  prefixlen_v6: 116
  mtu: 9000
  cmp_hash: SRC_ADDR
```

2. Install the VLAN CRs:

```
kubectl apply -f cnf_vlans.yaml
```

3. List the VLAN CRs:

```
kubectl get f5-big-net-vlan -n cnf-gateway
```

In this example, the VLAN CRs are installed:

```
NAME
vlan-client
vlan-server
```

4. If the Debug Sidecar is enabled (the default), you can verify the **f5-tmm** container's interface configuration:

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- ip a
```

The interfaces should appear at the bottom of the list:

```
8: clientside: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000
    inet 192.160.10.100/24 brd 192.168.10.0 scope global client
       valid_lft forever preferred_lft forever
    inet6 2002::192:168:10:100/112 scope global
```

```
       valid_lft forever preferred_lft forever
9: serverside: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000
    link/ether 1e:80:c1:e8:81:15 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.100/24 brd 10.10.10.0 scope global server
       valid_lft forever preferred_lft forever
    inet6 2002::10:10:10:100/112 scope global
       valid_lft forever preferred_lft forever
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Calico router

CNFs CRs

Overview

Cloud-native Network Functions (CNFs) Custom Resource Definitions (CRDs) extend the Kubernetes API; enabling AFM and TMM to be configured using CNFs Custom Resources (CRs). CNFs CRs configure AFM and TMM to support low-latency 5G application traffic, and apply networking configurations such as interface IP addresses and static routes.

This document describes the available CNFs CRs, and lists them in the order they should be configured and installed.

Protection and NAT

Protection and NAT CRs can be referenced by Traffic Management CRs to protect applications from unauthorized and malignant network traffic.

- F5BigDdosPolicy - Denial of Service (DoS/DDoS) event detection and mitigation.
- F5BigFwPolicy - Granular stateful-flow filtering based on access control list (ACL) policies.
- F5BigIpsPolicy - Intelligent packet inspection protects applications from malignant network traffic.
- F5BigNatPolicy - Carrier-grade NAT (CG-NAT) using large-scale NAT (LSN) pools.

Traffic management

Traffic management CRs configure TMM to provide secure application layer gateway services to remote subscribers.

- F5BigContextSecure - Full proxy TCP and UDP application layer gateway services.
- F5BigZeroratingirule - Part of Zero-Rating DNS solution; enabling subscribers to bypass rate limits.
- F5BigPePolicy - Intelligently control, steer, and optimize subscriber traffic.
- F5BigClassificationprofile - Enable deep packet inspection to analyze and categorize subscriber traffic.
- F5BigDnsApp - High-performance DNS resolution, caching, and DNS64 translations.
- F5BigAlgFtp - File Transfer Protocol (FTP) application layer gateway services.
- F5BigAlgTftp - Trivial File Transfer Protocol (TFTP) application layer gateway services.
- F5BigAlgPptp - Point-to-Point Tunneling Protocol (PPTP) application layer gateway services.
- F5BigAlgRtsp - Real Time Streaming Protocol (RTSP) application layer gateway services.

Networking CRs

Networking CRs configure TMM's networking components such as network interfaces and static routes.

Available network management CRs:

- F5BigNetVlan - TMM interface configuration: VLANs, Self IP addresses, MTU sizes, etc.
- F5BigCneSnatpool - Modify the source IP address of egress packets to TMM self IP address.
- F5BigNetStaticroute - TMM static routing table management.

Profiles and global settings

Profiles and global setting CRs can be reference by CNFs **Traffic Management** CRs to customize and enhance traffic processing.

- F5BigTcpSetting - TCP options to fine-tune how application traffic is managed.
- F5BigUdpSetting - UDP options to fine-tune how application traffic is managed.
- F5BigFastl4Setting - Fastl4 option to fine-tune how application traffic is managed.

Event logging

Event logging CRs can be referenced by Traffic Management CRs to log a wide variety of application traffic events to remote logging servers.

- F5BigLogProfile - Specifies subscriber connection information sent to remote logging servers.
- F5BigLogHslpub - Defines remote logging server endpoints for the F5BigLogProfile.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Kubernetes Custom Resources
- Kubernetes Service

F5BigDdosPolicy

Overview

The F5BigDdosPolicy Custom Resource (CR) configures the Traffic Management Microkernel (TMM) Proxy Pod to protect applications and the TMM Pod from Denial of Service / Distributed Denial of Service (Dos/DDoS) attacks. Using custom packet signatures, the F5BigDdosPolicy inspects all application traffic processed by the TMM Proxy Pod, to detect, report and/or mitigate DoS/DDoS events.

This document guides you through understanding, configuring and installing a simple F5BigDdosPolicy CR.

CR Parameters

The tables below describe the F5BigDdosPolicy CR parameters used in this document. For the full list of parameters, refer to the F5BigDdosPolicy Reference.

allowList

Parameter	Description
`sourceAddressList`	Specifies the F5BigCneAddresslist by `metadata.name` containing the source IP addresses to exclude from DDoS detection/mitigation.

vectors.floodVectors.commonConfigVectors

Parameter	Description
`vectorType`	Specifies the type of DoS Flood Vector to detect and mitigate: **ipv6-frag-flood**. Refer to F5BigDdosPolicy Reference for a full list.
`state`	Specifies the reponse for a vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`detectionThresholdEps`	Specifies the attack detection threshold in PPS for the configured attack type. Default value **4294967295**.
`detectionThresholdPercentage`	Specifies the attack detection percentage increase for the configured attack type. Default value **4294967295**.
`rateLimit`	Specifies the rate limit in PPS for the configured attack. Default value **4294967295**.

vectors.dnsFloodVectors.commonConfigVectors

Parameter	Description
`vectorType`	The type of DNS Flood Vector: **dns-aaaa-query**. Refer to F5BigDdosPolicy Reference for a full list.

Parameter	Description
`state`	Specifies the reponse for a vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`detectionThresholdEps`	Specifies the attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`rateLimit`	Specifies the rate limit in EPS for the configured attack. The default value is **4294967295**.

CR Example

```
apiVersion: "dos.k8s.f5net.com/v1"
kind: F5BigDdosPolicy
metadata:
  name: "cnf-dns-ddos"
  namespace: "cnf-gateway"
hslPublisher: "cnf-hsl-pub"
vectors:
  floodVectors:
    commonConfigVectors:
    - vectorType: "ipv6-frag-flood"
      state: "detection-only"
      detectionThresholdEps: 1111
      detectionThresholdPercentage: 11
      rateLimit: 10
  dnsFloodVectors:
    commonConfigVectors:
    - vectorType: "dns-aaaa-query"
      state: "detection-only"
      rateLimit: 111
      perDstIpLimitEps: 111111
allowList:
  sourceAddressList: "outbound-nat"
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigDdosPolicy CR shortName is **ddos**.

View CR instance:

```
kubectl get ddos -n <namespace>
```

View CR configuration:

```
kubectl get ddos -n <namespace> -o yaml
```

Address lists

The F5BigCneAddresslist CR defines lists of IP addresses that can be referenced by the F5BigDdosPolicy CR.

Example:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCneAddresslist
metadata:
  name: outbound-nat
spec:
  addresses:
  - "2002::192:168:10:1-2002::192:168:10:10"
  - "2002::10:10:10:0/112"
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

Use these steps to install the example F5BigDdosPolicy CR, and the ***optional*** CNFs CRs. Each step offers a brief description of the example CR.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** The example F5BigLogHslpub CR specifies a remote server with IP/port **[2002::10:30:2:220]:514** and the **udp** protocol. Copy and paste the example into a YAML file:

   Note: *The F5BigLogHslpub CR will be referenced by the F5BigDdosPolicy.*

   ```
   apiVersion: k8s.f5net.com/v1
   kind: F5BigLogHslpub
   metadata:
     name: "cnf-hsl-pub"
     namespace: "cnf-gateway"
   spec:
     pool:
     - name: "cnf-hslpool"
       endpoint:
       - "[2002::10:30:2:220]:514"
     syslog:
     - name: "syslog-dest"
       format: "rfc5424"
       protocol: "udp"
       pool: "cnf-hslpool"
   ```

2. Install the F5BigLogHslpub CR:

   ```
   kubectl apply -f cnf-hsl-cr.yaml
   ```

   *In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

   ```
   I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
   F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
   ```

3. The example F5BigCneAddresslist specifies a range **2002::192:168:10:1-2002::192:168:10:10**, and a subnet **2002::10:10:10:0/112** of IPv6 addresses. Copy and paste the example into a YAML file:

   Note: *The F5BigCneAddresslist CR will be referenced by the F5BigDdosPolicy CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCneAddresslist
metadata:
  name: "outbound-nat"
  namespace: "cnf-gateway"
spec:
  addresses:
  - "2002::192:168:10:1-2002::192:168:10:10"
  - "2002::10:10:10:0/112"
```

4. Install the F5BigCneAddresslist CR:

```
kubectl apply -f cnf-address-list.yaml
```

   *In this example, the BIG-IP Controller logs indicate the F5BigCneAddresslist CR was **added/updated**:*

```
I0208 12:00:00:12345   1 event.go:282] Event(v1.ObjectReference{Kind:"F5AddressList",
F5AddressListProfile cnf-gateway/outbound-nat was added/updated
```

5. The example F5BigDdosPolicy mitigates **ipv6-frag-flood** and **dns-aaaa-query** attacks, and excludes the F5BigCneAddresslist IPs from the policy. Copy and paste the example into a YAML file:

```
apiVersion: "dos.k8s.f5net.com/v1"
kind: F5BigDdosPolicy
metadata:
  name: "cnf-dns-ddos"
  namespace: "cnf-gateway"
hslPublisher: "cnf-hsl-pub"
vectors:
  floodVectors:
    commonConfigVectors:
    - vectorType: "ipv6-frag-flood"
      state: "detection-only"
      detectionThresholdEps: 1111
      detectionThresholdPercentage: 11
      rateLimit: 10
  dnsFloodVectors:
    commonConfigVectors:
    - vectorType: "dns-aaaa-query"
      state: "detection-only"
      rateLimit: 111
      perDstIpLimitEps: 111111
allowList:
  sourceAddressList: "outbound-nat"
```

6. Install the F5BigDdosPolicy CR:

```
kubectl apply -f cnf-ddos-cr.yaml -n cnf-gateway
```

   *In this example, the BIG-IP Controller logs indicate the F5BigDdosPolicy CR was **added/updated**:*

```
I0208 12:00:00:12345   1 event.go:282] Event(v1.ObjectReference{Kind:"F5Dos",
F5Dos cnf-gateway/cnf-dns-ddos was added/updated
```

7. The F5BigDdosPolicy will inspect all application traffic processed by the TMM Proxy Pod. Continue to the **Additional CRs** and **Dos/DDoS Statistics** sections.

Additional CRs

To begin Dos/DDoS detection and mitigation, install one of the **Traffic management** CNFs CRs.

Dos/DDoS Statistics

Use the steps below to verify the F5BigDdosPolicy CR DoS/DDoS statistics:

1. Connect to the debug sidecar:

```
kubectl exec -it deploy/f5-tmm -c debug -n <namespace> -- bash
```

*In this example, the debug sidecar is in the **cnf-gateway** Namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify the DoS/DDoS statistics:

```
tmctl -d /var/tmstat/blade dos_stat -s con-
 ↪  text_name,vector_name,profile_name,protocol_type,status,attack_detected,attack_count,drops
```

```
context_name vector_name profile_name              protocol_type status
------------ ----------- ------------------------- ------------- ------
Device       L3/4 BDoS                             L4 BDoS       Ready
Device       UDP flood   /Common/dos-device-config Device        Ready

attack_detected attack_count drops
--------------- ------------ -----
              0            0     0
              1            2  8523
```

```
context_name vector_name                    profile_name
------------ ------------------------------ -------------------------
Device       L3/4 BDoS
Device       DNS A Query                    /Common/dos-device-config
Device       IPv6 fragment flood            /Common/dos-device-config
Device       IPv6 too many extension headers /Common/dos-device-config
Device       DNS AAAA Query                 /Common/dos-device-config

protocol_type status attack_detected attack_count  drops
------------- ------ --------------- ------------ -----
L4 BDoS       Ready                0            0      0
Device        Ready                0            0      0
Device        Ready                3            4  18523
Device        Ready                0            0      0
Device        Ready                0            0      0
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigIpsPolicy

Overview

The F5BigIpsPolicy Custom Resource (CR) provides deep packet inspection, protecting applications from malignant network packet types and network traffic patterns. The F5BigIpsPolicy can be referenced by the F5BigContextSecure or F5BigDnsApp CRs to protect various types of 5G workloads.

This document guides you through understanding, configuring and installing a simple F5BigIpsPolicy.

CR parameters

The tables below describe the F5BigIpsPolicy CR parameters.

metadata

Parameter	Description
`name`	The name of the IPS policy. This value is referenced by the traffic management [CNF CRs].
`namespace`	The Kubernetes namespace the IPS policy will install to.

spec

Parameter	Description
`stagingPeriod`	Specifies the autopublishing suggestion period (in minutes). The default is **10080**.
`stagingConfidence`	Specifies the autopublishing suggestion confidence (percentage): **0** to **100**. The default is **0**.
`loggingGlobal`	Enables logging any of the configured compliances and signatures: **true** (default) or **false**.
`services`	Specifies a list of protocol services containing associated port numbers, compliance checks and signatures for the service.
`services.name`	Specifies the name of the service. Currently, only **dns** is available.
`services.ports`	Specifies the port for the service.
`services.compliances`	Specifics a list of compliance check for the service including its config value type and config value.
`services.compliances.name`	Specifies the name of the compliance check. For example, **dns_disallowed_resource_records**. For a full list of compliances, refer to F5BigIpsPolicy Compliance Checks.
`services.compliances.valueType`	Specifies the compliance check config value type: **int**, **vector-int**, **string**, **vector-string**, **boolean**, **enum**, or **vector-enum**.
`services.compliances.value`	Specifies the compliance check config value.
`services.compliances.action`	Specifies the compliance check config action: **accept** (default), **reject**, or **drop**.

Parameter	Description
`services.compliances.logging`	Enables logging a matching compliance inspection: **global** (default), **enabled**, or **disabled**.
`services.signatures`	Specifics a list of signatures for this service.
`services.signatures.name`	Specifies the name of the attack signature. For example, **dns_query_amplification_attempt**. For a full list of signatures, refer to F5BigIpsPolicy Attack Signatures.
`services.signatures.action`	Specifies the signature action: **accept** (default), **reject**, or **drop**.
`services.signatures.logging`	Enables logging a matching signature inspection: **global** (default), **enabled**, or **disabled**.

CR Example

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigIpsPolicy
metadata:
  name: "cnf-ips"
  namespace: "cnf-gateway"
spec:
  services:
    - name: dns
      ports:
        - "53"
      compliances:
        - name: dns_disallowed_query_type
          valueType: string
          value: SOA
          action: reject
      signatures:
        - name: dns_named_version_attempt
          action: reject
        - name: dns_os_solaris_exploit_sparc_overflow_attempt
          action: reject
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigIpsPolicy CR shortName is **ipspol**.

View CR instance:

```
kubectl get ipspol -n <namespace>
```

View CR configuration:

```
kubectl get ipspol -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller Pods.
- A Linux based workstation.

Installation

Use these steps to install the example F5BigIpsPolicy CR, and the ***optional*** CNFs CRs. Each step offers a brief description of the example CR.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** The example F5BigLogHslpub CR specifies a remote server with IP/port **[2002::10:30:2:220]:514**, and the **udp** protocol. Copy and paste the example into a YAML file:

   Note: *The F5BigLogHslpub CR will be referenced by the F5BigLogProfile.*

   ```
   apiVersion: k8s.f5net.com/v1
   kind: F5BigLogHslpub
   metadata:
     name: "cnf-hsl-pub"
     namespace: "cnf-gateway"
   spec:
     pool:
     - name: "cnf-hslpool"
       endpoint:
       - "[2002::10:30:2:220]:514"
     syslog:
     - name: "syslog-dest"
       format: "rfc5424"
       protocol: "udp"
       pool: "cnf-hslpool"
   ```

2. Install the F5BigLogHslpub CR:

   ```
   kubectl apply -f cnf-hsl-cr.yaml
   ```

   *In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

   ```
   I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
   F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
   ```

3. ***Optional:*** The example F5BigLogProfile CR specifies Protocol Inspection events to send to the remote log server. Copy and paste the example into a YAML file:

   Note: *The F5BigLogProfile CR will be referenced by the F5BigContextSecure CR.*

   ```
   apiVersion: "k8s.f5net.com/v1"
   kind: F5BigLogProfile
   metadata:
     name: "cnf-log-profile"
     namespace: "cnf-gateway"
   spec:
     name: "cnf-logs"
     publisher: "cnf-hsl-pub"
     protocolInspection:
   ```

```
    enabled: true
    publisher: "cnf-hsl-pub"
    logPacket: true
```

4. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-log-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

5. The example F5BigIpsPolicy CR rejects **SOA** record queries, and rejects **dns_named_version_attempt** and **dns_os_solaris_exploit_sparc_overflow_attempt** packet signatures. The F5BigIpsPolicy will log all configured `compliances` and `signatures`when the `logging` parameter is set to **global** (default). Copy and paste the CR into a YAML file:

***Note:** The F5BigIpsPolicy CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigIpsPolicy
metadata:
  name: "cnf-ips-policy"
  namespace: "cnf-gateway"
spec:
  services:
    - name: dns
      ports:
        - "53"
      compliances:
        - name: dns_disallowed_query_type
          valueType: string
          value: SOA
          action: reject
      signatures:
        - name: dns_named_version_attempt
          action: reject
        - name: dns_os_solaris_exploit_sparc_overflow_attempt
          action: reject
```

6. Install the F5BigIpsPolicy CR:

```
kubectl apply -f cnf-ips-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigIpsPolicy CR was **added/updated**:*

```
I0208 12:00:00.12345   1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5ProtocolInspectionProfile",
ProtocolInspectionProfile cnf-gateway/cnf-ips-policy was added/updated
```

7. The exaample F5BigContextSecure CR accepts packets destined to the **2002::200:200:200:0/112** subnet on the **subscriber-vlan** interface, and references the installed CRs. Copy and paste the CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
```

```
  name: "cnf-ips-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "2002::200:200:200:0/112"
   destinationPort: 53
   ipProtocol: "any"
   profile: "fastL4"
   protocolInspectionProfile: "cnf-ips-policy"
   logProfile: "cnf-log-profile"
   vlans:
     vlanList:
       - "subscriber-vlan"
```

8. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-ips-context.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-ips-context was added/updated
```

9. Review the **Additional CRs** and **IPS statistics** sections.

Additional CRs

The F5BigIpsPolicy can also be referenced by the F5BigDnsApp CR.

IPS statistics

Use the steps below to verify the F5BigIpsPolicy CR statistics:

***Important:** IPS statistics are not available until a compliance or signature packet match occurs.*

1. Connect to the debug sidecar:

```
tmctl -d blade protocol_inspection_stats
```

*In this example, the IPS policy show **7** compliance check matches:*

```
insp_id insp_name
------- --------------------------------

10007 dns_disallowed_resource_records

vs_name
-------------------------------------------
cnf-gateway-cnf-dns-ips-context-secure-SecureContext_vs

prof_name                                          hit_count last_hit_time
-------------------------------------------------  --------- -------------
cnf-gateway-cnf-dns-ips-profileprotocolinspection          7    1644624084
```

2. You can also view the TMM logs to verify packet matching:

```
kubectl logs -f f5-tmm-5576f687d5-bv2kx -c f5-tmm -n cnf-gateway | \
grep -i 'COMPL CHECK'
```

*In this example, each of the log messages indicates the compliance check indicates the **id** and **action**:*

```
IPS: ips_insp_callback/807: COMPL CHECK MATCH: id=10007, ctx='SOA', action=reject,
 ↪  support_id = 0001a91800002717
IPS: ips_insp_callback/807: COMPL CHECK MATCH: id=10007, ctx='SOA', action=reject,
 ↪  support_id = 0000525400002717
IPS: ips_insp_callback/807: COMPL CHECK MATCH: id=10007, ctx='SOA', action=reject,
 ↪  support_id = 0001673e00002717
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Kubernetes Service

F5BigIpsPolicy Compliance Checks

The F5BigIpsPolicy Custom Resource (CR) supports the following compliance checks:

- disallowed_query_type
- dns_experimental_resource_records
- dns_malformed_pdu
- dns_obsolete_resource_records
- dns_disallowed_resource_records
- dns_illegal_query_flags
- dns_maximum_reply_length
- dns_rdata_overflow
- dns_domains_blacklist
- dns_invalid_query_type
- dns_maximum_request_length
- dns_unknown_resource_record_type.

F5BigIpsPolicy Attack Signatures

The F5BigIpsPolicy Custom Resource (CR) supports the following attack signatures:

- dns_blacklist_dns_reverse_lookup_response_for_known_malware_domain_spheral_ru_win_trojan_glupteba
- dns_nginx_dns_resolver_dos
- dns_dns_query_amplification_attempt
- dns_malformed_dns_query_with_http_content
- dns_named_authors_attempt
- dns_named_authors_attempt_1
- dns_named_version_attempt
- dns_named_version_attempt_1
- dns_os_linux_os_linux_x86_linux_overflow_attempt
- dns_os_linux_os_linux_x86_linux_overflow_attempt_1
- dns_os_linux_os_linux_x86_linux_overflow_attempt_admv2
- dns_os_other_os_other_x86_freebsd_overflow_attempt
- dns_os_solaris_exploit_sparc_overflow_attempt
- powerdns_authoritative_server_denial_of_service
- dns_server_other_bind_buffer_overflow_named_tsig_overflow_attempt
- dns_server_other_bind_buffer_overflow_named_tsig_overflow_attempt_1
- dns_server_other_bind_buffer_overflow_via_nxt_records
- dns_server_other_bind_buffer_overflow_via_nxt_records_named_overflow_adm
- dns_server_other_bind_buffer_overflow_via_nxt_records_named_overflow_admrocks
- dns_server_other_bind_named_overflow_attempt
- dns_spoof_query_response_ptr_with_ttl_of_1_min_and_no_authority
- dns_spoof_query_response_with_ttl_of_1_min_and_no_authority
- dns_tcp_inverse_query

F5BigClassificationprofile

Overview

The F5BigClassificationprofile Custom Resource (CR) integrates with the Intrusion Detection and Mitigation (IPS) Pod to intelligently catagorize 5G application traffic. The F5BigClassificationprofile is required when the F5BigPePolicy CR is configured to identify application traffic using the **classification** filter. The F5BigClassificationprofile and F5BigPePolicy CRs are referenced by the F5BigContextSecure CR to accept and process 5G application traffic.

This document guides you through configuring and installing a simple F5BigClassificationprofile CR.

CR parameters

metadata

Parameter	Description
`name`	The name of the Classification policy. This value is referenced by the F5BigContextSecure CR.
`namespace`	The Kubernetes namespace the Classification policy will install to.

spec

Parameter	Description
`name`	Specifies a name for the Classification Profile. This value is ***not*** used as a reference by other CRs.
`description`	Specifies descriptive text that identifies the Profile.
`enableApplicationDetection`	Enables application detection: **true** (default) or **false**.
`enableLogUnclassifiedDomain`	Enables logging unclassified domains: **true** or **false** (default).
`enableUrlCategorization`	Enables URL categorization: **true** or **false** (default).
`preset.allowReclassification`	Enables transactional flows: **true** (default) or **false**.
`preset.enableFlowBundling`	Enables the correlation of flows or transactions lacking attributes for classification decision to an abstract key with associated classification tokens: **true** (default) or **false**.
`preset.enableCacheResults`	Enables caching classification results for the L4 destination. When the same traffic passes through, the result is taken from the cache: **true** (default) or **false**.
`preset.analyzeDns`	Enables the classification engine to inspect DNS responses and use the IP addresses returned, to enhance classification results for otherwise-unknown data-plane traffic: **true** (default) or **false**. When false, data-plane traffic for which no existing signatures exist gets generic classification.
`preset.analyzeSslServerside`	Enables the classification engine to process SSL Server Side Hello to inspect ALPN (primarily for HTTP2 / SPDY subclassification). If false the DNS traffic gets generic classification: **true** (default) or **false**.
`logPublisher`	Specifies the F5BigLogHslpub CR to log classification events using the `metadata.name` parameter.

Parameter	Description
`enableIruleEvent`	Enables iRule Events triggered by application in this classification settings: **true** (default) or **false**.

CR Example

F5BigClassificationprofile

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigClassificationprofile
metadata:
  name: "cnf-class-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-class-profile"
  enableApplicationDetection: true
  enableLogUnclassifiedDomain: true
  logPublisher: "cnf-hsl-pub"
  preset:
    enableFlowBundling: true
    analyzeSslServerside: false
    enableCacheResults: true
    analyzeDns: true
```

CR shortName

CR shortName provide an easy way to view installed CRs, and their configuration parameter. The CR shortName can also be used to delete the CR instance. The F5BigClassificationprofile CR shortName is **dpiprof**.

View CR instance:

```
kubectl get dpiprof -n <namespace>
```

View CR configuration:

```
kubectl get dpiprof -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

Use these steps to install the example F5BigClassificationprofile CR, and the ***optional*** CNFs CRs. Each step offers a brief description of the example CR.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** The example F5BigLogHslpub CR specifies a remote server with IP/port **[2002::10:30:2:220]:514**, and the **udp** protocol. Copy and paste the example into a YAML file:

   The F5BigLogHslpub CR will be referenced by the F5BigClassificationprofile.

```
apiVersion: k8s.f5net.com/v1
kind: F5BigLogHslpub
metadata:
  name: "cnf-hsl-pub"
  namespace: "cnf-gateway"
spec:
  pool:
  - name: "hsl-pool"
    endpoint:
    - "[2002::10:30:2:220]:514"
  syslog:
  - name: "cnf-syslog"
    format: "rfc5424"
    protocol: "udp"
    pool: "hsl-pool"
```

2. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

   *In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

3. The example F5BigClassificationprofile enables important parameters such as **ApplicationDetection**, **UrlCategorization**, and **analyzeDns**. Copy the example CR into a YAML file:

   The F5BigClassificationprofile will be referenced by the F5BigContextSecure CR.

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigClassificationprofile
metadata:
  name: "cnf-class-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-class-profile"
  enableApplicationDetection: true
  enableUrlCategorization: true
  logPublisher: "cnf-hsl-pub"
  preset:
    enableFlowBundling: true
    analyzeSslServerside: false
    enableCacheResults: true
    analyzeDns: true
```

4. Install the F5BigClassificationprofile CR:

```
kubectl apply -f cnf-class-profile.yaml
```

   *In this example, the BIG-IP Controller logs indicate the F5BigClassificationprofile CR was **added/updated**:*

```
I0624 12:00:00.12347   1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5ClassificationProfile",
F5ClassificationProfile cnf-gateway/cnf-class-profile was added/updated
```

5. Continue to the F5BigPePolicy guide to reference the Classification profile, and begin managing subscriber traffic using policy based classification criteria.

Classification statistics

If the [TMM Debug] sidecar is enabled (default), use the steps below to verify F5BigClassificationprofile statistics.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

   ```
   kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
   ```

2. Verify the F5BigClassificationprofile statistics:

   ```
   tmctl -d blade gpa_classification_stats
   ```

   ```
   result count cec flbl srdb custom bytes_in bytes_out pkts_in pkts_out
   ------ ----- --- ---- ---- ------ -------- --------- ------- --------
   205        0   0    0    0      0        0         0       0        0
   216        0   0    0    0      0        0         0       0        0
   205.67    60  60    0    0      0    36879    328096     542      482
   ```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigPePolicy

Overview

The F5BigPePolicy Custom Resource (CR) configures the Traffic Management Microkernel (TMM) Proxy Pod to manage 5G subscriber traffic using policy based classification criteria such as application type, URL category, or higher level packet flow information. Subscriber traffic can be allowed or denied, and optimized with the F5BigTcpSetting CR based on the F5BigPePolicy rule configuration. A F5BigPeProfile CR is required and referenced by the F5BigContextSecure CR to accept and process 5G application traffic using one or more F5BigPePolicy CRs.

This document guides you through understanding, configuring and installing the F5BigPePolicy and F5BigPeProfile CRs.

F5BigPePolicy CR parameters

The table below describes the F5BigPePolicy CR parameters used in this document. For the full list of parameters, refer to the F5BigPePolicy Reference.

spec.rule

Parameter	Description
`name`	Specifies the name of the F5BigPePolicy rule. A policy can contain multiple rules.
`precendence`	Specifies the precedence, or order for processing rules and actions: **1** to **1024**.

spec.rule.filter.classification

Important: *A F5BigClassificationprofile is required when using the `classification` feature.*

Parameter	Description
`name`	Specifies the name of the classification.
`match`	Specifies a traffic matching criteria: **match** (default), or **no-match**.
`category`	Specifies the type of traffic: **any** (default), **Web**, **Audio_Video**, **Encrypted**, **File_Download_Servers**, **Search_Engines**, **Network_Management_and_Services**, **News_and_Media**, and **Advertisements**.
`application`	Specifies the application type: **any** (default), **amazon**, **apple**, **tcp**, **udp**, **http**, **ssl**, **youtube**, **google**, **ftp**, **cnn**, and **amazon_adv**.

spec.rule.filter.flow

Parameter	Description
`match`	Specifies a traffic matching criteria: **match** (default), or **no-match**.

Parameter	Description
`protocol`	Specifies the protocol type for matching subscriber traffic: **any** (default), **tcp**, or **udp**.
`ipType`	Specifies the IP protocol type for matching subscriber traffic: **any** (default), **ipv4**, or **ipv6**.
`destinationAddress`	Specifies the destination IP address for matching subscriber traffic. The default is **0.0.0.0/0**.
`destinationPort`	Specifies the destination port for matching subscriber traffic. The default is **0**.
`sourceAddress`	Specifies the source IP address for matching subscriber traffic. The default is **0.0.0.0/0**.

spec.rule.action

Parameter	Description
`gate`	Specifies whether to allow (enabled) or deny (disabled) subscriber traffic that matches a PE rule: **Enabled** (default) or **Disabled**.
`tcpOptimizationUplink`	Specifies the F5BigTcpSetting CR applied to the uplink traffic that matches the rule.
`tcpOptimizationDownlink`	Specifies the F5BigTcpSetting CR applied to the downlink traffic that matches the rule.

F5BigPeProfile CR parameters

The table below describes the F5BigPeProfile CR `spec` parameters used in this document.

Parameter	Description
`description`	A description of the F5BigPeProfile CR.
`globalPolicy.highPrecendence`	Specifies a list of F5BigPePolicy CRs to apply as high precedence on subscriber traffic.
`globalPolicy.lowPrecendence`	Specifies a list of F5BigPePolicy CRs to apply as low precedence on subscriber traffic.
`unknownSubscriberpolicy`	Specifies a list of F5BigPePolicy CRs to apply to unknown subscriber traffic.

CR Examples

F5BigPePolicy

```
apiVersion: k8s.f5net.com/v1
kind: F5BigPePolicy
metadata:
  name: "cnf-pe-policy"
  namespace: "cnf-gateway"
```

```
spec:
  rule:
    - name: "stream-rule-1"
      reportingProfile: "cnf-log-profile"
      publisher: "cnf-hsl-pub"
      precedence: 1
      filter:
        classification:
          - application: amazon
            category: Audio_Video
            match: match
      action:
        gate: Enabled
        tcpOptimizationDownlink: tcp-high-bw-profile
        tcpOptimizationUplink: tcp-high-bw-profile
    - name: "stream-rule-2"
      precedence: 2
      filter:
        flow:
          - match: "match"
            protocol: "any"
            ipType: "any"
            sourceAddress: "2002::10:10:10:10/112"
            destinationAddress: "2002::192:168:100:0/112"
            destinationPort: 443
      action:
        gate: "Disabled:
```

F5BigPeProfile

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigPeProfile
metadata:
  name: "cnf-pe-profile"
  namespace: "cnf-gateway"
spec:
  description: "web profile"
  globalPolicy:
    highPrecedence:
      - "cnf-pe-policy"
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigPePolicy and F5BigPeProfile CR shortNames are **pepol** and **peprof** respectively.

View CR instance:

```
kubectl get pepol -n <namespace>
kubectl get peprof -n <namespace>
```

View CR configuration:

```
kubectl get pepol -n <namespace> -o yaml
kubectl get peprof -n <namespace> -o yaml
```

PE Logging

A F5BigLogProfile CR can be configured to log subscriber connection information such as subscriber ID, call duration, destination IP address and port, etc., when the connection matches an F5BigPePolicy CR `rule`. This section demonstrates two methods for logging subscriber connection data. ***Note:*** *The F5BigPePolicy references both a F5BigLogProfile CR, and a F5BigLogHslpub CR separately.*

Reporting Fields

The `reportingFields` parameter provides options that are easily viewed while configuring the F5BigLogProfile, however, the logging format on the remote server is more difficult to view.

In this example, the F5BigLogProfile is configured to capture and log the packet's source and destination information.

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
name: "cnf-log-profile"
namespace: "cnf-gateway"
spec:
pe:
reportingFields:
- "Source IP"
- "Source Port"
- "Destination IP"
- "Destination Port"
reportingType: flow-reporting
```

The `reportingFields` log entires will appear similar to the following on the remote server:

```
Apr  4 15:48:36 f5-tmm-5896659874-bzfc8 tmm[33] 10.20.2.220,57236,10.30.2.220,80
Apr  4 15:48:59 f5-tmm-5896659874-bzfc8 tmm[33] 10.20.2.220,59720,10.30.2.220,53
```

Format Script

The `formatScript` parameter provides options that are more difficult to view while configuring the F5BigLogProfile, however, the logging format on the remote server is easier to view.

In this example, the F5BigLogProfile is configured to capture and log the packet's source and destination information.

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  pe:
```

```
    formatScript: return (src-ip:[PEM::flow stats reported src-ip],src-port:[PEM::flow
     ↪  stats reported src-port],dst-ip:[PEM::flow stats reported
     ↪  dst-ip],dst-port:[PEM::flow stats reported dst-port],timestamp:[PEM::flow stats
     ↪  reported timestamp])
    reportingType: flow-reporting
```

The `formatScript` log entires will appear similar to the following on the remote server:

```
Apr  4 15:52:42 f5-tmm-5896659874-bzfc8 tmm[33]
 ↪  (src-ip:10.20.2.220,src-port:57240,dst-ip:10.30.2.220,dst-port:80)
Apr  4 15:52:43 f5-tmm-5896659874-bzfc8 tmm[33]
 ↪  (src-ip:10.20.2.220,src-port:59934,dst-ip:10.30.2.220,dst-port:53)
```

For a full list of parameter options, refer to the **spec.pe** section of the F5BigLogProfile Reference.

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

Use these steps to install the example F5BigPePolicy CR, and the ***optional*** CNFs CRs. Each step offers a brief description of the example CR.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** The example F5BigLogHslpub CR specifies a remote server with IP/port **[2002::10:30:2:220]:514**, and the **udp** protocol. Copy and paste the example into a YAML file:

   Note: *The F5BigLogHslpub CR will be referenced by both the F5BigLogProfile and the F5BigClassificationprofile CRs.*

   ```
   apiVersion: k8s.f5net.com/v1
   kind: F5BigLogHslpub
   metadata:
     name: "cnf-hsl-pub"
     namespace: "cnf-gateway"
   spec:
     pool:
     - name: "hsl-pool"
       endpoint:
       - "[2002::10:30:2:220]:514"
     syslog:
     - name: "cnf-syslog"
       format: "rfc5424"
       protocol: "udp"
       pool: "hsl-pool"
   ```

2. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

3. ***Optional:*** The example F5BigLogProfile CR specifies policy enforcement events to such as **Source IP** and **Destination IP**, and sends them to the remove logging server. Copy and paste the example into a YAML file:

   Note: *The F5BigLogProfile CR will be referenced by the F5BigPePolicy CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  pe:
    reportingFields:
      - "Source IP"
      - "Source Port"
      - "Destination IP"
      - "Destination Port"
    reportingType: flow-reporting
```

4. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-log-profile.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

5. The example F5BigTcpSetting CR increases the **sendBuffer** and **proxyBuffer** sizes to increase TCP throughput. Copy the example CR into a YAML file:

   Note: *The F5BigTcpSetting CR will be referenced by the F5BigPeProfile.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigTcpSetting
metadata:
  name: "tcp-high-bw-profile"
  namespace: "cnf-gateway"
spec:
  sendBufferSize: 150000
  receiveWindowSize: 70000
  proxyBufferHigh: 20000
  proxyBufferLow: 5000
  idleTimeout: 150
  resetOnTimeout: false
```

6. Install the F5BigTcpSetting CRs:

```
kubectl apply -f cnf-tcp-high-bw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigTcpSetting CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5TcpSetting",
TcpSetting cnf-gateway/tcp-high-bw-profile was added/updated
```

7. The example F5BigClassificationprofile enables important parameters such as **ApplicationDetection**, **UrlCategorization**, and **analyzeDns**. Copy the example CR into a YAML file:

   Note: *The F5BigClassificationprofile will be referenced by the F5BigContextSecure CR, and references the F5BigLogHslpub CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigClassificationprofile
metadata:
  name: "cnf-class-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-class-profile"
  enableApplicationDetection: true
  enableUrlCategorization: true
  logPublisher: "cnf-hsl-pub"
  preset:
    enableFlowBundling: true
    analyzeSslServerside: false
    enableCacheResults: true
    analyzeDns: true
```

8. Install the F5BigClassificationprofile CR:

```
kubectl apply -f cnf-class-profile.yaml
```

   *In this example, the BIG-IP Controller logs indicate the F5BigClassificationprofile CR was **added/updated**:*

```
I0624 12:00:00.12347   1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5ClassificationProfile",
F5ClassificationProfile cnf-gateway/cnf-class-profile was added/updated
```

9. The example F5BigPePolicy applies the higher bandwidth F5BigTcpSetting CR to **Audio_Video** application traffic, and **Disables** (denies) application traffic from source IP subnet **2002::10:10:10:10/112**, destined to the IP subnet **2002::192:168:100:0/112**. Copy the example CR into a YAML file:

   Note: *The F5BigPePolicy references the F5BigLogProfile, F5BigLogHslpub, and F5BigTcpSetting CRs.*

```
apiVersion: k8s.f5net.com/v1
kind: F5BigPePolicy
metadata:
  name: "cnf-pe-policy"
  namespace: "cnf-gateway"
spec:
  rule:
    - name: "stream-rule-1"
      reportingProfile: "cnf-log-profile"
      publisher: "cnf-hsl-pub"
      precedence: 1
      filter:
        classification:
          - application: amazon
            category: Audio_Video
            match: match
```

```
      action:
        gate: Enabled
        tcpOptimizationDownlink: tcp-high-bw-profile
        tcpOptimizationUplink: tcp-high-bw-profile
    - name: "stream-rule-2"
      precedence: 2
      filter:
        flow:
          - match: "match"
            protocol: "any"
            ipType: "any"
            sourceAddress: "2002::10:10:10:10/112"
            destinationAddress: "2002::192:168:100:0/112"
            destinationPort: 443
      action:
        gate: "Disabled"
```

10. Install the F5BigPePolicy CR:

```
kubectl apply -f cnf-pe-policy-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigPePolicy CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5PemPolicy",
PemPolicy cnf-gateway/cnf-hsl-pub was added/updated
```

11. Copy the example F5BigPeProfile CR into a YAML file:

***Note:** The F5BigPeProfile references the F5BigPePolicy, and will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigPeProfile
metadata:
  name: "cnf-pe-profile"
  namespace: "cnf-gateway"
spec:
  description: "web profile"
  globalPolicy:
    highPrecedence:
      - "cnf-pe-policy"
```

12. Install the F5BigPeProfile CR:

```
kubectl apply -f cnf-pe-profile-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigPeProfile CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5PemProfile",
PemProfile cnf-gateway/cnf-hsl-pub was added/updated
```

13. The example F5BigContextSecure CR listens for connections destined to the **any** IPv6 subnet on port **443** received on the **subscriber-vlan** interface. Copy and paste the example into a YAML file:

***Note:** The F5BigContextSecure CR references the F5BigPeProfile and F5BigClassificationprofile CRs.*

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
```

```
  name: "cnf-pe-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "::/0"
   destinationPort: 443
   ipProtocol: "any"
   profile: "fastL4"
   pemProfile: "cnf-pe-profile"
   classificationProfile: "cnf-class-profile"
   vlans:
     vlanList:
       - "subscriber-vlan"
```

14. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-context-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-pe-context was added/updated
```

15. Review the **Policy Enforcement statistics** section.

Policy Enforcement statistics

If the [TMM Debug] sidecar is enabled (default), use the steps below to verify F5BigPePolicy match statistics.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

   ```
   kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
   ```

2. Verify the F5BigPePolicy statistics:

   ```
   tmctl -d blade pem_actions_stat -s pass,drop,tcpopt_to_net,tcpopt_to_sub
   ```

   In this example, optimization is applied to both uplink (tcpopt_to_net) and downlink (tcpopt_to_sub) traffic.

```
pass drop tcpopt_to_net tcpopt_to_sub
---- ---- ------------- -------------
  6    0             6             6
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigPePolicy classification applications

The F5BigPePolicy Custom Resource (CR) supports the following classification applications:

- any
- fetlife
- tuberel
- xvideos
- youporn
- xhamster
- bootytape
- hardsextube
- bongacams
- porn555
- cam4
- redtube
- keezmovies
- bigupload
- txxx
- xvideoslive
- xnxx
- dmm_jp
- pornhub
- livejasmin
- camzap
- captora
- moatads
- sekindo
- supersonic
- segment
- google_ads
- doubleclick_ads
- criteo
- triplelift
- shorte_st
- facebook_ads
- localytics
- mailchimp
- pitchengine
- appnexus
- converto
- onclickads
- addthis
- scorecardsearch
- tvb_ads
- socialvibe
- aol_ads
- umeng
- demandbase
- adexc
- upsight

- onclkds
- tidaltv
- popads
- common_adv
- intercom
- adcolony
- adultadworld
- outbrain_ads
- amazon_adv
- 51_la
- scorecardresearch
- wistia
- popcash
- yahoo_adv
- uservoice
- adfly
- taboola_ads
- truelife_ads
- hubspot
- oracle_ads
- clickrouter
- marketo
- telerik
- yodlee
- xero_upload
- acronis_snap_deploy
- socialtext
- microsoft_socl
- windowslive
- yahoo_maps
- arab_truecaller
- cedexis
- ldap
- alpha_anywhere
- daap
- winamp_remote
- flashplugin_update
- google_skymap
- wasalny
- windows_azure
- baidu_maps
- gfan
- clicktools
- yahoo_smallbusiness
- bime
- callidus
- clearslide
- corba
- appguru
- samsung
- siftscience

- rsync
- solarwinds
- google_installer
- apple_spotloght
- microsoft
- gdbremote
- pusher
- adobe_update
- ntrglobal
- autodesk
- arcgis_online
- arcgis
- esri
- apple_vpp
- alfresco_management
- alfresco
- autodesk_a360
- microsoft_groups
- hockeyapp
- daydream
- hanaro_speedtest
- mlab
- rbb_speedtest
- lifesize
- microsoft_flow
- ibm
- apple_location_services
- google_cache
- meta4
- microsoft_dynamic_crm
- g_suite
- google_support
- google_maps
- google_update
- dell_download
- firefox_update
- samsung_update
- microsoft_product_activation
- paloalto_update
- agiloft
- office365_update
- wtp
- wtls
- wsp
- vmware
- ms_iis
- apache_tomcat
- apache
- auth0
- proofhq
- livechat

- wufoo
- evernote
- hp_update
- samsung_apps
- google_play
- sogou_map
- baidu_ime
- evernote_upload
- say_hi
- signnow
- yandex_navigator
- g2crowd
- daummypeople_uploads
- prezi
- google_apis
- dell
- 360_yunpan
- workday
- vmware_vsphere_web_client
- toshiba
- uptodown
- justdevelopit
- exchange_unified_messaging
- blackberry_app_world
- google_cloud
- apple_siri
- ibm_connections
- hulft
- tstore
- fandango
- msn_apps
- iec_60870_5_104
- qlik
- qlik_sense_cloud
- hitachi
- gooddata
- python
- android
- android_pay
- tesla
- syslog
- google_compression_proxy
- puffin_browser
- adobe_logs
- qq_download
- adobe_connect
- iqfeed
- openair
- facebook_code
- wakoopa
- dopool

- walkme
- google_location_service
- google_appengine
- here
- nend
- gravatar
- python_easy_install
- new_relic
- taleo
- schneider_electric
- apc
- jobvite
- mqtt
- liveperson
- kiwoom
- bitmoji
- windows_media_player
- trueservice
- truecloud
- acrobat
- splunk
- synology
- techsmith
- cddb
- unity3d
- nianticlabs
- adobe_creative_cloud
- http2
- dynamics_365
- soasta
- activesync
- acronis
- manage_engine
- adselfservice_plus
- salesforce
- adobe_meeting_rc
- softether
- mozilla
- naver
- onlinedown
- avira_update
- kaspersky_update
- pooq_downloads
- windows_marketplace
- conduit
- ios_ota_update
- apns
- zoho
- office365
- pathview
- appneta

- pagerduty
- ca_technologies
- nginx
- folder_plus
- esignlive
- oracle_e-business_suite
- burner
- medellia
- sumologic_upload
- amqp
- informatica
- freshbooks
- informatica_cloud
- infront
- gplex_pc
- netsuite
- soap
- google_cloud_print
- azure_media_serives
- azure_rms
- google_cloud_shell
- microsoft_webdav
- appstore
- dymamics_365
- cvs
- tivoconnect
- webaim
- source_forge
- alisoft
- jenkins
- apt-get
- yahoo_developer
- iso_8583
- landex
- reimageplus
- shopify
- google_analytics
- gstatic
- ytimg
- union_procedure_call
- skycn
- vmware_blast
- fc2
- google_chrome
- cups
- appland
- rightscale
- waze
- miop
- constant_contact_upload
- constant_contact

- autodesk_a360_upload
- endnote
- avaya
- canon
- canon_bnjp
- bacnet
- apache_solr
- adobe
- zendesk
- vmware_horizon_view
- f5_bigip
- google_developers
- amazon_kindle
- divx
- peoplesoft
- usejump
- zoho_crm
- facebook_developers
- csdn
- aliyun
- adobe_analytics
- spdy
- mapquest
- speedtest
- security_server_user_agent
- splunk_cloud
- samanage
- servicenow
- centurylink
- viglink
- progress
- symantec_installation_manager
- cip_ethernet_ip
- lookout
- daummypeople_downloads
- azure_media_services
- canon_bjnp
- cloudsigma
- evalesco_sysorb_web_client
- source_forge_download
- mcafee_update
- clarizen
- yum
- lindenlab
- sosta
- sohu
- windows_appstore
- apple_photo_stream
- pubnub
- rackspace
- xrea

- evalesco_sysorb
- wordfast
- sumologic
- logentries
- ubuntu
- landesk
- collectd
- oracle_cloud
- redhat
- apple_spotlight
- zillow
- google_spaces
- amazon_aws_console
- remind
- go_speed
- filehippo
- adobe_portfolio
- gao_de_maps
- peerguardian
- cloudshare
- engineyard
- grab
- trello
- citrix
- trendmicro_safesync
- schmedley
- secure_access
- avg_update
- panda_update
- office_store
- jaspersoft
- xbox_marketplace
- greenhouse
- siemens
- solarwinds_update
- simplehelp_update
- simplehelp
- teamup_upload
- teamup
- truecaller
- dynatrace
- dynatrace_app_monitoring
- better_biz_bur
- ipp
- dnf
- http_file_flash
- x1_com
- microsoft_powerapps
- softsonic
- google_gen
- google_vr

- hewlett_packard
- weebly
- billdesk
- fsecure_update
- lpd
- chrome_webstore
- appshopper
- android_update
- avast_update
- opera_mini
- appcelerator
- apple_maps
- sfax_upload
- sfax_download
- square
- qlik_view
- sfax
- saasu
- successfactors
- oliveoffice
- scribol
- xero
- jquery
- distcc
- sideroak
- nateon_out
- fsecure
- drupal
- msn
- jwplayer_cdn
- adp
- trello_upload
- skytap
- google_weblight
- pbworks
- opera_update
- nokia_ovi
- ammyy_admin
- ning
- themeforest
- apache_server
- mpquest
- return_stun
- office365_remote_analyser
- microsoft_graph
- silent_circle
- webrtc
- bitstrips
- oracle
- itunes_mediadownload
- vagrant

- vagrant_download
- hashicorp
- webalo
- torch_browser
- folding_at_home
- srvloc
- track_it
- heroix_longitude
- microsoft_cryptoapi
- java_update
- windows_update
- apple_update
- chrome_update
- amazon_aws
- iminent
- http2_ng
- giop
- benchbee
- tachy
- freecast
- slacker
- aim_audio
- melon_audio
- iheartradio
- baidu_audio
- amazon_audible
- paltalk_audio
- zello
- scanner_radio
- audio
- streamon
- mymusic
- abacast
- teachertube_audio
- realplayer
- we7
- vtuner
- streamaudio
- wo_streaming
- bugs_music_audio_stream
- shoutcast
- uberconference_call
- aol_radio
- xm_radio
- windowsmedia
- winamp
- podcasts
- http_file_audio
- cfox
- icecast
- archive_audio

- iheartradio_streaming
- soundcloud_streaming
- soundcloud_upload
- slacker_streaming
- soundcloud
- tunein
- youtube_hd
- icall
- xanga
- hipchat_video
- aim_video
- youtube
- sccp
- rtsp
- rtp
- rtcp
- gomtv_vod
- ali_wangwang_audio_video
- tvants
- sagetv_regular
- sapo_video
- jjang_live
- mpegts
- rdt
- ppstream
- itunes
- ctv
- mediasetspain_streaming
- abc_australia_view
- mogulus
- tstore_vod
- rtsp_data
- nate_hoppin_mediadownload
- srtcp
- srtp
- h225
- h245
- wedisk_streaming
- wowza
- shutterstock
- yi_home_camera
- naver_streaming
- tagoo
- sip
- youtube_video
- 51_com_video
- ppfilm
- abcnews_video
- google_duo
- pandora_uploads
- daum_potplayer

- rtmp
- rtmpe
- asf
- rtsp_tunnelled
- gracenote
- plex_tv
- youtube_safety_mode
- youtube_upload
- nba_leaguepass
- revver
- quicktime
- nonico
- msn_video
- sony_online_games
- facetime
- mms
- mgcp
- naver_blog
- egloos
- blogster
- google_blog
- livedoor
- livejournal
- tistory
- tumblr
- over_blog
- blogspot
- blogger
- skyrock
- kakao_blog
- blogdetik
- uber
- realtor
- tvdirect
- uber_eats
- bloomberg_professional
- 11st_kr
- google_finance
- esignal
- ali_wangwang_file_transfer
- trulia
- ifeng_finance
- google_ventures
- bigadda
- dontstayin
- slickdeals
- stylelife_jp
- bellmedia
- yelp
- pricerunner
- interpark

- ali_wangwang
- macys
- amazon_flex
- seosprint
- gmarket
- alimama
- intuit
- ikea
- groupon
- thomsonreuters
- bambusero
- flipkart
- mbtrade
- foursquare
- tvdirect_tv
- offerup
- dmm
- mi
- tgfone
- bookmyshow
- moneycontrol
- businessweekly
- pttrm
- rakuten
- cpall
- forbes
- letgo
- businessweek
- priceminister
- monex
- aliexpress
- wish
- amazon
- amazon_smile
- bloomberg
- tmall
- walmart
- homedepot
- booking
- alibaba
- bestbuy
- target
- etsy
- kakaocorp
- costco
- colmex
- meettheboss
- ebay
- yahoo_realestate
- snapdeal
- tokopedia

- alipay
- 1688com
- yahoo_finance
- momoshop
- hotline
- google_capital
- baofeng
- lyft
- cdiscount
- dangdang
- fortinet_webfilter
- ms_eop
- bitdefender_update
- onelogin
- pingidentity
- beyondtrust
- truste
- tenebril
- brightcloud
- verisign
- verisign_cert_check
- giganews
- sophos
- 360_update
- your-freedom
- websense
- kaspersky
- nod32
- microsoft_cert_check
- digicert_cert_check
- entrust_cert_check
- entrust
- identrust_cert_check
- identrust
- globalsign_cert_check
- symantec_endpoint_manager
- duo_security
- microsoft_auth_service
- ms_spynet
- 360_download
- 360_securutyapps
- ‘360’
- 360_totalsecurity
- radiusim
- netwitness
- webroot
- tenable
- tenable_cloud
- tenable_nessus
- okta
- bitdefender

- virustotal_api_upload
- virustotal_upload
- virustotal
- sophos_update
- 360_securityapps
- cloudmark_desktop
- cloudmark
- razor2
- trendmicro_update
- nod32_update
- globalsign
- panda
- fortinet
- dcc
- trendmicro
- mcafee_epo_admin
- digicet
- symantec_av_update
- symantec
- mcafee
- nshc
- forticlient_update
- hinet
- 1and1
- biglobe
- dynamicintranet
- acer
- kakaku
- chromebook
- fon
- apple
- lgdacom
- huawei
- chumby
- akamai
- keycdn
- highwinds
- limelight_networks
- demonbucket
- amazon_cloudfront
- bootstrapcdn
- cloudflare
- kaltura
- coralcdn_user
- maxcdn
- edgecast
- turner_cdn
- sapo_stratus
- fastly
- mongodb
- maxdb

- memcached
- marklogic_server
- quickbase
- gds_db
- postgres
- tds
- filemaker_server
- drda
- firebase
- filemaker_pro
- zoho_db
- mysql
- redis
- tns
- db2
- cubrid
- apache_cassandra
- couchbase_server
- marklogic
- filemaker_update
- plaxo
- fubar
- match
- faceparty
- gays
- meetic
- pof
- lifeknot
- adultfriendfinder
- zoosk
- outeverywhere
- spark_dating
- babylon
- college_blender
- lynda
- livemocha
- quizlet
- howstuffworks
- magister
- khanacademy
- mit_edu
- sciencestage
- eduzones
- clickview_video
- blackboard
- reverso
- canvaslms
- clickview
- instructure
- sakai
- pluralsight

- trueplookpanya
- quizlet_audio
- wikispaces
- google_classroom
- ted
- wikidot
- isakmp
- tor
- quic
- https
- ssl
- ssh
- packetix_vpn
- ipsec
- ign
- garena
- deviant_art
- digitalverse
- google_arts_and_culture
- jibjab
- timewarner
- hbo
- smilebox
- 9gag
- mangahere
- ck101
- life_tw
- kapanlagi
- bomb01
- smilebox_audio
- smilebox_video
- starz
- starz_streaming
- tvb
- natecyworld
- reverbnation
- nbc
- windowsmedia_center
- poco
- bigbrother
- hbo_video
- 91com
- giphy
- vice
- diodeo
- ifunny
- ozock
- truevisions_streaming
- truevisions
- truelife_streaming
- truelife

- truecorp
- cbc
- disney
- turner
- wetpaint
- teepr
- ifunny_video
- abs_cbn
- gfycat
- nhn
- artisto
- disney_channel
- disney_junior
- wittyfeed
- pullbbang
- pullbbang_video
- theladbible
- bet365
- disneyxd
- cartoonnetwork
- facebook_apps
- livestream_fbapp
- facebook_chat
- facebook_messeneger_chat
- facebook_messenger_chat
- facebook_chat_typing
- facebook_messenger_typing
- facebook_posting
- facebook_post_delete
- facebook_events
- playbuggle_fbapp
- jellysplash_fbapp
- mesmogames_fbapp
- watersplash_fbapp
- thunderrun_fbapp
- klondikegame_fbapp
- thetrainstation_fbapp
- cp_backgammon_new_fbapp
- monsterlink_fbapp
- celebritycrossword_fbapp
- farmvilleharvestswap_fbapp
- kelimelik_fbapp
- candyvalley_fbapp
- playfamilyguy_fbapp
- belotecom_fbapp
- shakethesky_fbapp
- bingodrive_fbapp
- goldenfrontier_fbapp
- landleven_fbapp
- pet_city_fbapp
- mypotfarm_fbapp

- yahtzzy_fbapp
- seaportgame_fbapp
- ligaultras_fbapp
- godgamemj_fbapp
- jackpotjoyslots_fbapp
- piratetreasures_fbapp
- jigsawworld_fbapp
- lost_bubble_fbapp
- fruit_land_fbapp
- blackdiamondcasino_fbapp
- playoceanquest_fbapp
- thetribez_fbapp
- kitchenscramble_fbapp
- monster_world_fbapp
- rockncash_fbapp
- forgeofempires_fbapp
- charmfarm_fbapp
- slots_farm_fbapp
- ocean_wars_fbapp
- bingolane_fbapp
- akamonslots_fbapp
- dutyofhero_fbapp
- kingsroadgame_fbapp
- mmhafbog_fbapp
- bubbles_iq_fbapp
- hotshotcasino_fbapp
- scatterslots_fbapp
- playpudding_fbapp
- diggysadventure_fbapp
- slotspharaohsway_fbapp
- crystalisland_fbapp
- magicpuzzles_fbapp
- fairwayblast_fbapp
- kingdomsofcamelot_fbapp
- castle_age_fbapp
- ghosttales_fbapp
- gardensoftime_fbapp
- tropicstorm_fbapp
- la_cosa_nostra_fbapp
- chumbacasino_fbapp
- taongafarm_fbapp
- wonkaslots_fbapp
- slotica_slots_fbapp
- facebook_games
- sugarwitch_fbapp
- twtexas_fbapp
- infinityslots_fbapp
- solitairelive_fbapp
- fazenda_feliz_fbapp
- familyfeudtwo_fbapp
- islandexperiment_fbapp

- solitairesafari_fbapp
- farmgame_tw_fbapp
- wordox_the_game_fbapp
- playfarm_fbapp
- royal_de_fbapp
- royalstory_fr_fbapp
- playhousehold_fbapp
- wizardofozslots_fbapp
- riddlestone_fbapp
- letsvegas_fbapp
- playpigdragon_fbapp
- frozenfreefall_fbapp
- fruitplanet_fbapp
- omgfortune_fbapp
- bubbleisland_fbapp
- stormfall_fbapp
- lostjewels_fbapp
- solitairecastle_fbapp
- tavlaplus_fbapp
- playyoworld_fbapp
- soldiersinc_fbapp
- spartawarofempires_fbapp
- totaldominationgame_fbapp
- parchisplayspace_fbapp
- dorfleben_fbapp
- spinitrich_fbapp
- goldfishcasinoslots_fbapp
- happyacres_fbapp
- cookingtale_fbapp
- super_city_game_fbapp
- twdoudizhu_fbapp
- luckyslotsgame_fbapp
- knightsbrides_fbapp
- crazy_cake_swap_fbapp
- gametower_fbapp
- avataria_fbapp
- canakyuzbirokey_fbapp
- vivaslots_fbapp
- vegas_downtown_slots_fbapp
- magerealm_fbapp
- canakokey_fbapp
- bingo_showdown_fbapp
- farmtown_fbapp
- pirates_game_fbapp
- under_control_fbapp
- startrekgame_fbapp
- leagueofangels_fbapp
- takefiveslots_fbapp
- fishworld_fbapp
- wartune_fbapp
- coral_isle_fbapp

- monsterhexa_fbapp
- playpengle_fbapp
- bestfiends_fbapp
- papapear_fbapp
- mirrorballslots_fbapp
- alphabettysaga_fbapp
- happyfishbowl_fbapp
- com_octro_rummy_fbapp
- realtexasholdem_fbapp
- jurassicparkbuilder_fbapp
- best_casino_fbapp
- villagelifegame_fbapp
- oldvegasslots_fbapp
- bingooo_fbapp
- tltexas_fbapp
- tastytale_fbapp
- doubleubingo_fbapp
- solitaireblitz_fbapp
- panic_room_fbapp
- wordswithfriends_fbapp
- cookiejam_fbapp
- criminalcase_fbapp
- petrescuesaga_fbapp
- triviacrack_fbapp
- candycrush_fbapp
- candycrushsoda_fbapp
- farmheroes_fbapp
- livepool_fbapp
- solitairethreearena_fbapp
- liveholdem_fbapp
- slingoadventure_fbapp
- csihiddencrimes_fbapp
- socialempires_fbapp
- zebramahjong_fbapp
- dreamlandstory_fbapp
- clock_maker_fbapp
- warcommander_fbapp
- heroesofglorybattle_fbapp
- yuzbirplus_fbapp
- panda_pop_fbapp
- doubledowncasino_fbapp
- agar_io_fbapp
- topeleven_fbapp
- slotomania_fbapp
- farmville_two_fbapp
- texas_holdem_fbapp
- dragoncity_fbapp
- dessertshop_fbapp
- praiabingo_fbapp
- liberatorsgame_fbapp
- bigfishcasino_fbapp

- avengersalliance_fbapp
- goldenmanager_fbapp
- citygirllife_fbapp
- guitarflash_fbapp
- safariescape_fbapp
- socialwarsgame_fbapp
- oyun_okey_fbapp
- casinostar_fbapp
- marketland_fbapp
- blood_strike_fbapp
- jackpotpartycasino_fbapp
- caesars_fbapp
- playforestrescue_fbapp
- pool_live_fbapp
- diamonddash_fbapp
- diamonddiggersaga_fbapp
- teenpattigold_fbapp
- pearls_peril_fbapp
- ck_islands_fbapp
- com_octro_teenpatti_fbapp
- bejeweledblitz_fbapp
- clash_of_kings_fbapp
- heart_of_vegas_fbapp
- bingoblitz_fbapp
- juicejam_fbapp
- soccerstars_miniclip_fbapp
- monsterlegends_fbapp
- toyblast_fbapp
- monsterbusters_fbapp
- mahjongtrails_fbapp
- poker_wsop_fbapp
- basketball_stars_fbapp
- houseoffun_fbapp
- doubleucasino_fbapp
- bookoflifegame_fbapp
- dragonsofatlantis_fbapp
- yahtzeewithbuddies_fbapp
- hititrich_fbapp
- okeyplus_fbapp
- vegas_app_fbapp
- angrybirds_fbapp
- mobwars_fbapp
- thepets_fbapp
- doubleluck_fbapp
- playairport_fbapp
- bakeryblitz_fbapp
- newrockcity_fbapp
- gamethrones_fbapp
- bingobyryzing_fbapp
- clickfuncasino_fbapp
- cross_stitch_world_fbapp

- tetris_battle_fbapp
- bubblewitch_fbapp
- onlinesoccermanager_fbapp
- pyramidsolitairesaga_fbapp
- bubblecoco_fbapp
- playmyvegas_fbapp
- txpoker_fbapp
- legend_es_fbapp
- is_cool_fbapp
- poker_italia_fbapp
- mistresses_fbapp
- turn_poker_fbapp
- lets_fish_fbapp
- the_secret_society_fbapp
- sunshinebay_fbapp
- myfarmthegame_fbapp
- flowershopfun_fbapp
- spadesplus_fbapp
- livescrabble_fbapp
- bingoisland_fbapp
- playtpirslots_fbapp
- dreamlike_mix_fbapp
- doutorbingo_fbapp
- ginrummyplus_fbapp
- nordsheroes_fbapp
- dragon_blood_fbapp
- astrogarden_fbapp
- songpop_fbapp
- dummyrummy_th_fbapp
- quickhitslots_fbapp
- cafeland_fbapp
- pnixgamesbowlingking_fbapp
- word_finder_fbapp
- thronerush_fbapp
- dicewithbuddies_fbapp
- candycrushjelly_fbapp
- scramblewf_fbapp
- angry_words_fbapp
- ztest_ck_fbapp
- belote_multijoueur_fbapp
- dragonsriseofberk_fbapp
- solitairearena_fbapp
- solitaireinwonder_fbapp
- solitairetales_fbapp
- slagalica_fbapp
- solitaireatlantis_fbapp
- megapolis_game_fbapp
- family_farm_fbapp
- ea_scrabble_closed_fbapp
- geniesandgems_fbapp
- onthefarm_fbapp

- charmking_fbapp
- highfivecasino_fbapp
- happyfarm_ar_fbapp
- rumble_app_fbapp
- scrubbydubbysaga_fbapp
- zooworldclassic_fbapp
- battlepirates_fbapp
- thekingoftowers_fbapp
- ilovecalcio_fbapp
- buggle_two_fbapp
- pepperpanicsaga_fbapp
- cuteanimal_fbapp
- playroyalstory_fbapp
- mutants_gg_fbapp
- facebook_mail
- fashionguide
- cosmopolitan
- aufeminin
- flying_file
- 500images
- carbonite
- xunlei
- nomadesk_upload
- nomadesk_download
- nomadesk
- fufox
- yosemite_backup
- docs_com
- olleh_ucloud
- extmatrix
- daum_cloud
- mumble
- hipchat_upload
- aim_transfer
- issuu_upload
- hightail_downloading
- huddle_upload
- http_file_vdi
- amazon_cloud_drive
- smartfile_korea
- paltalk_transfer
- ymsg_transfer
- hightail_uploading
- onedrive_share
- icloud_sync
- icloud_photos
- icloud_drive
- adobe_connet_file_share
- onedrive_upload
- onedrive_download
- webhard

- wedisk
- ulozto
- totodisk
- transferbigfiles
- ziddu
- uploadcare
- onehub
- korean_network_storage
- leapfile
- slack_file_transfer
- microsoft_bits
- zoho_share
- icloud
- aimini
- mypcbackup
- clip2net_upload
- justcloud
- druva
- efolder
- datto
- bacula
- ifolder
- zippyshare
- mediafire
- imageshack
- hotfile
- fluxiom
- gigaup
- bitshare
- microsoft_download
- gigatribe
- smugmug
- shoebox
- onedrive
- cubby
- data_bin
- demonsaw
- asus_webstorage_upload
- idrive_upload
- jjangfile
- file_io
- netload
- smugmug_upload
- egnyte_sync
- file_dropper
- esnips
- uploaded
- crocko
- tftp
- google_storage
- netfolder

- amazon_cloud_drive_upload
- aerofs
- adrive_upload
- acrobat_upload
- adobe_creative_cloud_uploading
- uploadmirrors
- acronis_download
- screencast_download
- sharebox
- ftps
- ibackup
- okurin
- multiupload
- evault
- qq_file_transfer
- dropbox_sync
- dropbox_download
- google_cloud_storage_upload
- filesanywhere
- google_cloud_storage
- goodsync
- gmail_drive
- asus_webstorage
- arcserve
- iscsi
- taku_file_bin
- spanning
- rayfile
- filesovermiles
- storix
- cloudme
- sugarsync
- aspera
- aol_filetransfer
- dropsend
- droplr
- dochub_upload
- dochub
- dropbox_upload
- filehost
- nortononline_backup
- pruna
- smb
- box_uploading
- ndmp
- depositfiles
- divshare
- diino
- clip2net
- dropbox
- mozy

- yandex_disk
- rapidshare
- sendspace
- fileflyer
- zshare
- megaupload
- afp
- naver_ndrive
- crashplan
- adobe_connect_file_share
- watchdox
- weiyun
- teamdrive
- trueshare
- uc_yun
- uploadmirrios
- solodfiles_upload
- syncplicity
- flying_file_transfer
- bigfile
- mailbigfile
- yourfilehost
- savefrom
- idrive
- adrive
- box_net
- egnyte
- filemail
- google_drive
- google_photos
- jungledisk
- solidfiles_upload
- uploading_com
- zbigz
- yahoo_box
- wiredrive
- wetransfer
- weiyun_file_transfer
- qnext
- torrentz
- openload_co_upload
- openload_co
- directdownloadlinks
- uploading
- http_file_executable
- clubnex
- fileserve
- hightail
- screencast_upload
- screencast
- http_file_archive

- http_file_data
- http_file_image
- accellion
- kiteworks
- 4sync
- 4sync_file_transfer
- 4shared_file_transfer
- screencast_streaming
- fileload
- onehub_upload
- quatrix
- maytech
- netfile
- quatrix_upload
- quatrix_download
- revconnect
- webdav
- nfs
- badongo
- mashare
- unitrends
- backblaze
- file_post
- app_statement_2
- sharefile
- promptfile
- signiant
- sharevault
- send_anywhere_file_transfer
- send_anywhere
- sendthisfile_file_upload
- sendthisfile
- solidfiles
- slideshare_upload
- idisk
- freakshare
- netbios
- zippyshare_download
- zippyshare_upload
- dl_free
- backupify
- filepost
- clubbox
- xbox_one_games
- jabber_transfer
- 4shared
- 2shared
- filecatalyst
- flashget
- sosbackup
- megashares

- tappin
- live_storage
- ftp_data
- ftp
- mega
- tcloud
- join_me_file_transfer
- motleyfool
- compass
- bolt
- aastocks
- sbi_bank
- chase
- yandex_money
- discover
- fidelity
- paytm
- wooribank
- factset
- mymarkets
- robinhood_stock_trading
- paypal
- icicibank
- kbstar
- apple_stocks
- ragingbull
- americanexpress
- google_wallet
- wellsfargo
- citigroup
- hdfcbank
- tradestation
- bankofamerica
- baac
- stockstar
- google_payments_center
- metatrader
- capitalone
- fix
- steam_download
- zhengtugame
- wolfteam
- cabal
- wow
- xboxlive
- ladbrokes
- ladbrokes_poker
- empire_four_kingdoms
- torch_browser_games
- pokemon_duel
- zynga

- lineage2
- everquest
- cellufun
- left4dead2
- zynga_cv
- destiny
- hirezstudios
- ageofconan
- we_dancing_online
- secondlife
- ruliweb
- teamfortress
- hangame
- fifa_mobile_soccer
- hyves_games
- itsmy
- deer_hunter_2014
- valve_source_engine
- unrealtournament3
- gameloft
- puzzle_and_dragon
- akinator
- farmville
- quake
- golf_clash
- playdemic
- baidu_games
- 51_com_games
- nexon
- xfire
- aeriagames
- planetside2
- point_blank
- unreal
- discordapp
- wolfenstein
- winamax
- gamer_tw
- eagames
- guildwars
- guildwars2
- arenanet
- tales_runner
- dota2
- dc_universe_online
- ogplanet
- stick_cricket
- drakensang_online
- neverwinter
- hearthstone
- path_of_exile

- gamepedia
- entropia
- gamesfly
- party_poker
- callofduty
- evony
- apple_game_center
- netmarble
- gamesmomo
- blizzard
- ubisoft_uplay
- ubisoft
- heros_of_the_storm
- realmofmad_god
- pokemongo
- poker_stars_update
- fruit_ninja
- talking_tom
- playstation
- slither_io
- nintendo
- nintendo_wfc
- psn
- armed_heros
- daybreak
- mobage
- minecraft
- maplestory
- war_rock
- league_of_legends
- origin
- roblox
- angeldust
- rift
- trionworlds
- clubpenguin
- dofus
- zynga_poker
- zynga_wwf
- lunplay
- cstrike
- dragonnest
- clawbert
- dancing_line
- goodgames
- battlefield
- dena
- terraria
- wsop
- aion
- coc_ta

- cartoonnetwork_games
- origin_downloads
- firefall
- globalagenda
- hero_online
- granadoespada
- battle_net
- chicken_scream
- crashofcars
- armorgames
- crossfire
- demonware
- bomberclone
- roblox_update
- zynga_games
- mailru_games
- rolling_sky
- nexongames
- msn_games
- miniclip_games
- xfire_games
- lotro
- final_fantasy_xiv
- xbox_games
- combatarms
- notdoppler
- powerrangers_legacy_wars
- ballz
- ketchaap
- addictinggames
- agame
- asheronscall
- poker_stars
- all_slots_casino
- xbox
- yahoo_games
- runescape
- pogo
- fifa13
- fifa12
- playonline
- popkart
- puzzlepirates
- ourgame
- seven_knights
- needforspeed
- halflife
- ntt_games
- knight_online
- max_payne3
- mocospace

- gree
- steam
- subspace
- easports
- star_craft_2
- epicgames
- ultima
- swtor
- swtor_download
- spiral_knights
- ragnarok
- streetfighter
- sanguo_mobile
- diablo_3
- pokemon
- gamestop
- little_fighter
- konaminet
- spirit_guardain
- super_mario_run
- king
- subway_surfers
- rovio
- regnum
- ea_games
- game_trailers
- dingit_tv
- baidu_hi_games
- dingit_tv_video
- candycrushsaga
- popcap_games
- plants_vs_zombies_2
- pogo_games
- twitch
- wargaming_net
- eve_online
- plants_vs_zombies
- quizup
- com2us
- paradise_paintball
- onlive
- i_gamer
- babycenter
- webmd_video
- xywy
- msn_health_and_fitness
- vgo
- apple_health
- hl7
- fitbit
- nih_gov

- walgreens
- webmd
- intacct
- insightly_crm
- highrise_crm
- batchbook
- onepagecrm
- sap_jam
- sap
- oracle_crm_ondemand
- siebel_crm
- modbus
- dnp3
- bugzilla
- base_crm
- l2tp
- trunk_1
- tlsp
- sscopmce
- tcf
- sun_nd
- sprite_rpc
- snp
- srp
- sps
- ddp
- dccp
- ddx
- rsvp
- mobility_header
- mpls_in_ip
- nsfnet_igp
- trunk_2
- vmtp
- wb_expak
- visa
- wb_mon
- wsn
- sm
- smp
- gre
- vrrp
- ospf_igp
- echo
- dcn_meas
- crudp
- emcon
- dgp
- dfs
- fibre_channel
- stp

- st
- fire
- ggp
- gmtp
- hmp
- iatp
- ifmp
- il
- i_nlsp
- ipcomp
- vines
- pipe
- activenet
- 3pc
- aris
- argus
- host
- cpnx
- crtp
- br_sat_mon
- cbt
- bbn_rcc_mon
- bna
- compaq_peer
- cphb
- cftp
- chaos
- sctp
- igmp
- qnx
- pvp
- pup
- ptp
- prm
- private_enc
- pnni
- nvp_ii
- sat_mon
- scps
- secure_vmtp
- skip
- rvd
- sat_expak
- ipv6_opts
- ipv6_route
- ipv6_frag
- ipv6_nonxt
- xtp
- iso_tp4
- esp
- tp_plus_plus

- rdp
- igp
- ipip
- egp
- pgm
- udplite
- uti
- eigrp
- iplt
- ipcv
- ipx_in_ip
- ippc
- iso_ip
- irtp
- kryptolan
- leaf_1
- larp
- sdrp
- lan
- idpr
- idpr_cmtp
- idrp
- ipv4_isis
- xnet
- xns_idp
- dsr
- hopopt
- mtp
- mux
- mfe_nsp
- mobile
- leaf_2
- merit_inp
- netblt
- narp
- discard
- rsvp_e2e_ignore
- shim6
- hip
- rohc
- wesp
- ax25
- encap
- any_0hop
- micp
- scc_sp
- iptm
- ah
- swipe
- manet
- etherip

- mycat
- sony
- soft4fun
- lg
- gadugadu
- gmail_chat
- aim_express
- ymsg_webmessenger
- ircs
- irc
- jabber
- oovoo
- weibo
- yugma
- skype_login
- oist
- sina_uc
- viber_chat
- hipchat_chat
- hipchat
- freepp
- tinychat
- qq_chat
- facebook_messenger_file_transfer
- viber_public_chats
- whatsapp_desktop
- airaim
- omegle_chat
- omegle
- lifesize_chat
- msnmobile
- ibm_connections_chat
- iloveim
- icq2go
- msn_shell
- misleee
- baidu_hi_file_transfer
- baidu_hi_audio_video
- minus
- paltalk_chat
- kakaotalk_voice
- wechat_media_message
- wechat_message
- wechat_voice
- koolim
- comm
- teamspeak_v3
- 050plus
- line
- baidu_yun
- daum_mypeople

- heywire
- daum_mypeople_chat
- nateon_pc2pc
- ali_wangwang_chat
- heytell
- ymsg_conf
- ymsg
- paltalk
- skype_c2c
- netmeeting_ils
- messengerfx
- mibbit
- nimbuzz_web
- skype_update
- messageme
- chaton
- skype_photo
- skype_video_msg
- skype_p2p
- skype_data
- skype_out
- hookt
- tuenti
- hangouts_chats
- lync_online
- mplus_messenger
- bobsled
- imo
- hike
- misslee
- popo_im
- gtalk
- chikka
- apple_imessage
- clicksandwhistles
- line_transfer
- softros_messenger
- webqq
- camfrog
- mightytext
- zoho_im
- yuuguu
- campfire
- whatsapp_file_transfer
- go_sms_pro
- apple_bonjour
- cgi_irc
- yoono
- baidu_hi
- tt_talk
- oicq

- whatsapp
- hangouts
- path
- qq
- whatsapp_chat
- purplevrs
- viber
- amazon_chime_chat
- aim
- zalo
- skype
- nateon
- pinger
- sapo_messenger
- facebook_messenger
- fetion
- viber_file_transfer
- zas_communicator
- gcm
- telegram
- ip_messenger
- irc_dcc
- google_allo
- boldchat
- aims
- omegle_typing
- vivox
- ebuddy
- voxer
- mibbit_chat
- trillian
- cryptocat
- kik
- laiwang
- softros_messenger_file_transfer
- icq
- icq_chat
- icq_filetransfer
- kakaotalk
- mitalk
- wechat
- chatango
- telstra
- verizonwireless
- verizon
- truemove
- nbtc_th
- i_kool
- cattelecom
- google_fiber
- comcast

- tencent
- talkatone
- line_call
- jajah
- vippie
- viber_call
- tribair_call_data
- tribair
- touch_to_talk
- justalk_voice
- appear_in
- vsee_call
- hangouts_voice
- facebook_voice
- whatsapp_voice
- teamspeak
- talk360
- libon
- daum_mypeople_voice
- alicall
- slack_voice
- airtel_talk
- qq_call
- justalk
- zalo_call
- line2
- foocall
- whos_calling
- vsee
- aol_voip
- magicjack
- messagenet
- alizila
- teltel
- line_out
- voxox
- net2phone
- whozcalling
- wirofon
- rebtel
- pronto_dialer
- nymgo
- iax
- ventrilo
- vbuzzer
- voxofon
- talkbox
- wicall
- monster
- glassdoor
- indeed

- headhunter
- 104_tw
- naukri
- weather_desktop
- msn_weather
- hk_observatory
- accuweather
- weatherbug
- google_weather_plugin
- weather
- gaiaonline
- msn_money
- google_groups
- tapatalk
- avito_ru
- microsoft_people
- yahoo_groups
- craigslist
- bitauto
- zhihu
- tianya
- dcinside
- kaskus
- live_groups
- vbulletin
- msn_groups
- yahoo_answers
- howard_forums
- ravelry
- imesh
- groove_music
- torch_browser_music
- kuwo
- shazam
- 8tracks
- hulkshare
- buzznet
- gogoyoko
- apple_music
- spotify
- grooveshark
- imeem
- qrocity
- jango
- yandex_music
- lastfm
- rapgenius
- eyny
- milkmusic
- google_play_music
- spotify_streaming

- napster
- amazon_prime_music
- musica
- rdio
- kugou
- musical_ly
- bugs_music
- xbox_music
- musicovery
- deezer
- soundhound
- musical_ly_video
- allmusic
- kkbox
- mog
- naver_streaming_music
- musicbox
- amie_street
- mtv_music
- deezer_streaming
- jango_streaming
- rhapsody
- amazon_mp3
- wlccp
- upnp
- krb5
- ali_wangwang_remote_control
- mdns
- rusers
- rstat
- nagios_core
- rlp
- sunrpc
- jumpdesktop
- x11
- radius
- pcp
- nat_pmp
- bootp
- hsrp
- xot
- pim
- portmapper
- time
- stun
- tcp
- telnet
- msdp
- zabbix
- whois
- http

- munin_web_interface
- coap
- munin
- udp
- tibcordv
- arubanetworks
- ripng
- netop
- nagios
- diameter
- ssdp
- snmp
- rmcp
- rwho
- dcerpc
- cotp
- dhcp
- chargen
- aruba_papi
- ident
- ipv6_icmp
- icmp
- mygreenpc
- netbios_ns
- iperf
- dhcpv6
- wins
- ncp
- netop_on_demand
- netop_remote_control
- ntrglobal_remote
- cisco
- barracuda_networks
- netbios_ss
- netbios_dg
- daytime
- f5
- sflow
- tacacs_plus
- dns
- ocsp
- netflow
- ntp
- nntps
- nntp
- scep
- ip6
- ip
- lwapp
- llmnr
- meraki

- nbns
- wccp
- mount
- h248_text
- epmd
- cnn_video
- kbs_live
- bbc
- ifeng
- nrk
- businessinsider
- cbs_news
- elpais
- news_au
- aol
- conservativetribune
- mingpao
- ctv_news
- sevenwestmedia
- rte
- wsj
- heavy
- cp24
- vivanews
- mashable
- apple_news
- huanqiu
- topbuzz
- westernjournalism
- npr
- npr_audio
- ltn_tw
- kompas
- donga
- detik
- appledaily_tw
- okezone
- gmw
- globo_video
- bleacherreport
- bleacherreport_video
- msn_news
- cbs_news_streaming
- chinadaily
- vox
- abc_australia
- providr
- patch
- latimes
- nextmedia
- nytimes

- nydailynews
- heraldm
- sky_news
- topbuzz_video
- bingnews
- uusee
- caijing
- buzzfeed
- mediasetspain
- baidu_news
- wsj_video
- blastingnews_video
- globo
- milliyet
- reuters
- nia
- mynet
- echoroukonline
- aljazeera
- linkedin_pulse
- blastingnews
- day_az
- hao123
- sapo
- washingtonpost
- huffingtonpost
- medium
- indiatimes
- abcnews
- xinhuanet
- apple_news_widget
- univision
- usatoday
- flipboard
- ntd_tv
- ipla
- nownews
- udn_tw
- libero
- guardian
- feedly
- patch_video
- ndtv
- rottentomatoes
- digg
- dailymail
- daum
- foxnews
- breitbart
- merdeka
- liputan6

- cnbc
- cnbc_video
- msnbc
- nbc_news
- google_news
- cnlive
- cnn
- cntv
- clubic
- cnet
- china_com
- china_cn
- clien
- livenewschat
- tribunnews
- ettoday
- bittorrent_tracker
- kgrid
- ares
- pando
- bittorrent_dht
- kazaa
- manolito
- vuze
- goboogy
- gnutella
- fring
- dc_plus
- winmx
- moopolice
- 1337x
- asiandvdclub
- bemaniso
- bitenova
- bithq
- bitme
- bitmetv
- bitseduce
- deepnet
- live_mesh
- cinemageddon
- central-torrent
- kat
- bittorrent
- bitlord
- extratorrent
- extremebits
- dreamora
- danishbits
- crazysaloon
- v-share

- poco_data
- namipan
- bearshare
- qqmusic
- qtrax
- raulken
- scenehd
- share
- winny
- tribler
- gnunet
- soribada
- stealthnet
- blubster
- slsk
- sync
- bleep
- foxy
- bittornado
- bitspirit
- amule
- bitcomet
- fileguri
- vagaa
- rarbg
- kugoo
- directconnect
- edonkey
- openft
- tudou_speedup
- utp
- adc
- jxta
- mute
- thepiratebay
- ares_chat
- ares_transfer
- filetopia
- applejuice
- rutracker
- swepiracy
- tamilthunder
- usabit
- bitvaulttorrent
- bitsoup
- browntracker
- vtunnel
- tor_onion_service
- cyberghost_vpn
- glype
- tunnelbear

- ghostsurf
- speedvpn
- securitykiss
- dotvpn
- pagekite
- hexatech_vpn
- logmein_hamachi
- fsecure_freedome
- http_proxy
- fly_proxy
- hidemyass
- zenmate
- i2p
- opendoor
- guardster
- telex
- lantern
- tor2web
- privax
- droidvpn
- phproxy
- hola
- cgi_proxy
- jap
- spotflux
- authentic8_silo
- avoidr
- bypassthat
- kproxy
- megaproxy
- 4everproxy
- wikimedia
- mendeley
- feidian
- morningstar_posting
- morningstar
- google_earth
- wikipedia
- wikihow
- wikihow_video
- wikia
- zoho_wiki
- nationalgeographic
- emaps
- researchgate
- wikidata
- wiktionary
- answers
- docstoc
- quora
- stackoverflow

- ehow
- namu_wiki
- alexa
- youseemore
- wikiquote
- wikibooks
- 123people
- about
- nasa
- elsevier
- zwiki
- wordreference
- thesaurus_com
- w3schools
- dictionary_com
- reference_com
- tikiwiki
- mediawiki
- mylife
- archive
- gopher
- ask
- rip
- bgp
- bing_desktop
- zum
- yahoo_douga
- bing
- baidu
- bing_bar
- google
- sogou
- yandex_images
- yandex
- yahoo_korea
- yahoo_search
- google_custom_search
- coccoc
- google_scholar
- elastic_search
- mywebsearch
- torrentdownloads
- stumbleupon
- google_safebrowsing
- web_de
- web_crawler
- haosou
- myway
- goo_jp
- gougou
- google_books

- yahoo
- officedepot
- cj_mall
- google_shopping
- jingdong
- yandex_market
- zumi
- amazon_primenow
- jingdong_video
- ebates
- newegg
- weloveshopping
- amazon_fresh
- google_store
- gearbest
- taobao
- made_by_google
- slrclub
- mouthshut
- flipp
- olleh
- google_trusted_store
- google_express
- instagram
- heystax
- friends_reunited
- fotolog
- flixster
- gamerdna
- just
- cloob
- classmates
- cafemom
- epernicus
- elftown
- draugiem
- faces
- exploroo
- experience_project
- airtime
- facebook
- yahoo_maktoob
- 51_com
- lokalisten
- goodreads
- grono
- gather
- geni
- habbo
- hospitality_club
- blackplanet

- bigtent
- badoo
- asmallworld
- athlinks
- tango_file_transfer
- daily_booth
- daily_strength
- local_guides_connect
- imvu
- afterschool
- facebook_social_plugin
- indaba_music
- internations
- imgur_upload
- snapchat
- irc_galleria
- google_plus
- kiwibox
- pixiv
- yoics
- imgur
- yammer
- weourfamily
- sayclub
- triberr
- dudu
- gifboom
- ameba_now
- vkontakte
- vampirefreaks
- twitter
- tribe
- meetme
- bubbly
- odnoklassniki_apps
- delicious
- sonico
- shelfari
- ryze
- renren
- twitter_analytics
- flickr_upload
- pinterest
- travbuddy
- studivz
- linkedin_mobile
- flixwagon_video
- mailru_myworld
- kaixin
- ameba
- instagram_upload

- askfm
- skout
- we_heart_it
- atarashii-ayakashi
- facebook_upload
- wixi
- flixwagon
- cyworld
- flickr
- hi5
- viadeo
- linkedin
- myspace_upload
- me2day
- pinterest_post_delete
- pinterest_posting
- myspace_post_delete
- myspace_posting
- beetalk
- yogrt
- tango
- sina
- pixnet
- meetup
- meetin
- mixi
- facebook_typing
- meinvz
- google_plus_posting
- linkedin_postings
- google_plus_post_delete
- linkedin_post_delete
- xt3
- xing
- netlog
- friendster
- odnoklassniki_messaging
- myspace
- ppomppu
- doshow
- yy
- keek
- orkut
- odnoklassniki
- chatroulette
- chinaren
- anobii
- tagged
- bebo
- twtkr
- eskimi

- beetalk_voice
- regram
- band
- kakao_story
- bbm_channels
- plurk
- qzone
- proxono
- proxeasy
- flipagram
- yahoo_geocities
- wordpress
- yahoo_biz
- yahoo7
- tastemade
- zamzar
- hong_kong_toolbar
- fonts_com
- kapook
- xml
- webbiographies
- windowslivespace
- wiserearth
- stocktwits
- disqus
- trippy
- howstuffworks_images
- books_iread
- care2
- airtime_surveys
- hootsuite
- pantip
- topix
- sanook
- orb
- iapp
- shockwave_flash
- avatars_united
- rbb_today
- twitter_plugin
- linkedin_plugin
- google_translate
- airelive
- globaltv
- google_calendar
- justin_tv
- listography
- librarything
- jammerdirect
- kaioo
- zamzar_upload

- godaddy
- fedex
- ancestry
- dek-d
- wpengine
- yahoo_toolbar
- webshots
- megavideo
- mobile_me
- meevee
- tstore_downloads
- fotki
- filestube
- glide
- yesky
- mop
- xml-sitemaps
- yoku
- google_plugin
- tstore_apps
- samknows
- calameo_upload
- calameo
- allrecipes
- rss
- sharepoint_document
- octopz
- photobucket
- local_guides
- 4chan
- wasabi
- radiko
- iask_cn
- rbc_cn
- truehits
- snapfish_upload
- lifebuzz
- gnutella_web_caching
- seesmic
- seeqpod
- shutterfly
- japanpost
- oneworldtv
- po_st
- sharethemusic
- scispace
- kroobannok
- true_af
- krumanow
- google_toolbar
- icap

- voip_line_downloads
- ameba_ownd
- skydrive
- socialtv
- thekitchn
- surveymonkey
- peercast
- reddit
- teachstreet
- mthai
- mono-mobile
- togetheroncampus
- alicezone
- eventbrite
- eventbrite_upload
- zoho_survey
- word_online
- box_editing
- twitpic
- zoho_people
- sky
- kauriworld
- youth_china_federation
- tunewiki
- 2ch
- msn_world
- feedburner
- facescal
- perfspot
- livemapp
- rssing
- google_docs_uploading
- flipagram_video
- circumventor
- google_opinion_rewards
- issuu
- picsart
- yousendit
- gogobox
- funshion
- google_desktop
- okwave
- ups
- usps
- yik_yak
- wickr
- whisper
- tinder
- wickr_messaging
- google_picasa
- multiply

- shutterfly_upload
- shareplan
- storify
- qook
- my_yahoo
- yahoo_notepad
- yahoo_calendar
- xuite
- nexopia
- google_what_browser
- muxlim
- myanimelist
- mychurch
- myheritage
- my_opera
- ubuntu_one
- skyblog
- msn_search
- google_myactivity
- passportstamp
- partnerup
- ngo_post
- oneclimate
- advogato
- surrogafier
- google_domains
- wix
- fasttrack
- palringo
- live_me
- scribd
- made_with_code
- snapfish
- google_sites
- qapacity
- quarterlife
- ufc_tv
- sportchosun
- bt_sports
- eurosport
- bbc_sports
- nbcsports
- espncricinfo
- espncricinfo_video
- mlb_tv
- tsn
- 7sports
- 7tennis
- naver_sports
- espnstar
- foxsports

- sportsseoul
- msn_sports
- foxsoprts_go
- nike
- nba
- nfl
- tpl
- foxsports_au
- nrl
- espn_watch
- sportsillustrated
- mpora
- afl_live
- wwe
- espn
- foxsports_go
- nfl_video
- nfl_streaming
- nhl_gamecenter_live
- nfl_sunday_ticket
- nfl_gamepass
- nhl
- ncaa
- cbc_sports
- t_online
- telenet_be
- tiscali
- blackberry
- softbank
- project_fi
- sprint
- att
- chunghwa
- t_freemium
- tu_me
- yahoo_travel
- travelocity
- couch_surfing
- ibibo
- airasia
- airbnb
- goibibo
- ana_airways
- travellerspoint
- concur
- ryanair
- concursolutions
- kayak
- tripadvisor
- skyscanner
- google_trips

- rezgo
- irctc_railways
- southwest
- expedia
- ctrip
- teredo
- capwap
- freenet
- gtp
- gtpv
- pptp
- ultrasurf
- logmein
- hotspotshield
- psiphon
- http_tunnel
- socks5
- socks4
- rpc_over_http
- dostupest
- hamachi
- openvpn
- gpass
- gtpv2
- steganos_vpn
- unknown
- abc_tv
- explorer_tv
- lovefilm
- espn2
- espn3
- channel5_demand5
- channel4_4od
- kylintv
- animelab
- presto
- 9now
- tenplay
- ifilm
- zattoo
- lynda_video
- readytalk_video
- conferendo
- netflix_video
- yahoo_screen
- metacafe
- hulu
- howcast
- hbo_go
- cnet_tv
- woopie

- afreeca_video_live
- mnet
- blinkx
- ebs
- 6cn
- filmaffinity
- purple
- dailymotion_video
- flixster_video
- foxsports_video
- howstuffworks_video
- howstuffworks_video_control
- mixbit
- qik_video
- muzu_tv
- google_videos
- melon_vod
- mbc_live
- mun2
- baitu
- mgoon
- mytv
- mgo
- ustream_video_broadcast
- redboxinstant
- cw_tv
- sbs_gorealra
- sbs_streaming
- tru_tv
- tv_guide
- pandora_audio
- epix
- air_video
- itunes_streaming
- sbs_ondemand
- microsoft_stream
- vudu
- tvnz
- tving
- gyao
- hovrs
- supersoccer
- brighttalk_video
- letv_video
- google_video
- rogersanyplacetv
- ku6_video
- wtv
- wimp
- ae_tv
- sage_tv

- myvideo
- olleh_tv
- graboid
- fiji
- godtube
- fox
- actvila
- redbull_tv
- animebw
- 56com
- yandex_video
- letv
- mbc
- breitbart_video
- rutube
- hottv
- btv_sk
- naver_streaming_vod
- octoshape
- stupid_video
- myspace_video
- yupptv
- imdb
- freeetv
- flumotion
- indowebster
- hgtv
- divx_webplayer
- channel5
- dropcam
- youview
- yomvi
- msn_video_streaming
- meerkat
- periscope
- photobucket_video
- videodetective
- amazon_prime_video
- blinkbox
- baidu_video
- kontiki
- niconico
- daum_tvpot_live
- redboxinstant_video
- popcorn_time
- vudu_streaming
- veohtv_upload
- youku_streaming
- italkbb
- vevo_streaming
- youku_upload

- nate_hoppin
- nate_hoppin_vod
- nate_streaming
- video
- thunder_kankan
- tnt_tv
- svt
- tbs
- telly
- spb_tv
- niconico_douga
- pandora_tv
- ooyala
- pandora
- eyejot
- earthcam
- crunchyroll
- cbs
- everyon
- amazon_chime_video
- twitter_video
- melon
- btv
- 3dxplorer
- pooq
- veohtv_streaming
- twitch_streaming
- teachertube_video
- tvu_network
- hulu_plus
- sbs_au
- facebook_video
- uplynk
- ulive
- uplus_box
- tvnplayer
- tvkoo
- jwplayer
- quickflix
- live_ly
- adnstream
- mtv
- break
- sky_video
- daum_tvpot
- mgoon_streaming
- mgoon_upload
- foxtel
- office365_video
- kakao_story_video
- video_jug

- vidmax
- vgtv
- viddler
- ameba_fresh
- stagevu
- plus7
- teachertube
- uplus_box_streaming
- ovguide
- qvod
- douyu
- dailymotion
- youtube_tv
- aol_streaming
- ustream_video
- vimeo_video
- vimeo_video_control
- tubitv
- tv2_norway
- tv_streamyx
- crackle
- jaman
- afreeca
- livestation
- wowtv
- witv
- daum_tvpot_vod
- ishoot
- clarovideo
- cntv_cctv_vod
- thevideos_tv
- veetle
- hbo_now
- tv4play
- tudou
- ustream
- mbc_vod
- channel4
- pptv
- pps
- youth_china_federation_video
- xfinity_tv
- xbox_live_video
- ku6
- baidu_player
- cctv_vod
- sina_video
- nextguide
- tving_downloads
- funshion_video
- iqiyi_video

- brightcove_video
- vidyo_video
- earthcam_video
- dropshots
- foxnews_video
- tvteka
- 6play
- tubitv_video
- naver_cast
- ionair
- sky_go
- cinemanow
- amazon_video
- youku
- gomtv
- slingbox
- putlocker
- live_ly_video
- brighttalk
- babelgum
- sky_on_demand
- sony_tv
- imdb_video
- tv4_sweden
- tv3_sweden
- tv_catchup
- mediaset
- kartinatv
- turkcelltv
- yahoo_video
- iqiyi
- vimeo
- tu
- tv
- videosurf
- videobash
- sohu_video
- paltalk_video
- xinhuanet_video
- keyholetv
- facebook_live
- vine
- pplive
- thevideos_tv_video
- cbs_streaming
- ontv_streaming
- ontv
- http_file_video
- screen_junkies
- bambuser
- viewster

- useetv
- rai
- yahoo_douga_video
- flash
- conviva
- netmovies
- nbc_streaming
- naver_streaming_live
- qqlive
- youtube_gaming
- samsung_videohub
- htv
- hot_tv
- simple_tv
- rottentomatoes_video
- metacafe_video
- blip_tv_video
- cnet_video_request
- livestream
- coolstreaming
- bilibili
- pandora_tv_client
- own3d_tv
- now_tv
- pdbox
- directv
- 123movies
- go90
- qdown
- voddler
- sopcast
- veohtv
- tvuplayer
- cnbc_prime
- netflix
- silverlight
- bbc_player
- itv
- bondisk
- afreeca_video
- nlive_tv
- archive_video
- brightcove
- cine21
- cuptv
- stan
- aereo
- abc_streaming
- aol_on
- disney_videos
- livestation_streaming

- stagevu_video
- qqvideo
- vevo
- blip_tv
- blockbuster
- ytn
- bluejeans_meeting
- bluejeans
- zoom
- anydesk
- lotusnotes
- nomachine_remote_control
- webrdp
- chrome_remote_desktop
- office_mobile
- office_calendar
- pastebin
- rsupport
- bitbucket_upload
- remoteview
- rsh
- premiere_global_services
- imeet_file_share
- gotomeeting_video
- groupwise
- horde
- cgp
- teamviewer
- remotecall
- amazon_chime_webclient
- amazon_chime_meeting
- traceroute
- rlogin
- vnc
- mosh
- moinmoin
- google_forms
- podio
- tibbr
- uberconference
- ms_rdp
- git
- jira
- office365_planner
- powerpoint_online
- excel_online
- sway
- slack
- onenote
- zoho_notebook
- zoho_planner

- zoho_sheet
- zoho_show
- lotus_live
- docusign
- yandex_translate
- windows_push_notifications
- websocket
- yahoo_web_analytics
- web_crawler_bot_facebook
- web_crawler_bot_alexa
- web_crawler_bot_google
- supremo
- bing_translator
- via3
- pivotaltracker_upload
- logmeinrescue_remote_control
- on24_upload
- on24
- alpemix
- office_delve
- google_docs
- fogbugz
- github
- diply
- readytalk
- mikogo
- sharepoint_blog
- sharepoint_calendar
- ibm_meetings
- eroom_net
- ica
- anydesk_remote_control
- webex_whiteboard
- google_slides
- google_keep
- sharepoint_online
- subversion
- amazon_chime
- diskshare
- logmeinrescue
- microsoft_teams
- atlassian_bamboo
- eroom_hosting
- showmypc
- sharepoint_admin
- sharepoint
- vidyo
- rally_software
- alpemix_remote_control
- jive_software
- zoom_meeting

- git_upload
- sharepoint_wiki
- fuze
- spotnet
- tableau
- tableau_desktop
- tableau_public
- tableau_reader
- adobe_echosign
- sugar_crm
- radmin
- microsoft_forms
- techinline
- officehard
- rexec
- rdmplus
- slideshare
- pcoip
- bitcoin
- imeet
- webex_weboffice
- chromium
- web_crawler_bot
- remoteview_update
- ammyy_admin_remote_control
- ammyy_admin_chat
- ultrahook
- web_crawler_bot_baidu
- web_crawler_bot_yahoo
- web_crawler_bot_bing
- web_crawler_bot_sogou
- web_crawler_bot_yandex
- beamyourscreen
- bomgar
- vyew
- tokbox
- zoho_meeting
- gotomeeting
- live_meeting
- globalmeet
- convo
- convo_upload
- convo_chat
- igloo_software
- moxtra
- pivotaltracker
- hpe_myroom
- bitbucket
- atlassian
- 163com
- pcanywhere_remote_control

- gatherplace
- airdroid
- netsupport_manager
- nomachine
- webex
- gotomypc
- anyterm
- apache_jserv
- bbcp
- deskmetrics
- gomeetnow
- perforce
- clicky
- hive
- pentaho
- asana
- crossloop
- office_online
- uberconference_screenshare
- finger
- finch
- hamicloud
- jedi
- cloud9
- huddle
- join_me
- isl_alwayson
- islonline
- zoho_writer
- springcm
- dameware_mini_remote
- atom
- fcc_speed_test
- hp_eprint_service
- litecoin
- fcc_speed_test_download
- fcc_speed_test_upload
- lastpass
- localtunnel
- thinkfree
- mercurial
- google_spreadsheets
- pcanywhere
- zinc
- proofhub
- chatter
- proofhub_file_transfer
- smartsheet_upload
- smartsheet
- tigertext
- simplenote

- pushbullet
- ngrok
- roboform
- huawei_dbank
- pinterest_messaging
- melduim
- xdmcp
- pcvisit
- projectplace
- google_drawings
- facebook_workplace
- screenconnect
- gotoassist
- pivotaltracker_download
- fastviewer
- confluence
- splashtop_remote
- isl_light
- pingsta
- nate_mail
- laposte_webmail
- qq_mail
- telenet_be_mail
- kaixin_mail
- korea_mail
- hushmail_upload
- ilohamail
- atmail
- chinaren_mail
- naver_mail
- ymail_classic
- secureserver_mail
- openwebmail
- sendgrid
- roundcube
- t_online_mail
- ibm_verse
- gmail
- mail2000
- mail_189
- yandex_mail
- ibm_connections_mail
- mailinator
- usermin
- ymail2
- pop3
- owa
- pop3s
- icloud_mail
- amos
- zoho_mail

- squirrelmail
- startmail
- outlook
- smtp
- smtps
- imp
- imaps
- imap
- mailru_agent
- mailru_webagent
- mailru_mail
- zimbra
- gmail_inbox
- daum_mail
- aol_mail
- 51_com_mail
- hinet_mail
- tiscali_mail
- web_de_mail
- ning_mail
- amazon_ses
- comcast_webmail
- cox_webmail
- zimbra_desktop
- mailru
- ymail_mobile_new
- 1and1_mail
- mail_com
- mimp
- livemail_mobile
- zimbra_standard
- ymail_mobile
- dimp
- mapi_over_http
- fastmail
- cryptoheaven
- gmail_mobile
- gmx
- gmail_basic
- live_hotmail
- mapi
- hiworks
- hushmail
- ms_exchange

F5BigPePolicy classification categories

The F5BigPePolicy Custom Resource (CR) supports the following classification categories:

- any (default)
- Adult_Content
- Advertisements
- Application_and_Software_Services
- Audio
- Audio_Video
- Blogs_and_Personal_Sites
- Business_and_Economy
- Computer_Security
- Computer_and_Internet
- Content_Delivery_Networks
- Database
- Dating
- Education
- Encrypted
- Entertainment
- Facebook_Apps
- Facebook_Chat
- Facebook_Commenting
- Facebook_Events
- Facebook_Games
- Facebook_Mail
- Fashion_and_Beauty
- File_Download_Servers
- Financial_Data_and_Services
- Games
- Health_and_Medicine
- Hosted_Business_Applications
- IP_Layer_Protocols
- Information_Technology
- Instant_Messaging
- Internet_Communication
- Internet_Telephony
- Job_Search
- Local_Information
- Message_Boards_and_Forums
- Music
- Network_Management_and_Services
- News_and_Media
- Peer-to-Peer_File_Sharing
- Proxy_Avoidance
- Reference_and_Research
- Routing
- Search_Engines
- Shopping
- Social_Networking
- Social_Web_-_Various

- Sports
- Telecommunication
- Travel
- Tunneling
- Unknown
- Video
- Web_Collaboration
- Web_based_email

F5BigPePolicy URL Categories

The F5BigPePolicy Custom Resource (CR) supports the following URL categories:

- any (default)
- abortion
- abused-drugs
- adult-content
- advanced-malware-payloads
- advertisements
- alcohol-and-tobacco
- auctions
- bot-networks
- business-and-economy
- cheating
- computer-and-internet
- computer-security
- content-delivery-networks
- cult-and-occult
- dating
- dead-sites
- dynamic-content
- educational-institutions
- entertainment
- fashion-and-beauty
- financial-data-and-services
- gambling
- games
- government
- gross
- hacking
- hate-and-racism
- health-and-medicine
- home-and-garden
- hunting-and-fishing
- illegal-or-questionable
- image-and-video-search
- individual-stock
- internet-communication
- internet-portals
- internet-watch-foundation
- job-search
- keyloggers-and-monitoring
- kids
- legal
- local-information
- marijuana
- military
- motor-vehicles
- music
- news-and-media

- nudity
- online-greeting
- parked-domain
- pay-to-surf
- peer-to-peer-file-sharing
- personal-network-storage-and-backup
- personals-and-dating
- philosophy
- phishing-and-other-frauds
- private-ip-addresses
- proxy-avoidance
- questionable
- real-estate
- recreation-and-hobbies
- reference-and-research
- religion
- search-engines
- sex-education
- shareware-and-freeware
- shopping
- social-networking
- society-and-lifestyles
- spam-urls
- sports
- spyware-and-adware
- streaming-media
- swimsuits-and-apparel
- training-and-tools
- translation
- travel
- uncategorized
- unknown

F5BigFwPolicy

Overview

The F5BigFwPolicy Custom Resource (CR) applies industry-standard firewall rules to the Traffic Management Microkernel (TMM), ensuring that only connections initiated by trusted clients will be accepted. When applying a new F5BigFwPolicy CR configuration, firewall rules are first sent to the Application Firewall Management (AFM) Pod to be compiled into a binary large object (BLOB), improving processing performance. Once the firewall BLOB is compiled, it is sent to the TMM Proxy Pod to begin inspecting and filtering network packets.

This document guides you through understanding, configuring and installing a simple F5BigFwPolicy CR.

CR parameters

The tables below describe the F5BigFwPolicy CR parameters

metadata

Parameter	Description
`name`	The name of the Firewall Policy. This value is referencd by CNF Traffic Management CRs.
`namespace`	The Kubernetes Namespace the firewall Policy will install to.

spec

Parameter	Description
`rule.name`	The name of the firewall rule. A policy can contain multiple firewall rules.
`rule.ipProtocol`	Specifies the IP protocol against which the packet will be compared. The default value is **“any”**. The [F5BigFwPolicy IP Protocols] document contains the full list of supported protocols.
`rule.action`	Specifies the action that will be applied to packet that matches ACL rule: **“accept”**, **“drop”**, or **“reject”**.
`rule.logging`	Enables ACL rule match logging: **true** or **false** (default).
`rule.source.addresses`	Specifies a list of IPv4 or IPv6 source addresses against which the packet will be compared: host **2002::10:10:10:1**, subnet **2002::10:10:0:0/96**, or range **2002::10:10:10:1-2002::10:10:10:20**.
`rule.source.addressLists`	Specifies the F5BigCneAddressList by `metadata.name` against which the packet will be compared. See the **Address and Port Lists** section below.
`rule.source.ports`	Specifies a list of source service ports or port ranges against which the packet will be compared. Port **0** is a valid value, that functions as a service port, ***not as a wildcard***.

Parameter	Description
`rule.source.portLists`	Specifies the F5BigCnePortList by `metadata.name` against which the packet will be compared. See the **Address and Port Lists** section below.
`rule.source.vlans`	Specifies a list of F5BigNetVlans in an array.
`rule.destination.addresses`	Specifies a list of IPv4 or IPv6 destination addresses against which the packet will be compared: host **2002::10:10:10:1**, subnet **2002::10:10:0:0/96**, or range **2002::10:10:10:1-2002::10:10:10:20**.
`rule.destination.ports`	Specifies a list of destination service ports or port ranges against which the packet will be compared. Port **0** is a valid value, that functions as a service port, ***not as a wildcard***.
`rule.destination.addressLists`	Specifies the F5BigCneAddressList by `metadata.name` against which the packet will be compared. See the **Address and Port Lists** section below.
`rule.destination.portLists`	Specifies the F5BigCnePortList by `metadata.name` against which the packet will be compared. See the **Address and Port Lists** section below.

CR Example

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFwPolicy
metadata:
  name: "cnf-fw-policy"
  namespace: "cnf-gateway"
spec:
  rule:
    - name: allow-10-20-http
      action: "accept"
      logging: true
      ipProtocol: tcp
      source:
        addresses:
          - "2002::10:20:0:0/96"
      destination:
        ports:
          - "80"
    - name: allow-10-30-ftp
      action: "accept"
      logging: true
      ipProtocol: tcp
      source:
        addresses:
          - "2002::10:30:0:0/96"
      destination:
        ports:
          - "20"
          - "21"
    - name: drop-all
```

```
      action: "drop"
      logging: true
      ipProtocol: any
      source:
        addresses:
          - "::0/0"
          - "0.0.0.0/0"
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigFwPolicy CR shortName is **fwpol**.

View CR instance:

```
kubectl get fwpol -n <namespace>
```

View CR configuration:

```
kubectl get fwpol -n <namespace> -o yaml
```

Address and Port lists

Complex lists of IP addresses and service ports can be configured using the F5BigCneAddresslist and F5BigCnePortlist CRs. The address and port list CRs can then be referenced by the F5BigDdosPolicy CR.

***Address list**:*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCneAddresslist
metadata:
  name: allow-ipv6
  namespace: cnf-gateway
spec:
  addresses:
   - "2002::192:168:10:1-2002::192:168:10:10"
   - "2002::10:10:10:0/112"
```

***Port list**:*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCnePortlist
metadata:
  name: allow-5000s
spec:
  ports:
  - "5000-5500"
```

Firewall mode

CNFs default firewall mode controls how network packets are handled when either of these conditions are met:

- None of the installed CNFs CRs reference a F5BigFwPolicy.
- A CNFs CR does reference a F5BigFwPolicy, however, packets do not match any of the rules.

The table below describes each of the default firewall mode settings:

Mode	Behavior
`accept`	Network packets are accepted and processed by TMM. This is the default setting.
`drop`	Network packets are silently dropped.
`reject`	Network packets are rejected. For TCP connections, a **RST** (reset) packet is sent in reponse.

By default, the firewall mode **accepts** all network packets not matching a F5BigFwPolicy firewall rule. You can modify this behavior prior to installing the BIG-IP Controller, using the `defaultFirewallRule.action` Helm parameter. For more information, see **step 6** in the **Installation** section of the BIG-IP Controller guide.

```
afm:
  defaultFirewallRule:
    action: accept
    log: true
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

Use these steps to install the example F5BigFwPolicy CR, and the ***optional*** CNFs CRs. Each step offers a brief description of the example CR.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** The example F5BigLogHslpub CR specifies a remote server with IP/port **[2002::10:30:2:220]:514**, and the **udp** protocol. Copy and paste the example into a YAML file:

   Note: *The F5BigLogHslpub CR will be referenced by the F5BigLogProfile.*

   ```
   apiVersion: k8s.f5net.com/v1
   kind: F5BigLogHslpub
   metadata:
     name: "cnf-hsl-pub"
     namespace: "cnf-gateway"
   spec:
     pool:
   ```

```
  - name: "hsl-pool"
    endpoint:
    - "[2002::10:30:2:220]:514"
  syslog:
  - name: "cnf-syslog"
    format: "rfc5424"
    protocol: "udp"
    pool: "hsl-pool"
```

2. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

3. ***Optional:*** The example F5BigLogProfile CR specifies firewall events such as **aclMatchAccept** and **aclMatchDrop**, and sends them to the remove logging server. Copy and paste the example into a YAML file:

   Note: *The F5BigLogProfile CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-logs"
  firewall:
    enabled: true
    network:
      publisher: "cnf-hsl-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
        tcpEvents: true
        translationFields: true
```

4. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-log-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

5. The example F5BigFwPolicy CR allows HTTP port **80** and FTP ports **20** and **21** from source IP subnet **2002::10:30:0:0/96**. Copy and paste the example into a YAML file:

   Note: *The F5BigFwPolicy CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFwPolicy
metadata:
  name: "cnf-fw-policy"
```

```
  namespace: "cnf-gateway"
spec:
  rule:
    - name: allow-http
      action: "accept"
      logging: true
      ipProtocol: tcp
      source:
        addresses:
          - "2002::10:20:0:0/96"
      destination:
        ports:
          - "80"
    - name: allow-ftp
      action: "accept"
      logging: true
      ipProtocol: tcp
      source:
        addresses:
          - "2002::10:30:0:0/96"
      destination:
        ports:
          - "20"
          - "21"
    - name: drop-all
      action: "drop"
      logging: true
      ipProtocol: any
      source:
        addresses:
          - "::0/0"
          - "0.0.0.0/0"
```

6. Install the F5BigFwPolicy CR:

```
kubectl apply -f cnf-fw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigFwPolicy CR was **added/updated**:*

```
I0202 12:00:00.12346   1 event.go:282
 ↪  Event(v1.ObjectReference{Kind:"F5FirewallPolicy",
FirewallPolicy cnf-gateway/cnf-fw-policy was added/updated
```

7. ***Optional:*** The example F5BigContextSecure CR listens for connections destined to the **2002::200:200:200:0/112** subnet on the **subscriber-vlan** interface, and references all of the installed CRs. Copy and paste the example into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "2002::200:200:200:0/112"
   destinationPort: 0
   firewallEnforcedPolicy: "cnf-fw-policy"
```

```
  logProfile: "cnf-log-profile"
  ipProtocol: "any"
  profile: "fastL4"
  vlans:
    vlanList:
      - "subscriber-vlan"
```

8. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-context.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-context was added/updated
```

9. Continue to the **Additional CRs** and **Firewall statistics** sections.

Additional CRs

The F5BigFwPolicy can also be referenced by these CNFs CRs:

- F5BigAlgFtp - File Transfer Protocol (FTP) application layer gateway services.
- F5BigAlgTftp - Trivial File Transfer Protocol (TFTP) application layer gateway services.
- F5BigAlgPptp - Point-to-Point Tunneling Protocol (PPTP) application layer gateway services.
- F5BigAlgRtsp - Real Time Streaming Protocol (RTSP) application layer gateway services.

Firewall statistics

If the [TMM Debug] sidecar is enabled (default), use the steps below to verify firewall filtering statics.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify the F5BigFwPolicy statistics:

```
tmctl -d blade fw_rule_stat
```

```
context_type context_name
------------ ----------------------------------------
virtual      cnf-gateway-cnf-fw-policy-SecureContext_vs

rule_name                           micro_rules counter last_hit_time action
----------------------------------- ----------- ------- ------------- ------
allow-10-20-http-firewallpolicyrule           1       2    1638572860      2
allow-10-30-ftp-firewallpolicyrule            1       5    1638573270      2
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigNatPolicy

Overview

The F5BigNatPolicy Custom Resource (CR) is a carrier-grade network address translation (CG-NAT) solution, using large-scale NAT (LSN) pools to support high-volume, low latency 5G workloads. The F5BigNatPolicy provides NAT44, NAT46, NAT64, and NAT66 IP address translations, and can advertise allocated NAT IP addresses to BGP peers, ensuring connections responses are routed properly and efficiently. Once installed and configured, the F5BigNatPolicy can be referenced by any of the Traffic Management [CNF CRs].

NAT implementations

The NAT46 and NAT64 implementations require multiple CNFs CRs with specific CR configurations. Refer to the CNFs NAT implementation guides for assistance:

- CNFs NAT64
- CNFs NAT46

This document guides you through understanding, configuring and installing a simple F5BigNatPolicy CR.

CR parameters

The tables below describe the F5BigNatPolicy CR parameters.

metadata

Parameter	Description
`name`	The name of the NAT policy. This value is referenced by the traffic management [CNF CRs].
`namespace`	The Kubernetes namespace the NAT policy will install to.

spec.sourceTranslation

Parameter	Description
`name`	Specifies the NAT Source Translation object name.
`type`	Specifies the type of translation to be performed: **static-nat**, **static-pat** or **dynamic-pat**.
`addresses`	Specifies a list of IPv4 or IPv6 addresses: host **2002::10:10:10:1**, subnet **2002::10:10:0:0/96**, or range **2002::10:10:10:1-2002::10:10:10:20**.
`port`	Specifies a list of service ports or service port ranges. **“8000”**, **“8000-10000”**
`icmpEcho`	Enables ICMP echo responses on translation IP addresses: **true** or **false** (default).

Parameter	Description
`proxyArp`	Enables proxy arp for translation IP addresses: **true** or **false** (default).
`patMode`	Specifies the translation mode of operation. **napt** (default) or **pba**. This parameter applies only if `sourceTranslation.type` is **dynamic-pat**.
`excludeAddresses`	Specifies a list of IPv4 or IPv6 Addresses to exclude from translation in the NAT pool: host **2002::10:10:10:1**, subnet **2002::10:10:0:0/96**, or range **2002::10:10:10:1-2002::10:10:10:20**.
`inbound.mode`	Enables inbound connections: **none**(default) or **endpoint-independent-filtering**. This parameter applies only if the `sourceTranslation.type` is **dynamic-pat**.
`inbound.eifTimeout`	Specifies the timeout in seconds for **endpoint independent filtering**: **3** (default) to **300**.
`mapping.mode`	Specifies the translated address mapping for setting successful translations: **none** (default), **address-pooling-paired**, or **endpoint-independent-mapping**. This parameter applies only if `sourceTranslation.type` is **dynamic-pat**.
`mapping.timeout`	Specifies the duration in seconds after which successfully translated address **mappings** expire. The default is **300** and the max is **31536000**.
`vlans.vlanList`	Specifies a list of F5BigNetVlans in an array to apply NAT.
`vlans.disableListedVlans`	When enabled, apply NAT on all F5BigNetVlans **except** those defined in the `sourceTranslation.vlans.vlanList` paramter: **true** (default) or **false**.
`routeAdvertisement`	Enables route advertisements for translated IP addresses: **true** or **false** (default).
`hairpinMode`	Enables clients in the same private network to connect using their private destination. TMM translates the connection using the public destination address, but does not send the packet through the public network: **true** or **false** (default).
`clientConnectionLimit`	Specifies the number of translated concurrent connections allowed per subscriber. The default is **0** and the max is **65535**.

spec.sourceTranslation.portBlockAllocation

Parameter	Description
`blockIdleTimeout`	Specifies the amount of time in seconds that an assigned block of ports remains available when idle before it times out: **30** to **31536000**. The default is **3600**.
`blockLifeTime`	Specifies the lifetime in seconds of a block of ports: **0** (default) to **31536000**.

Parameter	Description
`blockSize`	Specifies the number of ports per block. Each block is assigned to one client. A client can use all ports in a block multipled by the `clientBlockLimit` value, up to the connection limit if one is set: **1** to **31536000**. The default is **64**.
`clientBlockLimit`	Specifies the number of blocks that can be assigned to a client: **0** to **31536000**. The default is **1**.
`zombieTimeout`	Specifies the timeout duration for a zombie port block, which is a timed out port block with one or more active connections: **0** (default) to **31536000**. When the timeout duration expires, connections using the zombie block are killed and the zombie port block becomes an available port block. The default is **0**, which corresponds to an infinite timeout. The setting is ignored if the `blockLifetTme` value is **0**.

spec.rule

Parameter	Description
`name`	Name of the rule.
`description`	Description of the rule.
`ipProtocol`	Specifies the IP protocol against which the packet will be compared: **tcp**, **udp** or **any** (default).
`source.vlans`	Specifies a list of VLAN against which the packet will be compared.
`source.addresses`	Specifies a list of IPv4 or IPv6 addresses against which the packet will be compared: host **2002::10:10:10:1**, subnet **2002::10:10:0:0/96**, or range **2002::10:10:10:1-2002::10:10:10:20**.
`source.addressLists`	Specifies a list of address-list names F5BigCneAddresslist by `metadata.name` against which the packet will be compared.
`source.ports`	Specifies a list of source service ports or port ranges against which the packet will be compared. Port **0** is not a valid value, and is not allowed. “1000-1200”
`source.portLists`	Specifies a list of F5BigCnePortlist by `metadata.name` against which the packet will be compared. **[“source-portlist-1”]**
`destination.addresses`	Specifies a list of IPv4 or IPv6 addresses against which the packet will be compared: host **2002::10:10:10:1**, subnet **2002::10:10:0:0/96**, or range **2002::10:10:10:1-2002::10:10:10:20**.
`destination.addressLists`	Specifies a list of F5BigCneAddresslist by `metadata.name` against which the packet will be compared. **“dest-addr-list-1”**
`destination.ports`	Specifies a list of destination service ports or port ranges against which the packet will be compared. Port **0** is not a valid value, and is not allowed.
`destination.portLists`	Specifies a list of F5BigCnePortList by `metadata.name` against which the packet will be compared.

Parameter	Description
`sourceTranslation`	Specifies the `spec.sourceTranslation.name` parameter to reference.

CR Example

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNatPolicy
metadata:
  name: "cnf-66-nat"
  namespace: "cnf-gateway"
spec:
  sourceTranslation:
    - name: "nat66-dynamic"
      type: "dynamic-pat"
      addresses:
        - "2002::300:300:300:0/112"
      port: "8000-8050"
      icmpEcho: true
      proxyArp: true
      mapping:
        mode: "endpoint-independent-mapping"
        timeout: 300
      inbound:
        eifTimeout: 90
      routeAdvertisement: true
  rule:
    - name: dynamic-nat-66
      ipProtocol: tcp
      source:
        addresses:
          - "2002::100:100:100:0/112"
      destination:
        addresses:
          - "2002::200:200:200:0/112"
      sourceTranslation: "nat66-dynamic"
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigNatPolicy CR shortName is **natpol**.

View CR instance:

```
kubectl get natpol -n <namespace>
```

View CR configuration:

```
kubectl get natpol -n <namespace> -o yaml
```

NAT IP addresses

IP Allocation

When the F5BigNatPolicy is installed, the BIG-IP Controller reserves, and allocates blocks of IP addresses to each of the TMM Proxy PODs to use for NAT. When IP addresses are used and then no longer in use by the TMM, the reserved NAT IP addresses are released for use by other TMM Pods.

BGP Advertisement

When the TMM Proxy Pod installs with the **f5-tmm-routing** container, IP addresses used for NAT client connections are advertised to upstream BGP peers(by setting routeAdvertisement to True), enabling server responses to route back to TMM. Alternatively, you can configure appropriate routes on upstream devices, however, this method does not scale as well, and is more error-prone. For BGP configuration assistance, refer to the BGP Overview.

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- Installed the dSSM Database.
- A Linux based workstation.

Installation

Use these steps to install the example F5BigNatPolicy CR, and the ***optional*** CNFs CRs. Each step offers a brief description of the example CR.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** The example F5BigLogHslpub CR specifies a remote server with IP/port **[2002::10:30:2:220]:514** and the **udp** protocol. Copy and paste the example into a YAML file:

   Note: *The F5BigLogHslpub CR will be referenced by the F5BigLogProfile.*

   ```
   apiVersion: k8s.f5net.com/v1
   kind: F5BigLogHslpub
   metadata:
     name: "cnf-hsl-pub"
     namespace: "cnf-gateway"
   spec:
     pool:
     - name: "cnf-hslpool"
       endpoint:
       - "[2002::10:30:2:220]:514"
     syslog:
     - name: "syslog-dest"
       format: "rfc5424"
       protocol: "udp"
       pool: "cnf-hslpool"
   ```

2. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

3. ***Optional:*** The example F5BigLogProfile CR specifies NAT events such as connection **start** and **end** to send to the remote log server. Copy and paste the CR nto a YAML file:

   Note: *The F5BigLogProfile CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-logs"
  nat:
    enabled: true
    logSubscriberID: true
    publisher: "cnf-hsl-pub"
    inbound:
      start:
        mode: "enabled"
      end:
        mode: "enabled"
    quotaExceeded:
      mode: "enabled"
    errors:
      mode: "enabled"
```

4. The example F5BigNatPolicy CR specifies that subscribers with source IP in the **2002::100:100.100:0/112** subnet, connecting to destinations in the **2002::200:200:200:0/112** subnet, will have their source IP address translated using addresses in the **2002::300:300:300:0/112** subnet. Copy and paste the example into a YAML file:

   Note: *The F5BigNatPolicy CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNatPolicy
metadata:
  name: "cnf-66-nat"
  namespace: "cnf-gateway"
spec:
  sourceTranslation:
    - name: "nat66-dynamic"
      type: "dynamic-pat"
      addresses:
        - "2002::300:300:300:0/112"
      port: "8000-8050"
      icmpEcho: true
      proxyArp: true
      mapping:
        mode: "endpoint-independent-mapping"
        timeout: 300
```

```
      inbound:
        eifTimeout: 90
      routeAdvertisement: true
  rule:
    - name: dynamic-nat-66
      ipProtocol: tcp
      source:
        addresses:
          - "2002::100:100:100:0/112"
      destination:
        addresses:
          - "2002::200:200:200:0/112"
      sourceTranslation: "nat66-dynamic"
```

5. Install the F5BigNatPolicy CR:

```
kubectl apply -f nat-policy.yaml -n <namespace>
```

*In this example, the BIG-IP Controller logs indicate the F5BigNatPolicy CR was **added/updated**:*

```
I0202 12:00:00.12345   1 event.go:282 Event(v1.ObjectReference{Kind:"F5NatPolicy",
NatPolicy cnf-gateway/cnf-nat-policy was added/updated
```

6. The example F5BigContextSecure CR listens for connections destined to the **2002::200:200:200:0/112** subnet on the **subscriber-vlan** interface, and references the installed CRs. Copy and paste the CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-nat-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "2002::200:200:200:0/112"
   destinationPort: 0
   ipProtocol: "any"
   profile: "fastL4"
   natPolicy: "cnf-66-nat"
   vlans:
     vlanList:
       - "subscriber-vlan"
```

7. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-context-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-nat-context was added/updated
```

8. Review the **Additional CRs** and **NAT statistics** sections.

Additional CRs

The F5BigNatPolicy can also be referenced by these CNFs CRs:

- F5BigAlgFtp - File Transfer Protocol (FTP) application layer gateway services.
- F5BigAlgTftp - Trivial File Transfer Protocol (TFTP) application layer gateway services.
- F5BigAlgPptp - Point-to-Point Tunneling Protocol (PPTP) application layer gateway services.
- F5BigAlgRtsp - Real Time Streaming Protocol (RTSP) application layer gateway services.

NAT statistics

If the TMM Debug Sidecar is enabled (default), use the steps below to verify NAT connection statics.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n <namespace> -- bash
```

2. Verify F5BigNatPolicy connection statistics:

```
tmctl -d blade fw_nat_rule_stat
```

```
context_type context_name                          rule_name
------------ ------------------------------------  --------------------------
virtual      cnf-cnf-context-secure-SecureContext_vs  10-20-subnet-natpolicyrule

micro_rules counter last_hit_time action
----------- ------- ------------- ------
          1       8    1643836695      0
```

```
tmctl -d blade fw_nat_trans_stat -s type,name,translation_requests
```

```
type         name        translation_requests
------------ ----------- --------------------
fw_src_trans transparent                    0
fw_dst_trans transparent                    8
fw_src_trans automap                        0
```

3. Verify the F5BigNatPolicy client IP address mappings:

```
lsndb list all
```

```
Client                                                 Connections
-------------------------------------------------------------
0 client with 0 connection found.
LSN Persistence Entries
Client                                                 Translation
-------------------------------------------------------------
10.20.2.220:52110                                10.200.2.8:8265
10.20.2.220                                      10.200.2.8
2 persist entries found.
LSN port block allocations
Client                                                 Port block
-------------------------------------------------------------
0 port block entries found.
LSN Inbound Mapping Entries
Translation                                              Client
-------------------------------------------------------------
```

```
10.200.2.8:8265                          10.20.2.220:52110
10.200.2.7:8397                          10.20.2.220:52106
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigContextSecure

Overview

The F5BigContextSecure Custom Resource (CR) configures the Traffic Management Microkernel (TMM) to perform as an application layer gateway (ALG) for low-latency 5G workloads. The F5BigContextSecure CR provides granular connection management using the following F5 protocol profiles: TCP, UDP and FastL4, and is an integral part of the DNS44, DNS46, and DNS64 implementations.

This document guides you through understanding, configuring and installing a simple F5BigContextSecure CR.

CR parameters

The tables below describe the F5BigSecureContext CR parameters used in this document, refer to the F5BigContextSecure Reference for the full list of parameters.

spec

Parameter	Description
`destinationAddress`	Creates an IPv4 virtual server address that listens for ingress connections: host **10.10.10.50**, subnet **10.10.10.0/24”**.
`ipv6destinationAddress`	Creates an IPv6 virtual server address that listens for ingress connections: host **“4001::1”**, subnet **“4001::/64”**.
`destinationPort`	Defines the service port for ingress connections. **any** (default).
`ipProtocol`	Specifies the virtual server IP protocol: **tcp**, **udp**, or **any** (default).
`profile`	Specifies the profile to be used by the virtual server: **tcp**, **udp**, **fastl4**(default), or **ipother**.
`fastL4Settings.profileName`	Specifies how TMM handles connections using the F5BigFastl4Setting CR’s `metadata.name` value.
`tcpSettings.clientSide`	Specifies how TMM handles clientside TCP connections using the F5BigTcpSetting CR’s `metadata.name` value.
`tcpSettings.serverSide`	Specifies how TMM handles serverside TCP connections using the F5BigTcpSetting CR’s `metadata.name` value.
`udpSettings.clientSide`	Specifies how TMM handles clientside UDP connections using the F5BigUdpSetting CR’s `metadata.name` value.
`udpSettings.serverSide`	Specifies how TMM handles serverside UDP connections using the F5BigUdpSetting CR’s `metadata.name` value.
`snat.type`	Specifies the type of address translation: **none** (default), **automap**, or **snat**.
`snat.pool`	When `snat.type` is **snat**, specifies the F5BigCneSnatpool CR to reference using the `spec.name` parameter.
`vlans.vlanList`	Specifies one or more F5BigNetVlan CRs using the `metadata.name` parameter, that listen for application traffic.

CR Example

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "2002::200:200:200:0/112"
   destinationPort: 0
   ipProtocol: "tcp"
   profile: "tcp"
   tcpSettings:
     clientSide: "tcp-high-bw-profile"
     serverSide: "tcp-high-bw-profile"
   vlans:
     vlanList:
       - "subscriber-vlan"
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigContextSecure CR shortName is **secctx**.

View CR instance:

```
kubectl get secctx -n <namespace>
```

View CR configuration:

```
kubectl get secctx -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

Use these steps to install the example F5BigContextSecure CR, and the ***optional*** CNFs F5BigTcpSetting CR. Each step offers a brief description of the example CR.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** The example F5BigTcpSetting CR increases a number of packets buffers to increase performance. Copy and paste the example into a YAML file:

   Note: *The F5BigTcpSetting CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigTcpSetting
metadata:
  name: "tcp-high-bw-profile"
  namespace: "cnf-gateway"
spec:
  sendBufferSize: 150000
  receiveWindowSize: 70000
  proxyBufferHigh: 20000
  proxyBufferLow: 5000
  idleTimeout: 150
  resetOnTimeout: false
```

2. Install the F5BigTcpSetting CR:

```
kubectl apply -f cnf-tcp-high-bw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigTcpSetting CRs were **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5TcpSetting",
TcpSetting cnf-gateway/tcp-high-bw-profile was added/updated
```

3. The example F5BigContextSecure CR listens for connections destined to IP addresses in the **2002::200:200:200:0/112** subnet, and only on the **subscriber-vlan** interface. The CR also references the F5BigTcpsettings profile. Copy and paste the example into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "2002::200:200:200:0/112"
   destinationPort: 0
   ipProtocol: "tcp"
   profile: "tcp"
   tcpSettings:
     clientSide: "tcp-high-bw-profile"
     serverSide: "tcp-high-bw-profile"
   vlans:
     vlanList:
       - "subscriber-vlan"
```

4. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-context.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-context was added/updated
```

5. Continue to the **Additional CRs** and **ContextSecure statistics** sections.

Additional CRs

The F5BigContextSecure CR can also reference these CNFs CRs:

- F5BigFwPolicy - Granular network packet filtering using access control lists.
- F5BigNatPolicy - Carrier-grade NAT (CG-NAT) functionality.
- F5BigDnsApp - High-performance DNS resolution, caching, and DNS64 translations.
- F5BigPePolicy - Intelligently control, steer, and optimize subscriber traffic.
- F5BigIpsPolicy - DNS packet inspection for protection against malignant network traffic.
- F5BigCneSnatpool - Provides TMMs with additional IP addresses for source IP address translation.
- F5BigLogProfile - Capture and send traffic processing events to remote logging servers.

ContextSecure statistics

If the [TMM Debug] sidecar is enabled (default), use the steps below to verify firewall filtering statics.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

   ```
   kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
   ```

2. Verify the F5BigContextSecure statistics:

   Clientside connections:

   ```
   tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
   ```

   ```
   name                                        clientside.tot_conns
   ------------------------------------------- --------------------
   cnf-gateway-cnf-context-SecureContext_vs                       8
   ```

   Serverside connections:

   ```
   tmctl -d blade virtual_server_stat -s name,serverside.tot_conns
   ```

   ```
   name                                        serverside.tot_conns
   ------------------------------------------- --------------------
   cnf-gateway-cnf-context-SecureContext_vs                       8
   ```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigDnsApp

Overview

The F5BigDnsApp Custom Resource (CR) configures the Traffic Management Microkernel (TMM) to provide high-performance DNS resolution, caching and DNS64 translation mapping. The F5BigDnsApp CR can also reference the F5BigIpsPolicy to intelligently protect applications from malignant network traffic, and the F5BigDnsCache CR to optimize DNS lookup performance with query caching.

This document guides you through understanding, configuring and installing a simple F5BigDnsApp, and the optional F5BigDnsCache, F5BigIpsPolicy and F5BigLogProfile CRs.

CR parameters

The tables below describe the F5BigDnsApp and F5BigDnsCache CR parameters used in this document.

F5BigDnsCache

The table below describes the F5BigDnsCache CR `spec` parameters used in this document. For the full list of parameters, refer to the F5BigDnsCache Reference.

Note: *DNS responses remain cached for the duration of the DNS record TTL.*

Parameter	Description
`cacheType`	The DNS cache type: **transparent**.
`transparent.localZones.name`	The Fully Qualified Domain Name for a localZone.
`transparent.localZones.zoneType`	The zone type for the localZone: **deny**, **refuse**, **static**, **transparent** (default), **type-transparent**, or **redirect**.
`transparent.localZones.records`	An array of records for this localZone.

F5BigDnsApp

The table below describes the F5BigDnsApp CR `spec` parameters used in this document. For the full list of parameters, refer to the F5BigDnsApp Reference.

Parameter	Description
`destination.address`	Specifies the IPv4 address used by clients to resolve DNS queries.
`destination.ipv6Address`	Specifies the IPv6 address used by clients to resolve DNS queries.
`destination.port`	Specifies the service port used to resolve DNS queries. The default is **53**.
`pool.members`	Specifies a list of endpoint DNS servers used to resolve DNS queries.
`pool.members.address`	Specifies an endpoint, or DNS server used to resolve DNS queries.
`pool.members.port`	Specifies the endpoint service port used to resolve DNS queries. The default value is **53**.
`logProfile`	Specifies the F5BigLogProfile to be used.
`dns.useLocalBind`	Enables local bind DNS server: **true** or **false** (default).

Parameter	Description
`dns.dnsCache`	Enables caching when referencing a F5BIGDnscache CR by `metadata.name`.
`monitors.dns.enabled`	Enables monitoring the `pool.members` availability: **true** or **false** (default).
`monitors.dns.queryName`	Specifies a fully qualified domain name the monitor sends in the DNS query probe.
`monitors.dns.queryType`	Specifies the type of DNS query to send type that the monitor sends in DNS query probe: **a** (default) or **aaaa**.
`monitors.dns.recv`	The IP address that the monitor looks for in the DNS server response to the DNS query probe.
`monitors.dns.icmp.enabled`	Enables sending ICMP probes to verify the `pool.members` availability: **true** or **false** (default).

CR Examples

F5BigDnsCache

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigDnsCache
metadata:
  name: "cnf-dnscache"
  namespace: "cnf-gateway"
spec:
  cacheType: transparent
  transparent:
    localZones:
      - name: example.com
        zoneType: static
        records:
          - example.com. IN AAAA 2002::10:11:12:13
```

F5BigDnsApp

```
apiVersion: "dns.k8s.f5net.com/v1"
kind: F5BigDnsApp
metadata:
  name: "cnf-dnsapp"
  namespace: "cnf-gateway"
spec:
  ipProtocol: "udp"
  destination:
   ipv6Address: "2002::192:168:100:201"
   port: 53
  snat:
   type: "automap"
  dns:
    dnsCache: "cnf-dnscache"
    dns64Mode: "secondary"
    dns64Prefix: "64:ff9b::"
```

```
    dns64AdditionalSectionRewrite: "v4-only"
  pool:
    members:
      - address: "2002::10:10:10:100"
      - address: "2002::10:10:10:101"
  monitors:
    dns:
      enabled: true
      queryName: "webapp.net."
      queryType: "aaaa"
      recv: "2002::10:10:20:200"
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigDnsApp and F5BigDnsCache CR shortNames are **dnsapp** and **dnscache** respectively.

View CR instance:

```
kubectl get dnsapp -n <namespace>
kubectl get dnscache -n <namespace>
```

View CR configuration:

```
kubectl get dnsapp -n <namespace> -o yaml
kubectl get dnscache -n <namespace> -o yaml
```

DNS Monitors

Prior to configuring and applying F5BigDnsApp monitors to Service endpoints, it is important to understand the CR's `timeout` and `interval` parameters, and their recommended configuration. The parameters behave as follows:

- `timeout` is ***only*** observed when it is ***less than*** the `interval`: Endpoints are marked down when unanswered probes exceed the configured `timeout`.
- `timeout` is ***not*** observed when it is ***greater than*** the `interval`: Endpoints are marked down when unanswered probes exceed the configured `interval`.

Note: *F5 recommends setting the `timeout` value to* **the same or less than**_ the `interval` value._

Requirements

Ensure you have:

- Installed the [CNF Controller] Pods.
- A Linux based workstation.

Installation

Use the following steps to install the F5BigDnsApp CR.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** To capture and send DNS and IPS events to remote logging servers, copy the example F5BigLogHslpub CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigLogHslpub
metadata:
  name: "cnf-hsl-pub"
  namespace: "cnf-gateway"
spec:
  pool:
  - name: hsl-pool
    endpoint:
    - "2002::192:168:10:200:514"
  syslog:
  - name: "cnf-syslog"
    distribution: "adaptive"
    format: "rfc5424"
    pool: "hsl-pool"
    protocol: "udp"
```

2. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

3. ***Optional:*** To define the type of DNS and IPs events to capture, copy the F5BigLogProfile CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "dns-log"
  publisher: "cnf-hsl-pub"
  dns:
    enabled: true
    publisher: "cnf-hsl-pub"
    responseLogging: true
    queryId: true
  protocolInspection:
    enabled: true
    publisher: "cnf-hsl-pub"
    logPacket: true
```

4. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

5. ***Optional:*** The example F5BigIpsPolicy CR rejects **SOA** record queries, and rejects **dns_named_version_attempt** and **dns_os_solaris_exploit_sparc_overflow_attempt** packet signatures. Copy and paste the CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigIpsPolicy
metadata:
  name: "cnf-ips"
  namespace: "cnf-gateway"
spec:
  services:
    - name: dns
      ports:
        - "53"
      compliances:
        - name: dns_disallowed_query_type
          valueType: string
          value: SOA
          action: reject
      signatures:
        - name: dns_named_version_attempt
          action: reject
        - name: dns_os_solaris_exploit_sparc_overflow_attempt
          action: reject
```

6. Install the F5BigIpsPolicy CR:

```
kubectl apply -f cnf-ips-policy.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigIpsPolicy CR was **added/updated**:*

```
I0208 12:00:00.12345   1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5ProtocolInspectionProfile",
F5ProtocolInspectionProfile cnf-gateway/cnf-ips was added/updated
```

7. ***Optional:*** Copy the F5BigDnsCache CR into a YAML file:

*In this example, the DNS cache creates an **AAAA** record, and returns authoritative DNS responses for the **example.com** domain.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigDnsCache
metadata:
  name: "cnf-dnscache"
  namespace: "cnf-gateway"
spec:
  cacheType: transparent
  transparent:
    localZones:
      - name: example.com
        zoneType: static
        records:
          - example.com. IN AAAA 2002::10:11:12:13
```

8. Install the F5BigDnsCache CR:

```
kubectl apply -f cnf-dnscache-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigDnsCache CR was **added/updated**:*

```
I0208 12:00:00.12345   1 event.go:282] Event(v1.ObjectReference{Kind:"F5Dnscache",
F5Dnscache cnf-gateway/cnf-dnscache was added/updated
```

9. Copy the F5BigDnsApp into a YAML file:

   *In the example below, clients can use **192.168.100.201** or **2002::192:168:100:201** as their DNS resolver IP address.*

```
apiVersion: "dns.k8s.f5net.com/v1"
kind: F5BigDnsApp
metadata:
  name: "cnf-dnsapp"
  namespace: "cnf-gateway"
spec:
  ipProtocol: "udp"
  destination:
   address: "192.168.100.201"
   ipv6Address: "2002::192:168:100:201"
   port: 53
  snat:
   type: "automap"
  dns:
    dnsCache: "cnf-dnscache"
    dns64Mode: "secondary"
    dns64Prefix: "64:ff9b::"
    dns64AdditionalSectionRewrite: "v4-only"
  udp:
    allowNoPayload: true
  pool:
    members:
      - address: "2002::10:10:10:100"
      - address: "2002::10:10:10:101"
  monitors:
    dns:
      enabled: true
      queryName: "webapp.net."
      queryType: "aaaa"
      recv: "2002::10:10:20:200"
    icmp:
      enabled: true
```

10. Install the F5BigDnsApp CR:

```
kubectl apply -f cnf-dnsapp-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigDnsApp CR was **added/updated**:*

```
I0208 12:00:00.12345   1 event.go:282] Event(v1.ObjectReference{Kind:"F5Dns",
F5Dns cnf-gateway/cnf-dnsapp was added/updated
```

Traffic statistics

If you installed the CNF Controller with the Debug Sidecar enabled, connect to the sidecar to view the DNS statistics.

1. Log in to the TMM Debug container:

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. View the IPS statstics:

```
tmctl -d blade protocol_inspection_stats
```

*In this example, IPS disallowed MX records has matched **11** times, blacklisted domains **3**:*

```
insp_id insp_name                       vs_name
------- ------------------------------- ---------------------------------------
  10007 dns_disallowed_resource_records cnf-gateway-cnf-dnsapp-virtual_server
  10009 dns_domains_blacklist           cnf-gateway-cnf-dnsapp-virtual_server

prof_name                                      hit_count last_hit_time
---------------------------------------------- --------- -------------
cnf-gateway-dns-ips-profileprotocolinspection         11    1645748374
cnf-dateway-dns-ips-profileprotocolinspection          3    1645748620
```

3. View the DNS caching statistics:

```
tmctl -d blade dns_cache_resolver_stat -s name,queries,responses,msg.hits,msg.inserts
```

*In this example, **55** queries have been process, and **7** domain names have been added to the DNS cache.*

```
name                           queries responses msg.hits msg.inserts
------------------------------ ------- --------- -------- -----------
cnf-gateway-cnf-dnscache            55        48       48           7
```

4. View the DNS resolution statistics:

```
tmctl -d blade profile_dns_stat -s name,vs_name,queries
```

In this example, 55 successful DNS queries have been processed:

```
name                               vs_name                                 queries
---------------------------------- --------------------------------------- -------
cnf-gateway-cnf-dnsapp-profile_dns cnf-gateway-cnf-dnsapp-virtual_server      55
```

Monitor status

When the F5BigDnsApp has a `monitor` configured, the Service Proxy TMM Pod logs pool member status change messages similar to the following:

```
kubectl logs -f f5-tmm-7599d547fc-g2zqd -n cnf-gateway | grep 'Pool Member Status'
```

```
f5-tmm-7599d547fc-g2zqd tmm[34]: 01010057:3: Pool Member Status Change: pool member
 ↪  2002::10:10:10:100:53 in cnf-gateway-cnf-dnsapp-pool is up\n"
f5-tmm-7599d547fc-g2zqd tmm[34]: 01010057:3: Pool Member Status Change: pool member
 ↪  2002::10:10:10:100:53 in cnf-gateway-cnfn-dnsapp-pool is down\n"
f5-tmm-7599d547fc-g2zqd tmm[34]: 01010057:3: Pool Member Status Change: pool member
 ↪  2002::10:10:10:100:53 in cnf-gateway-cnf-dnsapp-pool is up\n"
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Kubernetes Service

F5BigZeroratingirule

Overview

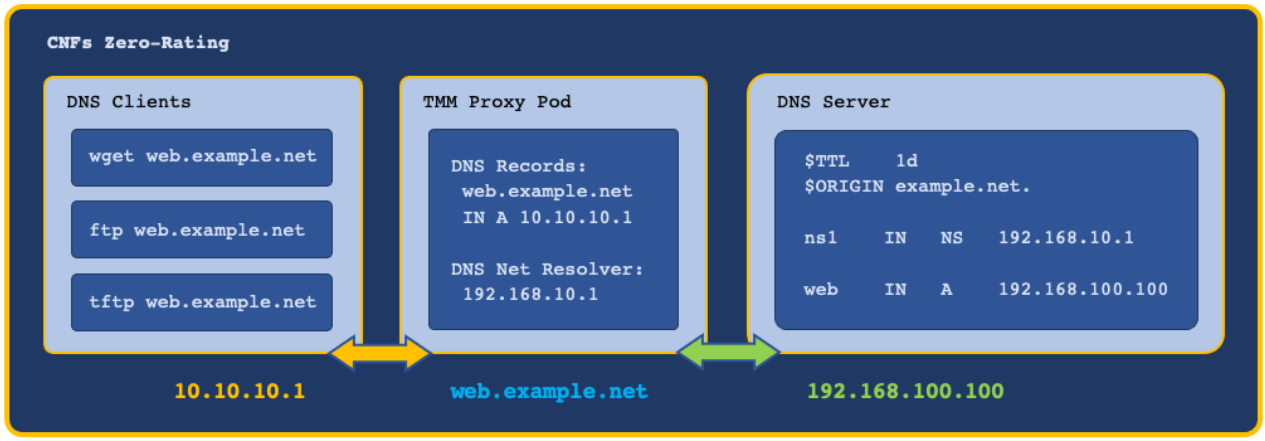
The F5BigZeroratingirule Custom Resource (CR) is an integral part of the Cloud-Native Network Functions (CNFs) Zero-Rating DNS solution; enabling subscriber access to applications with no impact, or increase, to billing quotas. CNFs Zero-Rating solution accomplishes this by ***steering*** application traffic using ***split-view DNS***. When configured as a DNS resolver, the Traffic Management Microkernel (TMM) Proxy Pod provides clientside DNS responses using custom IP addresses, bypassing quota management, then resolving and routing traffic to the DNS domain name on the server-side.

The CNFs Zero-Rating solution requires the CRs listed below:

- **F5BigDatagroup**
- **F5BigDnsCache**
- **F5BigDnsApp**
- **F5BigContextSecure**

This document guides you through understanding, configuring and installing a simple CNFs Zero-Rating solution.

Zero-Rating example:



CR parameters

F5BigZeroratingirule

The table below describes the F5BigZeroratingirule CR `spec` parameters.

Parameter	Description
`dnsResolver`	Specifies the F5BigDnsCache by `metadata.name` that will be used by TMM will resolve and cache external DNS queries.
`dataGroup`	Specifies the F5BigDatagroup CR by `metadata.name` that will provide IP address to domain name mapping.
`debugZeroRating`	Enables zero rating iRule logging: **true** or **false** (default).

F5BigDatagroup

The table below describes the F5BigDatagroup CR `spec` parameters.

Parameter	Description
`recordType`	Specifies the type of the datagroup record. The current available option is **address**.
`records`	Specifies a list of **key** and **value** pairs. The **key** represents an IPv4 or IPv6 address, and the **value** represents the domain name.

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigContextSecure CR shortName is **secctx**.

View CR instance:

```
kubectl get secctx -n <namespace>
```

View CR configuration:

```
kubectl get secctx -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the [CNF Controller] Pods.
- A Linux based workstation.

Installation

Use the following steps to install the F5BigDnsApp and F5BigDnsCache CRs that will create a DNS record in the transparent cache for the domain **example.com**..

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. The F5BigDatagroup CR defines the clientside domain name to IP address mappings. Copy the example CR into a YAML file:

   Note: *The CR provides IPv4 and IPv6 clientside mappings for the* ***web.example.net*** *domain name.*

   ```
   apiVersion: "k8s.f5net.com/v1"
   kind: F5BigDatagroup
   metadata:
     name: "cnf-datagroup"
     namespace: "cnf-gateway"
   spec:
     recordType: "address"
     records: [
        { "key": "10.20.2.37",
   ```

```
      "value": "web.example.net"
    },
    { "key": "2002::10:20:2:37",
      "value": "v6web.example.net"
    }
  ]
```

2. The **transparent** F5BigDnsCache CR defines the DNS record types for the clientside mappings. Copy the example **transparent** F5BigDnsCache CR into a YAML file:

   Note: *The CR defines the DNS **A** and **AAAA** DNS record types for the clientside mappings.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigDnsCache
metadata:
  name: "cnf-transparent-cache"
  namespace: "cnf-gateway"
spec:
  cacheType: transparent
  transparent:
    localZones:
    - name: example.net
      zoneType: static
      records:
        - web.example.net. IN A 10.20.2.37
        - v6web.example.net. IN AAAA 2002::10:20:2:37
```

3. Install the F5BigDatagroup CR:

```
kubectl apply -f cnf-datagroup-cr.yaml
```

   *In this example, the BIG-IP Controller logs indicate the F5BigDatagroup CR was **added/updated**:*

```
I0223 12:00:00.12345  1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:\"F5BigDatagroup\",
F5BigDatagroup cnf-gateway/cnf-datagroup was added/updated
```

4. Install the **transparent** F5BigDnsCache CR:

```
kubectl apply -f cnf-transparent-dns.yaml
```

   *In this example, the BIG-IP Controller logs indicate the F5BigDnsCache CR was **added/updated**:*

```
I0208 12:00:00.12345   1 event.go:282] Event(v1.ObjectReference{Kind:"F5Dnscache",
F5Dnscache cnf-gateway/cnf-transparent-cache was added/updated
```

5. The **net-resolver** F5BigKDnsCache CR defines both a domain name, and the domain name server to query. Copy one of the example F5BigDnsCache CRs into a YAML file: *Example 1* queries and caches all domains, while *Example 2* queries and caches two specific domains:

   Example 1:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigDnsCache
metadata:
  name: cnf-resolver-cache
  namespace: cnf-gateway
spec:
```

```
  cacheType: net-resolver
  netResolver:
    forwardZones:
      - forwardZone: .
        nameServers:
          - ipAddress: 10.30.2.1
            port: 53
```

Example 2:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigDnsCache
metadata:
  name: cnf-resolver-cache
  namespace: cnf-gateway
spec:
  cacheType: net-resolver
  netResolver:
    forwardZones:
      - forwardZone: example.net
        nameServers:
          - ipAddress: 10.30.2.1
            port: 53
      - forwardZone: internal.org
        nameServers:
          - ipAddress: 10.10.10.1
            port: 53
```

6. Install the **net-resolver** F5BigDnsCache CR:

```
kubectl apply -f cnf-netresolv-dns.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigDnsCache CR was **added/updated**:*

```
I0208 12:00:00.12345   1 event.go:282] Event(v1.ObjectReference{Kind:"F5Dnscache",
F5Dnscache cnf-gateway/cnf-resolver-cache was added/updated
```

7. The F5BigZeroratingirule CR groups the clientside domain name mappings and the serverside DNS resolver. Copy the example F5BigZeroratingirule CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigZeroratingirule
metadata:
  name: "cnf-zerorate-irule"
  namespace: "cnf-gateway"
spec:
  dnsResolver: "cnf-resolver-cache"
  dataGroup: "cnf-datagroup"
  debugZeroRating: true
```

8. Install the F5BigZeroratingirule CR:

```
kubectl apply -f cnf-zerorate-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigZeroratingirule CR was **added/updated**:*

```
I0223 12:00:00.12345  1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:\"F5BigZeroratingirule\",
F5SPKZERORATINGIRULE cnf-gateway/cnf-zerorate-irule was added/updated
```

9. The F5BigDnsApp CR specifies the IP address that subscribers will use to resolve DNS. The CR also references the **transparent** F5BigDnsCache CR. Copy the example F5BigDnsApp CR into a YAML file:

```
apiVersion: "dns.k8s.f5net.com/v1"
kind: F5BigDnsApp
metadata:
  name: "cnf-dnsapp"
  namespace: "cnf-gateway"
spec:
  destination:
    address: "10.20.22.94"
    port: 53
  ipProtocol: "udp"
  dns:
    useLocalBind: false
    dnsCache: "cnf-transparent-cache"
  udp:
    allowNoPayload: true
```

10. Install the F5BigDnsApp CR:

```
kubectl apply -f cnf-dnsapp-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigDnsApp CR was **added/updated**:*

```
I0224 12:00::00.12345  1 event.go:282] Event(v1.ObjectReference{Kind:\"F5Dns\",
F5Dns cnf-gateway/cnf-dnsapp was added/updated
```

11. The F5BigContextSecure CR will process subscriber application traffic. The CR also references the F5BigZeroratingirule CR. Copy the example F5BigContextSecure CR into a YAML file:

***Important:** Set the `destinationAddress` and `ipv6destinationAddress` to the IP address subnets used in the DNS mappings.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigContextSecure
metadata:
  name: "cnf-secure-context"
  namespace: "cnf-gateway"
spec:
  destinationAddress: "10.20.2.37/32"
  ipv6destinationAddress: "2002::10:20:2:37/128"
  destinationPort: 80
  ipProtocol: "tcp"
  profile: "tcp"
  iRules: [ "cnf-zerorate-irule" ]
```

12. Install the F5BigContextSecure CR:

```
kubectl apply -f cnf-context-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0224 12:00::00.12345  1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:\"F5SecureContext\",
SecureContext cnf-gateway/cnf-secure-context was added/updated
```

Traffic statistics

If you installed the CNF Controller with the Debug Sidecar enabled, connect to the sidecar to view the DNS statistics.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

   ```
   kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
   ```

2. Verify the DNS statistics:

   ```
   tmctl -d blade dns_cache_resolver_stat -s
    ↪  name,cache_index,queries,responses,responses_rate
   ```

   ```
   name                              cache_index queries responses responses_rate
   --------------------------------- ----------- ------- --------- --------------
   cnf-gateway-cnf-resolver-cache              0       9         9              0
   cnf-gateway-cnf-transparent-cache           0      13        13              0
   ```

3. Verify the Application traffic statstics:

   ```
   tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
   ```

   ```
   name                                            clientside.tot_conns
   ----------------------------------------------- --------------------
   cnf-gateway-cnf-secure-context-SecureContext_vs                    5
   cnf-gateway-cnf-secure-context-SecureContext_vs                    4
   cnf-gateway-big-dns-virtual_server                                 9
   ```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigLogProfile

Overview

The F5BigLogProfile Custom Resource (CR) is used to specify traffic processing events that the Traffic Management Microkernel (TMM) should capture, and send to one or more remote logging servers. The F5BigLogProfile specifically handles events that occur when TMM is processing traffic with any of the Protection or NAT [CNF CRs]. The F5BigLogHslpub CR specifies the remote logging destionation IP address and service port, and the logging format. The F5BigLogHslpub CR should be configured and installed first, and then referenced in the F5BigLogProfile CR configuration.

This document guides you through understanding, configuring and installing a simple F5BigLogHslPub and F5BigLogProfile CR.

CR parameters

F5BigLogHslpub

The table below describe the CR `spec` parameters. Configure and install the F5BigLogHslpub CR prior to installing the F5BigLogProfile CR.

Parameter	Description
`loadBalancingMethod`	Defines the load balancing mode used to distribute traffic across multiple pool members: **ROUND_ROBIN** (default), or **RATIO_LEAST_CONN_MEMBER**
`pool.name`	A user defined name for the HSL logging pool.
`pool.endpoint`	Specifies a list of one or more IP address and service ports for logging endpoint(s).
`syslog.name`	A user defined name for the syslog configuration.
`syslog.format`	Specifies the logging format: **rfc5424** (default), **rfc3164**, or **legacy-bigip**.
`syslog.protocol`	Specifies the protocol to use when connecting to the logging endpoint: **udp** (default) or **tcp**.
`syslog.distribution`	Specifies the distribution method used to send messages to pool members: **adaptive** (default) - connections to pool members are added as required to provide enough logging bandwidth. This can have an undesirable effect of logs accumulating on only one pool member when it provides sufficient logging bandwidth on its own. **balanced** - sends each successive log to a new pool member, balancing the logs among them according to the pool's load balancing method. **replicate** - replicates each log to all pool members, for redundancy.
`syslog.pool`	Specifies a pool of logging enpoints using the `spec.pool.name` value.

F5BigLogProfile

The table below describes only the parameters used in this document. For the full list of CR parameters, refer to the F5BigLogProfile Reference.

nat

Parameter	Description
`enabled`	Enables firewall NAT events: **true** or **false** (default).
`outbound.start.mode`	Enables event log entries at start of the translation event for a NAT client: **disabled** (default), **enabled**, and **backup**.
`outbound.start.includeDestAddrPort`	Include the destination IP address and port in the log message: **true** (default) or **false**.
`outbound.end.mode`	Enables event log entries at end of translation event for a NAT client: **disabled** (default), **enabled**, or **backup**.
`outbound.end.includeDestAddrPort`	Include the destination IP address and port in the log message: **true** (default) or **false**.
`inbound.start.mode`	Enables log entries at the start of the incoming connection event for a translated endpoint: **disabled** (default), **enabled**, or **backup**.
`errors.mode`	Enables event log entries when a NAT translation errors occur: **disabled** (default), **enabled**, or **backup**.
`publisher`	Specifies the name of the log publisher used for logging Network Address Translation events.

firewall

Parameter	Description
`enabled`	Enables logging of firewall event messages: **true** or **false** (default).
`trafficStats.activeFlows`	Enables logging the number of active flows on client side: **true** or **false**.
`trafficStats.reapedFlows`	Enables logging the number of reaped flows on client side: **true** or **false** (default).
`trafficStats.missedFlows`	Enables logging the number of TCP packets (non SYN/ACK) were dropped because of the flow table lookup failed: **true** or **false** (default).
`trafficStats.publisher`	Specifies the name of the log publisher to be used for trafficStats log messages.
`network.events.aclMatchAccept`	Enables logging the packets that match ACL rules configured when the F5BigFwPolicy `action` parameter is set to **accept**: **true** or **false** (default).
`network.events.aclMatchDrop`	Enables logging the packets that match when the F5BigFwPolicy `action` parameter is set to **drop**: = Drop: **true** or **false** (default).
`network.events.aclMatchReject`	Enables logging the packets that match when the F5BigFwPolicy `action` parameter is set to **reject**: **true** or **false** (default).
`network.events.translationFields`	Enables logging the translated server side fields including the Source Address/Port, Destination Address/Port, IP Protocol, and Vlan: **true** or **false** (default).
`network.publisher`	Specifies the name of the log publisher to be used for network log messages.

CR Examples

F5BigLogHslpub

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogHslpub
metadata:
  name: "cnf-hsl-pub"
  namespace: "cnf-gateway"
spec:
  pool:
    - name: "pool1"
      endpoint:
       - "10.10.10.100:514"
  syslog:
  - name: "syslog1"
    protocol: "tcp"
    distribution: "adaptive"
    pool: "pool1"
```

F5BigLogProfile

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  firewall:
    enabled: true
    trafficStats:
      publisher: "cnf-hsl-pub"
      reapedFlows: true
      missedFlows: true
      activeFlows: true
    network:
      publisher: "cnf-hsl-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
        tcpEvents: true
        aclToBoxDeny: true
       translationFields: true
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigLogProfile and F5BigLogHslpub CR shortNames are **logprof** and **hslpub** respectively.

View CR instance:

```
kubectl get logprof -n <namespace>
kubectl get hslpub -n <namespace>
```

View CR configuration:

```
kubectl get logprof -n <namespace> -o yaml
kubectl get hslpub -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller Pods.
- A Linux based workstation.

Installation

The F5BigLogHslpub defines the remote logging endpoints that receive logging data, and the F5BigLogProfile defines the types of events to capture and send. Use the following steps to configure and install both of the logging CRs.

1. Install the F5BigLogHslpub CR:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogHslpub
metadata:
  name: "cnf-hsl-pub"
  namespace: "cnf-gateway"
spec:
  pool:
    - name: "pool1"
      endpoint:
       - "10.10.10.100:514"
       - "10.10.10.101:514"
  syslog:
  - name: "syslog1"
    format: "rfc5424"
    protocol: "tcp"
    distribution: "adaptive"
    pool: "pool1"
```

2. Install the F5BigLogProfile CR:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  nat:
    enabled: true
    logSubscriberID: true
    publisher: "cnf-hsl-pub"
    inbound:
```

```
      start:
        mode: "enabled"
      end:
        mode: "enabled"
    quotaExceeded:
      mode: "enabled"
    errors:
      mode: "enabled"
  firewall:
    enabled: true
    trafficStats:
      publisher: "cnf-hsl-pub"
      reapedFlows: true
      missedFlows: true
      activeFlows: true
    network:
      publisher: "cnf-hsl-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
        tcpEvents: true
        aclToBoxDeny: true
       translationFields: true
```

3. Continue to the **Next step** section to begin using the F5BigLogProfile.

Next step

Select and install one of the Traffic Management [CNF CRs] to begin processing application traffic:

- F5BigContextSecure - Full proxy TCP and UDP application layer gateway services.
- F5BigAlgFtp - File Transfer Protocol (FTP) application layer gateway services.
- F5BigAlgTftp - Trivial File Transfer Protocol (TFTP) application layer gateway services.
- F5BigAlgPptp - Point-to-Point Tunneling Protocol (PPTP) application layer gateway services.
- F5BigAlgRtsp - Real Time Streaming Protocol (RTSP) application layer gateway services.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigAlgFtp

Overview

The F5BigAlgFtp Custom Resource (CR) configures the Traffic Management Microkernel (TMM) to provide File Transfer Protocol (FTP) application layer gateway (ALG) services. The F5BigAlgFtp CR supports both active and passive modes.

This document guides you through configuring and installing a simple F5BigAlgFtp CR.

CR Parameters

The table below describe the F5BigAlgFtp CR `spec` parameters:

Parameter	Description
`destinationAddress`	The destination IPv4 address of the application.
`ipv6destinationAddress`	The destination IPV6 address of the application.
`destinationPort`	The destination service port of the application. The default value is **21**.
`natPolicy`	Specifies an F5BigNatPolicy CR using the `metadata.name` value.
`firewallEnforcedPolicy`	Specifies an F5BigFwPolicy CR using the `metadata.name` value.
`logProfile`	Specifies an F5BigLogProfile CR using the `metadata.name` value.
`idleTimeout`	Specifies the number of seconds that a connection can remain idle before it is eligible for deletion: **0** - **4294967295**. The default is **300**.
`vlans.vlanList`	Specify a list of VLANs to accept traffic on. Traffic can also be denied using the `disableListedVlans` parameter.
`vlans.disableListedVlans`	Denies traffic specified with the `vlanList` parameter: `true` (default) or `false`.
`tcpSettings.clientSide`	Specifies how TMM handles clientside TCP connections using the F5BigTcpSetting CR's `metadata.name` value.
`tcpSettings.serverSide`	Specifies how TMM handles serverside TCP connections using the F5BigTcpSetting CR's `metadata.name` value.
`irules`	Reference to a list of iRules.
`ftpSession.translateExtended`	Enables automatically translating RFC2428 extended requests EPSV and EPRT to PASV and PORT when talking to IPv4 servers: **true** (default) or **false**.
`ftpSession.inheritParentProfile`	When enabled, the data channel inherits the TCP profile used by the control channel. When disabled, the data channel uses FastL4 only: **true** (default) or **false**.
`ftpSession.dataPort`	Specifies data channel port for the FTP profile. The default value is **20**.
`ftpSession.allowActiveMode`	Enables FTP Active Transfer mode: **true** (default) or **false**.
`ftpSession.allowFtpsMode`	Specifies the FTPS mode field to use: **DISALLOW** (default) or **ALLOW**.

Parameter	Description
`ftpSession.createWildCardVS`	Creates a wild card virtual server (0.0.0.0/0) on TMM: **true** or **false** (default). This parameter is only used when `ftpsMode` is set to **ALLOW**.

CR Example

```
apiVersion: k8s.f5net.com/v1
kind: F5BigAlgFtp
metadata:
  name: "cnf-ftp"
  namespace: "cnf-gateway"
spec:
   destinationAddress: "0.0.0.0/0"
   ipv6destinationAddress: "::/0"
   destinationPort: 21
   logProfile: "cnf-log-profile"
   firewallEnforcedPolicy: "cnf-firewall-policy"
   natPolicy: "cnf-nat-policy"
   tcpSettings:
     clientSide: "cnf-tcp-profile"
     serverSide: "cnf-tcp-profile"
   ftpSession:
     translateExtended: true
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigAlgFtp CR shortName is **algftp**.

View CR instance:

```
kubectl get algftp -n <namespace>
```

View CR configuration:

```
kubectl get algftp -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

This installation procedure provides ***Optional*** examples, representing each of the CNFs Custom Resources (CRs) that can be referenced by the F5BigAlgFtp CR. Use the steps below to configure TMM:

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** Copy the example F5BigNatPolicy CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNatPolicy
metadata:
  name: "cnf-nat-policy"
  namespace: "cnf-gateway"
spec:
  sourceTranslation:
    - name: "dynamic-trans"
      type: "dynamic-pat"
      addresses:
        - "10.200.2.1-10.200.2.10"
      port: "8000-8500"
      patMode: "napt"
      inbound:
        eifTimeout: 200
        mode: "endpoint-independent-filtering"
      mapping:
        mode: "endpoint-independent-mapping"
        timeout: 60
      routeAdvertisement: true
  rule:
    - name: 10-20-subnet
      ipProtocol: tcp
      source:
        addresses:
         - "10.20.2.0/24"
      sourceTranslation: "dynamic-trans"
```

2. Install the F5BigNatPolicy CR:

```
kubectl apply -f cnf-nat-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigNatPolicy CR was **added/updated**:*

```
I0202 12:00:00.12345   1 event.go:282 Event(v1.ObjectReference{Kind:"F5NatPolicy",
NatPolicy cnf-gateway/cnf-nat-policy was added/updated
```

3. ***Optional:*** Copy the example F5BigFwPolicy into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFwPolicy
metadata:
  name: "cnf-fw-policy"
  namespace: "cnf-gateway"
spec:
  rule:
    - name: allow-10-30-ftp
      action: "accept"
      logging: true
```

```
      ipProtocol: tcp
      source:
        addresses:
          - "10.30.2.0/24"
      destination:
        ports:
          - "20"
          - "21"
```

4. Install the F5BigFwPolicy CR:

```
kubectl apply -f cnf-fw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigFwPolicy CR was **added/updated**:*

```
I0202 12:00:00.12346   1 event.go:282
 ↪  Event(v1.ObjectReference{Kind:"F5FirewallPolicy",
FirewallPolicy cnf-gateway/cnf-fw-policy was added/updated
```

5. ***Optional:*** Copy the example F5BigLogHslpub CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigLogHslpub
metadata:
  name: "cnf-hsl-pub"
  namespace: "cnf-gateway"
spec:
  pool:
  - name: hsl-pool
    endpoint:
    - "10.30.2.220:514"
  syslog:
  - name: "cnf-syslog"
    distribution: "adaptive"
    format: "rfc5424"
    pool: "hsl-pool"
    protocol: "udp"
```

6. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

7. ***Optional:*** Copy the example F5BigLogProfile CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-logs"
  nat:
```

```
    enabled: true
    logSubscriberID: true
    publisher: "cnf-hsl-pub"
    inbound:
      start:
        mode: "enabled"
      end:
        mode: "enabled"
    quotaExceeded:
      mode: "enabled"
    errors:
      mode: "enabled"
  firewall:
    enabled: true
    trafficStats:
      publisher: "cnf-hsl-pub"
      reapedFlows: true
      missedFlows: true
      activeFlows: true
    network:
      publisher: "cnf-hsl-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
        tcpEvents: true
        aclToBoxDeny: true
        translationFields: true
```

8. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

9. ***Optional:*** Copy the FF5BigTcpSetting CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigTcpSetting
metadata:
  name: "cnf-tcp-profile"
  namespace: "cnf-gateway"
spec:
  nagle: true
  pushFlag: auto
  earlyRetransmit: false
```

10. Install the F5BigTcpSetting CR:

```
kubectl apply -f cnf-tcp-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigTcpSetting CR was **added/updated**:*

```
I0202 12:00:00.12349   1 event.go:282 Event(v1.ObjectReference{Kind:"F5TcpSetting",
TcpSetting cnf-gateway/tcp-client was added/updated
```

11. Copy the F5BigAlgFtp CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigAlgFtp
metadata:
  name: "cnf-ftp"
  namespace: "cnf-gateway"
spec:
   destinationAddress: "0.0.0.0/0"
   ipv6destinationAddress: "::/0"
   destinationPort: 21
   logProfile: "cnf-log-profile"
   firewallEnforcedPolicy: "cnf-firewall-policy"
   natPolicy: "cnf-nat-policy"
   tcpSettings:
     clientSide: "cnf-tcp-profile"
     serverSide: "cnf-tcp-profile"
   ftpSession:
     translateExtended: true
```

12. Install the F5BigAlgFtp CR:

```
kubectl apply -f f5-cnf-ftp.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigAlgFtp CR was **added/updated**:*

```
I0203 12:00:00:1245    1 event.go:282] Event(v1.ObjectReference{Kind:"F5FTP",
F5FTP cnf-gateway/cnf-ftp was added/updated
```

Traffic statistics

If you have installed the [TMM Debug] container, use the following steps to gather traffic processing statistics for the F5BigNatPolicy, F5BigFwPolicy and F5BigContextSecure CRs.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify F5BigNatPolicy connection statistics:

```
tmctl -d blade fw_nat_rule_stat
```

```
context_type context_name                          rule_name
------------ ------------------------------------  -------------------------
virtual      cnf-cnf-context-secure-SecureContext_vs  10-20-subnet-natpolicyrule

micro_rules counter last_hit_time action
----------- ------- ------------- ------
          1       8    1643836695      0
```

```
tmctl -d blade fw_nat_trans_stat -s type,name,translation_requests
```

```
bash type          name         translation_requests ------------ -----------
-------------------- fw_src_trans transparent                  0 fw_dst_trans
```

```
transparent                    8 fw_src_trans automap                    0##
```

Traffic statistics

If you have installed the [TMM Debug] container, use the following steps to gather traffic processing statistics for the F5BigNatPolicy, F5BigFwPolicy and F5BigContextSecure CRs.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify F5BigNatPolicy connection statistics:

```
tmctl -d blade fw_nat_rule_stat
```

```
context_type context_name                          rule_name
------------ ------------------------------------  --------------------------
virtual   cnf-cnf-context-secure-SecureContext_vs  10-20-subnet-natpolicyrule

micro_rules counter last_hit_time action
----------- ------- ------------- ------
          1       8    1643836695      0
```

```
tmctl -d blade fw_nat_trans_stat -s type,name,translation_requests
```

```
type         name        translation_requests
------------ ----------- --------------------
fw_src_trans transparent                    0
fw_dst_trans transparent                    8
fw_src_trans automap                        0
```

3. Verify the F5BigNatPolicy client IP address mappings:

```
lsndb list all
```

```
Client                                                Connections
------------------------------------------------------------
0 client with 0 connection found.
LSN Persistence Entries
Client                                                Translation
------------------------------------------------------------
10.20.2.220:52110                               10.200.2.8:8265
10.20.2.220                                     10.200.2.8
2 persist entries found.
LSN port block allocations
Client                                                Port block
------------------------------------------------------------
0 port block entries found.
LSN Inbound Mapping Entries
Translation                                             Client
------------------------------------------------------------
10.200.2.8:8265                              10.20.2.220:52110
10.200.2.7:8397                              10.20.2.220:52106
```

4. Verify the F5BigFwPolicy statistics:

```
tmctl -d blade fw_rule_stat
```

```
context_type context_name
------------ ----------------------------------------
virtual      cnf-gateway-cnf-fw-policy-SecureContext_vs

rule_name                           micro_rules counter last_hit_time action
----------------------------------- ----------- ------- -------------- ------
allow-10-20-http-firewallpolicyrule           1       2     1638572860      2
allow-10-30-ftp-firewallpolicyrule            1       5     1638573270      2
```

5. Verify the F5BigAlgFtp statistics:

```
tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
```

```
name                                          clientside.tot_conns
--------------------------------------------- --------------------
cnf-gateway-cnf-ftp-ftp-virtual-server        7
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- CNFs CRs.

F5BigAlgTftp

Overview

The F5BigAlgTftp Custom Resource (CR) configures the Traffic Management Microkernel (TMM) to provide secure File Transfer Protocol (FTP) application layer gateway (ALG) services.

This document guides you through understanding, configuring and installing a simple F5BigAlgTftp CR.

CR Parameters

The table below describes the CR `spec` parameters:

Parameter	Description
`destinationAddress`	The destination IPv4 address of the application.
`ipv6destinationAddress`	The destination IPv6 address of the application.
`destinationPort`	The destination server port of the application. The default is **69**.
`natPolicy`	Specifies a NAT Policy by `metadata.name`.
`firewallEnforcedPolicy`	Specifies a Firewall ACL Policy by `metadata.name`.
`logProfile`	Specifies a Log Profile by `metadata.name`.
`tftpSession.idleTimeout`	Specifies the number of seconds that a connection can remain idle before deletion: **1** to **4294967295**. The default is **30**.
`vlans.vlanList`	Specify a list of VLANs to accept traffic on. Traffic can also be denied using the `disableListedVlans` parameter.
`vlans.disableListedVlans`	Denies traffic specified with the `vlanList` parameter: `true` (default) or `false`.

CR Example

```
apiVersion: k8s.f5net.com/v1
kind: F5BigAlgTftp
metadata:
  name: "cnf-tftp"
  namespace: "cnf-gateway"
spec:
   destinationAddress: "0.0.0.0/0"
   ipv6destinationAddress: "::/0"
   destinationPort: 69
   natPolicy: "cnf-nat-policy"
   logProfile: "cnf-log-profile"
   firewallEnforcedPolicy: "cnf-fw-policy"
   tftpSession:
     idleTimeout: 35
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigAlgTftp CR shortName is **algtftp**.

View CR instance:

```
kubectl get algtftp -n <namespace>
```

View CR configuration:

```
kubectl get algtftp -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

This installation procedure provides ***Optional*** examples, representing each of the CNFs Custom Resources (CRs) that can be referenced by the F5BigAlgTftp CR. Use the steps below to configure TMM:

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** Copy the example F5BigNatPolicy CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNatPolicy
metadata:
  name: "cnf-nat-policy"
  namespace: "cnf-gateway"
spec:
  sourceTranslation:
    - name: "dynamic-trans"
      type: "dynamic-pat"
      addresses:
        - "10.200.2.1-10.200.2.10"
      port: "8000-8500"
      patMode: "napt"
      inbound:
        eifTimeout: 200
        mode: "endpoint-independent-filtering"
      mapping:
        mode: "endpoint-independent-mapping"
        timeout: 60
      routeAdvertisement: true
  rule:
    - name: 10-20-subnet
      ipProtocol: udp
      source:
        addresses:
```

```
        - "10.20.2.0/24"
    sourceTranslation: "dynamic-trans"
```

2. Install the F5BigNatPolicy CR:

```
kubectl apply -f cnf-nat-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigNatPolicy CR was **added/updated**:*

```
I0202 12:00:00.12345   1 event.go:282 Event(v1.ObjectReference{Kind:"F5NatPolicy",
NatPolicy cnf-gateway/cnf-nat-policy was added/updated
```

3. ***Optional:*** Copy the example FF5BigFwPolicy CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFwPolicy
metadata:
  name: "cnf-fw-policy"
  namespace: "cnf-gateway"
spec:
  rule:
    - name: allow-10-30-tftp
      action: "accept"
      logging: true
      ipProtocol: udp
      source:
        addresses:
          - "10.30.2.0/24"
      destination:
        ports:
          - "69"
```

4. Install the F5BigFwPolicy CR:

```
kubectl apply -f cnf-fw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigFwPolicy CR was **added/updated**:*

```
I0202 12:00:00.12346   1 event.go:282
 ↪  Event(v1.ObjectReference{Kind:"F5FirewallPolicy",
FirewallPolicy cnf-gateway/cnf-fw-policy was added/updated
```

5. ***Optional:*** Copy the example F5BigLogHslpub CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigLogHslpub
metadata:
  name: "cnf-pub"
  namespace: "cnf-gateway"
spec:
  pool:
  - name: hsl-pool
    endpoint:
    - "10.30.2.220:514"
  syslog:
  - name: "cnf-syslog"
    distribution: "adaptive"
```

```
    format: "rfc5424"
    pool: "hsl-pool"
    protocol: "udp"
```

6. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

7. ***Optional:*** Copy the example F5BigLogProfile CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-logs"
  nat:
    enabled: true
    logSubscriberID: true
    publisher: "cnf-pub"
    inbound:
      start:
        mode: "enabled"
      end:
        mode: "enabled"
    quotaExceeded:
      mode: "enabled"
    errors:
      mode: "enabled"
  firewall:
    enabled: true
    trafficStats:
      publisher: "cnf-pub"
      reapedFlows: true
      missedFlows: true
      activeFlows: true
    network:
      publisher: "cnf-pub-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
        aclToBoxDeny: true
        translationFields: true
```

8. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-log-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

9. Copy the F5BigAlgTftp CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigAlgTftp
metadata:
  name: "cnf-tftp"
  namespace: "cnf-gateway"
spec:
   destinationAddress: "0.0.0.0/0"
   ipv6destinationAddress: "::/0"
   destinationPort: 69
   natPolicy: "cnf-nat-policy"
   logProfile: "cnf-log-profile"
   firewallEnforcedPolicy: "cnf-fw-policy"
   tftpSession:
     idleTimeout: 35
```

10. Install the F5BigAlgTftp CR:

```
kubectl apply -f f5-cnf-tftp.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigAlgTftp CR was **added/updated**:*

```
I0203 12:00:00.1234519   1 event.go:282] Event(v1.ObjectReference{Kind:"F5TFTP",
F5TFTP cnf-gatway/cnf-tftp was added/updated
```

Traffic statistics

If you have installed the [TMM Debug] container, use the following steps to gather traffic processing statistics for the F5BigNatPolicy, F5BigFwPolicy and F5BigContextSecure CRs.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify F5BigNatPolicy connection statistics:

```
tmctl -d blade fw_nat_rule_stat
```

```
context_type context_name                          rule_name
------------ ------------------------------------  --------------------------
virtual   cnf-cnf-context-secure-SecureContext_vs  10-20-subnet-natpolicyrule

micro_rules counter last_hit_time action
----------- ------- ------------- ------
          1       8    1643836695      0
```

```
tmctl -d blade fw_nat_trans_stat -s type,name,translation_requests
```

```
type         name        translation_requests
------------ ----------- --------------------
```

```
fw_src_trans transparent                    0
fw_dst_trans transparent                    8
fw_src_trans automap                        0
```

3. Verify the F5BigNatPolicy client IP address mappings:

```
lsndb list all
```

```
Client                                              Connections
-------------------------------------------------------------
0 client with 0 connection found.
LSN Persistence Entries
Client                                              Translation
-------------------------------------------------------------
10.20.2.220:52110                                10.200.2.8:8265
10.20.2.220                                      10.200.2.8
2 persist entries found.
LSN port block allocations
Client                                              Port block
-------------------------------------------------------------
0 port block entries found.
LSN Inbound Mapping Entries
Translation                                            Client
-------------------------------------------------------------
10.200.2.8:8265                             10.20.2.220:52110
10.200.2.7:8397                             10.20.2.220:52106
```

4. Verify the F5BigFwPolicy statistics:

```
tmctl -d blade fw_rule_stat
```

```
context_type context_name
------------ ------------------------------------------
virtual      cnf-gateway-cnf-fw-policy-SecureContext_vs

rule_name                            micro_rules counter last_hit_time action
------------------------------------ ----------- ------- ------------- ------
allow-10-30-tftp-firewallpolicyrule            1       5    1638573270      2
```

5. Verify the F5BigAlgFtp statistics:

```
tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
```

```
name                                               clientside.tot_conns
-------------------------------------------------- --------------------
cnf-gateway-cnf-tftp-alg-virtual-server             7
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigAlgRtsp

Overview

The F5BigAlgRtsp Custom Resource (CR) configures the Traffic Management Microkernel (TMM) to provide secure Real Time Streaming Protocol (RTSP) application layer gateway (ALG) services. The F5BigAlgRtsp CR supports both active and passive modes.

This document guides you through understanding, configuring and installing a simple F5BigAlgRtsp CR.

CR Parameters

The table below describe the CR `spec` parameters.

Parameter	Description
`destinationAddress`	The externally-facing destination IP address of the application.
`ipv6destinationAddress`	The externally-facing IPV6 destination IP address of the application.
`destinationPort`	The externally-facing destination port address of the application. The default is **554**.
`natPolicy`	Specifies a NAT Policy by `metadata.name`.
`firewallEnforcedPolicy`	Specifies a Firewall ACL Policy by `metadata.name`.
`logProfile`	Specifies a Log Profile by `metadata.name`.
`vlans.vlanList`	Specify a list of VLANs to accept traffic on. Traffic can also be denied using the `disableListedVlans` parameter.
`vlans.disableListedVlans`	Denies traffic specified with the `vlanList` parameter: `true` (default) or `false`.
`rtspSession.idleTimeout`	Number of seconds without traffic before a connection is eligible for deletion: **0** to **4294967295**. The default is **300**.
`rtspSession.maxHeaderSize`	Specifies the maximum header size: **0** to **4294967295**. The default is **4096**.
`rtspSession.maxQueuedData`	Specifies the maximum queued size: **0** to **4294967295**. The default is **32768**.
`rtspSession.unicastRedirect`	When enabled, specifies that the client can select the destination port for the streamed data: **true** or **false** (default).
`rtspSession.multicastRedirect`	When enabled, If you are using multicast streams, specifies that the client has permission to supply a different destination IP address for the streamed data: **true** or **false** (default).
`rtspSession.sessionReconnect`	When enabled, specifies that the system persists a resumed control connection to the correct server: **true** or **false** (default).
`rtspSession.realHTTPPersistence`	When enabled, specifies that the system automatically persists Real Networks-tunneled RTSP data over HTTP, which is over the RTSP port: **true** (default) or **false**.
`rtspSession.checkSource`	When enabled, specifies that the system examines the origin of the message to determine whether the message came from the client or the server: **true** (default) or **false**.

Parameter	Description
`rtspSession.rtpPort`	Specifies data channel port used for RTP. The default is **0**.
`rtspSession.rtcpPort`	Specifies data channel port used for RTCP. The default is **0**.

CR Example

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigAlgRtsp
metadata:
  name: "cnf-rtsp"
  namespace: "cnf-gateway"
spec:
  destinationAddress: "0.0.0.0/0"
  ipv6destinationAddress: "::/0"
  destinationPort: 554
  natPolicy: "cnf-nat-policy"
  firewallEnforcedPolicy: "cnf-fw-policy"
  logProfile: "cnf-log-profile"
  rtspSession:
    idleTimeout: 500
    maxHeaderSize: 6144
    maxQueuedData: 32768
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigAlgRtsp CR shortName is **algrtsp**.

View CR instance:

```
kubectl get algrtsp -n <namespace>
```

View CR configuration:

```
kubectl get algrtsp -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

This installation procedure provides ***Optional*** examples, representing each of the CNFs Custom Resources (CRs) that can be referenced by the F5BigAlgRtsp CR. Use the steps below to configure TMM:

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** Copy the example F5BigNatPolicy CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNatPolicy
metadata:
  name: "cnf-nat-policy"
  namespace: "cnf-gateway"
spec:
  sourceTranslation:
    - name: "dynamic-trans"
      type: "dynamic-pat"
      addresses:
        - "10.200.2.1-10.200.2.10"
      port: "8000-8500"
      patMode: "napt"
      inbound:
        eifTimeout: 200
        mode: "endpoint-independent-filtering"
      mapping:
        mode: "endpoint-independent-mapping"
        timeout: 60
      routeAdvertisement: true
  rule:
    - name: 10-20-subnet
      ipProtocol: tcp
      source:
        addresses:
         - "10.20.2.0/24"
      sourceTranslation: "dynamic-trans"
```

2. Install the F5BigNatPolicy CR:

```
kubectl apply -f cnf-nat-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigNatPolicy CR was **added/updated**:*

```
I0202 12:00:00.12345   1 event.go:282 Event(v1.ObjectReference{Kind:"F5NatPolicy",
NatPolicy cnf-gateway/cnf-nat-policy was added/updated
```

3. ***Optional:*** Copy the example F5BigFwPolicy CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFwPolicy
metadata:
  name: "cnf-fw-policy"
  namespace: "cnf-gateway"
spec:
  rule:
    - name: allow-10-30-tftp
      action: "accept"
      logging: true
      ipProtocol: tcp
      source:
        addresses:
          - "10.30.2.0/24"
      destination:
        ports:
```

```
          - "554"
```

4. Install the F5BigFwPolicy CR:

```
kubectl apply -f cnf-fw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigFwPolicy CR was **added/updated**:*

```
I0202 12:00:00.12346   1 event.go:282
 ↪  Event(v1.ObjectReference{Kind:"F5FirewallPolicy",
FirewallPolicy cnf-gateway/cnf-fw-policy was added/updated
```

5. ***Optional:*** Copy the example FF5BigLogHslpub CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigLogHslpub
metadata:
  name: "cnf-pub"
  namespace: "cnf-gateway"
spec:
  pool:
  - name: hsl-pool
    endpoint:
    - "10.30.2.220:514"
  syslog:
  - name: "cnf-syslog"
    distribution: "adaptive"
    format: "rfc5424"
    pool: "hsl-pool"
    protocol: "udp"
```

6. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

7. ***Optional:*** Copy the example F5BigLogProfile CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-logs"
  nat:
    enabled: true
    logSubscriberID: true
    publisher: "cnf-pub"
    inbound:
      start:
        mode: "enabled"
      end:
```

```
        mode: "enabled"
    quotaExceeded:
      mode: "enabled"
    errors:
      mode: "enabled"
  firewall:
    enabled: true
    trafficStats:
      publisher: "cnf-pub"
      reapedFlows: true
      missedFlows: true
      activeFlows: true
    network:
      publisher: "cnf-pub-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
        aclToBoxDeny: true
        translationFields: true
```

8. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-log-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

9. Copy the F5BigAlgRtsp CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigAlgRtsp
metadata:
  name: "cnf-rtsp"
  namespace: "cnf-gateway"
spec:
  destinationAddress: "0.0.0.0/0"
  ipv6destinationAddress: "::/0"
  destinationPort: 554
  natPolicy: "cnf-nat-policy"
  firewallEnforcedPolicy: "cnf-fw-policy"
  logProfile: "cnf-log-profile"
  rtspSession:
    idleTimeout: 500
    maxHeaderSize: 6144
    maxQueuedData: 32768
```

10. Install the F5BigAlgRtsp CR:

```
kubectl apply -f f5-cnf-rtsp.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigAlgRtsp CR was **added/updated**:*

```
I0203 12:00:00.1234519   1 event.go:282] Event(v1.ObjectReference{Kind:"F5RTSP",
F5RTSP cnf-gatway/cnf-rtsp was added/updated
```

Traffic statistics

If you have installed the [TMM Debug] container, use the following steps to gather traffic processing statistics for the F5BigNatPolicy, F5BigFwPolicy and F5BigContextSecure CRs.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify F5BigNatPolicy connection statistics:

```
tmctl -d blade fw_nat_rule_stat
```

```
context_type context_name                          rule_name
------------ ------------------------------------  --------------------------
virtual      cnf-cnf-context-secure-SecureContext_vs  10-20-subnet-natpolicyrule

micro_rules counter last_hit_time action
----------- ------- ------------- ------
          1       8    1643836695      0
```

```
tmctl -d blade fw_nat_trans_stat -s type,name,translation_requests
```

```
type         name        translation_requests
------------ ----------- --------------------
fw_src_trans transparent                    0
fw_dst_trans transparent                    8
fw_src_trans automap                        0
```

3. Verify the F5BigNatPolicy client IP address mappings:

```
lsndb list all
```

```
Client                                                Connections
------------------------------------------------------------------
0 client with 0 connection found.
LSN Persistence Entries
Client                                                Translation
------------------------------------------------------------------
10.20.2.220:52110                               10.200.2.8:8265
10.20.2.220                                     10.200.2.8
2 persist entries found.
LSN port block allocations
Client                                                Port block
------------------------------------------------------------------
0 port block entries found.
LSN Inbound Mapping Entries
Translation                                             Client
------------------------------------------------------------------
10.200.2.8:8265                               10.20.2.220:52110
10.200.2.7:8397                               10.20.2.220:52106
```

4. Verify the F5BigFwPolicy statistics:

```
tmctl -d blade fw_rule_stat
```

```
context_type context_name
------------ ------------------------------------------
virtual      cnf-gateway-cnf-fw-policy-SecureContext_vs

rule_name                            micro_rules counter last_hit_time action
------------------------------------ ----------- ------- ------------- ------
allow-10-30-rtsp-firewallpolicyrule            1       5    1638573270      2
```

5. Verify the F5BigAlgRtsp statistics:

```
tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
```

```
name                                            clientside.tot_conns
----------------------------------------------- --------------------
cnf-gateway-cnf-rtsp-alg-virtual-server          7
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigAlgPptp

Overview

The F5BigAlgPptp Custom Resource (CR) configures the Traffic Management Microkernel (TMM) to provide secure Point-to-Point Tunneling Protocol (PPTP) application layer gateway (ALG) services.

This document guides you through understanding, configuring and installing a simple F5BigAlgPptp CR.

CR Parameters

The table below describes the CR `spec` parameters:

Parameter	Description
`destinationAddress`	The destination IPv4 address of the application.
`ipv6destinationAddress`	The destination IPv6 address of the application.
`destinationPort`	The destination server port of the application. The default is **1723**.
`natPolicy`	Specifies a NAT Policy by `metadata.name`.
`firewallEnforcedPolicy`	Specifies a Firewall ACL Policy by `metadata.name`.
`logProfile`	Specifies a Log Profile by `metadata.name`.
`vlans.vlanList`	Specify a list of VLANs to accept traffic on. Traffic can also be denied using the `disableListedVlans` parameter.
`vlans.disableListedVlans`	Denies traffic specified with the `vlanList` parameter: `true` (default) or `false`.

CR Example

```
apiVersion: k8s.f5net.com/v1
kind: F5BigAlgPptp
metadata:
  name: "cnf-pptp"
  namespace: "cnf-gateway"
spec:
   destinationAddress: "0.0.0.0/0"
   ipv6destinationAddress: "::/0"
   destinationPort: 1723
   natPolicy: "cnf-nat-policy"
   logProfile: "cnf-log-profile"
   firewallEnforcedPolicy: "cnf-fw-policy"
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigAlgPptp CR shortName is **algpptp**.

View CR instance:

```
kubectl get algpptp -n <namespace>
```

View CR configuration:

```
kubectl get algpptp -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

This installation procedure provides ***Optional*** examples, representing each of the CNFs Custom Resources (CRs) that can be referenced by the F5BigAlgPptp CR. Use the steps below to configure TMM:

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** Copy the example F5BigNatPolicy CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNatPolicy
metadata:
  name: "cnf-nat-policy"
  namespace: "cnf-gateway"
spec:
  sourceTranslation:
    - name: "dynamic-trans"
      type: "dynamic-pat"
      addresses:
        - "10.200.2.1-10.200.2.10"
      port: "8000-8500"
      patMode: "napt"
      inbound:
        eifTimeout: 200
        mode: "endpoint-independent-filtering"
      mapping:
        mode: "endpoint-independent-mapping"
        timeout: 60
      routeAdvertisement: true
  rule:
    - name: 10-20-subnet
      ipProtocol: tcp
      source:
        addresses:
         - "10.20.2.0/24"
      sourceTranslation: "dynamic-trans"
```

2. Install the F5BigNatPolicy CR:

```
kubectl apply -f cnf-nat-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigNatPolicy CR was **added/updated**:*

```
I0202 12:00:00.12345   1 event.go:282 Event(v1.ObjectReference{Kind:"F5NatPolicy",
NatPolicy cnf-gateway/cnf-nat-policy was added/updated
```

3. ***Optional:*** Copy the example F5BigFwPolicy into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFwPolicy
metadata:
  name: "cnf-fw-policy"
  namespace: "cnf-gateway"
spec:
  rule:
    - name: allow-10-30-tftp
      action: "accept"
      logging: true
      ipProtocol: tcp
      source:
        addresses:
          - "10.30.2.0/24"
      destination:
        ports:
          - "1723"
```

4. Install the F5BigFwPolicy CR:

```
kubectl apply -f cnf-fw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigFwPolicy CR was **added/updated**:*

```
I0202 12:00:00.12346   1 event.go:282
 ↪  Event(v1.ObjectReference{Kind:"F5FirewallPolicy",
FirewallPolicy cnf-gateway/cnf-fw-policy was added/updated
```

5. ***Optional:*** Copy the example F5BigLogHslpub CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigLogHslpub
metadata:
  name: "cnf-pub"
  namespace: "cnf-gateway"
spec:
  pool:
  - name: hsl-pool
    endpoint:
    - "10.30.2.220:514"
  syslog:
  - name: "cnf-syslog"
    distribution: "adaptive"
    format: "rfc5424"
    pool: "hsl-pool"
    protocol: "tcp"
```

6. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

7. ***Optional:*** Copy the example F5BigLogProfile CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-logs"
  nat:
    enabled: true
    logSubscriberID: true
    publisher: "cnf-pub"
    inbound:
      start:
        mode: "enabled"
      end:
        mode: "enabled"
    quotaExceeded:
      mode: "enabled"
    errors:
      mode: "enabled"
  firewall:
    enabled: true
    trafficStats:
      publisher: "cnf-pub"
      reapedFlows: true
      missedFlows: true
      activeFlows: true
    network:
      publisher: "cnf-pub-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
        aclToBoxDeny: true
        translationFields: true
```

8. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-log-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

9. Copy the F5BigAlgPptp CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigAlgPptp
```

```
metadata:
  name: "cnf-pptp"
  namespace: "cnf-gateway"
spec:
   destinationAddress: "0.0.0.0/0"
   ipv6destinationAddress: "::/0"
   destinationPort: 1723
   natPolicy: "cnf-nat-policy"
   logProfile: "cnf-log-profile"
   firewallEnforcedPolicy: "cnf-fw-policy"
```

10. Install the F5BigAlgPptp CR:

```
kubectl apply -f f5-cnf-pptp.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigAlgPptp CR was **added/updated**:*

```
I0203 12:00:00.1234519   1 event.go:282] Event(v1.ObjectReference{Kind:"F5PPTP",
F5PPTP cnf-gatway/cnf-pptp was added/updated
```

Traffic statistics

If you have installed the [TMM Debug] container, use the following steps to gather traffic processing statistics for the F5BigNatPolicy, F5BigFwPolicy and F5BigContextSecure CRs.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify F5BigNatPolicy connection statistics:

```
tmctl -d blade fw_nat_rule_stat
```

```
context_type context_name                          rule_name
------------ ------------------------------------  --------------------------
virtual      cnf-cnf-context-secure-SecureContext_vs  10-20-subnet-natpolicyrule

micro_rules counter last_hit_time action
----------- ------- ------------- ------
          1       8    1643836695      0
```

```
tmctl -d blade fw_nat_trans_stat -s type,name,translation_requests
```

```
type         name        translation_requests
------------ ----------- --------------------
fw_src_trans transparent                    0
fw_dst_trans transparent                    8
fw_src_trans automap                        0
```

3. Verify the F5BigNatPolicy client IP address mappings:

```
lsndb list all
```

```
Client                                                Connections
-------------------------------------------------------------
0 client with 0 connection found.
LSN Persistence Entries
Client                                                Translation
-------------------------------------------------------------
10.20.2.220:52110                               10.200.2.8:8265
10.20.2.220                                     10.200.2.8
2 persist entries found.
LSN port block allocations
Client                                                Port block
-------------------------------------------------------------
0 port block entries found.
LSN Inbound Mapping Entries
Translation                                              Client
-------------------------------------------------------------
10.200.2.8:8265                           10.20.2.220:52110
10.200.2.7:8397                           10.20.2.220:52106
```

4. Verify the F5BigFwPolicy statistics:

```
tmctl -d blade fw_rule_stat
```

```
context_type context_name
------------ ------------------------------------------
virtual      cnf-gateway-cnf-fw-policy-SecureContext_vs

rule_name                             micro_rules counter last_hit_time action
------------------------------------ ----------- ------- ------------- ------
allow-10-30-pptp-firewallpolicyrule             1       5    1638573270      2
```

5. Verify the F5BigAlgPptp statistics:

```
tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
```

```
name                                              clientside.tot_conns
------------------------------------------------ --------------------
cnf-gateway-cnf-pptp-alg-virtual-server           7
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigTcpSetting

The F5BigTcpSetting Custom Resource (CR) provides many options to fine-tune how Traffic Management Microkernel (TMM) handles TCP connections. Once configured and installed, the F5BigTcpSetting CR can then be referenced by one of the [CNF CRs] listed in the Additional CRs section below.

This document guides you through understanding, configuring and installing a simple F5BigTcpSetting CR.

CR Parameters

The table below describes the CR `spec` parameters:

Parameter	Description
`abc`	Enables increasing the congestion window by basing the increase amount on the number of previously unacknowledged bytes that each ACK covers: **true** (default) or **false**.
`ackOnPush`	Enables performance improvement to Windows and MacOS peers who are writing out on a very small end buffer: **true** or **false**.
`autoNagle`	Specifies how the system applies Nagle's algorithm to reduce the number of short segments on the network: **auto** (default), **enabled**, or **disabled**. When **auto**, the use of Nagle is based on network conditions.
`autoProxyBufferSize`	When enabled, specifies that the system uses the network measurements to set the optimal proxy buffer size. The default is **false**.
`autoReceiveWindowSize`	When enabled, specifies that the system uses the network measurements to set the optimal receive window size. The default is **false**.
`autoSendBufferSize`	When enabled, specifies that the system uses the network measurements to set the optimal send buffer size. The default is **false**.
`cmetricsCache`	Specifies, when enabled, that the system uses a cache for storing congestion metrics. The default is **true**.
`cmetricsCacheTimeout`	Specifies the time, in seconds, for which entries in the congestion metrics cache are valid. The default value is 0, which defers to the sys db variable route.metrics.timeout.
`congestionControl`	Specifies the algorithm to use to share network resources among competing users to reduce congestion: **high-speed** (default), **bbr**, **cdg**, **chd**, **cubic**, **illinois**, **new-reno**, **reno**, **scalable**, **vegas**, **westwood**, or **woodside**.
`delayedAcks`	When enabled, the traffic management system allows coalescing of multiple ACK responses. The default value is **true**.
`dsack`	Enables the Selective ACKs (SACK) option to acknowledge duplicate segments: **true** (default) or **false**.

Parameter	Description
`earlyRetransmit`	When enabled, specifies that the system uses early fast retransmits (as specified in RFC 5827) to reduce the recovery time for connections that are receive-buffer or user-data limited. The default value is **true**.
`ecn`	Enables the TCP flags CWR and ECE to notify its peer of congestion and congestion counter-measures: **true** (default) or **false**.
`enhancedLossRecovery`	Enables enhanced loss recovery to recover from random packet losses more effectively: **true** (default) or **false**.
`fastOpen`	When enabled, permits TCP Fast Open, allowing properly equipped TCP clients to send data with the SYN packet. The default value is **true**.
`idleTimeout`	Specifies the number of seconds that a connection is idle before the connection is eligible for deletion. The default value is **300**.
`initCWND`	Specifies the initial congestion window size for connections to this destination. The actual window size is this value multiplied by the MSS (Maximal Segment Size) for the same connection: **0** to **64**. The default value is **16**.
`initRWND`	Specifies the initial receive window size for connections to this destination. The actual window size is this value multiplied by the MSS (Maximal Segment Size) for the same connection: **0** to **64**. The default value is **16**.
`ipDFMode`	Describe the Don't Fragment (DF) bit setting in the outgoing packet's IP Header. Available options: **clear**, **pmtu** (default), **preserve**, and **set**.
`ipTTLMode`	Describe the outgoing packet's IP Header TTL value modes. Available options: **decrement**, **preserve**, **proxy** (default), and **set**.
`ipTTLV4`	Specifies the outgoing IPV4 Header TTL value for ip-ttl-mode 'set'. The default value is **225**.
`ipTTLV6`	Specifies the outgoing IPV6 Header TTL value for ip-ttl-mode 'set'. The default value is **64**.
`limitedTransmit`	Enables limited transmit recovery revisions for fast retransmits to reduce the recovery time for connections on a lossy network: **true** (default) or **false**.
`maxRetrans`	Specifies the maximum number of retransmissions of data segments that the system allows. The default value is **8**.
`maxSegmentSize`	Specifies the largest amount of data that the system can receive in a single TCP segment, not including the TCP and IP headers. If the value is 0 (zero), the system calculates the value from the MTU. The default value **1460**.
`md5SignaturePassphrase`	Enables a plaintext passphrase which may be between 1 and 80 characters in length, and is used in a shared-secret scheme to implement the spoof-prevention parts of RFC2385.
`minimumRTO`	Specifies the minimum TCP retransmission timeout in milliseconds. The default value is **1000**.

Parameter	Description
`nagle`	Enables Nagle's algorithm to reduce the number of short segments on the network: **true** or **false** (default).
`packetLossIgnoreBurst`	Specifies the probability of performing congestion control when multiple packets in a row are lost even if the pkt-loss-ignore-rate was not exceeded **0** to **32**. The default valie is **0**, meaning that the system should perform congestion control if any packets are lost. Higher values decrease the chance of performing congestion control.
`packetLossIgnoreRate`	Specifies the threshold of packets lost per million at which the system should perform congestion control: **0** to **1000000**. The default value is **0**.
`proxyBufferHigh`	Specifies the highest level at which the receive window is closed: **0** to **4294967295**. The default value is **16384**.
`proxyBufferLow`	Specifies the lowest level at which the receive window is closed: **0** to **4294967295**. The default value is **4096**.
`proxyMSS`	Specifies, when enabled, that the system advertises the same mss to the server as was negotiated with the The default is **false**.
`proxyOptions`	Specifies, when enabled, that the system advertises an option, such as a time-stamp to the server only if it was negotiated with the The default is **false**.
`pushFlag`	When default, specifies that the system sets PUSH flag when sending the last segment in the send buffer. When none, specifies that the system never sets PUSH flag for TCP packets. When one, specifies that the system sets one PUSH flag for the FIN segment. When auto, specifies that the system sets PUSH flag based on the application/network conditions. Available options: **auto**, **default** (default), **none**, and **one**.
`ratePace`	Enables rate pace TCP data transmission: **true** (default) or **false**.
`ratePaceMaxRate`	If not 0, the maximum rate in bytes per second that TCP connections will be paced to **0** to **4294967295**. The default value is **0**.
`receiveWindowSize`	Specifies the size of the receive window, in bytes. The default value is **65535**.
`resetOnTimeout`	Specifies whether to reset connections on timeout. The default is **true**.
`retransmitThreshold`	Specifies the number of duplicate ACKs (retransmit threshold) to start fast recovery. Higher values decrease the likelihood of performing fast recovery in a network with high packet reordering: **3** to **255**. The default is **3**.
`selectiveAcks`	Specifies, when enabled, that the system negotiates RFC2018-compliant Selective Acknowledgments with peers: **true** (default) or **false**.
`selectiveNack`	Specifies whether Selective Negative Acknowledgment is enabled or not: **true** or **false** (default).
`sendBufferSize`	Specifies the size of the buffer, in bytes. The default value is **131072**.

Parameter	Description
`slowStart`	Enables larger initial window sizes to help reduce round trip times: **true** (default) or **false**.
`synCookieEnable`	Enables SYN Cookies: **true** (default) or **false**.
`synMaxRetrans`	Specifies the maximum number of retransmissions of SYN segments that the system allows: **0** to **4294967295**. The default value is **3**.
`synRTOBase`	Specifies the initial RTO (Retransmission TimeOut) base multiplier for SYN retransmission, in milliseconds. This value is modified by the exponential backoff table to select the interval for subsequent retransmissions: **0** to **5000**. The default value is **3000**.
`tailLossProbe`	When enabled, specifies that the system uses tail loss probe to reduce the number of retransmission timeouts. The default is **true**.
`verifiedAccept`	When enabled, a SYN-ACK will be sent only if the server port is open. Not compatible with iRules. The default is **false**.

CR Example

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigTcpSetting
metadata:
  name: "cnf-tcp-optimize"
  namespace: "cnf-gateway"
spec:
  proxyBufferHigh: 128000
  proxyBufferLow: 128000
  idleTimeout: 150
  receiveWindowSize: 128000
  resetOnTimeout: false
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigTcpSetting CR shortName is **tcpset**.

View CR instance:

```
kubectl get tcpset -n <namespace>
```

View CR configuration:

```
kubectl get tcpset -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

Use the steps below to install the F5BigTcpSetting CR.

1. The example F5BigTcpSetting CR increases the **proxyBuffer** sizes, **idleTimeout** and **receiveWindowSize** to improve performance. Copy and paste the example into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigTcpSetting
metadata:
  name: "cnf-tcp-optimize"
  namespace: "cnf-gateway"
spec:
  proxyBufferHigh: 128000
  proxyBufferLow: 128000
  idleTimeout: 150
  receiveWindowSize: 128000
  resetOnTimeout: false
```

2. Install the F5BigTcpSetting CR:

```
kubectl apply -f cnf-tcp-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigTcpSetting CR was **added/updated**:*

```
I0202 12:00:00.12349   1 event.go:282 Event(v1.ObjectReference{Kind:"F5TcpSetting",
TcpSetting cnf-gateway/cnf-tcp-optimize was added/updated
```

3. The example F5BigContextSecure CR listens for connections destined to IP addresses in the **2002::200:200:200:0/112** subnet, using the **tcp** protocol, and only on the **subscriber-vlan** interface. The CR also references the F5BigTcpsettings profile. Copy and paste the example into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "2002::200:200:200:0/112"
   destinationPort: 0
   ipProtocol: "tcp"
   profile: "tcp"
   tcpSettings:
     clientSide: "cnf-tcp-optimize"
     serverSide: "cnf-tcp-optimize"
   vlans:
     vlanList:
       - "subscriber-vlan"
```

4. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-context.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-context was added/updated
```

5. The TMM Proxy Pod can now process application traffic using the F5BigTcpSetting CR.

Additional CRs

The F5BigTcpSetting CR can be references by the [CNF CRs] listed below:

- F5BigContextSecure - Full proxy TCP and UDP application layer gateway services.
- F5BigAlgFtp - File Transfer Protocol (FTP) application layer gateway services.
- F5BigDnsApp - High-performance DNS resolution, caching, and DNS64 translations.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigUdpSetting

The F5BigUdpSetting CR provides many option to fine-tune how Traffic Management Microkernel (TMM) handles UDP connections. Once configured and installed, the F5BigUdpSetting CR can then be referenced by one of the [CNF CRs] listed in the Additional CRs section below.

This document guides you through understanding, configuring and installing a simple F5BigUdpSetting CR.

CR Parameters

The table below describes the CR `spec` parameters:

Parameter	Description
`allowNoPayload`	Allows UDP datagrams with no payload: **true** or **false** (default).
`bufferMaxBytes`	Specifies the ingress buffer byte limit: **0** to **16777215**. The default is **655350**.
`bufferMaxPackets`	Specifies the ingress buffer packet limit: **0** to **255**. The default value is **0**.
`datagramLoadBalancing`	Specifies the system load balances UDP traffic packet-by-packet and does not treat UDP packets from the same source and port as part of a connection: **true** or **false** (default).
`idleTimeout`	The number of seconds that a connection is idle before the connection is eligible for deletion: **0** to **4294967295**. The default value is **60**.
`ipDFMode`	Describe the outgoing packet Don't Fragment (DF) bit. Modes: **pmtu** - Set the packet DF big based on the ip pmtu setting. **preserve** - Preserve the incoming packet DF bit. **set** - Set the outgoing UDP packet DF bit. **clear** - Clear the outgoing UDP packet DF bit.
`ipTOSToClient`	Specifies the Type of Service level assigned to UDP packets sent to clients: **0** to **65535**. The default value is **0**.
`linkQOSToClient`	Specifies the Quality of Service level assigned to UDP packets sent to clients: **0** to **65535**. The default value is **0**.
`ipTTLMode`	Describe the outgoing packet TTL. Modes are: **Proxy** - Set the IPv4 TTL to 255 and IPv6 to 64. **Preserve** - Preserve the original IP TTL value. **Decrement** - Set IP TTL to original packet TTL minus 1. **Set** - Set IP TTL to values from ip-ttl-v4 and ip-ttl-v6 in the same profile.
`ipTTLV4`	Specifies the outgoing IPv4 Header TTL value when IP TTL Mode is set: **0** to **255**. The default value is **225**.
`ipTTLV6`	Specifies the outgoing IPv6 Header TTL value when IP TTL Mode is set: **0** to **255**. The default value is **64**.
`noChecksum`	Enables checksum processing: **true** or **false** (default).
`proxyMSS`	Enables advertising the same MSS to the server as negotiated with the client: **true** or **false** (default).
`sendBufferSize`	The send buffer byte limit: **536** to **16777215**. The default value is **655350**.

CR Example

```
apiVersion: k8s.f5net.com/v1
kind: F5BigUdpSetting
metadata:
  name: "cnf-udp-optimize"
  namespace: "cnf-gateway"
spec:
  datagramLoadBalancing: true
  sendBufferSize: 15000000
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigUdpSetting CR shortName is **udpset**.

View CR instance:

```
kubectl get udpset -n <namespace>
```

View CR configuration:

```
kubectl get udpset -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

Use the steps below to install the F5BigUcpSetting CR.

1. The example F5BigUcpSetting CR modifies the **datagramLoadbalancing** and **sendBufferSize** parameters. Copy and paste the example into a YAML file:

   ```
   apiVersion: k8s.f5net.com/v1
   kind: F5BigUdpSetting
   metadata:
     name: "cnf-udp-optimize"
     namespace: "cnf-gateway"
   spec:
     datagramLoadBalancing: true
     sendBufferSize: 15000000
   ```

2. Install the F5BigUdpSetting CR:

   ```
   kubectl apply -f cnf-udp-cr.yaml
   ```

*In this example, the BIG-IP Controller logs indicate the F5BigUdpSetting CR was **added/updated**:*

```
I0202 12:00:00.12349   1 event.go:282 Event(v1.ObjectReference{Kind:"F5UdpSetting",
UdpSetting cnf-gateway/udp-client was added/updated
```

3. The example F5BigContextSecure CR listens for connections destined to IP addresses in the **2002::200:200:200:0/112** subnet, using the **udp** protocol, and only on the **subscriber-vlan** interface. The CR also references the F5BigTcpsettings profile. Copy and paste the example into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "2002::200:200:200:0/112"
   destinationPort: 0
   ipProtocol: "udp"
   profile: "udp"
   udpSettings:
     clientSide: "cnf-udp-optimize"
     serverSide: "cnf-udp-optimize"
   vlans:
     vlanList:
       - "subscriber-vlan"
```

4. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-context.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-context was added/updated
```

5. The TMM Proxy Pod can now process application traffic using the F5BigUdpSetting CR.

Additional CRs

The F5BigTcpSetting CR can be references by the [CNF CRs] listed below:

- F5BigContextSecure - Full proxy TCP and UDP application layer gateway services.
- F5BigDnsApp - High-performance DNS resolution, caching, and DNS64 translations.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigFastl4Setting

The F5BigFastl4Setting CR currently provides one option to fine-tune how Traffic Management Microkernel (TMM) handles connections. Once configured and installed, the F5BigFastl4Setting CR can then be referenced by the F5BigContextSecure Custom Resource (CR).

This document guides you through understanding, configuring and installing a simple F5BigFastl4Setting CR.

CR Parameters

The table below describes the CR `spec` parameters:

Parameter	Description
`idleTimeout`	Specifies the number of seconds that a connection is idle before the connection is eligible for deletion: **0** to **4294967295**. The default value is **300**.

CR Example

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFastl4Setting
metadata:
  name: "cnf-fastl4-optimize"
  namespace: "cnf-gateway"
spec:
  idleTimeout: 150
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Installation

Use the steps below to install the F5BigTcpSetting CR.

1. Copy the example F5BigTcpSetting CR modifies the **idleTimeout** period. Copy and paste the example into a YAML file:

   ```
   apiVersion: "k8s.f5net.com/v1"
   kind: F5BigFastl4Setting
   metadata:
     name: "cnf-fastl4-optimize"
     namespace: "cnf-gateway"
   spec:
     idleTimeout: 150
   ```

2. Install the CR:

```
kubectl apply -f cnf-fastl4-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigTcpSetting CR was **added/updated**:*

```
I0202 12:00:00.12349   1 event.go:282 Event(v1.ObjectReference{Kind:"F5TcpSetting",
TcpSetting cnf-gateway/cnf-tcp-optimize was added/updated
```

3. The example F5BigContextSecure CR listens for connections destined to IP addresses in the **2002::200:200:200:0/112** subnet, using **any** protocol, and only on the **subscriber-vlan** interface. The CR also references the F5BigFastl4Setting profile. Copy and paste the example into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "2002::200:200:200:0/112"
   destinationPort: 0
   ipProtocol: "any"
   profile: "fastL4"
   fastL4Settings:
     profileName: "cnf-fastl4-optimize"
   vlans:
     vlanList:
       - "subscriber-vlan"
```

4. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-context.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-context was added/updated
```

5. The TMM Proxy Pod can now process application traffic using the F5BigFastl4Setting CR.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigNetVlan

Overview

The F5BigNetVlan Custom Resource (CR) configures the Traffic Management Microkernel (TMM) network interface settings: VLAN tags, Self IP addresses, Maximum Transmission Size (MTU), bonding, and packet hashing algorithms.

This document guides you through understanding, configuring and deploying a simple F5BigNetVlan CR.

Scaling TMM

When scaling the TMM Proxy Pod beyond a single instance in the namespace, the `spec.selfip_v4s` and `spec.selfip_v6s` parameters must be configured to provide unique self IP addresses to each TMM replica. The first self IP address in the list is applied to the first TMM Pod, the second IP address to the second TMM Pod, continuing through the list.

Parameters

The table below describes the CR's `spec` parameters:

Parameter	Description
`name`	The name of the VLAN object in the TMM configuration.
`tag`	The tagging ID applied to the VLAN object.
`bonded`	Combine multiple interfaces into a single bonded interface (true/false). The default false (disabled).
`interfaces`	One or more interfaces to associate with the VLAN object.
`selfip_v4s`	Specifies a list of IPv4 Self IP addresses associated with the VLAN. Each TMM replica receives an IP address in the element order.
`prefixlen_v4`	The IPv4 self IP address subnet mask.
`selfip_v6s`	Specifies a list of IPv6 Self IP addresses associated with the VLAN. Each TMM replica receives an IP address in the element order.
`prefixlen_v6`	The IPv6 self IP address subnet mask.
`allowed_services`	Specifies a list of protocols and the protocol service ports this VLAN accepts.
`allowed_services.protocol`	Specifies the protocol traffic the VLAN accepts.
`allowed_services.port`	Specifies the service port traffic the VLAN accepts.
`mtu`	Maximum transmission unit in bytes: (1500 to 8000). The default is 1500. ***Important:*** *You must also set the BIG-IP Controller `TMM_DEFAULT_MTU` parameter to the same value when modifying the default.*
`trunk_hash`	The hashing algorithm used to distribute packets across bonded interfaces: **src-dst-mac** combines MAC addresses of the source and destination. **dst-mac** the MAC address of the destination. **index** combine ports of the source and the destination. **src-dst-ipport** combine IP addresses and ports of the source and the destination (default).

Parameter	Description
`cmp_hash`	Specifies how traffic will be disaggregated. Use the **SRC_ADDR** value for the subscriber (upstream) facing VLAN and the **DST_ADDR** value for the application (downstream) facing VLAN. Do not use the **SRC_DST_ADDR_PORT** value.
`auto_lasthop`	Disables the auto last hop feature that sends return traffic to the MAC address transmitting the request: **AUTO_LASTHOP_ENABLED**, **AUTO_LASTHOP_DISABLED** or **AUTO_LASTHOP_DEFAULT**.

Important: *To optimize network performance, set the `cmp_hash` parameter values as follows: set **SRC_ADDR** on the subscriber (upstream) VLAN, and **DST_ADDR** on the application (downstream) facing VLAN.*

CR Examples

Subscriber VLAN:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetVlan
metadata:
  name: "subscriber-vlan"
  namespace: "cnf-gateway"
spec:
  name: clientside
  interfaces:
    - "1.1"
  selfip_v4s:
    - 10.10.10.100
    - 10.10.10.101
  prefixlen_v4: 24
  selfip_v6s:
    - 2002::10:10:10:100
    - 2002::10:10:10:101
  prefixlen_v6: 116
  mtu: 9000
  cmp_hash: SRC_ADDR
```

Application VLAN:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetVlan
metadata:
  name: "application-vlan"
  namespace: "cnf-gateway"
spec:
  name: serverside
  interfaces:
    - "1.2"
  selfip_v4s:
    - 192.168.10.100
    - 192.168.10.101
```

```
  prefixlen_v4: 24
  selfip_v6s:
    - 2002::192:168:10:100
    - 2002::192:168:10:101
  prefixlen_v6: 116
  mtu: 9000
  cmp_hash: DST_ADDR
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigNetVlan CR shortName is **vlan**.

View CR instance:

```
kubectl get vlan -n <namespace>
```

View CR configuration:

```
kubectl get vlan -n <namespace> -o yaml
```

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- A Linux based workstation.

Deployment

Use the following steps to deploy the example F5SPKVlan CR, and verify the Service Proxy TMM configuration.

1. Copy the F5BigNetVlan CRs into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetVlan
metadata:
  name: "subscriber-vlan"
  namespace: "cnf-gateway"
spec:
  name: clientside
  interfaces:
    - "1.1"
  selfip_v4s:
    - 10.10.10.100
    - 10.10.10.101
  prefixlen_v4: 24
  selfip_v6s:
    - 2002::10:10:10:100
    - 2002::10:10:10:101
  prefixlen_v6: 116
```

```
  mtu: 9000
  cmp_hash: DST_ADDR
---
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetVlan
metadata:
  name: "application-vlan"
  namespace: "cnf-gateway"
spec:
  name: serverside
  interfaces:
    - "1.2"
  selfip_v4s:
    - 192.168.10.100
    - 192.168.10.101
  prefixlen_v4: 24
  selfip_v6s:
    - 2002::192:168:10:100
    - 2002::192:168:10:101
  prefixlen_v6: 116
  mtu: 9000
  cmp_hash: SRC_ADDR
```

2. Install the CR:

```
kubectl apply -f cnf-vlan.yaml
```

3. List the VLAN CRs:

```
kubectl get f5-big-net-vlan -n cnf-gateway
```

In this example, the VLAN CR is installed:

```
NAME
subscriber-vlan
application-vlan
```

4. If the Debug Sidecar is enabled (the default), you can verify that TMM has been configured:

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- ip a
```

The interfaces should appear at the bottom of the list:

```
8: clientside: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000
    link/ether 1e:80:c1:e8:81:15 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.100/24 brd 192.168.10.0 scope global server
       valid_lft forever preferred_lft forever
    inet6 2002::192:168:10:100/112 scope global
       valid_lft forever preferred_lft forever
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigNetStaticroute

Overview

The F5BigNetStaticroute Custom Resource (CR) configures the Traffic Management Microkernal's (TMM) static routing table.

This document guides you through a basic F5BigNetStaticroute CR installation.

Parameters

The CR `spec` parameters used to configure the Service Proxy TMM static routing table are:

Parameter	Description
`destination`	The IPv4 Address routing destination.
`prefixlen`	The IPv4 address subnet mask.
`gateway`	The IPv4 address of the routing gateway.
`destination_v6`	The IPv6 Address routing destination.
`prefixlen_v6`	The IPv6 address subnet mask.
`gateway_v6`	The IPv6 address of the routing gateway.
`type`	Type of route to set. The default is gateway.

Example CR:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetStaticroute
metadata:
  name: "staticroute-172-16"
  namespace: "cnf-gateway"
spec:
  destination_v6: 2002::172:16:10:0
  prefixLen_v6: 116
  type: gateway
  gateway_v6: 2002::192:168:10:100
```

Requirements

Ensure you have:

- Uploaded the CNFs Software.
- Installed the BIG-IP Controller Pods.
- Have a Linux based workstation.

Installation

SR-IOV traffic

Use the steps below to create a new route for SR-IOV (network) traffic, pointing the **2002::172:16:10:0/116** subnet to a remote router on TMM's **serverside** VLAN interface:

1. Copy the *Example CR* into a YAML file named **cnf-sriov-route.yaml**::

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetStaticroute
metadata:
  name: "staticroute-172-16"
  namespace: "cnf-gateway"
spec:
  destination_v6: 2002::172:16:10:0
  prefixLen_v6: 116
  type: gateway
  gateway_v6: 2002::192:168:10:100
```

2. Install the F5BigNetStaticroute CR:

```
kubectl apply -f cnf-static-route.yaml
```

3. Verify the static route exists in the Service Proxy TMM container's routing table:

   *In this example, TMM is installed in the **cnf-gateway** Namespace:*

```
kubectl exec -it deploy/f5-tmm -c f5-tmm -n cnf-gateway -- ip -6 r
```

```
2002::10:10:10:100/116 dev clientside proto kernel
2002::192:168:10:100/116 dev serverside proto kernel
2002::172:16:10:0/116 via 2002::192:168:10:200 dev serverside
```

Cluster traffic

When enabling the [BIG-IP Contoller] `TMM_IGNORE_GATEWAYS` Helm parameter, cluster (Pod-to-Pod) traffic may fail. Use the steps below to manually create routes for cluster traffic. In the F5BigNetStaticroute CR example below, a new route is created pointing the DNS Service IP **fd74:ca9b:3a09:868c:172:18:0:800a** to the Calico default gateway on the TMM Proxy Pod's **eth0** interface:

1. Copy the CR into a YAML file named **cnf-cluster-route.yaml**::

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetStaticroute
metadata:
  name: "cnf-staticroute-dns"
  namespace: "cnf-gateway"
spec:
  destination_v6: fd74:ca9b:3a09:868c:172:18:0:800a
  dynamic: false
  prefixLen_v6: 116
  type: gateway
  gateway_v6: fe80::ecee:eeff:feee:eeee
  interface: eth0
```

2. Install the F5BigNetStaticroute CR:

```
kubectl apply -f cnf-cluster-route.yaml
```

3. Verify the static route exists in the Service Proxy TMM container's routing table:

   *In this example, TMM is installed in the **cnf-gateway** Namespace:*

```
kubectl exec -it deploy/f5-tmm -c f5-tmm -n cnf-gateway -- ip -6 r
```

```
2002::10:20:2:0/112 dev client proto kernel
2002::10:30:2:0/112 dev server proto kernel
fd74:ca9b:3a09:868c:172:18:0:6b58 dev eth0 proto kernel
fd74:ca9b:3a09:868c:172:18:0:8000/116 via fe80::ecee:eeff:feee:eeee dev eth0
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigCneSnatpool

Overview

The F5BigCneSnatpool Custom Resource (CR) configures the Traffic Management Microkernel (TMM) Proxy Pod to perform source network address translations (SNAT) on egress network traffic. When clients connect to resources through TMM, the source IP address of the egress packet is translated to one of the IP addresses in the SNAT pool. Once configured and installed, the F5BigCneSnatpool CR can be referenced by the F5BigContextSecure and F5BigDnsApp CRs to process traffic.

This document guides you through understanding, configuring and deploying a simple F5BigCneSnatpool CR.

Scaling TMM

When scaling the TMM Proxy Pod beyond a single instance in the namespace, the F5BigCneSnatpool CR must be configured to provide a SNAT pool to each TMM replica. The first SNAT pool is applied to the first TMM replica, the second snatpool to the second TMM replica, continuing through the list.

Important: *When configuring SNAT pools with multiple IP subnets, ensure all TMM replicas receive the same IP subnets.*

Example CR:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCneSnatpool
metadata:
  name: "egress-snatpool"
  namespace: cnf-gateway
spec:
  name: "egress_snatpool"
  addressList:
    - - 10.244.10.1
      - 10.244.20.1
      - 2002::10:244:10:1
      - 2002::10:244:20:1
    - - 10.244.10.2
      - 10.244.20.2
      - 2002::10:244:10:2
      - 2002::10:244:20:2
```

CR shortName

CR shortNames provide an easy way to view installed CRs, and their configuration parameters. The CR shortName can also be used to delete the CR instance. The F5BigCneSnatpool CR shortName is **snatpool**.

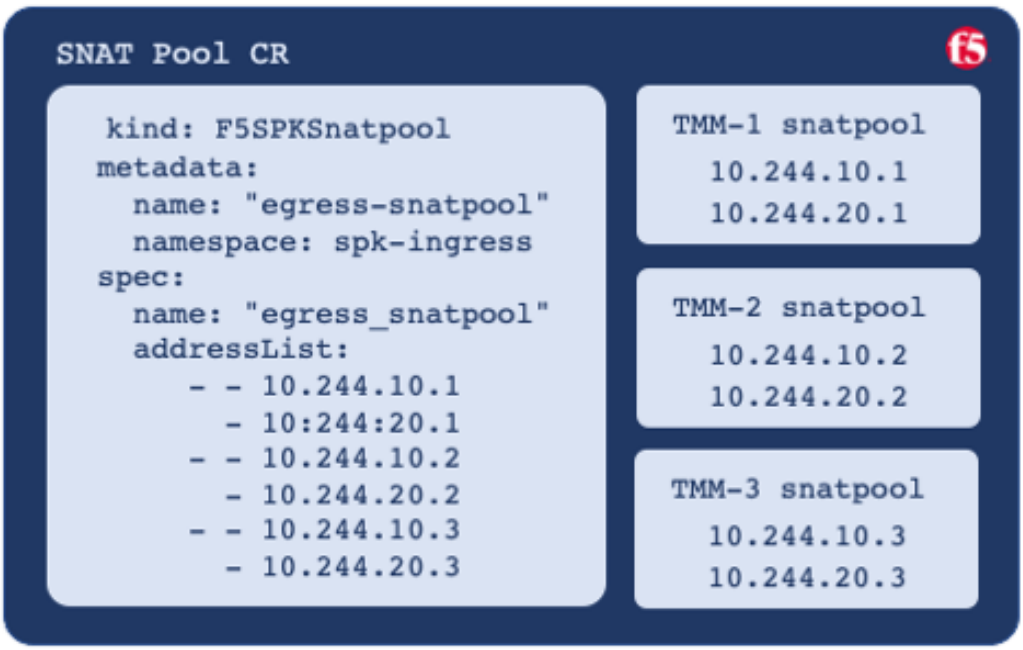
View CR instance:

```
kubectl get snatpool -n <namespace>
```

View CR configuration:

```
kubectl get snatpool -n <namespace> -o yaml
```

Example deployment:



Note: *The SNAT Pool CR supports both IPv4 and IPv6 addresses.*

Advertising address lists

By default, SNAT Pool IP addresses are not advertised (redistributed) to BGP neighbors. To advertise SNAT Pool IP addresses, you must configure a `prefixList` and `routeMaps` when installing the BIG-IP Controller. For configuration assistance, refer to the BGP Overview.

Requirements

Ensure you have:

- Created an external and internal F5BigNetVlan.
- A Linux based workstation.

Installation

Use the following steps to install the example F5SPKSnatpool CR.

1. Copy the example F5BigCneSnatpool into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCneSnatpool
metadata:
  name: "egress-snatpool"
  namespace: cnf-gateway
spec:
  name: "egress_snatpool"
```

```
  addressList:
    - - 10.244.10.1
      - 10.244.20.1
      - 2002::10:244:10:1
      - 2002::10:244:20:1
    - - 10.244.10.2
      - 10.244.20.2
      - 2002::10:244:10:2
      - 2002::10:244:20:2
```

2. Install the F5BigCneSnatpool CR:

```
kubectl apply -f cnf-snatpool-crd.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigCneSnatpool CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
F5Snatpool cnf-gateway/46_snatpool was added/updated
```

3. To verify the SNAT pool IP address mappings, obtain the name of the Controller's persistmap:

Note: *The persistmap maintains SNAT mappings after unexpected Pod restarts.*

```
kubectl get cm -n cnf-gateway | grep persistmap
```

```
persistmap-76946d464b-d5xvc
```

4. Verify the SNAT IP address mappings:

```
kubectl get cm persistmap-76946d464b-d5xvc \
-o "custom-columns=IP Addresses:.data.snatpoolMappings" -n cnf-gateway
```

*In this example, the SNAT IPs allocated to TMM are **10.244.10.1**, **10.244.20.1**, **2002::10:244:10:1** and **2002::10:244:20:1**.*

```
IP Addresses
{"ca93c77b-42bb-4b67-bf3a-
 ↪  d25128f3374b":"10.244.10.1,10.244.20.1,2002::10:244:1:10,2002::10:244:20.1"}
```

5. Continue to the **Next step** section to begin using the F5BigCneSnatpool.

Next step

Select and install one of the Traffic Management [CNF CRs] to begin processing application traffic:

- F5BigContextSecure - Full proxy TCP and UDP application layer gateway services.
- F5BigDnsApp - High-performance DNS resolution and caching.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigCneAddresslist

Overview

The F5BigCneAddresslist Custom Resource (CR) is useful for defining larger, more complex lists of IP address that can then be referenced by Cloud-Native Network Functions (CNFs) protection and NAT CRs:

- F5BigNatPolicy - Carrier-grade NAT (CG-NAT) using large-scale NAT (LSN) pools.
- F5BigFwPolicy - Granulalar packet filtering based on access control list (ACL) policies.
- F5BigDdosPolicy - Denial of Service (DoS/DDoS) event detection and mitigation.

This document guides you through creating a simple F5BigCneAddresslist.

Note: *The F5BigCnePortlist CR is useful for defining lists of service ports.*

Parameters

The CR `spec` parameters used to configure the Service Proxy TMM static routing table are:

Parameter	Description
`addresses`	The IPv4 or IPv6 addresses included in the address list: host **2002::10:10:10:1**, subnet **2002::10:10:0:0/96**, or range **2002::10:10:10:1-2002::10:10:10:20**.

Requirements

Ensure you have:

- Uploaded the [CNF Software].
- Installed the BIG-IP Controller Pods.
- Have a Linux based workstation.

Installation

Use the following steps to install the F5BigCneAddresslist CR:

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. Copy the example CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCneAddresslist
metadata:
  name: "outbound-nat"
  namespace: "cnf-gateway"
spec:
  addresses:
  - "2002::192:168:10:1-2002::192:168:10:10"
  - "2002::10:10:10:0/112"
```

2. Install the F5BigCneAddresslist CR:

```
kubectl apply -f cnf-address-list.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigCneAddresslist CR was **added/updated**:*

```
I0607 12:00:00.12345   1 event.go:282] Event(v1.ObjectReference{Kind:"F5AddressList",
F5AddressListProfile cnf-gateway/outbound-nat was added/updated
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

F5BigCnePortlist

Overview

The F5BigCnePortlist Custom Resource (CR) is useful for defining larger, more complex sets of service ports, that can then be referenced by any of the Cloud-Native Network Functions (CNFs) protection and NAT CRs:

- F5BigNatPolicy - Carrier-grade NAT (CG-NAT) using large-scale NAT (LSN) pools.
- F5BigFwPolicy - Granulalar packet filtering based on access control list (ACL) policies.
- F5BigDdosPolicy - Denial of Service (DoS/DDoS) event detection and mitigation.

This document guides you through creating a simple F5BigCnePortlist.

Note: *The F5BigCneAddresslist CR is useful for defining lists of IP addresses ports.*

Parameters

The CR `spec` parameters used to configure the Service Proxy TMM static routing table are:

Parameter	Description
`ports`	The service ports included in the port list. Port **0** is not a valid value, and is not allowed.

Requirements

Ensure you have:

- Uploaded the [CNF Software].
- Installed the BIG-IP Controller Pods.
- Have a Linux based workstation.

Installation

Use the following steps to install the F5BigCnePortlist CR:

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. Copy the example CR into a YAML file:

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCnePortlist
metadata:
  name: "allow-5000-80"
  namespace: "cnf-gateway"
spec:
  ports:
  - "5000-5500"
  - "80"
```

2. Install the F5BigCnePortlist CR:

```
kubectl apply -f cnf-port-list.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigCneAddresslist CR was **added/updated**:*

```
I0607 12:00:00.12345   1 event.go:282] Event(v1.ObjectReference{Kind:"F5PortList",
F5PortList cnf-gateway/allow-5000-80 was added/updated
```

Feedback

Provide feedback to improve this document by emailing spkdocs@f5.com.

CNFs NAT64

Overview

Cloud-Native Network Functions (CNFs) NAT64 provides the ability to process IPv6 to IPv4 application traffic, specifically between IPv6 only clients and IPv4 only servers. To provide NAT64, the Service Proxy Traffic Management Microkernel (TMM) Proxy Pod first translates DNS queries using the well-known NAT64 **64:ff9b::** prefix, by converting the IPv4 response into an IPv6 hexidecimal format, and appending the result to the host portion of the well-known NAT64 prefix. The TMM Proxy Pod then uses NAT to translate connections to **64:ff9b::/96** destinations, proxying the IPv6 client addresses and IPv4 application addresses.

Connection example

When the IPv6 only client requires a connection to IPv4 only server **www.64test.com**, the TMM Proxy Pod first performs a DNS query, and converts the response **40.40.40.1** to hex value **28282801**. The hex conversion is appended to the host portion of the well-known NAT64 prefix as **64:ff9b::2828:2801**. When the IPv6 client connects to **64:ff9b::2828:2801** through the TMM Proxy Pod, the network packet is sent to destination **40.40.40.1**, and connections between the endpoints continue to flow through the TMM Proxy Pod for the life of the connection

Required CNFs CRs

NAT64 requires CNFs Custom Resources (CRs) installed in this order:

- The F5BigDnsApp specifies a DNS listener used to translate and convert DNS queries.
- The F5BigNatPolicy specifies the IPv4 and IPv6 translation addresses.
- The ***optional*** F5BigFwPolicy filters subscriber connections by IP address.
- The ***optional*** F5BigLogProfile sends connection events to remote logging servers.
- The F5BigContextSecure processes and load balances subscriber connections.

This document describes, and guides you through the DNS64 CR implementation.

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- Installed the dSSM Database for F5BigNatPolicy configurations.
- A Linux based workstation.

Installation

Use this installation procedure to configure the TMM Proxy Pod for NAT64 functionality.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. ***Optional:*** The example F5BigLogHslpub CR specifies a remote server with IP/port **[2002::10:30:2:220]:514**, and the **udp** protocol. Copy and paste the example into a YAML file:

   Note: *The F5BigLogHslpub CR will be referenced by the F5BigLogProfile.*

```
apiVersion: k8s.f5net.com/v1
kind: F5BigLogHslpub
metadata:
  name: "cnf-hsl-pub"
  namespace: "cnf-gateway"
spec:
  pool:
  - name: "hsl-pool"
    endpoint:
    - "[2002::10:30:2:220]:514"
  syslog:
  - name: "cnf-syslog"
    format: "rfc5424"
    protocol: "udp"
    pool: "hsl-pool"
```

2. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

3. ***Optional:*** The example F5BigLogProfile CR captures NAT events such as connection **start** and **end**, and firewall events such as packet **match** and **drop**. Copy and paste the CR nto a YAML file:

   Note: *The F5BigLogProfile CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-logs"
  nat:
    enabled: true
    logSubscriberID: true
    publisher: "cnf-hsl-pub"
    inbound:
      start:
        mode: "enabled"
      end:
        mode: "enabled"
    quotaExceeded:
      mode: "enabled"
    errors:
      mode: "enabled"
  firewall:
    enabled: true
    network:
      publisher: "cnf-hsl-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
```

```
        tcpEvents: true
        translationFields: true
```

4. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-log-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/cnf-log-profile was added/updated
```

5. The example F5BigDnsApp CR receives DNS queries on IP **2002::10:20:2:10**, and sends the query to **10.30.2.220** for resolution. IPv4 only responses are converted and appended to the NAT64 well-known prefix **64:ff9b::**. Copy and paste the example CR into a YAML file:

***Note:** The F5BigDnsApp CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "dns.k8s.f5net.com/v1"
kind: F5BigDnsApp
metadata:
  name: "cnf-dns-64"
  namespace: "cnf-gateway"
spec:
  destination:
    ipv6Address: "2002::10:20:2:10"
    port: 53
  snat:
    type: "automap"
  dns:
    dns64Mode: "secondary"
    dns64Prefix: "64:ff9b::"
    dns64AdditionalSectionRewrite: "v4-only"
  ipProtocol: "udp"
  udp:
  pool:
    members:
      - address: "10.30.2.220"
        port: 53
```

6. Install the F5BigDnsApp CR:

```
kubectl apply -f cnf-dns-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigDnsApp CR was **added/updated**:*

```
I0202 12:00:00.12345   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Dns",
F5Dns cnf-gateway/cnf-dns-64 was added/updated
```

7. The example F5BigNatPolicy CR NATs subscriber connections sourced from IPv6 prefix **2002::10:20:2:0/112**, and destined to the IPv6 prefix **64:ff9b::0/96**. Subscribers source IP addresses will NAT to an IPv6 address within the **10.200.2.1-10.200.2.11** range, and be sent to the server's IPv4 address. Copy and paste the CR into a YAML file:

***Note:** The F5BigNatPolicy CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNatPolicy
metadata:
```

```
  name: "cnf-nat-64"
  namespace: "cnf-gateway"
spec:
  sourceTranslation:
    - name: "trans-64"
      type: "dynamic-pat"
      addresses:
        - "10.200.2.1-10.200.2.11"
      port: "8000-8500"
      routeAdvertisement: true
      icmpEcho: true
      proxyArp: true
  rule:
    - name: rule-ip64
      ipProtocol: any
      source:
        addresses:
         - "2002::10:20:2:0/112"
      destination:
        addresses:
         - "64:ff9b::0/96"
      sourceTranslation: "trans-64"
```

8. Install the F5BigNatPolicy CR:

```
kubectl apply -f cnf-nat-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigNatPolicy CR was **added/updated**:*

```
I0202 12:00:00.12345   1 event.go:282 Event(v1.ObjectReference{Kind:"F5NatPolicy",
NatPolicy cnf-gateway/cnf-nat-64 was added/updated
```

9. ***Optional:*** The example F5BigFwPolicy allows connections only from the IPv6 prefix **2002::10:20:2:0/112**. Copy and past the CR into a YAML file:

   Note: *The F5BigFwPolicy CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFwPolicy
metadata:
  name: "cnf-fw-64"
  namespace: "cnf-gateway"
spec:
  rule:
   - name: allow-10-20
     action: "accept"
     logging: true
     ipProtocol: any
     source:
       addresses:
         - "2002::10:20:2:0/112"
   - name: drop-all
     action: "drop"
     logging: true
     ipProtocol: any
     source:
```

```
      addresses:
        - "::/0"
```

10. Install the F5BigFwPolicy CR:

```
kubectl apply -f cnf-fw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigFwPolicy CR was **added/updated**:*

```
I0202 12:00:00.12346   1 event.go:282
 ↪  Event(v1.ObjectReference{Kind:"F5FirewallPolicy",
FirewallPolicy cnf-gateway/cnf-fw-64 was added/updated
```

11. The F5BigContextSecure CR listens for connections destined to the **64:ff9b::0/96** prefix on service port **80**, and processes application traffic by referencing the installed CRs. Copy and paste the CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-64-context"
  namespace: "cnf-gateway"
spec:
   ipv6destinationAddress: "::/0"
   destinationPort: 80
   firewallEnforcedPolicy: "cnf-fw-64"
   natPolicy: "cnf-nat-64"
   logProfile: "cnf-log-profile"
   ipProtocol: "any"
   profile: "fastL4"
```

12. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-64-context.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
SecureContext cnf-gateway/cnf-64-context was added/updated
```

13. Continue to the Traffic statistics section after testing the application.

Traffic statistics

If you have installed the [TMM Debug] container, use the following steps to gather traffic processing statistics for the F5BigNatPolicy, F5BigFwPolicy and F5BigContextSecure CRs.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify F5BigNatPolicy connection statistics:

```
tmctl -d blade fw_nat_rule_stat
```

```
context_type context_name                           rule_name
------------ ---------------------------------------  --------------------------
virtual   cnf-cnf-context-secure-SecureContext_vs  10-20-subnet-natpolicyrule

micro_rules counter last_hit_time action
----------- ------- ------------- ------
          1       8    1643836695      0
```

```
tmctl -d blade fw_nat_trans_stat -s type,name,translation_requests
```

```
type         name        translation_requests
------------ ----------- --------------------
fw_src_trans transparent                    8
fw_dst_trans transparent                    8
fw_src_trans automap                        0
```

3. Verify the F5BigNatPolicy client IP address mappings:

```
lsndb list all
```

```
Client                                                Connections
------------------------------------------------------------
0 client with 0 connection found.
LSN Persistence Entries
Client                                                Translation
------------------------------------------------------------
10.20.2.220:52110                               10.200.2.8:8265
10.20.2.220                                     10.200.2.8
2 persist entries found.
LSN port block allocations
Client                                                Port block
------------------------------------------------------------
0 port block entries found.
LSN Inbound Mapping Entries
Translation                                             Client
------------------------------------------------------------
10.200.2.8:8265                           10.20.2.220:52110
10.200.2.7:8397                           10.20.2.220:52106
```

4. Verify the F5BigFwPolicy statistics:

```
tmctl -d blade fw_rule_stat -s rule_name,counter,last_hit_time,action
```

```
rule_name                 counter last_hit_time action
------------------------- ------- ------------- ------
allow-4-firewallpolicyrule       1    1646355700      2
allow-6-firewallpolicyrule       1    1646355702      2
drop-4-firewallpolicyrule        0             0      0
drop-6-firewallpolicyrule        0             0      0
```

5. Verify the F5BigDnsApp DNS Profile statistics:

```
tmctl -d blade profile_dns_stat -s name,queries,responses
```

```
name                                queries responses
----------------------------------- ------- ---------
cnf-gateway-cnf-dns-64-profile_dns      20        20
```

6. Verify the F5BigContextSecure, and F5BigDnsApp virtual server statistics:

```
tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
```

```
name                                    clientside.tot_conns
--------------------------------------- --------------------
cnf-gateway-ipv64-vip-SecureContext_vs                    15
cnf-gateway-dns-64-virtual_server                         20
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Wikipedia NAT64 overvew.

CNFs NAT46

Overview

Cloud-Native Network Functions (CNFs) NAT46 provides the ability to process IPv4 to IPv6 application traffic, specifically between IPv4 only clients and IPv6 only servers. To provide NAT46, the Service Proxy Traffic Management Microkernel (TMM) Proxy Pod uses an IPv4 virtual server to receive connections, and load balancing connections to IPv6 pool members as the packet destinations. The Service Proxy TMM Pod translate the source IP address of egress network packets to IPv6 using source network address translation (SNAT).

Connection example

When the IPv4 client requires a connection to IPv6 server **www.46test.com**, the TMM Proxy Pod's virtual server receives the connection for **10.30.2.1**, and load balances the connection to one of the pool members using the pool member's IP address **2002::10:30:2:220** as the destination. When the IPv4 client connects to **10.30.2.1** through the TMM Proxy Pod, the network packet is sent to destination address **2002::10:30:2:220**, using the source IP address **2002::10:30:2:111** selected from the configured SNAT pool. Connections between the endpoints continue to flow through the TMM Proxy Pod for the life of the connection.

Required CNFs CRs

NAT46 requires CNFs Custom Resources (CRs) installed in this order:

- The F5BigNetVlan specifies clientside IPv4 and serverside IPv6 addresses.
- The F5BigCneSnatpool specifies IPv6 addresses used to translate subscriber connections.
- The ***optional*** F5BigFwPolicy filters application connections by IP address.
- The ***optional*** F5BigLogProfile sends connection events to remote logging servers.
- The F5BigContextSecure processes and load balances subscriber connections.

This document describes, and guides you through the DNS46 implementation.

Requirements

Ensure you have:

- Installed the BIG-IP Controller.
- Installed the dSSM Database for F5BigNatPolicy configurations.
- A Linux based workstation.

Installation

Use this installation procedure to configure NAT46.

Tip: *Open a second shell to view the CNFs Event Logs while installing.*

1. The example F5BigNetVlan CR configures IPv4 addresses on the clientside, and IPv6 addresses on the serverside for up to **2** TMM Proxy Pods. Copy and paste the example CR into a YAML file:

   Note: *You can configure both IPv4 and IPv6 addresses on each VLAN.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetVlan
metadata:
  name: "subscriber-vlan"
  namespace: "cnf-gateway"
spec:
  name: clientside
  interfaces:
    - "1.1"
  selfip_v4s:
    - 10.10.10.100
    - 10.10.10.101
  prefixlen_v4: 24
  cmp_hash: DST_ADDR
---
apiVersion: "k8s.f5net.com/v1"
kind: F5BigNetVlan
metadata:
  name: "application-vlan"
  namespace: "cnf-gateway"
spec:
  name: serverside
  interfaces:
    - "1.2"
  selfip_v6s:
    - 2002::192:168:10:100
    - 2002::192:168:10:101
  prefixlen_v6: 116
  mtu: 9000
  cmp_hash: SRC_ADDR
```

2. Install the F5BigNetVlan CR:

   *In this example, the BIG-IP Controller logs indicate the F5BigNetVlan CRs were **added/updated**:*

```
I0613 12:00:00:12345    1 event.go:282] Event(v1.ObjectReference{Kind:"F5Vlan",
F5Vlan cnf-gateway/subscriber-vlan was added/updated
```

```
I0613 12:00:00:12345    1 event.go:282] Event(v1.ObjectReference{Kind:"F5Vlan",
F5Vlan cnf-gateway/application-vlan was added/updated
```

3. The example F5BigCneSnatpool CR configures up to **2** TMM Pods with **4** IPv6 address to translate egress connections. Copy and paste the example CR into a YAML file:

   ***Note:** The F5BigCneSnatpool CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigCneSnatpool
metadata:
  name: "46-snatpool"
  namespace: "cnf-gateway"
spec:
  name: "46_snatpool"
  addressList:
    - - 2002::10:30:2:111
      - 2002::10:30:2:112
      - 2002::10:30:2:113
```

```
      - 2002::10:30:2:114
    - - 2002::10:30:2:115
      - 2002::10:30:2:116
      - 2002::10:30:2:117
      - 2002::10:30:2:118
```

4. Install the F5BigCneSnatpool CR:

```
kubectl apply -f f5-cnf-snatpool.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigCneSnatpool CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282] Event(v1.ObjectReference{Kind:"F5Snatpool",
F5Snatpool cnf-gateway/46_snatpool was added/updated
```

5. ***Optional:*** The example F5BigLogHslpub CR specifies a remote server with IP/port **[2002::10:30:2:220]:514**, and the **udp** protocol. Copy and paste the example into a YAML file:

   Note: *The F5BigLogHslpub CR will be referenced by the F5BigLogProfile.*

```
apiVersion: k8s.f5net.com/v1
kind: F5BigLogHslpub
metadata:
  name: "cnf-hsl-pub"
  namespace: "cnf-gateway"
spec:
  pool:
  - name: "hsl-pool"
    endpoint:
    - "[2002::10:30:2:220]:514"
  syslog:
  - name: "cnf-syslog"
    format: "rfc5424"
    protocol: "udp"
    pool: "hsl-pool"
```

6. Install the F5BigLogHslpub CR:

```
kubectl apply -f cnf-hsl-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogHslpub CR was **added/updated**:*

```
I0202 12:00:00.12347   1 event.go:282 Event(v1.ObjectReference{Kind:"F5Hslpub",
F5Hslpub cnf-gateway/cnf-hsl-pub was added/updated
```

7. ***Optional:*** The example F5BigLogProfile CR captures firewall events such as packet **match** and **drop**. Copy and paste the CR nto a YAML file:

   Note: *The F5BigLogProfile CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigLogProfile
metadata:
  name: "cnf-log-profile"
  namespace: "cnf-gateway"
spec:
  name: "cnf-logs"
  firewall:
```

```
    enabled: true
    network:
      publisher: "cnf-hsl-pub"
      events:
        aclMatchAccept: true
        aclMatchDrop: true
        tcpEvents: true
        translationFields: true
```

8. Install the F5BigLogProfile CR:

```
kubectl apply -f cnf-log-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigLogProfile CR was **added/updated**:*

```
I0202 12:00:00.12348   1 event.go:282 Event(v1.ObjectReference{Kind:"F5LogProfile",
LogProfile cnf-gateway/log_profile1 was added/updated
```

9. ***Optional:*** The example F5BigFwPolicy CR allows connectivity only from the IPv4 prefix **10.20.2.0/24**. Copy and paste the CR into a YAML file:

   Note: *The F5BigFwPolicy CR will be referenced by the F5BigContextSecure CR.*

```
apiVersion: "k8s.f5net.com/v1"
kind: F5BigFwPolicy
metadata:
  name: "cnf-fw-46"
  namespace: "cnf-gateway"
spec:
  rule:
   - name: allow-10-20
     action: "accept"
     logging: true
     ipProtocol: any
     source:
       addresses:
         - "10.20.2.0/24"
   - name: drop-all
     action: "drop"
     logging: true
     ipProtocol: any
     source:
       addresses:
         - "0.0.0.0/0"
```

10. Install the F5BigFwPolicy CR:

```
kubectl apply -f cnf-fw-cr.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigFwPolicy CR was **added/updated**:*

```
I0202 12:00:00.12346   1 event.go:282
 ↪  Event(v1.ObjectReference{Kind:"F5FirewallPolicy",
FirewallPolicy cnf-gateway/cnf-fw-46 was added/updated
```

11. The FF5BigContextSecure CR accepts packets on the **subscriber-vlan** interface destined to the **10.30.2.0/24** IPv4 prefix, and load balances the packets to the **2002::10:30:2:220** and **2002::10:30:2:221** destination IP ad-

dresses. A translated client source IPv6 address will be selected from the installed F5BigCneSnatpool. Copy and paste the CR into a YAML file:

```
apiVersion: k8s.f5net.com/v1
kind: F5BigContextSecure
metadata:
  name: "cnf-46-context"
  namespace: "cnf-gateway"
spec:
   destinationAddress: "10.30.2.0/24"
   destinationPort: 0
   ipProtocol: "any"
   profile: "fastL4"
   firewallEnforcedPolicy: "cnf-fw-46"
   logProfile: "cnf-log-profile"
   vlans:
      vlanList:
        - "subscriber-vlan"
   snat:
     type: "snat"
     pool: "46_snatpool"
   pool:
     members:
       - address: "2002::10:30:2:220"
       - address: "2002::10:30:2:221"
```

12. Install the F5BigContextSecure CR:

```
kubectl apply -f f5-cnf-46-context.yaml
```

*In this example, the BIG-IP Controller logs indicate the F5BigContextSecure CR was **added/updated**:*

```
I0202 12:00:00:12350    1 event.go:282]
 ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
F5SecureContext cnf-gateway/cnf-46-context was added/updated
```

13. Continue to the **Traffic statistics** section.

Traffic statistics

If you have installed the [TMM Debug] container, use the following steps to gather traffic processing statistics for the F5BigNatPolicy, F5BigFwPolicy and F5BigContextSecure CRs.

1. Log in to the TMM debug Pod:

   *In this example, the TMM debug container is in the **cnf-gateway** namespace:*

```
kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
```

2. Verify F5BigNatPolicy connection statistics:

```
tmctl -d blade fw_nat_rule_stat
```

```
context_type context_name                          rule_name
------------ ------------------------------------  --------------------------
virtual      cnf-cnf-context-secure-SecureContext_vs  10-20-subnet-natpolicyrule
```

```
micro_rules counter last_hit_time action
----------- ------- ------------- ------
          1       8    1643836695      0
```

```
tmctl -d blade fw_nat_trans_stat -s type,name,translation_requests
```

```
type         name        translation_requests
------------ ----------- --------------------
fw_src_trans transparent                    8
fw_dst_trans transparent                    8
fw_src_trans automap                        0
```

3. Verify the F5BigNatPolicy client IP address mappings:

```
lsndb list all
```

```
Client                                                Connections
------------------------------------------------------------
0 client with 0 connection found.
LSN Persistence Entries
Client                                                Translation
------------------------------------------------------------
10.20.2.220:52110                               10.200.2.8:8265
10.20.2.220                                     10.200.2.8
2 persist entries found.
LSN port block allocations
Client                                                Port block
------------------------------------------------------------
0 port block entries found.
LSN Inbound Mapping Entries
Translation                                              Client
------------------------------------------------------------
10.200.2.8:8265                               10.20.2.220:52110
10.200.2.7:8397                               10.20.2.220:52106
```

4. Verify the F5BigFwPolicy statistics:

```
tmctl -d blade fw_rule_stat -s rule_name,counter,last_hit_time,action
```

```
rule_name                  counter last_hit_time action
-------------------------- ------- ------------- ------
allow-4-firewallpolicyrule       1    1646355700      2
allow-6-firewallpolicyrule       1    1646355702      2
drop-4-firewallpolicyrule        0             0      0
drop-6-firewallpolicyrule        0             0      0
```

5. Verify the F5BigDnsApp DNS Profile statistics:

```
tmctl -d blade profile_dns_stat -s name,queries,responses
```

```
name                               queries responses
---------------------------------- ------- ---------
cnf-gateway-cnf-dns-64-profile_dns      20        20
```

6. Verify the F5BigContextSecure, and F5BigDnsApp virtual server statistics:

```
tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
```

```
name                                    clientside.tot_conns
--------------------------------------- --------------------
cnf-gateway-ipv64-vip-SecureContext_vs                    15
cnf-gateway-dns-64-virtual_server                         20
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Wikipedia NAT64 overvew.

BGP Overview

Overview

To scale the Cloud-Native Network Functions (CNF) Traffic Management Microkernel (TMM), learn and announce routing prefixes between networks, and to advertise subscriber NAT IP addresses to downstream routers, Border Gateway Protocol (BGP) sessions must be established on TMM's upstream and downstream interfaces. The TMM Proxy Pod's **f5-tmm-routing** container can be enabled and configured when installing the BIG-IP Controller. By default, the **f5-tmm-routing** container is disabled.

Review the sections below for BGP configuration assistance:

- Scaling TMM Pods
- Advertising IP addresses
- Filtering IP addresses
- Enabling BFD
- Troubleshooting

Note: *Review CNF BGP topology for a high-level overview.*

Parameters

The tables below describe the BIG-IP Controller's `tmm.dynamicRouting.tmmRouting.config` Helm parameters.

tmm.dynamicRouting

Parameter	Description
`enabled`	Enables the **f5-tmm-routing** container: **true** or **false** (default).
`exportZebosLogs`	Enables sending **f5-tmm-routing** logs to Fluentd Logging: **true** (default) or **false**.

tmm.dynamicRouting.tmmRouting.config.bgp

Configure and establish BGP peering relationships.

Parameter	Description
`asn`	The AS number of the **f5-tmm-routing** container.
`hostname`	The hostname of the **f5-tmm-routing** container.
`logFile`	Specifies a file used to capture BGP logging events: **/var/log/zebos.log**.
`debugs`	Sets the BGP logging level to debug for troublshooting purposes: **["bgp"]**. It is not recommended to run in debug level for extended periods.
`secret`	Set the name of the Kubernetes secret containing the BGP neighbor password. See the BGP Secret section below.
`neighbors.ip`	The IPv4 or IPv6 address of the BGP peer.
`neighbors.asn`	The AS number of the BGP peer.

Parameter	Description
`neighbors.password`	The BGP peer MD5 authentication password. **Note:** The password is stored in the **f5-tmm-dynamic-routing** configmap unencrypted.
`neighbors.ebgpMultihop`	Enables connectivity between external peers that do not have a direct connection (1-255).
`neighbors.acceptsIPv4`	Enables advertising IPv4 virtual server addresses to the peer (true / false). The default is **false**.
`neighbors.acceptsIPv6`	Enables advertising IPv6 virtual server addresses to the peer (true / false). The default is **false**.
`neighbors.softReconf`	Enables BGP4 policies to be activated without clearing the BGP session.
`neighbors.maxPathsEbgp`	The number of parallel eBGP (external peer) routes installed. The default is **2**.
`neighbors.maxPathsIbgp`	The number of parallel iBGP (internal peer) routes installed. The default is **2**.
`neighbors.fallover`	Enables bidirectional forwarding detection (BFD) between neighbors (true / false). The default is **false**.
`neighbors.routeMap`	References the `routeMaps.name` parameter, and applies the filter to the BGP neighbor.

tmm.dynamicRouting.tmmRouting.config.prefixList

Create prefix lists to filter specified IP address subnets.

Parameter	Description
`name`	The name of the `prefixList` entry.
`seq`	The order of the `prefixList` entry.
`deny`	Allow or deny the `prefixList` entry.
`prefix`	The IP address subnet to filter.

tmm.dynamicRouting.tmmRouting.config.routeMaps

Create route maps that apply to BGP neighbors, referencing specified prefix lists.

Parameter	Description
`name`	The name of the `routeMaps` object applied to the BGP neighbor.
`seq`	The order of the `routeMaps` entry.
`deny`	Allow or deny `routeMaps` entry.
`match`	The name of the referenced `prefixList`.

tmm.dynamicRouting.tmmRouting.config.bfd

Enable BFD and configure the control packet intervals.

Parameter	Description
`interface`	Selects the BFD peering interface.
`interval`	Sets the minimum transmission interval in milliseconds (50-999).
`minrx`	Sets the minimum receive interval in milliseconds (50-999).
`multiplier`	Sets the Hello multiplier value (3-50).

BGP Secrets

BGP neighbor passwords can be stored as Kubernetes secrets using the `bgpSecret` parameter described in the BGP Parameters section above. When using Secrets, the value must be the `neighbor.ip`, and the data must be the base64 encoded password. When using IPv6, replace any colon **:** characters, with dash **-* characters. For example:

```
apiVersion: v1
kind: Secret
metadata:
 name: bgp-secret
 namespace: cnf-gateway
data:
 10.1.2.3: c3dvcmRmaXNo
 2002--10-1-2-3: cGFzc3dvcmQK
```

Scaling TMM Pods

When scaling TMM Pods beyond a single instance in the Namespace, you must configure BGP with Equal-cost Multipath (ECMP) load balancing. With ECMP configured, each TMM replica establishes a peer relationship with the upstream (clientside) router, and each routing prefix TMM advertises will have multiple next-hop entries; one for each TMM self IP address. The upstream router should be configured with an ECMP algorithm that load balances connections across the TMM Pods. F5 recommends ***ECMP Resilient Hashing*** if this option is available.

1. When installing the [BIG-IP Contoller], set the `maxPathsEbgp` parameter to the maxiumum number of TMM replicas you intend to use:

   ***Note**: In this example, the `maxPathsEbgp` parameter configures* **up to_ 4_** TMM Pod replicas:_

   ```
   tmm:
     dynamicRouting:
       enabled: true
       tmmRouting:
         config:
           bgp:
             asn: 100
             maxPathsEbgp: 4
             maxPathsIbgp: 'null'
             hostname: cnf-bgp
             neighbors:
             - ip: 10.10.10.200
               asn: 200
               ebgpMultihop: 10
             - ip: 192.168.10.200
   ```

```
            asn: 400
            ebgpMultihop: 10
```

2. Once the BIG-IP Controller is installed, log in to the peer router and verify the advertised routing prefixes are being advertised with the TMM self-IPs as next hops:

```
show ip bgp
```

*In this example, the TMM replicas with self IP addresses **10.10.10.250** and **10.10.10.251** are advertising the **10.11.12.0/24 subnet**:*

```
Network             Next Hop         Metric     Path
10.11.12.0/24       10.10.10.250          0     200
                    10.10.10.251          0     200
```

3. The external peer routers should now distribute traffic flows to the TMM replicas based on the configured ECMP load balancing algorithm.

Advertising IP addresses

To ensure upstream routers use TMM as a gatway, and to ensure downstream routers correclty route back through TMM, BGP peering relationships should be established on TMM's upstream and downstream interfaces. With BGP relationships, TMM can advertise routes learned from the downstream router, and any destination IP addresses defined in the Traffic Management [CNF CRs] to the upstream router. TMM will also advertise IP addresses defined in the F5BigNatPolicy CR, to ensure the downstream router sends connections back through TMM.

1. When installing the [BIG-IP Contoller], set the `acceptsIPv4` and the `acceptsIPv6` parameters to advertise IPv4 and IPv6 destination addresses:

   ***Note**: In this example, the `acceptsIPv4` and `acceptsIPv6` parameters are set on the upstream BGP peer:*

```
tmm:
  dynamicRouting:
    enabled: true
    tmmRouting:
      config:
        bgp:
          asn: 100
          maxPathsEbgp: 4
          maxPathsIbgp: 'null'
          hostname: cnf-bgp
          neighbors:
          - ip: 10.10.10.200
            asn: 200
            ebgpMultihop: 10
            acceptsIPv4: true
            acceptsIPv6: true
          - ip: 192.168.10.200
            asn: 400
            ebgpMultihop: 10
```

2. Once the BIG-IP Controller is installed, log in to the peer router and verify the advertised routing prefixes are being advertised with the TMM self-IPs as next hops:

```
show ip bgp
```

*In this example, the TMM replicas with self IP addresses **10.10.10.250** and **10.10.10.251** are advertising the CR destination address **192.168.10.100**:*

```
show ip bgp
```

```
Network             Next Hop         Metric     Path
192.168.10.100/32   10.10.10.250          0     200
                    10.10.10.251          0     200
```

Filtering IP addresses

By default, all F5BigNatPolicy IP addresses are advertised (redistributed) to BGP neighbors. To advertise specific NAT IP addresses, configure a `prefixList` defining the IP addresses to advertise, and apply a `routeMap` to the BGP neighbor configuration referencing the `prefexList`. In the example below, only the **10.244.10.0/24** and **10.244.20.0/24** IP address subnets will be advertised to the BGP neighbor:

```
dynamicRouting:
  enabled: true
  tmmRouting:
    config:
      prefixList:
        - name: 10pod
          seq: 10
          deny: false
          prefix: 10.244.10.0/24 le 32
        - name: 20pod
          seq: 10
          deny: false
          prefix: 10.244.20.0/24 le 32

      routeMaps:
        - name: snatpoolroutemap
          seq: 10
          deny: false
          match: 10pod
        - name: snatpoolroutemap
          seq: 11
          deny: false
          match: 20pod

      bgp:
        asn: 100
        hostname: cnf-bgp
        neighbors:
        - ip: 10.10.10.200
          asn: 200
          routeMap: snatpoolroutemap
```

Once the BIG-IP Controller is installed, verify the expected SNAT pool IP addresses are being advertised.

1. Install the F5BigNatPolicy Custom Resource (CR).

2. Log in to the **f5-tmm-routing** container:

```
oc exec -it deploy/f5-tmm -c f5-tmm-routing -n <project> -- bash
```

*In this example, the **f5-tmm-routing** container is in the **cnf-gateway** Project:*

```
oc exec -it deploy/f5-tmm -c f5-tmm-routing -n cnf-gateway -- bash
```

3. Log in IMI shell and turn on privileged mode:

```
imish
en
```

4. Verify the SNAT pool IP addresses are being advertised:

```
show bgp ipv4 neighbors <ip address> advertised-routes
```

In this example, the NAT pool IP addresse lists are being advertised, and TMM's external interface is the next hop:

```
show bgp ipv4 neighbors 10.10.10.200 advertised-routes

     Network          Next Hop            Metric    LocPrf        Weight
*>   10.244.10.1/32   10.20.2.207         0         100           32768
*>   10.244.10.2/32   10.20.2.207         0         100           32768
*>   10.244.20.1/32   10.20.2.207         0         100           32768
*>   10.244.20.2/32   10.20.2.207         0         100           32768

Total number of prefixes 4
```

Enabling BFD

Bidirectional Forwarding Detection (BFD) rapidly detects loss of connectivity between BGP neighbors by exchanging periodic BFD control packets on the network link. After a specified interval, if a control packet is not received, the connection is considered down, enabling fast network convergence. The BFD configuration requires the interface name of the external BGP peer. Use the following command to obtain the external interface name:

```
kubectl get ingressroutevlan <external vlan> -o "custom-columns=VLAN Name:.spec.name"
```

1. When installing the [BIG-IP Contoller], configure the `bfd` for the interface to monitor:

***Note:** In this example, BFD is enabled on TMM's **external** interface:*

```
tmm:
  dynamicRouting:
    enabled: true
    tmmRouting:
      config:
        bgp:
          asn: 100
          maxPathsEbgp: 4
          maxPathsIbgp: 'null'
          hostname: cnf-bgp
          neighbors:
          - ip: 10.10.10.200
            asn: 200
            ebgpMultihop: 10
```

```
          acceptsIPv4: true
          acceptsIPv6: true
        - ip: 192.168.10.200
          asn: 400
          ebgpMultihop: 10
      bfd:
        interface: external
        interval: 100
        minrx: 100
        multiplier: 3
```

2. Once the BIG-IP Controller is installed, verify the BFD configuration is working.

   *In this example, the **f5-tmm-routing** container is in the **cnf-gateway** Namespace:*

```
kubectl exec -it deploy/f5-tmm -c f5-tmm-routing -n cnf-gateway -- bash
```

3. Log in IMI shell and turn on privileged mode:

```
imish
en
```

4. View the bfd session status:

   ***Note:** You can append the **detail** argument for verbose session information.*

```
show bfd session
```

   *In this example, the **Sess-State** is **Up**:*

```
BFD process for VRF: (DEFAULT VRF)
=================================================================================
Sess-Idx   Remote-Disc  Lower-Layer  Sess-Type   Sess-State  UP-Time   Remote-Addr
2          1            IPv4         Single-Hop  Up          00:03:16  10.10.10.200/32
Number of Sessions:    1
```

5. BGP should now quickly detect link failures between neighbors.

Troubleshooting

When BGP neighbor relationships fail to establish, begin troubleshooting by reviewing BGP log events to gather useful diagnostic data. If you installed the Fluentd logging collector, review the **Log file locations** and **Viewing logs** sections of the FLuentd Logging guide before proceeding to the steps below. If the Fluentd logging collector is not installed, use the steps below to verify the current BGP state, and enable and review log events to resolve a simple connectivity issue.

***Note:** BGP connectivity is established over TCP port **179**.*

1. Run the following command to verify the BGP state:

```
kubectl exec -it deploy/f5-tmm -c f5-tmm-routing -n cnf-gateway \
-- imish -e 'show bgp neighbors' | grep state
```

   *In this example, the **BGP state** is **Active**, indicating neighbor relationships are not currently established:*

```
 BGP state = Active
 BGP state = Active
```

2. To enable BGP logging, log in to the **f5-tmm-routing** container:

```
kubectl exec -it deploy/f5-tmm -c f5-tmm-routing -n cnf-gateway \
-- bash
```

3. Run the following commands to enter configuration mode:

```
imish
en
config t
```

4. Enable BGP logging:

```
log file /var/log/zebos.log
```

5. Exit configuration mode, and return to the shell:

```
exit
exit
exit
```

6. View the BGP log file events as they occur:

```
tail -f /var/log/zebos.log
```

In this example, the log messages indicate the peers (neighbors), are not reachable:

```
Jan 01 12:00:00 : BGP : ERROR [SOCK CB] Could not find peer for FD - 11 (error:107)
Jan 01 12:00:01 : BGP : INFO 10.20.2.206-Outgoing [FSM] bpf_timer_conn_retry: Peer
 ↪  down,
Jan 01 12:00:02 : BGP : ERROR [SOCK CB] Could not find peer for FD - 11 (error:107)
Jan 01 12:00:01 : BGP : INFO 10.30.2.206-Outgoing [FSM] bpf_timer_conn_retry: Peer
 ↪  down,
```

7. ***Fix:*** The tag ID on the F5BigNetVlan was set to the correct ID value:

*The messages indicate the neighbors are now **Up**. It can take up to two minutes for the relationships to establish:*

```
Jan 01 12:00:05 : BGP : ERROR [SOCK CB] Could not find peer for FD - 13 (error:107)
Jan 01 12:00:06 : BGP : INFO %BGP-5-ADJCHANGE: neighbor 10.20.2.206 Up
Jan 01 12:00:07 : BGP : ERROR [SOCK CB] Could not find peer for FD - 11 (error:107)
Jan 01 12:00:08 : BGP : INFO %BGP-5-ADJCHANGE: neighbor 10.30.2.206 Up
```

8. The BGP state should now be **Established**:

```
imish -e 'show bgp neighbors' | grep state
```

```
  BGP state = Established, up for 00:00:36
  BGP state = Established, up for 00:00:19
```

9. If the BGP state is still not established, and there are issues other than connectivity, set BGP logging to debug, and continue reviewing the lower-level log events:

```
debug bgp all
```

10. Once the BGP troubleshooting is complete, remove the BGP log and debug configurations:

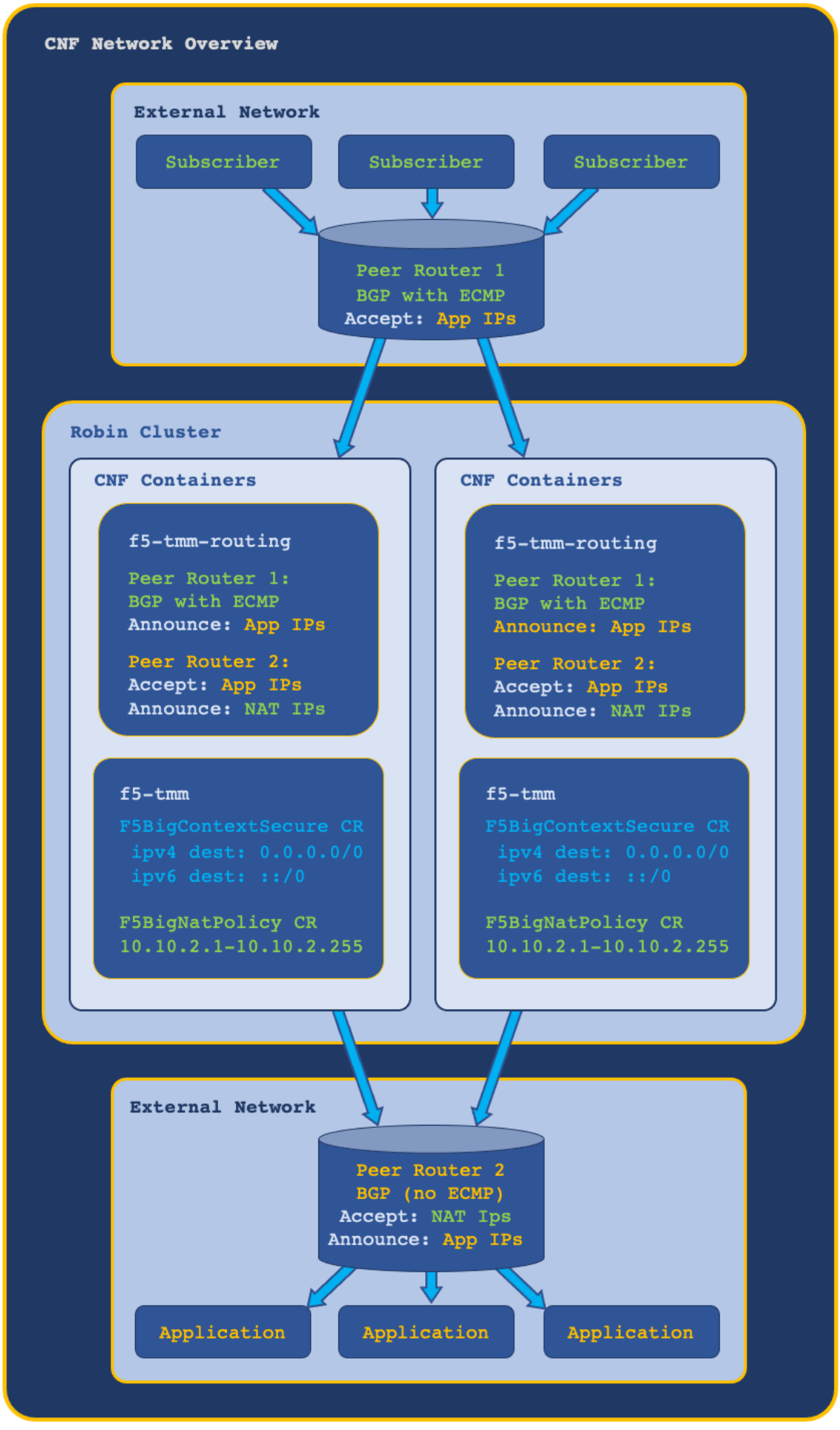
```
no log file
```

```
no debug bgp
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

CNF BGP Topology



Debug Sidecar

Overview

The TMM Proxy Pod's debug sidecar provides a set of command line tools for obtaining low-level, diagnostic data and statistics about the Service Proxy Traffic Management Microkernel (TMM). The debug sidecar deploys by default with the BIG-IP Controller.

Command line tools

The table below lists and describes the available command line tools:

Tool	Description
tmctl	Displays various TMM traffic processing statistics, such as pool and virtual server connections.
bdt_cli	Displays TMM networking information such as ARP and route entries.
tmm_cli	Sets the TMM logging level. For an example, see the **tmm_cli** section below.
mrfdb	Performs read and write dSSM Database operations.
configviewer	Displays a log of the configuration objects created and deleted using SPK Custom Resources (CRs).
qkview	Creates a diagnostic data TAR file for F5 support. See the **Qkview** section below.

Connecting to the sidecar

To connect to the debug sidecar and begin gathering diagnostic information, use the commands below.

1. Connect to the debug sidecar:

   *In this example, the debug sidecar is in the **cnf-gateway** Project:*

   ```
   kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
   ```

2. Execute one of the available diagnostic commands:

   *In this example, **ping** is used to test connectivity to a remote host with IP address **192.168.10.100**:*

   ```
   ping 192.168.10.100
   ```

   ```
   PING 192.168.10.100 (192.168.10.100): 56 data bytes
   64 bytes from 192.168.10.100: icmp_seq=0 ttl=64 time=0.067 ms
   64 bytes from 192.168.10.100: icmp_seq=1 ttl=64 time=0.067 ms
   64 bytes from 192.168.10.100: icmp_seq=2 ttl=64 time=0.067 ms
   64 bytes from 192.168.10.100: icmp_seq=3 ttl=64 time=0.067 ms
   ```

3. Type **Exit** to leave the debug sidecar.

Command examples

tmctl

Use the **tmctl** tool to query Service Proxy TMM for application traffic processing statistics.

Virtual server connections

To view **virtual server** connection statistics run the following command:

Client side statstics

```
tmctl -d blade virtual_server_stat -s name,clientside.tot_conns
```

Server side statstics

```
tmctl -d blade virtual_server_stat -s name,serverside.tot_conns
```

bdt_cli

Use the **bdt_cli** tool to query the Service Proxy TMM for networking data.

1. Connect to TMM referencing the gRPC channel SSL/TL certificates and key:

   ```
   bdt_cli -tls=true -use_fqdn=true -server_addr=tmm0:8850 \
           -ca_file=/etc/ssl/certs/ca_root.crt \
           -client_crt=/etc/ssl/certs/f5-ing-demo-f5ingress.crt \
           -client_key=/etc/ssl/private/f5-ing-demo-f5ingress.key
   ```

2. Once connected, enter a number representing the network data of interest:

   ```
   Enter the request type(number or string):
   1. check
   2. arp
   3. connection
   4. route
   5. exit
   ```

   The output will resemble the following:

   ```
   "2" looks like a number.
   Enter ArpRequest(override fields as necessary, defaults are listed here):
   e.g. {}
   ```

3. Select the **Enter** key again to view the networking data:

   ```
   name:169.254.0.254 ipAddr:169.254.0.254 macAddr:00:01:23:45:67:fe vlan:tmm expire:0
    ↪  status:permanent
   name:169.254.0.253 ipAddr:169.254.0.253 macAddr:00:98:76:54:32:10 vlan:tmm expire:0
    ↪  status:permanent
   name:169.254.0.1 ipAddr:169.254.0.1 macAddr:00:01:23:45:67:00 vlan:tmm expire:0
    ↪  status:permanent
   name:10.244.1.98 ipAddr:10.244.1.98 macAddr:22:22:fe:6d:59:e1 vlan:eth0 expire:0
    ↪  status:permanent
   name:10.20.200.210 ipAddr:10.20.200.210 macAddr:96:b3:23:d4:7c:69 vlan:net1 expire:0
    ↪  status:permanent
   ```

tmm_cli

By default, the **f5-tmm** container logs events at the **Notice** level. You can use the **tmm_cli** command to modify the logging level.

The logging levels listed below generally log messages from the lower severity levels as well.

1-Debug, 2-Informational, 3-Notice, 4-Warning, 5-Error, 6-Critical, 7-Alert, 8-Emergency

1. Connect to the debug sidecar:

   *In this example, the debug sidecar is in the **cnf-gateway** Project:*

   ```
   kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
   ```

2. To set the **f5-tmm** container's logging level to **Debug**, run the following command:

   ```
   tmm_cli -logLevel 1
   ```

   ```
   ok
   ```

   *The **f5-tmm** container will log an event message simlilar to the following:*

   ```
   Set bigdb var 'log.tmm.level'='Debug'
   ```

configviewer

Use the **configviewer** utility to show events related to installing [CNF CRs].

1. You must set the `CONFIG_VIEWER_ENABLE` parameter to `true` when deploying the BIG-IP Controller. For example:

   ```
   tmm:

     customEnvVars:
       - name: CONFIG_VIEWER_ENABLE
         value: "true"
   ```

2. After deploying a Custom Resource (CR), you can view the current configuration event with the following command:

   ***Note:** The example respresents a portion of the TMM configuration.*

   ```
   configviewer --ipport=tmm0:11211 --displayall
   ```

   ```
   GetAll Connect!
   GetAll Connect Complete!
   pattern: 006f40782e*
   binlookup config_viewer_bin
    Query: get/th /6552fc31.0/*

   ------------------------------------------------------------------------------
    ↪  --------------
   Config for pool_member_list updated at <some date / time>
   {
           "name": "apps-nginx-crd-pool-member-list",
           "id": "apps-nginx-crd-pool-member-list",
           "members": [
   ```

```
                "apps-nginx-crd-pool-member-10.244.1.22",
                "apps-nginx-crd-pool-member-10.244.1.23",
                "apps-nginx-crd-pool-member-10.244.2.21"
        ]
}
```

Qkview

The **qkview** utility collects diagnostic and logging information from the **f5-tmm** container, and stores the data in a Linux TAR file. If you enabled the Fluentd Logging collector, run the qkview utility on **f5-fluentd** container to gather log files from all of the SPK Pods. Qkview files are typically generated and sent to F5 for further analysis. Use the steps below to run the **qkview** utility, and copy the file to your local workstation.

1. Obtain the name of the TMM Proxy Pod:

   *In this example, the TMM Proxy Pod is in the **cnf-gateway** namespace.*

   ```
   kubectl get pods --selector app=f5-tmm -n cnf-gateway
   ```

   *In this example, the TMM Proxy Pod is named **f5-tmm-77b95f699f-5zv8n**.*

   ```
   NAME                      READY   STATUS
   f5-tmm-77b95f699f-5zv8n   5/5     Running
   ```

2. Connect to the debug sidecar:

   ```
   kubectl exec -it f5-tmm-77b95f699f-5zv8n -c debug -n cnf-gateway -- bash
   ```

   The shell prompt should contain the name of the TMM Proxy Pod.

   ```
   debuguser@f5-tmm-77b95f699f-5zv8n:~$
   ```

3. Run the **qkview** utility:

   ```
   qkview
   ```

   The command output should indicate the file was created and saved.

   ```
   Diagnostic snapshot file saved: qkview.20220511-185024.tar.gz
   ```

4. Copy the file to your workstation:

   ```
   kubectl cp <namespace>/<podname>:<file> ./<file> -c debug
   ```

   *In this example, the qkview named **qkview.20220511-185024.tar.gz** is copied to the local workstation.*

   ```
   kubectl cp cnf-gateway/f5-tmm-77b95f699f-k8bfh:qkview.20220511-185024.tar.gz \
   ./qkview.20220511-185024.tar.gz -c debug
   ```

5. Obtain the name of the Fluent logging Pod:

   ```
   kubectl get pods --selector run=f5-fluentd -n cnf-gateway
   ```

   *In this example the Fluentd logging Pod is named **f5-toda-fluentd-84f96b6757-v5wj9**.*

   ```
   f5-toda-fluentd-84f96b6757-v5wj9       1/1     Running
   ```

6. Connect to the Fluentd logging Pod:

```
kubectl exec -it f5-toda-fluentd-84f96b6757-v5wj9 -n cnf-gateway -- bash
```

7. Run the **qkview** utility:

```
qkview
```

The command output should indicate the file was created and saved.

```
Diagnostic snapshot file is saved: qkview.20220511-195129.tar.gz
```

8. Copy the file to your workstation:

```
kubectl cp <namespace>/<podname>:<file> ./<file>
```

*In this example, the qkview named **qkview.20220511-195129.tar.gz** is copied to the local workstation.*

```
kubectl cp cnf-gateway/ff5-toda-fluentd-84f96b6757-v5wj:qkview.20220511-195129.tar.gz
 ↪ \
./qkview.20220511-195129.tar.gz
```

Disabling the sidecar

The TMM debug sidecar installs by default with the SPK Controller. You can disable the debug sidecar by setting the `debug.enabled` parameter to `false` in the BIG-IP Controller Helm values file:

```
debug:
  enabled: false
```

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

CNFs Event Logs

Identifying errors while performing CNFs installation procedures will save considerable post-installation troubleshooting time; helping you to avoid complicated configuration reviews and log file analysis. This document demonstrates how to view CNFs event logs using the standard `kubectl logs` command, and how to look for errors that may occur during installations. The CNFs event logs may not be available when Fluentd has been configured to collect the Controller and Traffic Management Microkernel (TMM) logs. Review the Fluentd Logging overview for more information.

BIG-IP Controller

The BIG-IP Controller listens to the Kubernetes API for CNFs' Custom Resource (CR) installation events. When one of the CNFs CRs is installed, the BIG-IP Controller processes the CR data, logs both CR processing and configuration data, and if the configuration passes validation, configures the TMM Proxy Pod.

To filter the BIG-IP Contoller logs for errors while installing a CNFs CR, and also to gather the CR configuration logged during the installation, you must open two separate shells. Use the steps below to view the CNF BIG-IP Controller logs while installing a simple F5BigContextSecure CR:

1. Using ***shell 1***, obtain the name of the BIG-IP Controller Pod:

   *In this example, the BIG-IP Controller is in the **cnf-gateway** Namespace:*

   ```
   kubectl get pods -n cnf-gateway | grep ingress
   ```

   *In this example, the BIG-IP Controller name is **f5ingress-f5ingress-6c4ffd7886-zxpp7**:*

   ```
   f5ingress-f5ingress-6c4ffd7886-zxpp7   2/2     Running
   ```

2. Stream the Controller events in real time and filter for errors using the `-f` option and `grep` command. The `kubectl logs` command uses the Pod name twice; once with the hash suffix to target the Pod, and once without the hash to target the container:

   ```
   kubectl logs -f f5ingress-f5ingress-6c4ffd7886-zxpp7 -c f5ingress-f5ingress -n
    ↪  cnf-controller | grep -iE 'error|added'
   ```

3. Using ***shell 2***, install the CNF CR, and look for error log messages in ***shell 1***. Error log messages begin with a capital **E**:

   ```
   kubectl apply -f cnf-contextsecure-cr.yaml
   ```

   *An **Error** message is logged indicating the F5BigContextSecure CR `ipProtocol` value **any** does not match the `profile` value **tcp**:*

   ```
   E0127 12:00:00.12345  1 controller_securecontext.go:95] Error with
    ↪  AddOrUpdateSecureContext configuration for cnf-gateway/cnf-context-secure: Error
    ↪  Validating SecureContext: Profile 'tcp' incompatible with ipProtocol 'any'
   ```

4. ***Fix:*** The F5BigContextSecure `ipProtocol` parameter was set to **tcp** to match the `profile` parameter value, and the CR was reinstalled:

   ```
   kubectl apply -f cnf-contextsecure-cr.yaml
   ```

   *An **added/updated** message is logged indicating the F5BigContextSecure CR was successfully added to TMM:*

   ```
   I0202 12:00:00:1234    1 event.go:282]
    ↪  Event(v1.ObjectReference{Kind:"F5SecureContext",
   SecureContext cnf-gateway/cnf-context-secure was added/updated
   ```

5. Using ***shell 2***, capture the BIG-IP Controller configuration logs sent to TMM, using the `>filename.txt` and `--since=` options to capture and limit the amount of data:

   *In this example, the last **2** minutes of Controller logs are saved to the **controller-logs.txt** file:*

```
kubectl logs f5ingress-f5ingress-6c4ffd7886-zxpp7 -c f5ingress-f5ingress -n
 ↪  cnf-controller --since=2m > controller-logs.txt
```

6. Use an editor to view the Controller logging data and the configuration sent to TMM:

```
I0129 01:00:00.34760     1 controller_securecontext.go:124] Adding SecureContext:
 ↪  cnf-context-secure
I0129 01:00:00.36214     1 grpccfg2.go:97] Create mTLS secure context.
I0129 01:00:00.36300     1 grpccfg2.go:389] gRPC - Send GRPC Message:

                 {
                        "embedded": {
                        "@type": "type.googleapis.com/declTmm.create_msg",
                        "revision": 0,
                        "embedded": {
                             "@type": "type.googleapis.com/declTmm.virtual_server",
                           "id": "cnf-gateway-cnf-context-secure-SecureContext_vs",
                                "enabled": true,
                               "source_address_translation_type": "SRC_TRANS_NONE",
                                "default_pool": "",
                                "traffic_matching_criteria": "cnf-gateway-cnf-
                                 ↪  context-secure-SecureContext_tmc",
                                "security_nat_policy": "cnf-nat-policy-natpolicy",
                                "security_firewall_enforced_policy":
                                 ↪  "cnf-fw-policy-firewallpolicy",
                                "security_log_profile":
                                 ↪  "cnf-log-profile-securitylogprofile",
                                "no_translate": true,
                                "irules_reference": [
                                 ]
                        }
                        }
                 }
```

TMM Proxy Pod

The TMM Proxy Pod processes 5G workloads when configured with one or more of the Traffic Management CNFs CRs. To process traffic, the TMM Proxy Pod relies on supporting Pods, for example the dSSM Database. When connectivity failures occur between the PODs, traffic management processing failures may also occur. Use the steps below to filter the TMM logs for errors connecting to the dSSM (redis) Database.

***Note:** To modify the TMM logging level, review the **tmm_cli** section of the Debug Sidecar overview.*

1. Obtain the name of the TMM Proxy Pod(s):

   *In this example, the TMM Proxy Pod is in the **cnf-gateway** Namespace:*

```
kubectl get pods -n cnf-gateway | grep tmm
```

   The command output shows there are two TMMs in the Namespace:

```
f5-tmm-dcbc5c4b6-9ldbh                  4/4     Running
f5-tmm-dcbc5c4b6-qkrkh                  4/4     Running
```

2. View the TMM Proxy Pod logs, using the `grep` command to filter for specific events. The `kubectl logs` command uses the Pod name to target the Pod, and **f5-tmm** to target the container:

   In this case, NAT is not working, so the `grep` command will filter for dSSM (redis) database connectivity events:

   ```
   kubectl logs f5-tmm-dcbc5c4b6-9ldbh -n cnf-gateway -c f5-tmm | grep -i redis
   ```

   In this example, the TMM Proxy Pod is unable to connect to the dSSM (redis) database:

   ```
   Jan 01 12:00:00.12345 tmm[12]: redis_reconnect/1255: Retrying redis connect: 5
   redis_sentinel_conn_fail/1031: Connection establishment with REDIS SENTINEL server
    ↪  failed
   redis_reconnect_later/1280: Exceeded max no of retries
   ```

3. ***Fix:*** The TMM `SESSIONDB_EXTERNAL_SERVICE` Helm parameter was set to the correct dSSM hostname **f5-dssm-sentinel.cnf-gateway**, and the BIG-IP Controller was reinstalled. The TMM logs now show successfull connectivity:

   ```
   kubectl logs f5-tmm-dcbc5c4b6-9ldbh -n cnf-gateway -c f5-tmm | grep -i redis
   ```

   ```
   Jan 01 12:00:00:12345 tmm[12]: redis_sentinel_connected/1006: Connection
    ↪  establishment with REDIS SENTINEL server successful
   Jan 01 12:00:00:12346 tmm[12]: redis_pubsub_connected/1418: Pubsub connection
    ↪  established with REDIS server
   Jan 01 12:00:00:12347 tmm[12]: redis_connected/4846: Connection established with
    ↪  REDIS server
   ```

4. Use the `grep -i <string>` option to filter for additional events that may relate to current issues: **grpc**, **vlan**, **endpoint**, etc.

Logging Levels

By default, the **f5-tmm** container logs events at the **Notice** level. For troubleshooting purposes, you can change the default logging level using the BIG-IP Controller Helm values file, or on a currently installed and running **f5-tmm** container using the **tmm_cli** command. The available logging levels are listed below, use either of the procedures to change the default logging level.

1-Debug, 2-Informational, 3-Notice, 4-Warning, 5-Error, 6-Critical, 7-Alert, 8-Emergency

tmm_cli

Use the steps below to modify the logging level for a currently installed and running **f5-tmm** container.

1. Connect to the debug sidecar:

   *In this example, the debug sidecar is in the **cnf-gateway** Project:*

   ```
   kubectl exec -it deploy/f5-tmm -c debug -n cnf-gateway -- bash
   ```

2. To set the **f5-tmm** container's logging level to **Debug**, run the following command:

   ```
   tmm_cli -logLevel 1
   ```

```
ok
```

*The **f5-tmm** container will log an event message simlilar to the following:*

```
Set bigdb var 'log.tmm.level'='Debug'
```

Helm value

Use the steps below to modify the default logging level using the BIG-IP Controller Helm values.yaml file.

1. Edit the Helm values.yaml file, and add the `tmm.logLevel` parameter with the desired logging level value:

   *In the example below, the logging level is set to **Debug**:*

   ```
   tmm:
     logLevel: Debug
   ```

2. Install the BIG-IP Controller.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Review the **Troubleshooting** Section of the BGP Overview

Upgrading dSSM

Overview

The Cloud-Native Network Functions (CNFs) distributed Session State Management (dSSM) Sentinel and DB Pods can be upgraded using the typical Helm upgrade process. However, to ensure the process completes without service interruption, a custom ***dssm-upgrade-hook*** container is deployed during the upgrade. The upgrade process maintains all of the dSSM DB Pod session state data.

This document guides you through upgrading the dSSM database, and verifying the update results are successful.

Note: *If preserving data is not required, refer to the* ***Quick Upgrade*** *section to properly uninstall and upgrade the dSSM installation.*

Requirements

Ensure you have:

- A running CNF dSSM Database installation.
- A newer version of the CNF dSSM Helm chart.
- A workstation with Helm installed.

Procedures

Use the procedures below to upgrade the dSSM database, verify the results, and if required, rollback to the previous installation version.

Pre-upgrade status

Use the step below to verify the dSSM Pod cluster status, software version and persisted data. This will be useful to ensure the upgrade is succesful.

1. Verify the **STATUS** of the dSSM Pods is **Running**:

   In this example, the dSSM Pods are in the ***cnf-gateway*** *Namespace:*

   ```
   kubectl get pods -n cnf-gateway
   ```

   ```
   NAME                              READY   STATUS    RESTARTS
   f5-dssm-db-0                      2/2     Running   0
   f5-dssm-db-1                      2/2     Running   0
   f5-dssm-db-2                      2/2     Running   0
   f5-dssm-sentinel-0                2/2     Running   0
   f5-dssm-sentinel-1                2/2     Running   0
   f5-dssm-sentinel-2                2/2     Running   0
   ```

2. Verify the **f5-dssm-store** version:

   ```
   kubectl describe pods | grep Image: | grep -i dssm
   ```

   In this example, the ***f5-dssm-store*** *is* ***v1.6.1****:*

```
Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.6.1
Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.6.1
Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.6.1
Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.6.1
Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.6.1
Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.6.1
```

3. Log in to the dSSM database (DB):

```
kubectl exec -it f5-dssm-db-0 -- bash
```

4. Enter the Redis command line interface (CLI):

```
redis-cli --tls --cert /etc/ssl/certs/dssm-cert.crt \
--key /etc/ssl/certs/dssm-key.key \
--cacert /etc/ssl/certs/dssm-ca.crt
```

5. List the DB entries. The entries should be present after the upgrade.

```
KEYS *
```

```
1) "0073c3b6eft_dns4610.144.175.221"
2) "0073c3b6eft_dns4610.144.175.222"
3) "0073c3b6eft_dns4610.144.175.224"
4) "0073c3b6eft_dns4610.144.175.223"
5) "0073c3b6eft_dns4610.144.175.220"
```

Software upgrade

Use the steps below to upgrade the dSSM Sentinel and DB Pods.

Note: *The* **dssm-upgrade-hook**_ container logs valuable diagnostic data, opening a second shell to view the data is recommended._

1. To grant the **dssm-upgrade-hook** container access the K8S API, create two YAML files with the following code, and set the `namespace` parameter to the dSSM installation Project:

   Important: *The* ***dssm-upgrade-hook*** *will fail to complete the upgrade without proper access to the K8S API.*

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: pods-list
  namespace: cnf-gateway
rules:
- apiGroups: ["", "apps"]
  resources: ["pods", "statefulset", "statefulsets"]
  verbs: ["get", "delete", "list"]
```

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: pods-list
subjects:
- kind: ServiceAccount
  name: default
```

```
  namespace: cnf-gateway
roleRef:
  kind: Role
  name: pods-list
  apiGroup: rbac.authorization.k8s.io
```

2. Create the `Role` and `RoleBinding` objects:

```
kubectl create -f role.yaml
```

```
kubectl create -f role-binding.yaml
```

3. Verify the `Role` and `RoleBinding` objects have been created:

```
kubectl describe -f role.yaml
```

```
Name:         pods-list
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  pods       []                 []              [get delete]
```

```
kubectl describe -f role-bind.yaml
```

```
Name:         pods-list
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  pods-list
Subjects:
  Kind            Name     Namespace
  ----            ----     ---------
  ServiceAccount  default  spk-utilities
```

4. Obtain the **NAME** of the current dSSM Helm release:

```
helm list -n cnf-gateway
```

*In this example, the dSSM Helm release NAME is **f5-dssm**:*

```
NAME       NAMESPACE       REVISION     STATUS      CHART
f5-dssm    cnf-gateay      1            deployed    f5-dssm-0.9.0
```

5. Upgrade the dSSM database Pods using the newer version Helm chart:

***Note:** The **timeout** value is a precaution; cluster resources may cause the process to go beyond the default **300** seconds.*

```
helm upgrade f5-dssm <chart> --timeout 800s -n cnf-gateway
```

*In this example, the Helm chart version is **UK**:*

```
helm upgrade f5-dssm f5-dssm-0.17.0.tgz --timeout 800s -n cnf-gateway
```

6. To monitor the upgrade status, in the second shell, view the **dssm-upgrade-hook** container logs:

```
kubectl logs -f f5-dssm-hook -n cnf-gateway
```

*The upgrade logs should begin **similar** to the following:*

```
HELM-HOOK IS RUNNING
UPGRADING SENTINELS
Namespace is cnf-gateway
dssm-upgrade-hook IS RUNNING
```

*The upgrade logs should **end** similar to the following:*

```
DONE UPGRADING
Helm-hook pod is going down
pod "dssm-upgrade-hook" deleted
```

Post-upgrade status

Use the steps below to ensure the dSSM software upgrade was successful.

1. List the REVISION (version) of the dSSM Helm releases:

```
helm history f5-dssm -n cnf-gateway
```

```
REVISION    STATUS          CHART             APP VERSION     DESCRIPTION
1           superseded      f5-dssm-0.16.1    v0.16.1         Install complete
2           deployed        f5-dssm-0.17.0    v0.17.0         Upgrade complete
```

2. Verify the dSSM Pod **STATUS** is currently **Running**:

```
kubectl get pods -n cnf-gateway
```

```
NAME                       READY   STATUS
f5-dssm-db-0               2/2     Running
f5-dssm-db-1               2/2     Running
f5-dssm-db-2               2/2     Running
f5-dssm-sentinel-0         2/2     Running
f5-dssm-sentinel-1         2/2     Running
f5-dssm-sentinel-2         2/2     Running
```

3. Verify the **f5-dssm-store** version of the dSSM Pods:

```
kubectl describe pods -n cnf-gateway | grep Image: | grep -i dssm
```

```
 Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.17.0
 Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.17.0
 Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.17.0
 Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.17.0
 Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.17.0
 Image:         artifactory.f5net.com/f5-mbip-docker/f5-dssm-store:v1.17.0
```

4. Verify the dSSM Pod **STATUS** is currently **Running**:

   ***Note:** It may take a few minutes for the rollback to complete.*

```
kubectl get pods -n cnf-gateway
```

```
NAME                     READY   STATUS
f5-dssm-db-0             2/2     Running
f5-dssm-db-1             2/2     Running
f5-dssm-db-2             2/2     Running
f5-dssm-sentinel-0       2/2     Running
f5-dssm-sentinel-1       2/2     Running
f5-dssm-sentinel-2       2/2     Running
```

5. Log in to the dSSM database (DB):

```
oc exec -it f5-dssm-db-0 -n cnf-gateway -- bash
```

6. Enter the Redis command line interface (CLI):

```
redis-cli --tls --cert /etc/ssl/certs/dssm-cert.crt \
--key /etc/ssl/certs/dssm-key.key \
--cacert /etc/ssl/certs/dssm-ca.crt
```

7. List the DB entries. These entries should be the same as the pre-upgrade check.

```
KEYS *
```

```
1) "0073c3b6eft_dns4610.144.175.221"
2) "0073c3b6eft_dns4610.144.175.222"
3) "0073c3b6eft_dns4610.144.175.224"
4) "0073c3b6eft_dns4610.144.175.223"
5) "0073c3b6eft_dns4610.144.175.220"
```

8. Delete the `Role` and `RoleBinding` objects:

```
oc delete -f role-binding.yaml
```

```
oc delete -f role.yaml
```

Rollback

If the dSSM database is not performing as expected after the upgrade, rollback to the previous dSSM database version using the steps below:

1. List the current version of the dSSM database:

```
helm list -n cnf-gateway
```

*In this example, the dSSM database version is **v.17.0** and the **REVISION** version is **2**:*

```
NAME         NAMESPACE      REVISION    STATUS      CHART              APP VERSION
f5-dssm      cnf-gateway    2           deployed    f5-dssm-0.17.0     v0.17.0
```

2. Rollback the dSSM database to the previous **REVISION** (installation version):

```
helm rollback f5-dssm <revision> -n <namespace>
```

*In this example, the previous **REVISION** is **1**:*

```
helm rollback f5-dssm 1 -n cnf-gateway
```

3. List the Helm REVISION (installation versions) of the dSSM database:

```
helm history f5-dssm -n cnf-gateway
```

```
REVISION    STATUS         CHART             APP VERSION     DESCRIPTION
1           superseded     f5-dssm-0.16.1    v0.16.1         Install complete
2           superseded     f5-dssm-0.17.0    v0.17.0         Upgrade complete
3           deployed       f5-dssm-0.16.1    v0.16.1         Rollback to 1
```

4. Verify the dSSM Pod **STATUS** is currently **Running**:

   Note: *It may take a few minutes for the rollback to complete.*

```
kubectl get pods -n cnf-gateway
```

```
NAME                      READY   STATUS
f5-dssm-db-0              2/2     Running
f5-dssm-db-1              2/2     Running
f5-dssm-db-2              2/2     Running
f5-dssm-sentinel-0        2/2     Running
f5-dssm-sentinel-1        2/2     Running
f5-dssm-sentinel-2        2/2     Running
```

Quick Upgrade

The quick upgrade section provides a much easier way to upgrade the dSSM database if preserving data is not a requirement. Use the steps below to properly uninstall the current dSSM Database installation and then reinstall using Helm.

1. List the dSSM Helm release:

```
helm list -n <namespace>
```

   *In this exammple, the dSSM database release **f5-dssm** is installed in the **cnf-gateway** Namespace:*

```
helm list -n cnf-gateway
```

```
NAME     NAMESPACE    REVISION   STATUS     CHART           APP VERSION
f5-dssm  spk-utils    1          deployed   f5-dssm-0.16.1  v0.16.1
```

2. Uninstall the dSSM installation:

```
helm uninstall f5-dssm -n cnf-gateway
```

   The commaznd output will appear similar to the following:

```
release "f5-dssm" uninstalled
```

3. List the dSSM PVCs:

```
kubectl get pvc -n cnf-gateway
```

```
NAME                      STATUS   VOLUME
data-f5-dssm-db-0         Bound    pvc-933c17ae-4378-4eac-8d09-65848a1e164e
data-f5-dssm-db-1         Bound    pvc-c843c33b-c277-46f2-bcb1-4ee5db76ea4b
data-f5-dssm-db-2         Bound    pvc-d0f5441b-0e0c-4385-b558-a84e15fc44a9
```

4. Delete each of the PVCs using the PVC **NAME**:

```
kubectl delete pvc data-f5-dssm-db-0 -n cnf-gateway
```

The command output will appear similar to the following:

```
persistentvolumeclaim "data-f5-dssm-db-0" deleted
```

5. Once all of the PVCs have been deleted, reinstall the dSSM DBs using the dSSM Database installation guide.

Feedback

Provide feedback to improve this document by emailing cnfdocs@f5.com.

Supplemental

- Using Helm

BIG-IP Controller Reference

The BIG-IP Controller and Traffic Management Microkernel (TMM) configuration parameters. Each heading below represents the top-level parameter element. For example, to set the Controller's `watchNamespace`, use `controller.watchNamespace`.

controller

Parameters to configure the BIG-IP Controller.

Parameter	Description
`image.repository`	The domain name or IP address of the local container registry.
`watchNamespace`	The Namespace to watch for Service and CRD update events.
`fluentbit_sidecar.enabled`	Enables the fluentbit logging sidecar (true /false). The default is true.
`fluentbit_sidecar.image.repository`	The domain name or IP address of the local container registry.
`fluentbit_sidecar.fluentd.port`	The service port of the Fluend container. The default is 54321.

tmm

Parameters to configure TMM.

Parameter	Description
`image.repository`	The domain name or IP address of the local container registry.
`replicaCount`	Number of SPK TMMs desired in the replicaset.
`hostNetwork`	Enable TMM pods to use host network namespace.
`cniNetworks`	Comma-seperated list of CNI network interfaces used by TMM.
`robinNetworks`	Enables Robin networking: **true** or **false** (default).
`resources.limits.cpu`	The number of TMM threads to allocate.
`resources.limits.hugepages-2Mi`	The amount of hugepages to allocate: (1.5GB X TMM threads) + 512MB.
`resources.limits.memory`	The amount of memory to allocate: (1.5GB X TMM threads) + 512MB.
`serviceAccount.name`	Specifies the serviceAccount the TMM Pod will use. By default TMM uses the **default** serviceAccount.
`vxlan.enabled`	Enable VXLAN configuration for this TMM deployment (true/false).
`vxlan.name`	VXLAN tunnel name.
`vxlan.localIp`	VXLAN local IP address.
`vxlan.selfIp`	VXLAN self IP address.
`vxlan.port`	VXLAN port.
`vxlan.key`	VXLAN key.
`vxlan.staticRouteNodeNetmask`	Netmask for static routes to nodes.

Parameter	Description
`vxlan.staticRoutePoolMemberNetmask`	Netmask for static routes to pool members.

tmm.dynamicRouting

Parameters to configure BGP. For configuration assistance, refer to the BGP Overview.

Parameter	Description
`enabled`	Enable the TMM dynamic routing container.
`trouted.image.repository`	The domain name or IP address of the local container registry.
`tmmRouting.image.repository`	The domain name or IP address of the local container registry.
`tmmRouting.config.bgp.hostname`	Sets the BGP Hostname.
`tmmRouting.config.bgp.logFile`	Sets the name and location for the BGP log file.
`tmmRouting.config.bgp.debugs`	BGP array of debug.
`tmmRouting.config.bgp.asn`	TMM's BGP Autonomous System Number.
`tmmRouting.config.bgp.maxPathsEbgp`	BGP maximum number of paths for External BGP (2-64). Disable with 'null' value.
`tmmRouting.config.bgp.maxPathsIbgp`	BGP maximum number of paths for Internal BGP (2-64). Disable with 'null' value.
`tmmRouting.config.bgp.neighbors`	BGP router array of neighbors.
`tmmRouting.config.bgp.neighbors.ip`	BGP router neighbors IP.
`tmmRouting.config.bgp.neighbors.advertiseIPv4`	Advertise IPv4 virtual server addresses neighbors. true enables - empty string disables.
`tmmRouting.config.bgp.neighbors.advertiseIPv6`	Advertise IPv6 virtual server addresses to neighbors. true enables - empty string disables.
`tmmRouting.config.bgp.neighbors.ebgpMultiHop`	Sets the EBGP TTL (range: 1-255).
`tmmRouting.config.bgp.neighbors.password`	BGP router neighbors Password.
`tmmRouting.config.bgp.gracefulRestartTime`	BGP graceful restart time.
`tmmRouting.config.bgp.routeMap`	The name of the `routeMaps` use to filter neighbor routes.
`tmmRouting.config.prefixList.name`	The name of the `prefixList` entry.
`tmmRouting.config.prefixList.seq`	The order of the `prefixList` entry.
`tmmRouting.config.prefixList.deny`	Allow or deny the `prefixList` entry.
`tmmRouting.config.prefixList.prefix`	The IP address subnet to filter.
`tmmRouting.config.routeMaps.name`	The name of the `routeMaps` object applied to the neighbor
`tmmRouting.config.routeMaps.seq`	The order of the `routeMaps` entry.
`tmmRouting.config.routeMaps.deny`	Allow or deny the `routeMaps` entry.
`tmmRouting.config.routeMaps.match`	The name of the referenced `prefixList`.
`tmmRouting.config.bgp.neighbors.failoverBFD`	Enable BFD fallover between peers: true / false.
`tmmRouting.config.bfd.interface`	Sets the BFD peering interface.
`tmmRouting.config.bfd.interval`	Configures the BFD transmission interval (50-999).

Parameter	Description
`tmmRouting.config.bfd.minrx`	Configures BFD minimum receive interval (50-999).
`tmmRouting.config.bfd.multiplier`	Configures the BFD multiplier (3-50).

afm

Parameter	Description
`enabled`	Enables the Edge Firewall Pod: **true** or **false** (default).
`defaultFirewallRule.action`	Sets the Edge Firewall default firewall action: **accept** (default), **reject**, or **drop**.
`defaultFirewallRule.log`	Enables logging messages when a packet matches the `defaultFirewallRule.action`: **true** (default) or **false**.
`pccd.enabled`	Enables the Packet Classification Compiler daemon (PCCD): **true** or **false** (default).
`pccd.image.repository`	The domain name or IP address of the local container registry.
`fluentbit_sidecar.enabled`	Enables the fluentbit logging sidecar (true /false). The default is true.
`fluentbit_sidecar.image.repository`	The domain name or IP address of the local container registry.

ipsd

Parameter	Description
`enabled`	Enables the intrusion detection and protection system Pod: **true** or **false** (default).
`image.repository`	The domain name or IP address of the local container registry.

f5-toda-logging

Parameters to send TMM logging data to the Fluentd Logging container.

Note: *f5-toda-logging is a subchart of the Ingress Helm chart.*

Parameter	Description
`enabled`	Enable or disable TMM logging: true (default) or false.
`fluentD.host`	Sets the fluentd service name used as a target to send logging information.
`sidecar.image.repository`	Sidecar regitry name.
`tmstats.config.image.repository`	The path of f5-toda-tmstatsd image.

debug

Parameters for the Debug Sidecar.

Parameter	Description
`image.repository`	Debug registry name.

F5BigPePolicy Reference

The F5BigPePolicy Custom Resource (CR) configuration parameters. Each heading below represents the top-level parameter element. For example, to set the profile `state`, use `spec.spec`.

spec

Parameter	Description
`state`	Specifies the state of the F5BigPePolicy: **Enabled** (default) or **Disabled**.

spec.rule

Parameter	Description
`name`	Specifies the name of the F5BigPePolicy rule. A policy can contain multiple rules.
`precendence`	Specifies the precedence, or order for processing rules and actions: **1** to **1024**.
`reportingProfile`	Specifies the F5BigLogProfile CR to reference by `metadata.name` parameter.
`publisher`	Specifies the F5BigLogHslpub CR to reference by `metadata.name` parameter.

spec.rule.action

Parameter	Description
`gate`	Specifies whether to allow (enabled) or deny (disabled) subscriber traffic that matches a PE rule: **Enabled** (default) or **Disabled**.
`tcpOptimizationUplink`	Specifies the F5BigTcpSetting CR applied to the uplink traffic that matches the rule.
`tcpOptimizationDownlink`	Specifies the F5BigTcpSetting CR applied to the downlink traffic that matches the rule.
`ratePacing.udp.enabled`	Enables UDP ratepacing: **true** or **false** (default).
`ratePacing.udp.sendBuffer`	Specifies the UDP send buffer size used to store the UDP packets when the maximum rate is exceeded: **0** to **4294967295**. The default is **655350**. UDP packets will be later transmitted when the rate of the flow allows. When the send buffer exceeds the specified value, UDP packet will be dropped.
`ratePacing.udp.maxRate`	Specifies the UDP max rate on a per flow basis: **0** (default) to **4294967295**. A value of **0** disables rate limitation. When the maxRate 0 is called and some UDP packets are buffered in the send buffer, the packets will be output at the line rate and feature will be turned off.
`ratePacing.udp.debugUdpRatePacing`	Enables UDP rate pacing iRule logging: **true** or **false** (default).

spec.rule.filter.classification

This parameter set detects application traffic by type.

Parameter	Description
`name`	Specifies the name of the classification.
`match`	Specifies a traffic matching criteria: **match** (default), or **no-match**.
`category`	Specifies the type of traffic: **any** (default), **Web**, **Audio_Video**, **Encrypted**, **File_Download_Servers**, **Search_Engines**, **Network_Management_and_Services**, **News_and_Media**, and **Advertisements**.
`application`	Specifies the application type: **any** (default), **amazon**, **apple**, **tcp**, **udp**, **http**, **ssl**, **youtube**, **google**, **ftp**, **cnn**, and **amazon_adv**.

spec.rule.filter.flow

This parameter set detects application traffic by specific source and destination IP addresses and ports, and also DSCP values.

Parameter	Description
`name`	Specifies the name of the flow filter.
`match`	Specifies a traffic matching criteria: **match** (default), or **no-match**.
`dscp`	Specifies the DSCP value for matching subscriber traffic: **0** (default) to **63**.
`ecnDetection`	Enables ECN Detection, matching traffic with congestion encountered CE: **true** or **false** (default).
`protocol`	Specifies the protocol type for matching subscriber traffic: **any** (default), **tcp**, or **udp**.
`ipType`	Specifies the IP protocol type for matching subscriber traffic: **any** (default), **ipv4**, or **ipv6**.
`sourceVlan`	Specifies the source vlan for matching subscriber traffic. The default is **any**.
`sourceAddress`	Specifies the source IP address for matching subscriber traffic. The default is **0.0.0.0/0**.
`sourcePort`	Specifies the source port for matching subscriber traffic. The default is **0**.
`destinationAddress`	Specifies the destination IP address for matching subscriber traffic. The default is **0.0.0.0/0**.
`destinationPort`	Specifies the destination port for matching subscriber traffic. The default is **0**.

spec.rule.filter.url_catagorization

This parameter set detects application traffic by URL.

Parameter	Description
`name`	Specifies the name of the url_catagorization.
`match`	Specifies a traffic matching criteria: **match** (default), or **no-match**.
`category`	Specifies the URL category of the traffic. For example, **adult** or **business-and-economy**. The default is **any**. For a full list of categories, refer to F5BigPePolicy URL Categories.

F5BigDdosPolicy Reference

The F5BigDdosPolicy Custom Resource (CR) configuration parameters. Each heading below represents the top-level parameter element. For example, to set the `listType`, use `udpPortlist.listType`.

Parameter	Description
`hslPublisher`	Specifies the endpoint logging server to send logging messages. References the F5BigLogHslpub CR by `metadata.name` parameter.

udpPortlist

Parameter	Description
`listType`	Specifies whether to include or exclude the service ports used for UDP flood vector detection: **exclude-listed-ports** (default), or **include-listed-ports**.
`entries.port`	Specifies the service port(s) used for UDP flood vector detection.
`entries.matchDirection`	Specifies if packet matches are based on source port, destination port or either: **src**, **dst** or **either** (default).

allowList

Parameter	Description
`sourceAddressList`	Specifies a F5BigCneAddresslist CR by `metadata.name` containing the source IP addresses to be excluded from DoS detection/mitigation.
`entries.name`	Specifies a name for the allowlist.
`entries.ipProtocol`	Specifies the IP protocol allowed by the allowlist: **any** (default), **icmp**, **igmp**, **tcp**, **udp**.
`entries.entryType`	Specifies what the allowList match is based on: **destination-match**, **source-match**, **v4-all**, **v6-all**, or **all-ip**.
`entries.matchingAddress`	Specifies a destination IP address when `entryType` is **destination-match**, or source IP address when `entryType` is **source-match**.

Parameter	Description
`entries.destinationPort`	Specifies a destination service port the allowList matches. The default values is **0** for all ports.
`entries.sourceVlan`	Specifies the name of the source VLAN the allowList matches. The default value is **any** for all VLANs.

vectors.floodVectors.commonConfigVectors

Parameter	Description
`vectorType`	Specifies the type of DoS Flood Vector to detect and mitigate: **udp-flood**, **ether-brdcst-pkt**, **ether-multicst-pkt**, **arp-flood**, **ip-frag-flood**, **ipv6-frag-flood**, **tcp-rst-flood**, **icmpv4-flood**, **icmpv6-flood**, and **tcp-psh-flood**.
`state`	Specifies the system's response when a vector match occurs: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`detectionThresholdEps`	Specifies the attack detection threshold in Events Per Second (EPS). When EPS exceeds the threshold, the attack is logged and reported. The default value is **4294967295**.
`detectionThresholdPercentage`	Specifies the attack detection threshold by Events Per Second (EPS) percentage increase. The system compares the current EPS rate to the average rate from the last hour, and when the percentage is exceeded, the attack is logged and reported. The default value is **4294967295**.
`rateLimit`	Specifies the rate limit in Events Per Second (EPS). When EPS exceeds the threshold, excess events are dropped until the EPS rate no longer exceeds the threshold. The default value is **4294967295**.
`perSourceIpDetectionEps`	Specifies the attack detection threshold in EPS per source IP address. The default value is **4294967295**.
`perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured attack type per source IP. The default value is **4294967295**.

Parameter	Description
`perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured attack type per destination IP. The default value is **4294967295**.

vectors.ipV6errorVectors.commonConfigVectors

Parameter	Description
`vectorType`	Specifies the type of IPv6 DoS Error Vector to match: **dup-ext-hdr**, **bad-hop-cnt**, **bad-ipv6-ver**, **addr-len-gt-l2-len**, or **payload-len-ls-l2-len**.
`detectionThresholdEps`	Specifies the IPv6 attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`detectionThresholdPercentage`	Specifies the IPv6 attack detection percentage increase for the configured attack type. The default value is **4294967295**.

vectors.ipV6floodVectors.commonConfigVectors

Parameter	Description
`vectorType`	Specifies the type of IPv6 DoS Flood Vector to match: **l4-ext-hdrs-go-end**, and **bad-ext-hdr-order**.
`state`	Specifies the reponse for an IPv6 vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`detectionThresholdEps`	Specifies the IPv6 attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`detectionThresholdPercentage`	Specifies the IPv6 attack detection percentage increase for the configured attack type. The default value is **4294967295**.
`rateLimit`	Specifies the rate limit in EPS for the configured IPv6 attack type. The default value is **4294967295**.

Parameter	Description
`perSourceIpDetectionEps`	Specifies the IPv6 attack detection threshold in EPS for the configured attack type per source IP. The default value is **4294967295**.
`perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured IPv6 attack type source IP. The default value is **4294967295**.
`perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured IPv6 attack type per destination IP. The default value is **4294967295**.
`perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured IPv6 attack type per destination IP. The default value is **4294967295**.

vectors.ipV6floodVectors.specificConfigVectors

Parameter	Description
`lowHopCnt.state`	Specifies the reponse for a vector match: **detection-only** (default) or **mitigation**.
`lowHopCnt.detectionThresholdEps`	Specifies the attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`lowHopCnt.detectionThresholdPercentage`	Specifies the attack detection percentage increase for the configured attack type. The default value is **4294967295**.
`lowHopCnt.rateLimit`	Specifies the rate limit in EPS for the configured attack. The default value is **4294967295**.
`lowHopCnt.perSourceIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per source IP. The default value is **4294967295**.
`lowHopCnt.perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured attack type per source IP. The default value is **4294967295**.
`lowHopCnt.perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`lowHopCnt.perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`lowHopCnt.ipv6LowHopCount`	Specifies the minimum acceptable value for IPv6 Hop Count: **1** (default) through **4**.

Parameter	Description
`extHdrTooLarge.state`	Specifies the reponse for an IPv6 vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`extHdrTooLarge.detectionThresholdEps`	Specifies the attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`extHdrTooLarge.detectionThresholdPercentage`	Specifies the attack detection percentage increase for the configured attack type. The default value is **4294967295**.
`extHdrTooLarge.rateLimit`	Specifies the rate limit in EPS for the configured attack. The default value is **4294967295**.
`extHdrTooLarge.perSourceIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per source IP. The default value is **4294967295**.
`extHdrTooLarge.perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured attack type per source IP. The default value is **4294967295**.
`extHdrTooLarge.perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`extHdrTooLarge.perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`extHdrTooLarge.maxipv6ExtHdrSize`	Specifies the size at which an IPv6 Extension Header is considered oversized: **0** through **1024**. The default value is **128**.
`withExtHdrFrames.state`	Specifies the reponse for an IPv6 vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`withExtHdrFrames.detectionThresholdEps`	Specifies the attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`withExtHdrFrames.detectionThresholdPercentage`	Specifies the attack detection percentage increase for the configured attack type. The default value is **4294967295**.
`withExtHdrFrames.rateLimit`	Specifies the rate limit in EPS for the configured attack. The default value is **4294967295**.
`withExtHdrFrames.perSourceIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per source IP. The default value is **4294967295**.

Parameter	Description
`withExtHdrFrames.perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured attack type per source IP. The default value is **4294967295**.
`withExtHdrFrames.perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`withExtHdrFrames.perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`withExtHdrFrames.ipv6ExtHdrFrameType`	The IPv6 Header Frame type to match: **auth**, **dstopt**, **esp**, **frag**, **hbh**, **mobility**, **route**, and **All** (default).
`tooManyExtHdrs.state`	Specifies the reponse for an IPv6 vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`tooManyExtHdrs.detectionThresholdEps`	Specifies the attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`tooManyExtHdrs.detectionThresholdPercentage`	Specifies the attack detection percentage increase for the configured attack type. The default value is **4294967295**.
`tooManyExtHdrs.rateLimit`	Specifies the rate limit in EPS for the configured attack. The default value is **4294967295**.
`tooManyExtHdrs.perSourceIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per source IP. The default value is **4294967295**.
`tooManyExtHdrs.perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured attack type per source IP. The default value is **4294967295**.
`tooManyExtHdrs.perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`tooManyExtHdrs.perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`tooManyExtHdrs.maxIpv6ExtHdrs`	Specifies the number of IPv6 Extension Headers that are considered too many: **0** - **15**. The default value is **4**.

vectors.l4errorVectors.commonConfigVectors

Parameter	Description
`vectorType`	The type of layer 4 DoS Error Vector: **bad-udp-chksum** or **bad-udp-hrd**.
`detectionThresholdEps`	Attack detection threshold in pps for the Attack type in question. The default value is **4294967295**.
`detectionThresholdPercentage`	Attack detection percentage increase for the Attack type in question. The default value is **4294967295**.

vectors.dnsErrorVectors.commonConfigVectors

Parameter	Description
`vectorType`	The type of DNS DoS Error Vector: **dns-malformed**, **dns-qdcount-limit**, or **unsolicited-dns-response**.
`detectionThresholdEps`	Attack detection threshold in pps for the Attack type in question. The default value is **4294967295**.
`detectionThresholdPercentage`	Attack detection percentage increase for the Attack type in question. The default value is **4294967295**.

vectors.dnsFloodVectors.commonConfigVectors

Parameter	Description
`vectorType`	The type of DNS Flood Vector: **dns-a-query**, **dns-aaaa-query**, **dns-any-query**, **dns-ptr-query**, **dns-axfr-query**, **dns-cname-query**, **dns-ixfr-query**, **dns-mx-query**, **dns-ns-query**, **dns-other-query**, **dns-soa-query**, **dns-srv-query**, or **dns-txt-query**.
`state`	Specifies the reponse for a vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`detectionThresholdEps`	Specifies the attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`detectionThresholdPercentage`	Specifies the attack detection percentage increase for the configured attack type. The default value is **4294967295**.

Parameter	Description
`rateLimit`	Specifies the rate limit in EPS for the configured attack. The default value is **4294967295**.
`perSourceIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per source IP. The default value is **4294967295**.
`perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured attack type per source IP. The default value is **4294967295**.
`perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured attack type per destination IP. The default value is **4294967295**.

vectors.dnsFloodVectors.specificConfigVectors

Parameter	Description
`oversizedDns.state`	Specifies the reponse for a vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`oversizedDns.detectionThresholdEps`	Specifies the attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`oversizedDns.detectionThresholdPercentage`	Specifies the attack detection percentage increase for the configured attack type. The default value is **4294967295**.
`oversizedDns.rateLimit`	Specifies the rate limit in EPS for the configured attack. The default value is **4294967295**.
`oversizedDns.perSourceIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per source IP. The default value is **4294967295**.
`oversizedDns.perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured attack type per source IP. The default value is **4294967295**.
`oversizedDns.perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per destination IP. The default value is **4294967295**.

Parameter	Description
`oversizedDns.perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`oversizedDns.maxDnsSize`	Specifies the size at which a DNS packet is considered oversized: **256** through **8192**. The default value is **4096**.
`dnsNxdomainQuery.state`	Specifies the reponse for a vector match: **detection-only** (default) or **mitigation**. To disable, delete the custom resource.
`dnsNxdomainQuery.detectionThresholdEps`	Specifies the attack detection threshold in EPS for the configured attack type. The default value is **4294967295**.
`dnsNxdomainQuery.detectionThresholdPercentage`	Specifies the attack detection percentage increase for the configured attack type. The default value is **4294967295**.
`dnsNxdomainQuery.rateLimit`	Specifies the rate limit in EPS for the configured attack. The default value is **4294967295**.
`dnsNxdomainQuery.perSourceIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per source IP. The default value is **4294967295**.
`dnsNxdomainQuery.perSourceIpLimitEps`	Specifies the rate limit in EPS for the configured attack type per source IP. The default value is **4294967295**.
`dnsNxdomainQuery.perDstIpDetectionEps`	Specifies the attack detection threshold in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`dnsNxdomainQuery.perDstIpLimitEps`	Specifies the rate Limit in EPS for the configured attack type per destination IP. The default value is **4294967295**.
`dnsNxdomainQuery.dnsNXDomainLearnPeriod`	NEED DESCRIPTION **1** - **2147483647**. The default value is **7200**.
`dnsNxdomainQuery.dnsNXDomainPeriod`	NEED DESCRIPTION **1** - **2147483647**. The default value is **86400**.
`dnsNxdomainQuery.dnsNXDomainTrackerSize`	NEED DESCRIPTION **64** - **8000**. The default value is **320**.
`dnsNxdomainQuery.validDomains`	NEED DESCRIPTION

F5BigContextSecure Reference

The F5BigContextSecure Custom Resource (CR) configuration parameters. Each heading below represents the top-level parameter element. For example, to set the protocol `profile`, use `spec.profile`.

spec

Parameter	Description
`destinationAddress`	The advertised IPv4 address of the application.
`ipv6destinationAddress`	The advertised IPv6 address of the application.
`destinationPort`	Defines the service port for ingress connections.
`ipProtocol`	Specifies the virtual server IP protocol: **tcp**, **udp**, or **any** (default).
`profile`	Specifies the profile to be used by the virtual server: **fastl4** (default), **tcp**, **udp**, or **ipother**.
`fastL4Settings.profileName`	Specifies how TMM handles connections using the F5BigFastl4Setting CR's `metadata.name` value.
`tcpSettings.clientSide`	Specifies how TMM handles clientside TCP connections using the F5BigTcpSetting CR's `metadata.name` value.
`tcpSettings.serverSide`	Specifies how TMM handles serverside TCP connections using the F5BigTcpSetting CR's `metadata.name` value.
`udpSettings.clientSide`	Specifies how TMM handles clientside UDP connections using the F5BigUdpSetting CR's `metadata.name` value.
`udpSettings.serverSide`	Specifies how TMM handles serverside UDP connections using the F5BigUdpSetting CR's `metadata.name` value.
`natPolicy`	Specifies the F5BIGNatPolicy CR to reference using the `metadata.name` parameter.
`firewallEnforcedPolicy`	Specifies the F5BigFwPolicy CR to reference using the `metadata.name` parameter.
`pemProfile`	Specifies the F5BigPeProfile CR using the `metadata.name` parameter. Refer to F5BigPePolicy.
`protocolInspectionProfile`	Specifies the F5BigIpsPolicy CR using the `metadata.name` parameter.
`logProfile`	Specifies the F5BigLogProfile CR using the `metadata.name` parameter.
`iRules`	The F5BigZeroratingirule CR to reference.
`classificationProfile`	Specifies whether traffic sent to this virtual server is classified. The system automatically enables classification for virtual servers as needed when you create a policy enforcement listener.
`snat.type`	Specifies the type of address translation: **none** (default), **automap**, or **snat**.
`snat.pool`	When `snat.type` is **snat**, specifies the F5BigCneSnatpool CR to reference using the `spec.name` parameter. **"test_pool"**

Parameter	Description
`protocolInspectionProfile`	Specifies the F5BigIpsPolicy CR using the `metadata.name` parameter.
`vlans`	Specifies one or more F5BigNetVlan CRs using the `metadata.name` parameter, that listen for application traffic.
`vlans.disableListedVlans`	When enabled, accept traffic on all VLANs except those defined in the `vlans.vlanList` paramter: **true** (default) or **false**.
`vlans.vlanList`	Specifies a list of F5BigNetVlan CRs to listen for ingress traffic, using the CR's `metadata.name` parameter.
`loadBalancingMethod`	Specifies the load balancing method used to distribute traffic across pool members: **round-robin** distributes connections evenly across all pool members (default), and **ratio-least-connections** distributes connections first to members with the least number of active connections.
`fastL4.idleTimeout`	Specifies the number of seconds that a fastL4 connection can remain idle before deletion: **0** to **4294967295**. The default values is **300**.
`ipother.client.idleTimeout`	Specifies the number of seconds that an ipother client-side connection can remain idle before deletion: **0** to **4294967295**. The default is **60**.
`ipother.server.idleTimeout`	Specifies the number of seconds that an ipother server-side connection can remain idle before deletion: **0** to **4294967295**. The default is **60**.

spec.pool

Parameter	Description
`minActiveMembers`	Specifies the minimum number of members that must be available in one priority group: **0** to **65536**. The default is **0**.
`members`	Specifies a list of IP addresses and ports for the Service.
`members.address`	Specifies the IP address of the Service.
`members.port`	Specifies the port of the Service.
`members.priorityGroup`	Specifies the priority group for pool member: **0** to **8**. The default is **0**.

monitors

Parameter	Description
`icmp.interval`	Specifies, in seconds, the monitor check frequency. The default value is **5**.
`icmp.timeout`	Specifies, in seconds, the time in which the target must respond. The default value is **16**.

Parameter	Description
`icmp.serversslProfileName`	Specifies the server side SSL profile used to ping the target. The default is **_mon_ssl**.
`tcp.interval`	Specifies, in seconds, the monitor check frequency. The default values if **5**.
`tcp.timeout`	Specifies, in seconds, the time in which the target must respond. The default value is **16**.
`tcp.serversslProfileName`	Specifies the server side SSL profile used to ping the target. The default is **_mon_ssl**.

F5BigDnsCache Reference

The F5BigDnsCache Custom Resource (CR) configuration parameters. The F5BigDnsCache is referenced by the F5BigDnsApp CR and must be created first. Each heading below represents the top-level parameter element. For example, to set the DNS Cache `name`, use `spec.name`.

spec

Parameter	Description
`cacheType`	The type for this DNS cache: **transparent** or **net-resolver** or **resolver**.

spec.transparent

Parameter	Description
`answerDefaultZones`	Whether to answer queries for default zones such as **localhost**, **reverse 127.0.0.1 and ::1** or **AS112**: **true** or **false** (default).
`localZones.name`	The Fully Qualified Domain Name for a localZone.
`localZones.zoneType`	The zoneType for this localZone: **deny**, **refuse**, **static**, **transparent** (default), **type-transparent** or **redirect**.
`localZones.records`	Array of records for this localZone.
`msgCacheSize`	Number of bytes allocated for the message cache. The default is **1048576**.
`rrsetCacheSize`	Number of bytes allocated for the resource record set cache. The default is **10485760**.
`rrsetRotate`	Specifies which resource record set rotation method should be used on cache responses: **none** (default) or **query-id**.

spec.netResolver

Parameter	Description
`answerDefaultZones`	Whether to answer queries for default zones such as **localhost**, **reverse 127.0.0.1 and ::1** or **AS112**: **true** or **false** (default).
`forwardZones.forwardZone`	The Fully Qualified Domain Name for a Forward Zone. Use **.** for all domains.
`forwardZones.nameServers`	An array of Name Servers for a Forward Zone.
`forwardZones.nameServers.ipAddress`	The IPv4 or IPv6 address for a Name Server.
`forwardZones.nameServers.port`	Port for a Name Server. The default is **53**.
`msgCacheSize`	Number of bytes allocated for the message cache. The default is **1048576**.

Parameter	Description
`nameserverCacheCount`	description: Maximum number of DNS nameservers to cache. The default is **166536**.
`nameserverTTL`	Time to live, in seconds, for DNS nameservers in the cache. The default is **900**.
`negCacheSize`	Number of bytes allocated for the negative cache. The default is **1048576**.
`randomQueryNameCase`	Enables resolver to randomize the case of query names: **true** (default) or **false**.
`returnTruncatedWhenNoTcp`	Enables resolver to return a truncated response from an upstream DNS nameserver when useTcp is set to **false**: The default value is **false**.
`rrsetCacheSize`	Number of bytes allocated for the resource record set cache. The default is **10485760**.
`useIpv4`	Enables the resolver to issue IPv4 queries: **true** (default) or **false**.
`useIpv6`	Enables the resolver to issue IPv6 queries: **true** (default) or **false**.
`useTcp`	Enables the resolver to issue tcp queries: **true** (default) or **false**.
`useUdp`	Enables the resolver to issue udp queries: **true** (default) or **false**.

spec.resolver

Parameter	Description
`answerDefaultZones`	Whether to answer queries for default zones such as **localhost**, **reverse 127.0.0.1 and ::1** or **AS112**: **true** or **false** (default).
`forwardZones.forwardZone`	The Fully Qualified Domain Name for a Forward Zone. Use **.** for all domains.
`forwardZones.nameServers`	An array of Name Servers for a Forward Zone.
`forwardZones.nameServers.ipAddress`	The IPv4 or IPv6 address for a Name Server.
`forwardZones.nameServers.port`	Port for a Name Server. The default is **53**.
`localZones.name`	The Fully Qualified Domain Name for a localZone.
`localZones.zoneType`	The zoneType for this localZone: **deny**, **refuse**, **static**, **transparent** (default), **type-transparent** or **redirect**.
`localZones.records`	Array of records for this localZone.
`maxConcurrentQueries`	Specifies the maximum number of concurrent distinct queries used by the resolver. A query is identified by query name, type and class. If the number of distinct queries exceeds this limit, the resolver replaces the earliest query in the queue with the new query if it has been in the queue longer than the allowed time. The default is **1024**.
`maxConcurrentTcp`	Specifies the maximum number of concurrent TCP flows used by the resolver. The default is **20**.
`maxConcurrentUdp`	Specifies the maximum number of concurrent UDP flows used by the resolver. The default is **8192**.

Parameter	Description
`msgCacheSize`	Number of bytes allocated for the message cache. The default is **1048576**.
`nameserverCacheCount`	description: Maximum number of DNS nameservers to cache. The default is **166536**.
`nameserverTTL`	Time to live, in seconds, for DNS nameservers in the cache. The default is **900**.
`randomQueryNameCase`	Enables resolver to randomize the case of query names: **true** (default) or **false**.
`rootHints`	Specifies the IP addresses of DNS servers that the BIG-IP system considers authoritative for the DNS root nameservers. Important:By default, the BIG-IP system uses the DNS root nameservers published by InterNIC. Caution:When you add DNS root nameservers, the BIG-IP system no longer uses the default nameservers published by InterNIC, but instead uses the nameservers you add as authoritative for the DNS root nameservers.
`rrsetCacheSize`	Specifies the number of bytes allocated for the resource record set cache. The default is **10485760**.
`rrsetRotate`	Select which resource record set rotation method should be used on cache responses: **none** (default) or **query-id**.
`unwantedQueryReplyThreshold`	The system always rejects unsolicited replies. The default value of “0” (off) indicates the system does not generate SNMP traps or log messages when rejecting unsolicited replies. The default is **0**.
`useIpv4`	Enables the resolver to issue IPv4 queries: **true** (default) or **false**.
`useIpv6`	Enables the resolver to issue IPv6 queries: **true** (default) or **false**.
`useTcp`	Enables the resolver to issue tcp queries: **true** (default) or **false**.
`useUdp`	Enables the resolver to issue udp queries: **true** (default) or **false**.

cacheType setting

By caching DNS responses and answering queries from the cache, the BIG-IP system is able to immediately respond to subsequent client requests for the same resource. This enhances DNS performance in two significant ways. First, answering a DNS query from the cache is faster and has a very short latency, because the sooner a client gets a DNS response, the faster the client can access the Internet resource. Secondly, caching DNS responses reduces the number of queries that have to be resolved. The BIG-IP system uses the cache to resolve the same query from multiple clients handling many more queries per second than a typical DNS resolver.

Currently, only the Transparent, Net Resolver and Resolver cache types are supported.

Cache Type	Description
`Transparent Cache`	A transparent cache does not perform recursive resolution, but instead relies on another DNS resource for resolution.
`Net Resolver Cache`	A net-resolver cache is utilized by RESOLVER::name_lookup iRule and other non-DNS features of the Big-IP system.
`Resolver Cache`	A resolver cache performs recursive resolution to fill its cache.

Forward Zones

Applicable only to the DNS Resolver and Net Resolver. A forward zone contains a zone name and a list of IP addresses to upstream DNS nameservers capable of answering queries for the specified zone.

Forward Zone Statistics

Statistic	Description
`fwd.queries`	Number of queries sent to the DNS cache that matched a configured forward zone.
`fwd.responses`	Number of responses returned from a configured forward zone's nameserver, whether they were good responses or SERVFAIL.

Local Zones

Applicable only to the Transparent and Resolver caches, a local zone contains resource records that a DNS cache uses to resolve matching DNS queries with authoritative DNS responses. The zoneType attribute of the local zone determines how the cache handles a DNS query that does not match the local zone.

Parameter	Description
`deny`	For a non-matching query, the cache drops the DNS query. This is an example of a response to a non-matching query: DNS request timed out.
`redirect`	For a non-matching query, when the query is for a subdomain of the local zone, the cache returns the same response that it would for the local zone. For example, if you add the local zone siterequest.com, the cache returns the same response to queries for wiki.siterequest.com. and download.wiki.siterequest.com. This is an example of a response to a non-matching query: NOERROR rcode returned and example.com. NOT resolved as expected.
`refuse`	For a non-matching query, the cache returns a REFUSED message in the DNS response. This is an example of a response to a non-matching query: REFUSED rcode returned and example.com. NOT resolved as expected.
`static`	For a non-matching query, the cache returns a NODATA or NXDOMAIN in the DNS response, which also includes the SOA record if the local zone contains one. This is an example of a response to a non-matching query: NOERROR rcode returned and example.com. NOT resolved as expected.
`transparent`	Transparent is the default value. For a non-matching query, the cache performs a pass-through or iterative resolution of the DNS query. If the query matches, but no resource records are available, the cache returns a response with a NoData message. This is an example of a response to a non-matching query: NOERROR rcode returned and example.com. NOT resolved as expected.

Parameter	Description
`type-transparent`	For a non-matching query, or a query for a matching domain name, but with a request for a record of a different type, the cache performs a pass-through or iterative resolution of the DNS query; however, if the query matches, but no resource records are available, the cache does not return a response with a NoData message. This is an example of a response to a non-matching query: DNS request resolved to example.com. as expected.

Local Zone Statistics

Statistic	Description
`local.queries`	Number of queries that match a subdomain of a local zone. These might not be an exact local zone rrset match.
`local.responses`	Number of responses generated from an exact local zone's rrset match.

Sub-Caches

The DNS Cache is composed of a series of "sub-caches", each of which serves a specific purpose for caching the various records that make up a DNS response.

Message

The message sub-cache is used to cache the complete DNS response sent to the originating client. The parts of this response include the question section, answer section, authority section and additional records, as well as all relevant header information sent back to the client.

An entry in the message sub-cache contains a query name, a query type, a query class, and query flags. This entry will then store information related to resource records in the rrset sub-cache used to generate the response sent back to the originating client.

Message Sub-Cache Statistics

Statistic	Description
`msg.hits`	Number of times the message sub-cache was checked for an entry, and a valid (and unexpired) entry was found.
`msg.misses`	Number of times the message sub-cache was checked for an entry, and a valid (and unexpired) entry was not found.
`msg.inserts`	Number of times that a new entry that did not currently exist (or existed but was expired) in the message sub-cache was inserted into the message sub-cache.
`msg.updates`	Number of times that a valid (and unexpired) entry was updated in the message sub-cache.

Statistic	Description
msg.evictions	Number of times that a valid (and unexpired) entry was removed from the message sub-cache to make space for a newer response.
msg.evictions_mempressure	Number of times that a valid (and unexpired) entry was removed from the message sub-cache due to memory pressure constraints.
msg.ts_usecs_bytes_allocated	Internally used to determine the last time the message sub-cache size allocation was changed.
msg.bytes_allocated	Number of bytes currently allocated to the message sub-cache across the entire TMM process
msg.evictions_rate	Average number of entries evicted from the message sub-cache per second.
msg.modifications_rate	Average number of entries modified in the message sub-cache per second.

Resource Record Set

The resource record set (rrset) sub-cache contains records related to all rrsets that have been received by upstream DNS nameservers.

An entry in the resource record set sub-cache contains a domain name, an rrset type, and an rrset class. This entry will then store all relevant resource record set data related to the rrset type.

Resource Record Set Sub-Cache Statistics

Statistic	Description
rrset.hits	Number of times the resource record set sub-cache was checked for an entry, and a valid (and unexpired) entry was found.
rrset.misses	Number of times the resource record set sub-cache was checked for an entry, and a valid (and unexpired) entry was not found.
rrset.inserts	Number of times that a new entry that did not currently exist (or existed but was expired) in the resource record set sub-cache was inserted into the resource record set sub-cache.
rrset.updates	Number of times that a valid (and unexpired) entry was updated in the resource record set sub-cache.
rrset.evictions	Number of times that a valid (and unexpired) entry was removed from the resource record set sub-cache to make space for a newer response.
rrset.evictions_mempressure	Number of times that a valid (and unexpired) entry was removed from the resource record set sub-cache due to memory pressure constraints.
rrset.ts_usecs_bytes_allocated	Internally used to determine the last time the resource record set sub-cache size allocation was changed.
rrset.bytes_allocated	Number of bytes currently allocated to the resource record set sub-cache across the entire TMM process
rrset.evictions_rate	Average number of entries evicted from the resource record set sub-cache per second.

Statistic	Description
`rrset.modifications_rate`	Average number of entries modified in the resource record set sub-cache per second.

Nameserver

In addition to the information available in the NS Resource Record set, the nameserver cache contains useful for keeping track of pending queries to upstream DNS nameservers. It specifically keeps track of things like TTL, RTT, whether nameserver uses EDNS, and the Lameness of the particular nameserver.

An entry in the nameserve sub-cache contains a domain name and IP address, and it will store information about that zones TTL, RTT, EDNS Status, Lameness, and Timeout information.

Nameserver Sub-Cache Statistics

Statistic	Description
`nameserver.hits`	Number of times the nameserver sub-cache was checked for an entry, and a valid (and unexpired) entry was found.
`nameserver.misses`	Number of times the nameserver sub-cache was checked for an entry, and a valid (and unexpired) entry was not found.
`nameserver.inserts`	Number of times that a new entry that did not currently exist (or existed but was expired) in the nameserver sub-cache was inserted into the nameserver sub-cache.
`nameserver.updates`	Number of times that a valid (and unexpired) entry was updated in the nameserver sub-cache.
`nameserver.evictions`	Number of times that a valid (and unexpired) entry was removed from the nameserver sub-cache to make space for a newer response.
`nameserver.evictions_mempressure`	Number of times that a valid (and unexpired) entry was removed from the nameserver sub-cache due to memory pressure constraints.
`nameserver.ts_usecs_bytes_allocated`	Internally used to determine the last time the nameserver sub-cache size allocation was changed.
`nameserver.bytes_allocated`	Number of bytes currently allocated to the nameserver sub-cache across the entire TMM process
`nameserver.evictions_rate`	Average number of entries evicted from the nameserver sub-cache per second.
`nameserver.modifications_rate`	Average number of entries modified in the nameserver sub-cache per second.

Key

The key cache is used by the Validating Resolver (Not Yet Implemented) to store DNSKEYs.

An antry for the key sub-cache contains a domain name and a class, and it will store information about the key's TTL, algorithm, algorithm, and rrset type secured.

Note: Validating Resolver is not currently implemented.

Key Sub-Cache Statistics

Statistic	Description
`key.hits`	Number of times the key sub-cache was checked for an entry, and a valid (and unexpired) entry was found.
`key.misses`	Number of times the key sub-cache was checked for an entry, and a valid (and unexpired) entry was not found.
`key.inserts`	Number of times that a new entry that did not currently exist (or existed but was expired) in the key sub-cache was inserted into the key sub-cache.
`key.updates`	Number of times that a valid (and unexpired) entry was updated in the key sub-cache.
`key.evictions`	Number of times that a valid (and unexpired) entry was removed from the key sub-cache to make space for a newer response.
`key.evictions_mempressure`	Number of times that a valid (and unexpired) entry was removed from the key sub-cache due to memory pressure constraints.
`key.ts_usecs_bytes_allocated`	Internally used to determine the last time the key sub-cache size allocation was changed.
`key.bytes_allocated`	Number of bytes currently allocated to the key sub-cache across the entire TMM process
`key.evictions_rate`	Average number of entries evicted from the key sub-cache per second.
`key.modifications_rate`	Average number of entries modified in the key sub-cache per second.

A Mesh Of Query States

The DNS Cache uses a mesh of query states in order to track and store all resource record entries it has received from upstream DNS nameservers. This allows the cache to know what resource record sets it already has available and to make it easy to determine which record sets it is missing in order respond to an incoming client's DNS query.

These query states contain information about pending queries (per query-name, query-type, and query class) and maintain relationships with other pending queries which are waiting for information from this query. For example, there may be a query state waiting on a response for `www.example.com`, which is itself waiting for a response from an upstream DNS nameserver for `example.com`. In this case, `www.example.com` is a super-state waiting for the sub-query state of `example.com`. It is is possible that a query for `mail.example.com` gets a response saved to the DNS cache before the query to `example.com` gets a response. At this point, the query state for `mail.example.com` would know that it also needs the response for `example.com` and would wait for that information to be returned before it finishes generating it's response for `mail.example.com`. Once the response for `example.com` is received, then queries for both `www.example.com` and `mail.example.com` would be sent to the `example.com` nameserver.

Another benefit of these query states is they allow specific portions of a resolution to be cached, and that portion can then be retrieved locally without sending a query off-box to an upstream DNS server.

Continuing the above example, if after `www.example.com` and `mail.example.com` are fully resolved, and a query for `about.example.com` comes into the cache, the cache will already have information cached for `example.com` and can simply send it's query for `about.example.com` to `example.com`'s nameserver. It will search the relevant sub-caches for the information it is looking for, and only send queries to upstream DNS nameservers when it cannot find relevant information in these sub-caches.

This only happens when the cache is unable to provide an answer to the client's query from it's configured local zones or forward zones.

Note on Maximum Concurrent Queries Connections

The DNS Resolver configuration object "max-concurrent-queries" represents the maximum number of simultaneously pending DNS queries sent to upstream DNS nameservers. The number of "max-concurrent-queries" is evenly divided between two lists that hold all pending queries, a "forever" list and a "jostle" list. The purpose of these two lists is to allow some queries to run to completion while having another list of queries that can be cancelled and replaced with new queries. Specifically, an entry in the "forever" list will run to completion, while an entry in the "jostle" list will potentially be cancelled and replaced. When one of these lists is full (half of the configured max-concurrent-queries setting) then entries will be moved to maximize the number of entries in the "forever" list.

Mesh Statistics

Statistic	Description
`mesh.total_callbacks`	Number of upstream responses or timeouts handled.
`mesh.num_reply_states`	Number of current pending queries to upstream DNS nameservers in both the "jostle" list and "forever" list.
`mesh.forever_states`	Number of current pending queries to upstream DNS nameservers in the "forever" list.
`mesh.detached_states`	Number of pending states that have not received any replies from upstream DNS nameservers, and have no other mesh-states waiting for it's response.
`mesh.replies_sent`	Unused, replaced by `queries` statistic.
`mesh.stats_jostled`	Number of upstream queries dropped due to running out of space after query was sent and before a response was received.
`mesh.stats_dropped`	Number of upstream queries dropped before being sent.
`mesh.udp_queries`	Number of UDP queries sent to upstream DNS nameservers.
`mesh.tcp_queries`	Number of TCP queries sent to upstream DNS nameservers.
`mesh.responses`	Number of responses received for queries sent to upstream DNS nameservers.
`mesh.queries_rate`	Average number of queries per second send from the cache to upstream DNS nameservers.
`mesh.wait_response`	Number of pending queries sent to upstream DNS nameservers that have not yet received a response.
`mesh.udp_wait_send`	Number of UDP queries currently waiting to be sent to upstream DNS nameservers.
`mesh.tcp_wait_send`	Number of TCP queries currently waiting to be sent to upstream DNS nameservers.
`mesh.unwanted_replies`	Number of unwanted or unsolicited replies received from upstream DNS nameservers, or expected replies received over the wrong port.
`mesh.hits`	Incoming query was able to be resolved based upon already cached data, no further resolution required.

Statistic	Description
mesh.misses	Incoming query was not able to be resolved solely with previously cached data, further resolution required.
mesh.client_wait_response	Number of client queries waiting for a response from the cache.
mesh.dns64reqs	Number of DNS64 requests sent to upstream DNS nameservers.
mesh.dns64xlated	Number of responses where DNS64 AAAA prefix was used to synthesize an AAAA record from an A record.
mesh.dns64nodata	Number of NODATA responses to DNS64 queries send to upstream DNS nameservers.

Queries to Upstream DNS Nameservers

When the DNS cache is sending queries to upstream DNS nameservers, the queries are not sent by the Virtual Server configured to use the DNS cache. Instead, it communicates directly to the upstream DNS nameservers in order to send queries and receive responses. For TCP connections, the cache takes full ownership of these connections for their entire lifetime, establishing, tearing down, and timing out these connections. For UDP, the cache simply generates the packet and sends them out while awaiting for responses to sent queries.

Upstream DNS Nameserver Error Statistics

Statistic	Description
ns_resp_err.memory	Number of times a connection to an upstream DNS nameserver was unsuccessful due to an out of memory condition.
ns_resp_err.port_v4	Number of times a connection to an upstream DNS nameserver was unsuccessful due to not having any v4 ports available to use.
ns_resp_err.port_v6	Number of times a connection to an upstream DNS nameserver was unsuccessful due to not having any v6 ports available to use.
ns_resp_err.route_v4	Number of times a connection to an upstream DNS nameserver was unsuccessful due to not having a route to the specified v4 host.
ns_resp_err.route_v6	Number of times a connection to an upstream DNS nameserver was unsuccessful due to not having a route to the specified v4 host.
ns_resp_err.timeout	Number of times a connection to an upstream DNS nameserver was unsuccessful due to a connection timeout.
ns_resp_err.servfail	Number of responses from an upstream DNS nameserver returned with an rcode of SERVFAIL.
ns_resp_err.formerr	Number of responses from an upstream DNS nameserver returned with an rcode of FORMERR.
ns_resp_err.other	Number of times a connection to an upstream DNS nameserver was unsuccessful or was returned a response with an rcode that isn't covered by the previously defined conditions.

Upstream DNS Namserver UDP Connection Statistics

Statistic	Description
udp.pkts_in	Total number of UDP packets received from upstream DNS nameservers.
udp.pkts_out	Total number of UDP packets sent to upstream DNS nameservers.
udp.bytes_in	Total number of bytes received over UDP from upstream DNS nameservers.
udp.bytes_out	Total number of bytes sent over UDP to upstream DNS nameservers.
udp.max_conns	Maximum number of simultaneous pending UDP connections to upstream DNS nameservers.
udp.tot_conns	Total number of UDP connections established to upstream DNS nameservers.
udp.cur_conns	Current number of outstanding UDP connections established to upstream DNS nameservers.

Upstream DNS Nameserver TCP Connection Statistics

Statistic	Description
tcp.pkts_in	Total number of TCP packets received from upstream DNS nameservers.
tcp.pkts_out	Total number of TCP packets sent to upstream DNS nameservers.
tcp.bytes_in	Total number of bytes received over TCP from upstream DNS nameservers.
tcp.bytes_out	Total number of bytes sent over TCP to upstream DNS nameservers.
tcp.max_conns	Maximum number of simultaneous pending TCP connections to upstream DNS nameservers.
tcp.tot_conns	Total number of TCP connections established to upstream DNS nameservers.
tcp.cur_conns	Current number of outstanding TCP connections established to upstream DNS nameservers.

Other Statistics

The following statistics are also available:

Statistic	Description
name	Name of the DNS Cache.
cache_index	Internally used cache index number.
queries	Number of queries that have been sent to the cache.
responses	Number of responses that the cache was able to generate for queries it received.
responses_rate	Number of responses the cache generated within the previous 1 second of time.

Statistic	Description
response_time_sync	Running average time for synchronous cache responses.
response_time_async	Running average time for asynchronous cache responses.
cb_ctx_cnt	Number of currently pending queries sent to the DNS cache.
sync	Number of responses the cache was able to generate without needing to establish any connections to an upstream DNS nameserver.
async	Number of responses the cache generated using data that had to be obtained from upstream DNS nameservers.
failure_resolv	Number of malformed or corrupted responses received from from upstream DNS nameservers.
failure_server	Internal cache was unable to accept the query due to resource limitations such as an out of memory condition, or too many pending queries to upstream DNS nameservers.
failure_cf	Response generated by cache that cannot be sent to waiting client due to client connection being closed.
failure_send	Response generated by cache that cannot be sent to waiting client due to a resource constraint (such as out-of-memory), or client is in the process of closing their connection.
sec_unchecked	Number of rrsets or messages that have yet to be validated.
sec_bogus	Number of rrsets or messages that failed to validate (according to local policy) but should have been validated.
sec_indeterminate	Number of rrsets or messages that were found to be insecure, but not authoritatively so. Generally, this means that the rrset is not below a configured trust anchor.
sec_insecure	Number of rrsets or messages that were found to be authoritatively insecure. Generally, this means that the rrset is below a trust anchor, but also below a verified and insecure delegation.
sec_secure	Number of rrsets or messages that have been validated as secure against a local policy.
rpz_rewrites	Number of queries that were generated from a configured response-policy-zone.

F5BigDnsApp Reference

The F5BigDnsApp Custom Resource (CR) configuration parameters. Each heading below represents the top-level parameter element. For example, to set the virtual server destination address, use `spec.destination.address`.

Parameters

spec

Parameter	Description
`dns`	Specifies configuration of a Domain Name System (DNS) profile used by the virtual server. See `spec.dns` below for more parameter options.
`destination`	Specifies the destination IP address for clients to use as a DNS resolver. See `spec.destination` below for more parameter options.
`pool`	Specifies the load balancing pool configuration of the remote DNS servers used to resolve DNS queries.
`monitors`	Specifies the monitor configuration for the `pool` members. When a member is detected **down**, DNS queries will not be sent until the status changes to **up**. See `spec.monitors` below for more parameter options.
`snat`	Specifies Source Network Address Translation (SNAT) configuration used by the virtual server. See `spec.snat` below for more parameter options.
`tcpSettings.clientSide`	Specifies a client side F5BigTcpSetting CR referenced by the virtual server, using the `metadata.name` parameter.
`tcpSettings.serverSide`	Specifies a server side F5BigTcpSetting CR referenced by the virtual server, using the `metadata.name` parameter.
`udpSettings.clientSide`	Specifies a client side F5BigUdpSetting CR referenced by the virtual server, using the `metadata.name` parameter.
`udpSettings.serverSide`	Specifies a server side F5BigUdpSetting CR referenced by the virtual server, using the `metadata.name` parameter.
`vlans`	Specifies a F5BigNetVlan CR to reference that accepts network traffic, using the `metadata.name` parameter.
`loadBalancingMethod`	Specifies the load balancing algorithm used to load balance name resolution requests among the members: **round-robin** (default) distributes connections evenly across all pool members. **ratio-least-connections** distributes connections first to members with the least number of active connections. **weighted-round-robin** distributes connections across all pool members based on specified weights and **ratio-session** distributes connections according to the ratio of the number of sessions each pool member has active.
`ipProtocol`	Specifies the IP protocol for the virtual server to direct traffic: **udp** (default) or **tcp**.
`logProfile`	Specifies DNS F5BigLogProfile to be used.

Parameter	Description
`protocolInspectionProfile`	Specifies a F5BigIpsPolicy CR to reference using the `spec.name` parameter.

spec.destination

Parameter	Description
`address`	Specifies the virtual server's address. Any of this field and `ipv6Address` is required to be specified for virtual server destination.
`ipv6Address`	Specifies the virtual server's IPV6 address. Any of this field and `address` is required to be specified for virtual server destination.
`port`	Specifies the virtual server's port. The default is **53**.

spec.dns

Parameter	Description
`useLocalBind`	Enables local bind DNS server: **true** or **false** (default).
`dns64Mode`	Specifies the DNS64 mode: **disable** (default), **secondary**, **immediate**, and **v4-only**. See `spec.dns.dns64mode` below for more parameter options. Refer to the CNFs NAT64 guide for implementation assistance.
`dns64Prefix`	The IPv6 prefix used for DNS64 mapping; mapping A to AAAA type records. The default is **::**.
`dns64AdditionalSectionRewrite`	Sets DNS64 additional section rewriting. For AAAA and A records in additional section. This field specifies how they are being rewritten. The options are **disable** (default), **v6-only**, **v4-only**, **any**. See `spec.dns.dns64AdditionalSectionRewrite` below for more parameter options.
`dnsCache`	Indicates whether to allow queries to be answered non-authoritatively by a DNS cache. It enables caching when referencing a F5BIGDnscache CR (Custom Resource) by `metadata.name`. The default is empty which means caching is disabled.

spec.dns.dns64Mode

Value	Description
`disabled`	The BIG-IP system does not map IPv4 addresses to IPv6 addresses.

Value	Description
`secondary`	The BIG-IP system receives an AAAA query and forwards the query to a DNS server. The BIG-IP system then forwards the first good response from the DNS server to the client. If the system receives an A response first, it appends a 96-bit prefix to the record and forwards it to the client. If the system receives an AAAA response first, it simply forwards the response to the client. The system disregards the second response from the DNS server.
`immediate`	The BIG-IP system receives an AAAA query and forwards the query to a DNS server. Only if the server fails to return a response does the BIG-IP system send an A query. If the BIG-IP system receives an A response, it appends a 96-bit user-configured prefix to the record and forwards it to the client.
`v4-only`	The BIG-IP system receives an AAAA query, but forwards an A query to a DNS server. After receiving an A response from the server, the BIG-IP system appends a 96-bit user-configured prefix to the record and forwards it to the client. `Important:` Select this option only if you know that all your DNS servers are IPv4 only servers.

spec.dns.dns64AdditionalSectionRewrite

Value	Description
`disable`	The BIG-IP system does not perform additional rewrite.
`v6-only`	The BIG-IP system accepts only A records. The system appends the 96-bit user-configured prefix to a record and returns an IPv6 response to the client.
`v4-only`	The BIG-IP system accepts only AAAA records and returns an IPv6 response to the client.
`any`	The BIG-IP system accepts and returns both A and AAAA records. If the DNS server returns an A record in the Additional section of a DNS message, the BIG-IP system appends the 96-bit user-configured prefix to the record and returns an IPv6 response to the client.

spec.monitors

Note: *For all F5BigDnsApp monitors, F5 recommends setting the* `timeout` *value to be* **the same or less than**_ the `interval` value._

Parameter	Description
`dns`	DNS monitor configuration.
`icmp`	ICMP monitor configuration.
`tcp`	TCP monitor configuration.

spec.monitors.dns

Parameter	Description
`acceptRcode`	The RCODE required in the response for an 'up' status: **no-error** or **anything**. The default is **no-error**.
`aliasAddress`	The IP address of the resource that is the destination of this monitor.
`aliasPort`	The port of the resource that is the destination.
`answerContains`	The record types requred in the answer section of the response in order to mark the status of a node up: **query-type** (default), **any-type**, or **anything**.
`enabled`	Specifies whether this monitor is enabled or not: **true** or **false** (default).
`queryName`	The query name that the monitor sends a DNS query for. This is a required field of a DNS monitor.
`queryType`	The DNS query type that the monitor sends: **a** (default) or **aaaa**.
`recv`	The IP address that the monitor looks for in the DNS response's resource record sections.
`reverse`	Enables the monitor operates in reverse mode. When the monitor is in reverse mode, a successful receive string match marks the monitored object down instead of up: **true** or **false** (default).
`interval`	The value applies only when it is ***greater than*** the `timeout`: Endpoints are marked down when unanswered probes exceed the configured interval: **0** to **4294967295**. The default is **5**.
`timeout`	The value applies only when it is ***less than*** the `interval` value: Endpoints are marked down when unanswered probes exceed the configured timeout: **0** to **4294967295**. The default is **5**.
`timeUntilUp`	The amount of time, in seconds, after the first successful response before a node is marked up: **0** to **4294967295**. The default is **0**.
`upInterval`	The frequency, in seconds, at which the system issues the monitor check when the resource is up: **0** to **4294967295**. The default is **0**.

spec.monitors.icmp

Parameter	Description
`enabled`	Specifies whether this monitor is enabled or not: **true** or **false** (default).

Parameter	Description
`interval`	The value applies only when it is ***greater than*** the `timeout`: Endpoints are marked down when unanswered probes exceed the configured interval: **0** to **4294967295**. The default is **5**.
`timeout`	The value applies only when it is ***less than*** the `interval` value: Endpoints are marked down when an unanswered probes exceed the configured timeout: **0** to **4294967295**. The default is **5**.

spec.monitors.tcp

Parameter	Description
`enabled`	Specifies whether this monitor is enabled or not: **true** or **false** (default).
`interval`	The value applies only when it is ***greater than*** the `timeout`: Endpoints are marked down when unanswered probes exceed the configured interval: **0** to **4294967295**. The default is **5**.
`timeout`	The value applies only when it is ***less than*** the `interval` value: Endpoints are marked down when unanswered probe exceed the configured timeout: **0** to **4294967295**. The default is **5**.
`receiveDisableString`	The regular expression, when matched, disables the target.
`receiveString`	The regular expression, when matched, indicated the target is up.
`sendString`	Text string to send to the target.

spec.pool

Parameter	Description
`minActiveMembers`	Specifies the minimum number of members that must be available in one priority group: **0** (default) to **65535**.
`members`	Specifies a list of IP addresses and ports for the service. This is a required field for a pool.

spec.pool.members

Parameter	Description
`address`	Specifies the address of the service. This is a required field of a pool member.
`port`	Specifies the port of the service: **0** to **65535**. The default value is **53**.
`priorityGroup`	Specifies the port of the service: **0** (default) to **8**.

spec.snat

Parameter	Description
`type`	Specifies the type of source address translation to use: **none** (default), **snat**, or **automap**. When using **snat** a `snat.pool` must be defined.
`pool`	Specifies the name of a F5BigCneSnatpool. The name of F5BigCneSnatpool uses its CR (Custome Resource) `metadata.name` parameter. You can **only use** this option when automap and translation are not used.

spec.tcpSettings

Parameter	Description
`clientSide`	Specifies the name of client-side TCP profile F5BigTcpSetting. The name of F5BigTcpSetting uses its CR (Custome Resource) `metadata.name` parameter. If not specified, the default **sys-default-tcp** will be used.
`serverSide`	Specifies the name of server-side TCP profile F5BigTcpSetting. The name of F5BigTcpSetting uses its CR (Custome Resource) `metadata.name` parameter. If not specified, the default **sys-default-tcp** will be used.

spec.udpSettings

Parameter	Description
`clientSide`	Specifies the name of client-side UDP profile F5BigUdpSetting. The name of F5BigUdpSetting uses its CR (Custome Resource) `metadata.name` parameter. If not specified, the default **sys-default-dns-udp** will be used.
`serverSide`	Specifies the name of server-side UDP profile F5BigUdpSetting. The name of F5BigUdpSetting uses its CR (Custome Resource) `metadata.name` parameter. If not specified, the default **sys-default-dns-udp** will be used.

spec.vlans

Parameter	Description
`vlanList`	Specifies a list names of F5BigNetVlan that the virtual server will use to either accept traffic. The name of F5BigNetVlan uses its CR (Custome Resource) `metadata.name` parameter.
`vlanList.item`	A reference to a F5BigNetVlan name.
`disableListedVlans`	When enabled, accept traffic on all VLANs except those defined in the `vlans.vlanlist`. The parameter: **true** (default) or **false**.

F5BigLogProfile Reference

The F5BigLogProfile Custom Resource (CR) configuration parameters. Each heading below represents the top-level parameter element. For example, to set the profile `name`, use `spec.name`.

spec.algLogging

Parameter	Description
`enabled`	Enables logging of application layer gateway (ALG) event messages: **true** or **false** (default).
`publisher`	Name of the log publisher to use for DNS log messages.
`csvFormat`	Enable generating log entries in comma-separated-values (csv) format: **true** or **false** (default).

spec.algLogging.dataChannel

Parameter	Description
`start.mode`	Enables event log messages when the ALG data channel connection is established: **disabled** (default), or **enabled**.
`start.includeDestAddrPort`	Include the destination IP address and port when `dataChannel.start.mode` is enabled: **true** (default) or **false**.
`end.mode`	Enables event log messages when the ALG data channel connection is closed: **disabled** (default), or **enabled**.
`end.includeDestAddrPort`	Include the destination IP address and port when `dataChannel.start.mode` is enabled: **true** (default) or **false**.

spec.algLogging.controlChannel

Parameter	Description
`start.mode`	Enables event log messages when the ALG data channel connection is established: **disabled** (default), or **enabled**.
`start.includeDestAddrPort`	Include the destination IP address and port when `controlChannel.start.mode` is enabled: **true** (default) or **false**.
`end.mode`	Enables event log messages when the ALG data channel connection is closed: **disabled** (default), or **enabled**.
`end.includeDestAddrPort`	Include the destination IP address and port when `controlChannel.start.mode` is enabled: **true** (default) or **false**.

spec.dns

Parameter	Description
`enabled`	Enables logging of DNS event messages: **true** or **false** (default).
`description`	User defined description for the logging profile.
`queryLogging`	Enable DNS query logging: **true** (default) or **false**.
`responseLogging`	Enable DNS response logging: **true** or **false** (default).
`completeAnswer`	Include all the resource records in response log messages: **true** (default) or **false**.
`queryId`	Include the query id in the query and response messages: **true** or **false** (default).
`source`	Include the log message originator in the query and response messages: **true** (default) or **false**.
`timeStamp`	Include the timestamp in the query and response messages: **true** (default) or **false**.
`view`	Include the view in the query message: **true** (default) or **false**.
`publisher`	Name of the log publisher to use for DNS log messages.

spec.firewall

Parameter	Description
`enabled`	Enables logging of firewall event messages: **true** or **false** (default).
`flowspec.publisher`	Specifies the name of the log publisher to be used for the flowspec route injector log messages.

spec.firewall.ipIntelligence

Parameter	Description
`publisher`	Specifies the name of the log publisher used for IP Intelligence log messages.
`geo`	Enables logging of geo location in shun IP Intelligence event: **true** or **false** (default).
`rtbh`	Enables logging of Remote Triggered Black Hole (RTBH) IP Intelligence events: **true** or **false** (default).
`scrubber`	Enables logging of scrubber IP Intelligence events: **true** or **false** (default).
`shun`	Enables logging of shun IP Intelligence events: **true** or **false** (default).
`translation`	Enables logging of translated server side fields in IP Intelligence log messages. Translated fields include Source Address/Port, Destination Address/Port, IP Protocol, Route Domain and Vlan: **true** or **false** (default).
`aggregateRate`	Specifies the rate limit of all combined ipIntelligence log messages per second: **0** to **4294967295**. The default is **4294967295**.

spec.firewall.trafficStats

Parameter	Description
`activeFlows`	Enables logging the number of active flows on client side: **true** or **false**.
`reapedFlows`	Enables logging the number of reaped flows on client side: **true** or **false** (default).
`missedFlows`	Enables logging the number of TCP packets (non SYN/ACK) were dropped because of the flow table lookup failed: **true** or **false** (default).
`synCookies`	Enables logging the number of syncookies generated, accepted and rejected in the context globally and per virtual server. These log messages will be generated periodically: **true** or **false** (default).
`syncookiesWhitelist`	Enables logging the number of syncookies whitelist hits, accepted and rejected in the context globally and per virtual server. These log messages will be generated periodically: **true** or **false** (default).
`publisher`	Specifies the name of the log publisher to be used for trafficStats log messages.

spec.firewall.network

Parameter	Description
`publisher`	Specifies the name of the log publisher to be used for network log messages.
`aggregateRate`	Specifies the rate limit of all combined network log messages per second. Beyond this rate limit, log messages are not logged until the threshold drops below the specified rate: **0** to **4294967295**. The default is **4294967295**.
`log.aclMatchAccept`	Enables logging the packets that match ACL rules configured with action = Accept or action = Accept Decisively: **true** or **false** (default).
`events.aclMatchDrop`	Enables logging the packets that match ACL rules configured with action = Drop: **true** or **false** (default).
`events.aclMatchReject`	Enables logging the packets that match ACL rules configured with action = Reject: **true** or **false** (default).
`events.ipErrors`	Enables logging of IP error packets: **true** or **false** (default).
`events.tcpErrors`	Enables logging of TCP error packets: **true** or **false** (default).
`events.tcpEvents`	The default is **false**.
`events.translationFields`	Enables logging of translated server side fields in ACL match and TCP events. Translated fields include Source Address/Port, Destination Address/Port, IP Protocol, Route Domain and Vlan: **true** or **false** (default).
`log.geoAlways`	Enables logging the Geographic IP Location information fields in ACL match and TCP logging. Geographic information includes the country code of Source Address and Destination Address: **true** or **false** (default).
`log.uuidField`	Enables logging the ACL rule UUID field in ACL match and TCP logging. If the acl_rule_uuid field is explicitly specified in field-list or user-defined formats, UUID value will be logged regardless of state of this option: **true** or **false** (default).

Parameter	Description
`ratelimit.aclMatchAccept`	Specifies rate limits for the logging of packets that match ACL rules configured with action = Accept or action = Accept Decisively. This option is effective only if logging of this message type is enabled: **0** to **4294967295**. The default is **4294967295**.
`ratelimit.aclMatchDrop`	Specifies rate limits for the logging of packets that match ACL rules configured with action = Drop. This option is effective only if logging of this message type is enabled: **0** to **4294967295**. The default is **4294967295**.
`ratelimit.aclMatchReject`	Trate limits for the logging of packets that match ACL rules configured with action = Reject. This option is effective only if logging of this message type is enabled: **0** to **4294967295**. The default is **4294967295**.
`ratelimit.logIpErrors`	Specifies rate limits for the logging of IP error packets. This option is effective only if logging of this message type is enabled: **0** to **4294967295**. The default is **4294967295**.
`ratelimit.logTcpErrors`	Specifies rate limits for the logging of TCP error packets. This option is effective only if logging of this message type is enabled: **0** to **4294967295**. The default is **4294967295**.
`ratelimit.logTcpEvents`	Specifies rate limits for the logging of TCP events on client side. This option is effective only if logging of this message type is enabled: **0** to **4294967295**. The default is **4294967295**.
`format.fieldListDelimiter`	Specifies the delimiter string when storage format type is **field-list**. Special character $ should not be used in delimiter string as it is reserved for internal usage. The default value is **,**.
`format.type`	**none** (default), **field-list**, **user-defined**.
`format.userDefinedFieldList`	Specifies the format of log message in form of user defined string. This option is valid when storage format type is user-defined. The default value is **none**.
`format.networkFieldList`	Specifies a set of fields to be logged. This option is valid when storage format type is **field-list**.

spec.firewall.portMisuse

Parameter	Description
`publisher`	Specifies the name of the log publisher to be used for portMisuse log messages.
`aggregateRate`	Specifies the rate limit of all combined portMisuse log messages per second. Beyond this rate limit, log messages are not logged until the threshold drops below the specified rate. The default value is **4294967295**.

spec.nat

Parameter	Description
`enabled`	Enables logging of NAT event messages: **true** or **false** (default).

Parameter	Description
`publisher`	Specifies the name of the log publisher used for logging Network Address Translation events.
`lsnLegacyMode`	Enables LSN legacy CGNAT/LSN logging instead of the new Firewall NAT logging: **true** or **false** (default). LSN Legacy Mode can only log Dynamic PAT source translation events, cannot log Static NAT or Static PAT source translation events, cannot log Destination translation events, and does not support Firewall NAT logging features such as LocalDB, ArcSight, or Log Throttling.
`logSubscriberID`	Enables logging of subscriber IDs associated with a subscriber IP address: **true** or **false** (default).
`aggregateRateLimit`	Specifies the rate limit of all combined Network Address Translation log messages per second. The default value is **4294967295**.

spec.nat.outbound

Parameter	Description
`start.mode`	Enables event log entries at start of the translation event for a NAT client: **disabled** (default), **enabled**, and **backup**.
`start.includeDestAddrPort`	Include the destination IP address and port in the log message: **true** (default) or **false**.
`start.ratelimit`	Specifies rate limits for logging Outbound Start and corresponding logging network events: **0** to **4294967295**. The default is **4294967295**.
`start.formatType`	**none** (default), **field-list**, **user-defined**.
`start.delimiter`	Specifies a delimiter when the storage format type is **field-list**. The special character dollar sign, $ should not be used in delimiter string as it is reserved for internal usage. The default value is **,**.
`start.fieldList`	Specifies a set of network fields to be logged: **[“context_name”,“dest_ip”,“dest_port”,“event_name”,“protocol”]**.
`start.userDefinedFieldList`	Specifies a set of network fields to be logged: $context_name$**{dest_ip}**$dest_port$**{event_name}${protocol}**.
`end.mode`	Enables event log entries at end of translation event for a NAT client: **disabled** (default), **enabled**, or **backup**.
`end.includeDestAddrPort`	Include the destination IP address and port in the log message: **true** (default) or **false**.
`end.ratelimit`	Specifies rate limits for logging Outbound End and cooresponding events: **0** to **4294967295**. The default is **4294967295**.
`end.formatType`	**none** (default), **field-list**, **usr-defined**.
`end.delimiter`	Specifies the delimiter when storage format type is **field-list**. The special character dollar sign, $ should not be used in delimiter string as it is reserved for internal usage. The default value is **,**.
`end.fieldList`	Specifies a set of network fields to be logged: **[“context_name”,“dest_ip”,“dest_port”,“event_name”,“protocol”]**.

Parameter	Description
`end.userDefinedFieldList`	User-Defined-List specifies a set of network fields to be logged: $context_name$**{dest_ip}**$dest_port$**{event_name}${protocol}**.

spec.nat.inbound

Parameter	Description
`start.mode`	Enables log entries at the start of the incoming connection event for a translated endpoint: **disabled** (default), **enabled**, or **backup**.
`start.ratelimit`	Specifies rate limits for logging Inbound Start and cooresponding events: **0** to **4294967295**. The default is **4294967295**.
`start.formatType`	**none** (default), **field-list**, **user-defined**.
`start.delimiter`	Specifies the delimiter when storage format type is **field-list**. Note: The special character dollar sign, $ should not be used in delimiter string as it is reserved for internal usage. The default value is **,**.
`start.fieldList`	Specifies a set of network fields to be logged: **[“context_name”,“dest_ip”,“dest_port”,“event_name”,“protocol”]**.
`start.userDefinedFieldList`	Specifies a set of network fields to be logged: $context_name$**{dest_ip}**$dest_port$**{event_name}${protocol}**.
`end.mode`	Enables event log entries at the end of the incoming connection event for a translated endpoint: **disabled** (default), **enabled**, **backup**.
`end.ratelimit`	Specifies rate limits for logging Inbound End and cooresponding events: **0** to **4294967295**. The default is **4294967295**.
`end.formatType`	**none** (default), **field-list**, **user-defined**.
`end.delimiter`	Specifies the delimiter when storage format type is **field-list**. The special character dollar sign, $ should not be used in delimiter string as it is reserved for internal usage. The default value is **,**.
`end.fieldlist`	Specifies a set of network fields to be logged: **[“context_name”,“dest_ip”,“dest_port”,“event_name”,“protocol”]**.
`end.userDefinedFieldList`	Specifies a set of network fields to be logged: $context_name$**{dest_ip}**$dest_port$**{event_name}${protocol}**.

spec.nat.quotaExceeded

Parameter	Description
`mode`	Enables event log entries when a NAT client exceeds allocated resources: **disabled** (default), **enabled**, or **backup**.
`ratelimit`	Specifies the Quota Exceeded Rate Limit to set throttling rate limits for logging Quota exceeded network events: **0** to **4294967295**. The default value is **4294967295**.
`formatType`	**none** (default), **field-list**, or **user-defined**.

Parameter	Description
`delimiter`	Specifies a delimiter when storage format type is Field-List. Note: The special character dollar sign, $ should not be used in delimiter string as it is reserved for internal usage. The default value is **,**.
`fieldList`	Specifies a set of network fields to be logged: **[“context_name”,“dest_ip”,“dest_port”,“event_name”,“protocol”]**.
`userDefinedFieldList`	Specifies a set of network fields to be logged: $context_name$**{dest_ip}**$dest_port$**{event_name}${protocol}**.

spec.nat.errors

Parameter	Description
`mode`	Enables event log entries when a NAT translation errors occur: **disabled** (default), **enabled**, or **backup**.
`ratelimit`	Specifies rate limits for the logging Errors network and cooresponding events. The default value is **4294967295**.
`formatType`	**none** (default), **field-list**, or **user-defined**.
`delimeter`	Delimiter is valid when storage format type is field-list. he special character dollar sign, $ should not be used in delimiter string as it is reserved for internal usage. The default value is **,**.
`fieldList`	Specifies a set of network fields to be logged: **[“context_name”,“dest_ip”,“dest_port”,“event_name”,“protocol”]**.
`userDefinedFieldList`	Specifies a set of user defined network fields to be logged: $context_name$**{dest_ip}**$dest_port$**{event_name}${protocol}**.

spec.pe

Parameter	Description
`reportingType`	Specifies the report type: **session-reporting** or **flow-reporting**.
`reportingFields`	Specifies a list of reporting fields. For example **- “Source IP”**. For a full list, review the F5BigLogProfile Reporting Fields.
`formatScript`	Specifies a list of format scripts. For example **src-ip:[PEM::flow stats reported src-ip]**. For a full list, review the [F5BigLogProfile Format Script].
`usageVolumeThreshold.downlink`	Specifies the downlink usage volume threshold.
`usageVolumeThreshold.uplink`	Specifies the uplink usage volume threshold.
`usageVolumeThreshold.total`	Specifies the total usage volume threshold.
`intervalThreshold`	Specifies the interval threshold.

spec.protocolInspection

Parameter	Description
`enabled`	Enables logging of protocol inspection event messages: **true** or **false** (default).
`publisher`	Name of the log publisher to use for DNS log messages.
`logPacket`	Enables logging the packet of any payload matching the protocol inspection profile: **true** or **false** (default).

F5BigLogProfile Reporting Fields

The F5BigLogProfile Custom Resource (CR) supports the following reporting fields:

session-reporting

- **All**
- **Report ID**
- **Timestamp (s)**
- **Timestamp (ms)**
- **Record Type**
- **Subscriber ID**
- **Subscriber ID Type**
- **Application ID**
- **Volume (Uplink)**
- **Volume (Downlink)**
- **Report Version**
- **3GPP Parameters**
- **Calling Station ID**
- **Called Station ID**
- **Last Record Sent (s)**
- **Concurrent Flows**
- **New Flows**
- **Terminated Flows**
- **Total Transactions**
- **Successful Transactions**
- **Duration (s)**
- **Record Reason**
- **Gate Drop Count**

flow-reporting

- **All**
- **Report ID**
- **Timestamp (s)**
- **Timestamp (ms)**
- **Record Type**
- **Subscriber ID**
- **Subscriber ID Type**
- **Application ID**
- **Volume (Uplink)**
- **Volume (Downlink)**
- **Report Version**
- **Source IP**
- **Source Port**
- **Destination IP**
- **Destination Port**
- **IP Protocol**
- **URL Category ID**
- **Flow Start (s)**

- **Flow Start (ms)**
- **Flow Stop (s)**
- **Flow Stop (ms)**
- **Total Transactions**
- **Route Domain**
- **Vlan ID**
- **Flow Duration (s)**
- **Flow Duration (ms)**
- **SNI**
- **Video Content Provider**
- **Video Resolution**
- **Video Bit Rate**
- **Handshake RTT**

Software Releases

This document details the CNFs software releases to date by version, and lists the CNFs software images for each release.

1.0.0 LA

Supported Platforms

Robin CNP release 5.3.5-207

Software images

Container	Version
f5ingress	v6.0.13
tmm-img	v2.0.5
tmrouted-img	v0.8.21
f5-l4p-engine	v1.3.48
f5-nsec-ips-daemon	v1.4.14
f5-debug-sidecar	v5.54.3
f5-toda-tmstatsd	v1.7.7
f5-fluentbit	v0.2.0
f5-fluentbit	v1.2.29
f5dr-img	v0.5.7
f5-dssm-store	v1.21.0
f5-fluentd	v1.4.9
f5-dssm-upgrader	1.0.5
opentelemetry-collector	0.46.0

CRD bundle

Bundle	Version
f5-cnf-crds-n6lan	0.36.7

Feedback

Provide feedback to improve this document by emailing spkdocs@f5.com.