Gigamon®

Universal Cloud Tap-Container Deployment Guide

GigaVUE Cloud Suite

Product Version: 6.10

Document Version: 1.1

(See Change Notes for document updates.)

Gigamon Inc.
3300 Olcott Street
Santa Clara, CA 95054
408.831.4000

Change Notes

When a document is updated, the document version number on the cover page will indicate a new version and will provide a link to this Change Notes table, which will describe the updates.

Product Version	Document Version	Date Updated	Change Notes
6.10	1.1	03/28/2025	This deployment guide has been completely restructured for better usability and a streamlined deployment process. Key updates: • **Improved Content Flow** – Logical organization for easier navigation. • **Clearer Deployment Steps** – Simplified, actionable instructions. • **Use Case-Specific Enhancements** – Added use cases with tailored roadmaps, prerequisites, and next steps. • **Updated Visuals** – New diagrams, flow charts, and examples for clarity.
6.10	1.0	03/07/2025	The original release of this document with 6.10.00 GA.

Contents

Overview of Universal Cloud Tap - Container

Universal Cloud Tap - Container (UCT-C), earlier known as Universal Container Tap (UCT), is a containerized component deployed as a DaemonSet. This ensures that it runs on every worker node within a cluster. UCT-C provides network broker features in a containerized form and can perform traffic acquisition, basic filtering, and tunneling support. It is deployed as a Pod in the given worker node where the workloads run.

UCT-C is deployed by the Kubernetes orchestrator and not by GigaVUE-FM. The traffic acquisition process is initiated by UCT-C.

Following are the modules implemented in UCT-C:

- **Traffic Acquisition** - UCT-C supports traffic acquisition by replicating the traffic from the worker pods.
- **Filtering Module** - UCT-C provides basic filtering based on 5-Tuple. The filtering configuration is pushed by GigaVUE-FM.
- **Tunneling Modules** - UCT-C supports L2GRE, VXLAN, and TLS-PCAPng tunneling to send the tapped traffic to the GigaVUE V Series Nodes or tools.

Architecture of Universal Cloud Tap - Container

UCT-C enables the capture of network traffic directly from Kubernetes workload pods and facilitates its redirection to designated tunnel destinations, such as V Series appliances or monitoring and security tools.

The controller is deployed as a Kubernetes Deployment with a replica count set to 1, ensuring that only a single instance of the controller operates within the Kubernetes cluster. A corresponding Controller Service is created to enable seamless communication with the controller.

The Universal Cloud Tap - Container works with the following components:

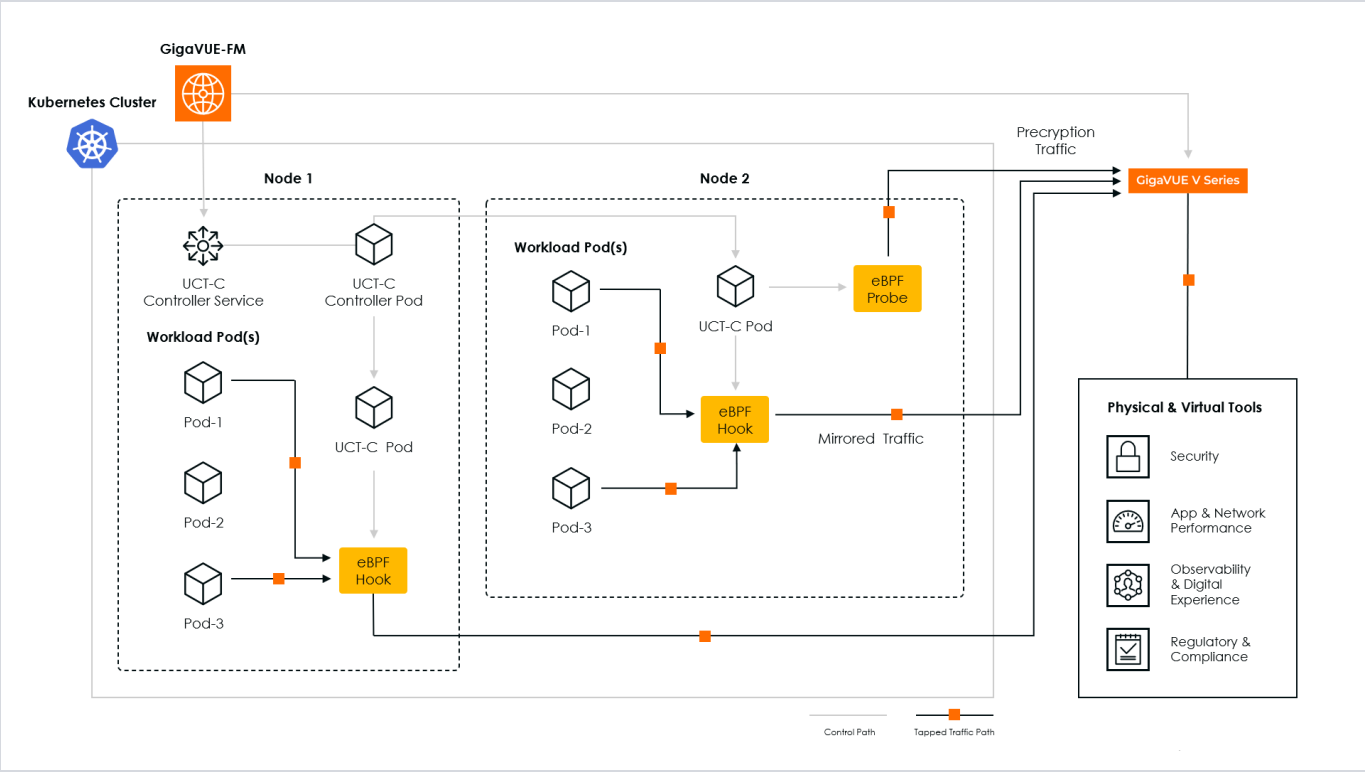
- **GigaVUE-FM Fabric Manager** is a browser-based fabric management and orchestration interface that provides a single pane of glass visibility, management, and orchestration of both the physical and virtual.

- **UCT-C Tap** is the primary UCT-C module that collects the workload traffic, filters it, and tunnels the filtered traffic directly to the tools or through the GigaVUE V Series Nodes. It also sends the traffic policy statistics and heartbeats to the UCT-C Controller. UCT-C Tap must run as a **privileged pod**.

  NOTE: UCT-C uses eBPF (extended Berkeley Packet Filter) to tap traffic from user pods. eBPF runs on the Linux kernel and requires privileged pod permission in Kubernetes. UCT-C Tap pods require SYS_ADMIN and NET_ADMIN privileges to attach eBPF Hooks, run commands in other namespaces, and run low-level networking commands.

- **UCT-C Controller** is the management component of UCT-C that controls and communicates with UCT-C Tap. UCT-C Controller collects the data from UCT-C Taps and sends the collected statistics and heartbeats to GigaVUE-FM.

The following diagram illustrates the various components involved in the UCT-C solution.



The UCT-C Controller acts as the central communication link between GigaVUE-FM and UCT-C Taps, managing registration, policy deployment, data collection, and statistics reporting. The process begins with the UCT-C Controller registering with GigaVUE-FM, after which the UCT-C Taps are also registered through the Controller. Once registered, GigaVUE-FM communicates with the UCT-C Taps exclusively through the UCT-C Controller, ensuring a structured flow of configuration and data.

When GigaVUE-FM deploys traffic policies, the controller receives and distributes them to the connected UCT-C Taps, instructing them on how to filter and process network traffic. The filtered network packets are then either tunneled directly to connected monitoring and security tools or sent through GigaVUE V Series Nodes deployed within a supported GigaVUE Cloud Suite environment.

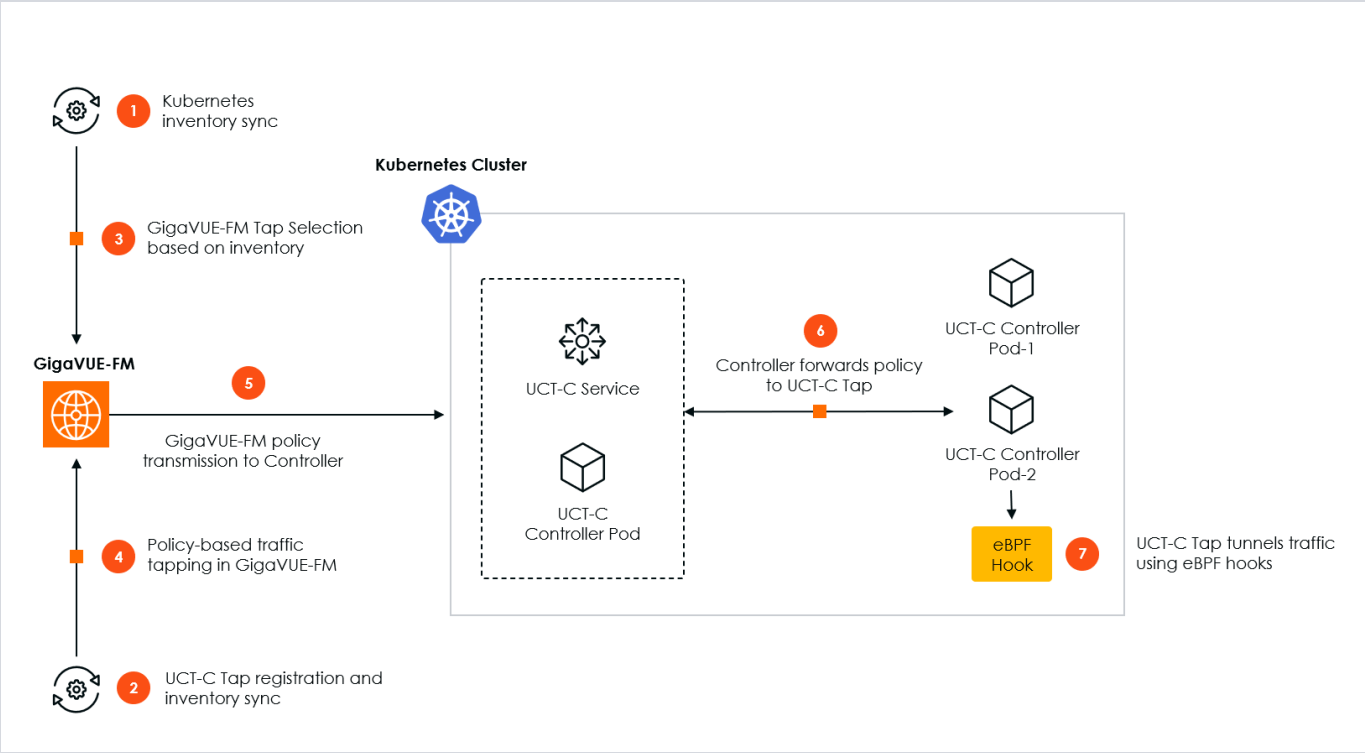
The UCT-C Controller also manages ongoing communication and monitoring by collecting data from the UCT-C Taps, including traffic statistics and system health information. It consolidates these details and transmits them back to GigaVUE-FM, ensuring continuous visibility into network activity. Additionally, the controller sends periodic heartbeats to confirm the status and availability of UCT-C Taps, enabling GigaVUE-FM to monitor the system's overall health and performance.

The UCT-C solution can tap mirrored and Precryption traffic (refer to the architecture diagram above). UCT-C provides multiple tunneling options to send the acquired traffic to external systems for further processing or analysis. The supported tunneling types are:

- **Layer 2 Generic Routing Encapsulation (L2GRE)**: A tunneling protocol that encapsulates Layer 2 traffic to transport it over an IP network.
- **Virtual Extensible LAN (VXLAN)**: A network virtualization technology that encapsulates Layer 2 Ethernet frames inside Layer 3 UDP packets. It is typically used to create a virtualized Layer 2 network that runs over a Layer 3 infrastructure, allowing for better scalability and flexibility in cloud environments.
- **Transport Layer Security - PCAPng (TLS-PCAPng)**: A secure tunneling method that captures traffic in a standardized PCAPng format, encrypted using TLS, ensuring the confidentiality and integrity of the data during transmission.

These tunneling protocols allow the UCT-C to send the captured traffic to the GigaVUE V Series Nodes or external network monitoring tools, ensuring that all network traffic data is securely transmitted and available for analysis.

Traffic Tapping in Kubernetes Using GigaVUE-FM



1. **Kubernetes inventory sync:**

   When the Controller starts, it gathers inventory information about the Kubernetes cluster in which it runs, including data on worker nodes, Kubernetes services, and Pods. The controller then sends this information to GigaVUE-FM. It also registers with the Kubernetes cluster to receive real-time notifications of any changes in the cluster's inventory. As changes occur, the controller pushes these updates to GigaVUE-FM.

2. **UCT-C Tap Registration and Inventory Sync:**

   When a UCT-C Tap starts on a node, it sends its registration message to the controller, which then forwards it to GigaVUE-FM. This way, GigaVUE-FM knows which UCT-C Tap runs on which worker node.

3. **GigaVUE-FM Tap Selection Based on Inventory:**

   Access to inventory information and knowing which UCT-C Taps are running on which nodes allow GigaVUE-FM to apply configuration policies and rules. You can choose which pods should be tapped for traffic based on Kubernetes resources such as nodes, namespaces, and pods.

4. **Policy-Based Traffic Tapping in GigaVUE-FM:**

When a user configures a policy for tapping, GigaVUE-FM determines the list of worker pods that need to be tapped based on the Source Selection criteria. It also identifies the nodes on which these pods are running. Subsequently, GigaVUE-FM requests the controller to configure the UCT-C Tap on each node to tap traffic from the specified worker pods.

5. **GigaVUE-FM Policy Transmission to Controller:**

   When GigaVUE-FM sends this policy to the controller.

6. **Controller Forwards Policy to UCT-C Tap:**

   The controller forwards the policy configuration to the UCT-C Tap, which is responsible for implementing the policy. The UCT-C Tap then applies the policy to tap traffic from the specified worker pods.

7. **UCT-C Tap Tunnels Traffic using eBPF Hooks:**

   After this, the UCT-C Tap adds eBPF hooks at the TC layer to get the traffic and tunnel it to the designated endpoint specified in the policy. Once this process is complete, traffic to and from these pods is tunneled to the destination outlined in the policy.

Communication between UCT-C and GigaVUE-FM

This section describes the different communication channels between UCT-C and GigaVUE-FM. Refer to the following section:

- Controller to GigaVUE-FM Communication
- GigaVUE-FM to Controller Communication
- Controller to Tap Communication
- Tap to Controller Communication

Controller to GigaVUE-FM Communication

The communication between the Controller and the GigaVUE-FM uses TLS or mTLS protocols. The process includes the following responsibilities:

- Identify how to reach GigaVUE-FM.
- Connect to GigaVUE-FM to register itself.
- Report the initial inventory to GigaVUE-FM.
- Report any changes in the inventory to GigaVUE-FM.
- Forward UCT-C Tap registration information to GigaVUE-FM.
- The Controller maintains a record of each tap in its memory table.
- Report the heartbeat of each UCT-C Tap instance to GigaVUE-FM.
- Report Volume Based Licensing (VBL) statistics to GigaVUE-FM.
- Report Pod/Policy statistics to GigaVUE-FM.

NOTE: Data packets from the selected source to the tunnels do not pass through the Controller

GigaVUE-FM to Controller Communication

GigaVUE-FM to Controller communication utilizes TLS or mTLS. The process includes the following responsibilities:

- Send Policy configuration requests for UCT-C Pod(s).
- Register UCT-C instances with the Controller for management
- Security through TLS and mTLS.

Controller to Tap Communication

The communication between the Controller and the Tap utilizes a secure channel, ensuring it is encrypted. The process includes the following responsibilities:

- Forwards policy configuration requests received from GigaVUE-FM to the Tap.
- Secure the communication channel.
- Traffic Forwarding and Routing Configuration.
- Traffic Filtering and Transformation Updates.

Tap to Controller Communication

The communication between the Tap and the Controller utilizes a secure channel, ensuring it is encrypted. Each tap looks up the Controller Kubernetes Service. Taps do not communicate directly with GigaVUE-FM; all communications are routed through the Controller.

The following actions occur during this process:

- Tap connects to the Controller to register with GigaVUE-FM.
- Sends heartbeat requests.
- Sends Volume Based Licensing (VBL) statistics.
- Sends pod or policy statistics.

Traffic Acquisition using UCT-C

You can acquire traffic from multiple container pod instances using UCT-C. The acquired traffic is then forwarded securely to the GigaVUE V Series Node for advanced processing, including core intelligence and additional GigaSMART capabilities, which can enhance network visibility, performance analysis, and security monitoring.

Refer to the following topics for more details about Traffic Acquisition using UCT-C:

- Precryption
- Secure Tunnels

Precryption

License: Requires **SecureVUE Plus** license.

Gigamon Precryption™ technology[1] redefines security for virtual, cloud, and containerized applications, delivering plain text visibility of encrypted communications to the full security stack without the traditional cost and complexity of decryption.s

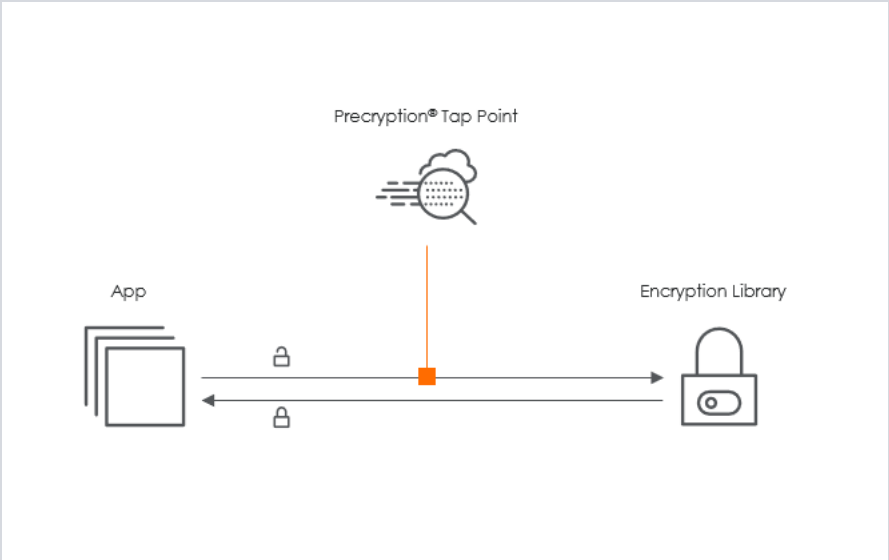
This section explains:

- How Gigamon Precryption Technology Works
- Why Gigamon Precryption
- Key Features
- Key Benefits
- Precryption Technology on Single Node
- Precryption Technology on Multi-Node
- Supported Platforms
- Prerequisites

How Gigamon Precryption Technology Works

Precryption technology leverages native Linux functionality to tap, or copy, communications between the application and the encryption library, such as OpenSSL.

[1] **Disclaimer**: The Precryption feature allows users to acquire traffic after it has been decrypted. This traffic can be acquired from both virtual machine (VM) and container-based solutions, and is then sent to the V Series product for further processing. The Precryption feature provides an option to use encrypted tunnels for communication between the acquisition (via UCT-C or UCT-V) of unencrypted traffic and the traffic processing (at the V Series) which will better safeguard the traffic while in transit. However, if a user does not use the option for encrypted tunnels for communication, decrypted traffic will remain unencrypted while in transit between the point of acquisition and processing. Please note that this information is subject to change, and we encourage you to stay updated on any modifications or improvements made to this feature. By using this feature, you acknowledge and accept the current limitations and potential risks associated with the transmission of decrypted traffic.



In this way, Precryption captures network traffic in plain text, either before it has been encrypted or after it has been decrypted. Precryption functionality doesn't interfere with the message's actual encryption or transmission across the network. There's no proxy, retransmissions, or break-and-inspect. Instead, this plaintext copy is forwarded to the Gigamon Deep Observability Pipeline for further optimization, transformation, replication, and tool delivery.

Precryption technology is built on GigaVUE® Universal Cloud Tap (UCT) and works across hybrid and multi-cloud environments, including on-prem and virtual platforms. As a bonus, UCT with Precryption technology runs independently of the application and doesn't have to be baked into the application development life cycle.

Why Gigamon Precryption

GigaVUE Universal Cloud Tap with Precryption technology is a lightweight, friction-free solution that eliminates blind spots present in modern hybrid cloud infrastructure. It provides East-West visibility into virtual, cloud, and container platforms. It delivers unobscured visibility into all encryption types, including TLS 1.3, without managing and maintaining decryption keys. IT organizations can now manage compliance, keep private communications private, architect the necessary foundation for Zero Trust, and boost security tool effectiveness by a factor of 5x or more.

Key Features

The following are the key features of this technology:

- Plain text visibility into communications with modern encryption (TLS 1.3, mTLS, and TLS 1.2 with Perfect Forward Secrecy).
- Plain text visibility into communications with legacy encryption (TLS 1.2 and earlier).

- Non-intrusive traffic access without agents running inside container workloads.
- Elimination of expensive resource consumption associated with traditional traffic decryption.
- Elimination of key management required by traditional traffic decryption.
- Zero performance impact based on cipher type, strength, or version.
- Support across hybrid and multi-cloud environments, including on-prem, virtual, and container platforms.
- Keep private communications private across the network with plaintext threat activity delivered to security tools.
- Integration with Gigamon Deep Observability Pipeline for the full suite of optimization, transformation, and brokering capabilities.

Key Benefits

The following are the key benefits of this technology:

- Eliminate blind spots for encrypted East-West (lateral) and North-South communications, including traffic that may not cross firewalls.
- Monitor application communications with an independent approach that enhances development team velocity.
- Extend security tools' visibility to all communications, regardless of encryption type.
- Achieve maximum traffic tapping efficiency across virtual environments.
- Leverage a 5–7x performance boost for security tools by consuming unencrypted data.
- Support a Zero Trust architecture founded on deep observability.
- Maintain privacy and compliance adherence associated with decrypted traffic management.

How Gigamon Precryption Technology Works

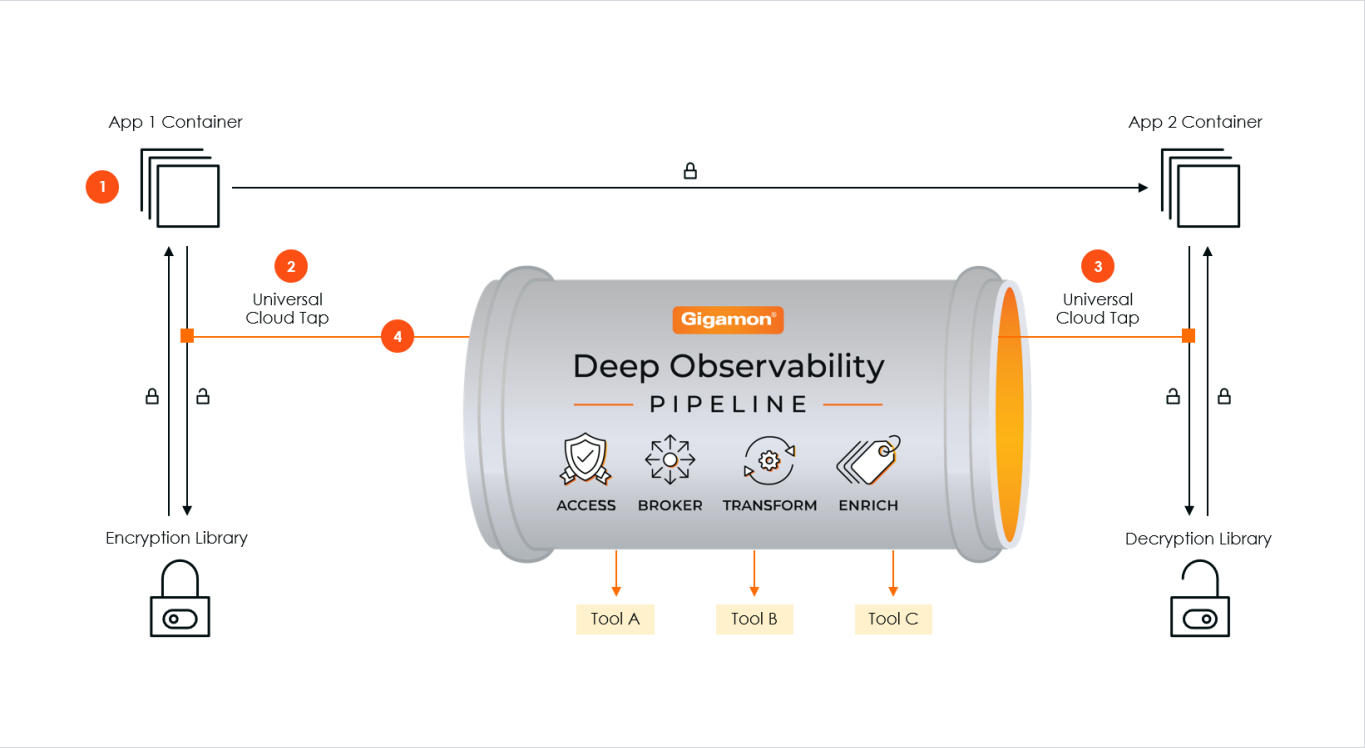
This section explains how Precryption technology works on single nodes and multiple nodes in the following sections:

- Precryption Technology on Single Node
- Precryption Technology on Multi-Node

Precryption Technology on Single Node

1. When any application needs to encrypt a message, it uses an encryption library, such as OpenSSL, to perform the actual encryption.
2. GigaVUE Universal Cloud Tap (UCT), enabled with Precryption technology, gets a copy of this message before it's encrypted on the network.
3. The encrypted message is sent to the receiving application with unmodified encryption—no proxy, no re-encryption, no retransmissions.
4. GigaVUE UCT creates packet headers as needed, encapsulates them in a tunnel, and forwards them to GigaVUE V Series in the deep observability pipeline. Gigamon optimizes, transforms, and delivers data to tools without further decryption.

Precryption Technology on Multi-Node



1. When any application needs to encrypt a message, it uses an encryption library, such as OpenSSL, to perform the actual encryption.
2. GigaVUE Universal Cloud Tap (UCT), enabled with Precryption, gets a copy of this message before it's encrypted on the network.
3. Optionally, GigaVUE UCT enabled with Precryption can also acquire a copy of the message from the server end after the decryption.
4. GigaVUE UCT creates packet headers as needed, encapsulates them in a tunnel, and forwards them to V Series in the deep observability pipeline. There, they are further enriched, transformed, and delivered to tools without further decryption.

Supported Platforms

VM environments: Precryption™ is supported on the following VM platforms where UCT-V is supported:

Platform Type	Platform
Public Cloud	• AWS • Azure • GCP (via Third Party Orchestration)
Private Cloud	• OpenStack • VMware ESXi (via Third Party Orchestration only) • VMware NSX-T (via Third Party Orchestration only)

Container environments: Precryption™ is supported on the following container platforms where UCT-C is supported:

Platform Type	Platform
Public Cloud	• EKS • AKS • GKE
Private Cloud	• OpenShift • Native Kubernetes (VMware)

Prerequisites

Points to Note

- OpenSSL version 1.0.2, version 1.1.0, version 1.1.1, and version 3.x.
- For UCT-C, worker pods should always have libssl installed to ensure that UCT-C Tap can tap the Precryption packets from the worker pods whenever libssl calls are made from the worker pods.
- For GigaVUE-FM, you must add port 5671 in the security group to capture the statistics.
- Port 9900 should be enabled in security group settings on the UCT-V controller to receive the statistics information from UCT-V.
- For UCT-C, you must add port 42042 and port 5671 to the security group.
- Precryption is supported only on Linux systems running Kernel version 4.18 or later.

License Prerequisite

- Precryption™ requires a SecureVUE Plus license.

Supported Kernel Version

Precryption is supported for Kernel Version 4.18 and above for all Linux and Ubuntu Operating Systems. For the Kernel versions below 4.18, refer to the following table:

Kernel-Version	Operating System
4.18.0-193.el8.x86_64	RHEL release 8.2 (Ootpa)
4.18.0-240.el8.x86_64	RHEL release 8.3 (Ootpa)
4.18.0-305.76.1.el8_4.x86_64	RHEL release 8.4 (Ootpa)
4.18.0-348.12.2.el8_5.x86_64	RHEL release 8.5 (Ootpa)
4.18.0-372.9.1.el8.x86_64	RHEL release 8.6 (Ootpa)
4.18.0-423.el8.x86_64	RHEL release 8.7 Beta (Ootpa)
4.18.0-477.15.1.el8_8.x86_64	RHEL release 8.8 (Ootpa)
5.3.0-1024-kvm	Ubuntu 19.10
4.18.0-305.3.1	Rocky Linux 8.4
4.18.0-348	Rocky Linux 8.5
4.18.0-372.9.1	Rocky Linux 8.6
4.18.0-425.10.1	Rocky Linux 8.7
4.18.0-477.10.1	Rocky Linux 8.8
4.18.0-80.el8.x86_64	CentOS 8.2
4.18.0-240.1.1.el8_3.x86_64	CentOS 8.3
4.18.0-305.3.1.el8_4.x86_64	CentOS 8.4
4.18.0-408.el8.x86_64	CentOS 8.5

For more details, refer to Gigamon TV.

Notes

- Refer to *Configure Precryption in UCT-V* topic in the respective GigaVUE Cloud Suite Guides for details on how to enable Precryption™ in VM environments.
- Refer to Configure Precryption in UCT-C for details on how to enable Precryption™ in container environments.

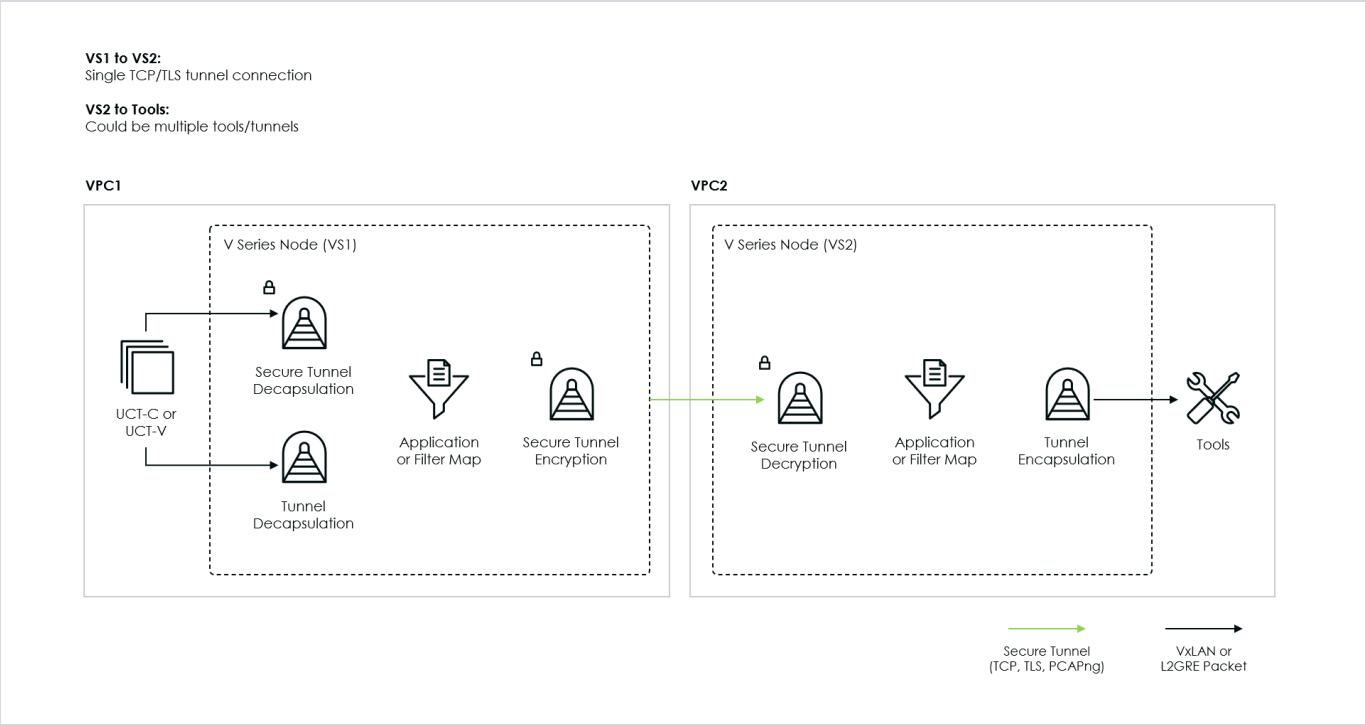
Secure Tunnels

Secure Tunnels securely transfer the cloud-captured packets on UCT-V and UCT-C to a GigaVUE V Series Node. The data from UCT-C is encapsulated in PCAPng format, and the encrypted data is sent over a TLS connection to a GigaVUE V Series Node.

Secure Tunnels can also transfer the captured packets from a GigaVUE V Series Node to another GigaVUE V Series Node or GigaVUE HC Series.

In the case of GigaVUE V Series Node to GigaVUE V Series node, the traffic from the GigaVUE V Series Node 1 is encapsulated using PCAPng format and transported to GigaVUE V Series Node 2, where the traffic is decapped. The secure tunnels between a V Series Node and a V Series Node have multiple use cases.

The GigaVUE V Series Node decapsulates and processes the packet as per the configuration. The decapsulated packet can be sent to the application, such as De-duplication, Application Intelligence, Load balancer, and tool. The Load Balancer on this node can send the packets to multiple V Series Nodes. In this case, the packets can be encapsulated again and sent over a secure tunnel.



Supported Platforms

Secure Tunnels are supported on:

- OpenStack
- Azure
- AWS
- VMware NSX-T (only for Third Party Orchestration)
- VMware ESXi (only for Third Party Orchestration)
- Nutanix (only for Third Party Orchestration)
- Google Cloud Platform (only for Third Party Orchestration)

For information about how to configure secure tunnels, refer to the section Configure Secure Tunnels in UCT-C.

Deployment Overview

Universal Cloud Tap - Container deployment follows a set of steps that include planning and preparation before the actual solution deployment. This section provides an overview of the activities associated with UCT-C deployment. The top-level deployment activities are:

Plan your Deployment	Address Prerequisites	Deploy UCT-C Solution
• Overview of Universal Cloud Tap - Container • Architecture of Universal Cloud Tap - Container • Traffic Acquisition using UCT-C • Security Requirements • Getting UCT-C Container Image • Uploading images in the local repository	• License Information • Supported Platforms for UCT-C • Network Ports Requirements • Compute Requirements • Kernel and CPU Requirements	• Launch GigaVUE-FM • Deploy UCT-C Solution in Kubernetes • Post Deployment • Configure UCT-C Solution using GigaVUE-FM • Configure UCT-C Features

Deployment Planning

When planning the deployment of UCT-C components, it is essential to ensure that all prerequisites, security requirements, and infrastructure considerations are met.

The following sections provide detailed insights into the key aspects of UCT-C deployment planning, including how to acquire and manage the container images, configure security protocols, and upload the images to a local repository for streamlined deployment across your infrastructure.

- Security Requirements
- Getting UCT-C Container Image
- Uploading images in the local repository

Security Requirements

Kubernetes Service Account

To create a Service Account with privileged access for tapping, run the below-listed commands:

```
kubectl create ns uctc
kubectl create sa gigamon -n uctc
```

Use the Gigamon service account in uctc-tap.yaml, which allows UCT-C to appear as a privileged pod.

If you are using the OpenShift Platform, refer to the following sections:

- **Using YAML**: In YAML deployment, for the Red Hat OpenShift Container Platform, use the command below. |

```
oc adm policy add-scc-to-user -z gigamon privileged -n uctc
```

- **Using Helm**: In a Helm deployment for the Red Hat OpenShift Container Platform, set the create value to "True" under securityContextConstraints in values.yaml. This configuration generates a customized Security Context Constraint (SCC) with the necessary permissions required for deploying the UCT-C solution on OpenShift.

```
securityContextConstraints:
create: True
name: "gigamon"
```

NOTE: Security Context is not required in other platforms.

Access and Permissions Required for Deployment

To deploy the solution, you should have the below permissions:

- If you use standard ports like 443 for Controller to GigaVUE-FM communication, the Controller should be launched with privileged access.
- You should have Privileged user access since UCT-C Tap pods require privileged access for Mirroring or Precryption.

Getting UCT-C Container Image

You should create a Secret with registry credentials to pull the image from the Gigamon Software portal to your local server. This is optional if the images are in a local registry that is not password or token protected.

Run the command below to create the required secret.

```
oc create secret generic gigamon --from-file=.dockerconfigjson=<file_name> --
type=kubernetes.io/dockerconfigjson -n uctc
```

Uploading images in the local repository

UCT-C Controller and Tap images will be available for every release build in the Gigamon software portal. Download the UCT-C images from the Gigamon Software portal to your local server. Untar the UCT-C image file and extract the images. Upload **gigamon.gigavue-uctc-cntlr-image.tar** and **gigamon.gigavue-uctc-tap-image.tar** to your local repository. You can use this file name in your deployment files.

NOTE: Contact Technical Support or Contact Sales for information on downloading the respective UCT-C build from the Gigamon software portal.

Deployment Prerequisites

The following are the prerequisites of UCT-C Solution to initiate UCT-C and GigaVUE-FM deployment with the required licenses and network requisites.

- **Kubernetes Knowledge**—To deploy and manage the UCT-C solution, you should have basic knowledge of the Kubernetes platform. Refer to Kubernetes-basics.
- **Knowledge of YAML Files** - A Kubernetes deployment YAML file is a configuration file that defines the desired state of a Kubernetes Deployment. The YAML file is used to create, update, or delete deployments in a Kubernetes cluster. YAML files are available as part of the UCT-C images. You should download and untar the UCT-C image to access the readme files.
  For more information on how to deploy UCT-C using YAML files, refer to YAML Files.
- **Knowledge of Helm Charts** - Helm is a tool for managing packages of pre-configured Kubernetes resources, known as Helm charts. Helm charts are available as part of the UCT-C images. You should download and untar the UCT-C image to access the readme files.
  For more information on how to deploy UCT-C using Helm Charts, refer to Helm Charts.
- **Familiarity with Containerized workloads** - Understanding containerized workloads is essential for effectively deploying UCT-C, as it requires interacting with containerized applications, acquiring network traffic from Kubernetes pods, and ensuring seamless integration with the orchestration platform. Key considerations include network interfaces, resource management, security, and dynamic scaling, ensuring UCT-C can capture and forward relevant traffic without impacting system performance.

Refer to the following sections for detailed information on the prerequisites required for a successful UCT-C deployment

- License Information
- Supported Platforms for UCT-C
- UCT-C Resource Requirements

License Information

All the UCT-C Taps deployed in your environment periodically report the statistics to UCT-C Controller. Then, the UCT-C Controller periodically reports the collective statistics of UCT-C Taps to GigaVUE-FM for Volume-Based Licensing.

In the Volume-Based Licensing scheme, a license entitles specific applications on your devices to use a specified amount of total data volume over the term of the license. The license distribution to individual nodes or devices becomes irrelevant for Gigamon accounting purpose. GigaVUE-FM tracks the total amount of data processed by the UCT-C and the overuse, if any.

Volume-based has a service period of 1 month. The service period defines the duration during which the license is active. Total usage or overage is monitored.

To purchase licenses with the Volume-Based License (VBL) option, contact our Sales. Refer to Contact Sales.

Supported Platforms for UCT-C

The following tables list the different platforms and their Kubernetes version, Container Runtime Interface (CRI), and Container Network Interface (CNI).

NOTE: As an end user, you must have an understanding and knowledge of your container services.

Platform	Kubernetes Version	CRI	CNI
Amazon Elastic Kubernetes Service (EKS)	1.27+	Containerd	VPC
Rancher	1.23 to 1.29	Containerd, CRI-dockerd	Calico, Flannel, Canal, Weave
Red Hat OpenShift	4.13 and 4.14	CRI-O	SDN, OVN
Kubernetes	1.27+	Containerd	Flannel, Cilium, Calico, Multus

Notes:

- UCT-C is platform and CNI independent software and supports other Kubernetes platforms such as Azure Kubernetes Service (AKS), GKE, and VMware Tanzu. If you have any issues or need further assistance, please contact Gigamon Technical Support.
- The Kubernetes versions listed for Red Hat OpenShift (4.13 and 4.14) correspond to OpenShift Container Platform (OCP) versions.

UCT-C Resource Requirements

This section describes how to initiate UCT-C and GigaVUE-FM deployment with the required resource requirements.

NOTE: You should have a Load Balancer and an Ingress Controller in your setup to effectively manage ingress traffic and ensure that applications within the cluster are accessible, secure, and performant.

Refer to the following sections for more details:

- Network Ports Requirements
- Compute Requirements
- Kernel and CPU Requirements

Network Ports Requirements

The following table describes the Kubernetes network requirements for UCT-C to work efficiently.

Universal Cloud Tap - Container deployed inside Kubernetes worker node				
Direction	**Protocol**	**Port**	**Destination CIDR**	**Purpose**
Outbound	TCP	42042	Any IP address	Allows UCT-C to send statistical information to UCT-C Controller.
Outbound	UDP	VXLAN (default 4789)	Any IP address	Allows UCT-C to tunnel traffic to the GigaVUE V Series Node or other destination.
UCT-C Controller deployed inside Kubernetes worker node				
Direction	**Protocol**	**Port**	**Source CIDR**	**Purpose**
Inbound	TCP	8443 (configurable)	GigaVUE-FM IP	Allows GigaVUE-FM to communicate with UCT-C Controller.

Direction	Protocol	Port	Destination CIDR	Purpose
Outbound	TCP	5671	Any IP address	Allows UCT-C Controller to send statistics to GigaVUE-FM.
Outbound	TCP	443	GigaVUE-FM IP	Allows UCT-C Controller to communicate with GigaVUE-FM.

The following table describes the ports that should be opened on GigaVUE-FM:

Direction	Port	Purpose
Inbound	443	GigaVUE-FM REST service port.
Outbound	4433	Allows GigaVUE-FM to communicate with UCT-C Controller.
Outbound	8443	Allows GigaVUE-FM to communicate with UCT-C Controller.
Inbound	5671	Allows UCT-C to send statistics to GigaVUE-FM through Rabbit-MQ port.

Compute Requirements

The following tables describes the minimum compute network requirements for UCT-C.

Compute Instances	vCPU	Memory	Disk Space
UCT-C Controller	1 vCPU	refer to the table below	—
UCT-C Tap	1 vCPU	1GB	—
GigaVUE V Series Node	4 vCPUs	8GB	20GB
GigaVUE V Series Proxy	1 vCPU	1GB	2GB
GigaVUE-FM	4 vCPUs	16GB	41GB

Compute Instances	Memory	Cluster Size
UCT-C Controller	256MB	less than 1000 pods
UCT-C Controller	512MB	2000 pods
UCT-C Controller	1GB	up to 5000 pods

Kernel and CPU Requirements for Universal Cloud Tap - Container

The minimum kernel version requirements for different platforms are as follows:

- **AWS EKS/Azure AKS/Native Kubernetes/OpenShift** - 5.4 and above
- **GKE** - 5.15 (Ubuntu OS)
- **VMware Tanzu Photon**- 4.19 (Photon+ OS)

NOTE: UCT-C solution and its configuration would be successful only if deployed on worker nodes with Ubuntu 5.14 or later. If UCT-C Taps are deployed on worker nodes running GKE's Container Optimized OS (COS), the policy deployment from UI will fail with an "eBPF attachment error."

The minimum CPU and RAM requirements for UCT-C and UCT-C Controller are as follows:

- **UCT-C TAP** - 1 vCPU and 1Gi
- **UCT-C Controller** - 1 vCPU and 1Gi

Deployment of UCT-C Solution

This chapter provides detailed information about deploying the UCT-C solution in a Kubernetes environment and configuring it in GigaVUE-FM.

Refer to the following sections for details:

- Launch GigaVUE-FM
- Deploy UCT-C Solution in Kubernetes
- Post Deployment
- Configure UCT-C Solution using GigaVUE-FM

Launch GigaVUE-FM

You can download the recent GigaVUE-FM image files from the Gigamon Customer Portal. After fetching the image, upload and launch GigaVUE-FM on your GigaVUE V Series Node supported cloud environment. Contact Technical Support of Gigamon for assistance or refer to GigaVUE Cloud Suite for more information on GigaVUE V Series configuration on the supported cloud environments.

NOTE: GigaVUE-FM should be up and ready before you deploy the UCT-C solution.

Deploy UCT-C Solution in Kubernetes

To deploy UCT-C Solution, complete the following steps:

1. Implement external access to the Kubernetes environment (for example, ingress, external public IPs, load balancers) to allow communication between the UCT-C Controller and GigaVUE-FM. Refer to UCT-C Resource Requirements.

2. Ensure that the firewall rules on Kubernetes nodes are met according to the Network Ports Requirements.
3. YAML files or HELM charts are available as part of the UCT-C images. You should untar the UCT-C image to access the readme files.
4. Add the UCT-C images to a private Docker registry or ensure the files can be pulled from the Docker Hub registry. You can spin up or down the UCT-C instances based on traffic load.

You can deploy the UCT-C Controller and Taps in Kubernetes using the YAML files, Helm Charts, or Red Hat OpenShift Platform using OpenShift UI. Refer to the following sections:

- YAML Files
- Helm Charts
- Red Hat OpenShift Platform using OpenShift UI

YAML Files

YAML files will be available for every release build in the Gigamon software portal. Download the respective UCT-C release build from the repository and untar the **.tgz** file. After you finish untarring the file, you can extract the YAML file and update the following fields in **uctc-controller.yaml** and **uctc-tap.yaml** files.

> **NOTE:** Contact Technical Support or Contact Sales for information on downloading the respective UCT-C build from the Gigamon software portal.

Deployment of UCT-C Controller

Follow these steps to deploy the UCT-C Controller:

1. Use the following command to unzip and untar the .tgz file:

   ```
   gunzip <name of the UCT-C Controller .tgz file>
   tar -xvf <name of the UCT-C Controller .tar file>
   ```

   After extracting the tar file, navigate to the YAML folder in the newly created **uctc-cntlr-<image version>-<build number>** folder and update the details given in the following steps.
2. Provide the created ***secret*** in the following section of the YAML file:

   ```
   imagePullSecrets:
   - name: <secret>
   ```
3. Provide the ***FM IP address*** in the following section of the YAML file:

   ```
   command:
   - /uctc-controller
   - <FM IPv4 or FM IPv6 or FM IPv4,FM IPv6(both with comma-separation notation)>
   - '443'
   ```

```
- '8443'
- '0'
- "/etc/gcbcerts"
- "gcb-cert.pem"
- "gcb-pvt-key.pem"
- "gcb-ca-root-cert.pem"
Refer to the following examples:
# example1: 192.168.0.10 (IPv4)
# example2: 2001:db8:abcd:ef01::5 (IPv6)
# example3: 192.168.0.10,2001:db8:abcd:ef01::5 (IPv4,IPv6)
```

4. If you provide both IPv4 and IPv6 addresses in the above argument, the following configurations will be used:

   ```
   - name: UCTC_CNTLR_FM_IP_CONFIG
   value: IPv4 or IPv6
   - name: UCTC_CNTLR_FALLBACK_CONFIG
   value: True or False
   ```

   IP CONFIG option allows the user to provide the preferred IP version. If the user does not provide any value, the default value, IPv4, will be used.

   When the preferred IP version fails to connect (example: IPv6), **FALLBACK CONFIG** will be used to connect to the other available IP version (example: IPv4). The default value, True, will be used to consider the Fallback mechanism.

   Notes:
   - Fallback configuration will be used during the node registration phase only.
   - Controller FM IP and FALLBACK configurations will be used only if you provide both IPv4 and IPv6.
   - If you provide IPv4 alone, only IPv4 connections will be considered, and UCTC_CNTLR_FM_IP_CONFIG and UCTC_CNTLR_FALLBACK_CONFIG will be disregarded. Similarly, if IPv6 alone is provided, only IPv6 connections will be considered.
   - Default values will be used if you do not provide any options.
   - In dual stack cluster, update the following field in **uctc-cntlr-service** to deploy a dual-stack setup. Spec:ipFamilies:- Ipv4- IPv6ipFamilyPolicy: PreferDualStack.

5. Provide ***External IP*** and ***Kubernetes Cluster API URL*** in the following section of the YAML file:

   ```
   env:
   - name: UCTC_CNTLR_SERVICE_NAME
   value: "GIGAMON_UCTC_CNTLR_SERVICE"
   - name: UCTC_CNTLR_EXT_IP_DNS
   ```

```
value: "<external IPv4 or external IPv6 or external IPv4,IPv6(both with comma-separation notation) or DNS"
Refer to the following examples:
# example1: 192.168.0.10 (IPv4)
# example2: 2001:db8:abcd:ef01::5 (IPv6)
# example3: 192.168.0.10,2001:db8:abcd:ef01::5 (IPv4,IPv6)
# example4: gigamon.example.com
- name: K8S_CLUSTER_ENDPOINT
value: <K8s Cluster API URL>
Refer to the following example:
# example: https://10.10.10.13:6443
- name: FM_FQDN
value: www.fm.gigamon.com
```

6. Update the namespace in the YAML file as required and run the following command:

   ```
   kubectl create -f uctc-controller.yaml
   ```

   Following the execution of the above command, when the UCT-C Controller pod is created successfully, the output (sample) will be as follows:

   ```
   gigamon@controller-2:~$kubectl create -f uctc-controller.yaml
   service/gigamon-uctc-cntlr-service created
   deployment.apps/uctc-cntlr-v1 created
   clusterrole.rbac.authorization.k8s.io/pods-list created
   clusterrolebinding.rbac.authorization.k8s.io/pods-list created
   ```

The following table describes all the field values in the YAML file.

Field Values	Description
Port: 443	The UCT-C Controller REST service port number.
Port: 42042	Port must be 42042. Allows UCT-C to send statistics information to UCT-C Controller.
GigaVUE-FM IP	The IP address of the GigaVUE-FM with which your UCT-C is connected.
UCT-C-Cntlr REST SVC Port	The UCT-C Controller REST service port number. This must be opened on your GigaVUE-FM to allow inbound traffic to Kubernetes. This allows GigaVUE-FM to communicate with the UCT-C Controller. Example: 8443 (configurable)
FM REST Svc Port	The GigaVUE-FM REST service port number. This must be opened on your Kubernetes to allow outbound traffic. This allows the UCT-C Controller to communicate with GigaVUE-FM. Example: 443

Field Values	Description
Ports: containerPort: 443 containerPort: 42042	Two ports must be opened. The first container port must be the same as UCT-C-Cntlr REST SVC Port, and the second container port must be port 42042. This allows UCT-C to send statistical data to the UCT-C Controller.
External LB balancer IP	The external load balancer IP/DNS value allows GigaVUE-FM to communicate with theUCT-C Controller within Kubernetes.
K8S cluster end-point	Kubernetes cluster end point for GigaVUE-FM to access the control plane. Example: https://<kubernetesapiserverurl>:6443

Deployment of UCT-C Tap

Follow these steps to deploy the UCT-C Tap:

1. Use the following command to Unzip and Untar the .tgz file:

   ```
   gunzip <name of the UCT-C Tap .tgz file>
   tar -xvf <name of the UCT-C Tap .tar file>
   ```

   After extracting the tar file, navigate to the YAML folder in the newly created **uctc-tap-<image version>-<build number>** folder and update the details given in the following steps.

2. Feed the created ***secret*** in the following section of the YAML file.

   ```
   imagePullSecrets:
   - name: <secret>
   ```

3. Update the namespace in the following section of the YAML file as required. This should be the same as the namespace in which theUCT-C controller is deployed.

   ```
   - name: UCTC_CNTLR_SVC_DNS
   value: gigamon-uctc-cntlr-service.<namespace>.svc.cluster.local
   ```

4. When UCT-C TAP gets both IPv4 and IPv6 from the above controller service DNS, the following configurations will be used:

   ```
   - name: UCTC_TAP_IP_CONFIG
   value: IPv4 or IPv6
   - name: UCTC_TAP_FALLBACK_CONFIG
   value: True or False
   ```

   IP CONFIG option allows the user to provide the preferred IP version. If the user does not provide any value, the default value IPv4, will be used.

   When the preferred IP version fails to connect (example: IPv6), the **FALLBACK CONFIG** will connect to the other available IP version (example: IPv4). The default value True will be used to consider the Fallback mechanism.

Notes:

- Fallback configuration will be considered during node registration only.
- If IPv4 alone is provided, only IPv4 connections will be considered, and UCTC_TAP_IP_CONFIG and UCTC_TAP_FALLBACK_CONFIG will be disregarded. Similarly, if IPv6 alone is provided, only IPv6 connections will be considered.
- Default values will be used if TAP gets dual IP's from the controller service DNS.

5. Edit the following **volumeMounts** as per your container Runtime.

```
volumeMounts:
  - name: socket
    mountPath: /var/run/containerd/containerd.sock
volumes:
  - name: socket
    hostPath:
      Path: /var/run/containerd/containerd.sock
```

Following are the socket location for commonly used CRIs,

```
docker - /var/run/docker.sock
containerd - /var/run/containerd/containerd.sock
cri-o - /var/run/crio/crio.sock
```

- Run the following command for deploying UCT-C Tap:

```
kubectl create -f uctc-tap.yaml -n <namespace where UCT-C tap has to be deployed>
```

Following the execution of the above command, when the UCT-C Tap pod is created successfully, the output (sample) will be as follows:

```
gigamon@controller-2:~$ kubectl create -f uctc-tap.yaml -n uctc
daemonset.apps/gigamon-uctc created
```

Helm Charts

Helm Charts will be available for every release build in the Gigamon software portal. To get started, download the respective UCT-C release build from the repository and untar the **.tgz** file. After untarring, extract the Helm Charts (**uct-cntlr-<version>.tgz** and **uct-tap-<version>.tgz**) and further update the fields mentioned in Deploy using Helm Chart before deployment.

Notes:

- Contact Technical Support or Contact Sales for information on downloading the respective UCT-C build from the Gigamon software portal.
- Support for two Helm Charts is deprecated from software version 6.7.00.

Using Helm with an External **values.yaml** File:

To deploy the chart in an external file, follow the below steps:

1. Download the original Helm chart from Gigamon software portal, either locally or to a remote Helm chart repository.
2. Untar/zip the chart, then copy the original **values.yaml** file to a new file named '**myvalues.yaml**'.
3. Edit the **myvalues.yaml** file with the necessary changes as mentioned in Deploy using Helm Chart.

Use the below deployment command to install the chart:

```
helm install uctc -f myvalues.yaml gigamon-gigavue-uctc-helm-6.10.tgz -n uct-ns
```

Deploy using Helm Chart

You can deploy UCT-C Controller and TAPs using Helm Chart. Follow the steps listed below to deploy the solution.

1. Use the command below to Unzip and Untar the .tgz file:

   ```
   gunzip <name of the UCT-C .tgz file>
   tar -xvf <name of the UCT-C .tar file>
   ```

   After extracting the tar file, navigate to the Helm folder in the newly created **uctc-<image version>-<build number>** folder and update the details given in the below steps.

2. Update the **imagePullSecrets**, **namespace**, **GigaVUE-FM IP**, **external load balancer IP,** and **Kubernetes API URL** in the following section of the **values.yaml file** present in the UCT-C directory.

   ```
   imagePullSecrets: [{name: secret}]
   namespace: uctc
   ```

   If only an IPv4 address is provided and IPv6 is not configured, GigaVUE-FM will use the IPv4 address exclusively for communication.

   ```
   fm_ip: "<FM IPv4>"
   ```

   If only an IPv6 address is provided and fm_ip is not configured, GigaVUE-FM will use the IPv6 address for communication.

   ```
   fm_ipv6: "<FM IPv6>"
   ```

   If both IPv4 and IPv6 are provided, **UCTC_CNTLR_FM_IP_CONFIG** will be used to choose the preferred IP stack.

```
ext_load_balancer: "<FM IPv4 or FM IPv6 or FM IPv4,FM IPv6(both with comma-separation notation>"
Refer to the below examples:
# example1: 192.168.0.10 (IPv4)
# example2: 2001:db8:abcd:ef01::5 (IPv6)
# example3: 192.168.0.10,2001:db8:abcd:ef01::5 (IPv4,IPv6)
k8s_cluster_url: "<url>"
# example: https://10.10.10.12:6443
```

3. If you provide two IPs, IPv4 and IPv6 in the fm_ip argument, the following configurations will be used:

```
# values: <IPv4 | IPv6>
uctc_tap_ip_config: "IPv4"
# values: <true | false>
uctc_tap_fallback_config: "false"
```

IP CONFIG option allows the user to provide the preferred IP version. If the user does not provide any value, the default value IPv4 will be used.

When the preferred IP version fails to connect (example: IPv6), the **FALLBACK CONFIG** will be used to connect to the other available IP version (example: IPv4). The default value, True, will be used to consider the Fallback mechanism.

Notes:

- Fallback configuration will be used during the node registration phase only.
- Controller FM IP and FALLBACK configurations will be used only if you provide both IPv4 and IPv6.
- Default values will be used if you do not provide any options.

4. Edit the following **volumeMounts** as per your container Runtime:

```
crisocketvolume:
mountPath: /var/run/containerd/containerd.sock
name: socket
```

The socket location for commonly used CRIs are as follows:

```
docker - /var/run/docker.sock
containerd - /var/run/containerd/containerd.sock
cri-o - /var/run/crio/crio.sock
```

5. Run the below command in the location where the UCT-C folder is present.

```
helm install uctc /uctc -n <Namespace>
```

NOTE: Users can skip the steps 1-5 and use the below command to directly deploy UCT-C Controller and TAPs using Helm Chart.
helm install uctc -n uctc ./uctc --set namespace=uctc --set serviceAccount.name=test --set imagePullSecrets[0].name=gigamon --

```
set uctcController.fm_ip=x.x.x.x --set uctcController.ext_load_balancer=x.x.x.x --set
uctcController.k8s_cluster_url=https://x.x.x.x:6443 --set uctcController.uctc_cntlr_fm_ip_
config=IPv4 --set
uctcTap.uctc_tap_ip_config=IPv4 --set uctcTap.cri_socket_path=/run/containerd/containerd.sock
```

Validate UCT-C Deployment

To validate the UCT-C deployment and check for any failures, set the validation value to "True" in values.yaml file. Two pods, **uctc-prevalidator-pod** and **uctc-postvalidator-pod** will be deployed, which will perform checks to ensure certain conditions are met for a successful deployment. If all the checks pass, the validator pods will clean up automatically, and UCT-C will be deployed successfully. However, if any check fails, the respective validator pod (either **uctc-prevalidator-pod** or **uctc-postvalidator-pod**) will remain in an error state, and the Helm installation will fail.

Follow the steps listed below if the installation fails due to a validation error:

1. Check Logs: To review the logs of the failure pod and identify the issue, use the following command and specify the corresponding failed pod name.

   ```
   kubectl logs < uctc-prevalidator-pod | uctc-postvalidator-pod > -n uctc
   ```

2. Delete the Helm Release: Use the following command to delete the failed Helm release.

   ```
   helm uninstall my-release -n uctc
   ```

3. Delete the Error Pod: After you identify the cause of failure, you can delete the failed validator pod (uctc-prevalidator-pod or uctc-postvalidator-pod depending on which pod has failed) and re-run the Helm installation. Use the command below to delete the failed validator pod.

   ```
   kubectl delete pod <uctc-prevalidator-pod | uctc-postvalidator> -n uctc
   ```

4. Run the command below in the location where the UCT-C folder is present.

   ```
   helm install uctc /uctc -n <Namespace>
   ```

   NOTE: If there are intermittent connectivity issues between GigaVUE-FM and the cluster, set 'validation' to false before installing the Helm chart to avoid false negative failures.

The following table lists the configurable parameters and their default values defined in the values.yaml file.

UCT-C Controller configuration

Parameter	Description	Default Value
uctcController.image.repository	UCT-C Controller Docker image repository.	gigamon/gigamon-gigavue-uctc-cntlr
uctcController.image.tag	UCT-C Controller Docker image tag.	XXX_IMAGE_TAG_XXX
uctcController.nameOverride	This value will override the default resource's name generated by the chart's templates.	uctc-cntlr
uctcController.fullnameOverride	The provided name will combine with the default resource's name.	
uctcController.podAnnotations	Annotations can be added based on the user requirements.	
uctcController.service.name	Name of the UCT-C Controller Service to be created.	uctc-cntlr-service
uctcController.service.type	Type of the service to be created.	ClusterIP
uctcController.fm_ip	IPv4 address of the GigaVUE-FM. If only IPv4 is provided and fm_ipv6 is not specified, GigaVUE-FM will default to using IPv4 for communication. If both fm_ipv4 and fm_ipv6 are provided, the preferred IP stack will be selected based on the configuration specified in uctc_cntlr_fm_ip_config.	
uctcController.fm_ipv6	IPv6 address of the GigaVUE-FM. If only IPv6 is provided and fm_ip is not specified, GigaVUE-FM will default to using IPv6 for communication. If both fm_ipv4 and fm_ipv6 are provided, the preferred IP stack will be determined based on the uctc_cntlr_fm_ip_config setting.	
uctcController.uctc_svc_rest_port	Port at which UCT-C Controller listens for GigaVUE-FM request.	8443
uctcController.fm_svc_rest_port	Port at which GigaVUE-FM listens for UCT-C Controller registration message.	443
uctcController.k8s_cluster_url	K8S cluster end-point (typically, master nodes with the default port of 6443).	
uctcController.uctc_cntlr_fm_ip_config	User's preferred IP stack for communication between the UCT-C Controller and a dual-stack GigaVUE-FM.	IPv4
uctcController.uctc_cntlr_fallback_config	If enabled as 'True,' the uctc_cntlr_fm_ip_config will utilize the alternative IP stack if the preferred one is unavailable.	True
uctcController.uctc_cntlr_inventory_batch_sz	This configuration is used to configure UCT-C inventory batch size in terms of	25

Parameter	Description	Default Value
	number of pods.	
uctcController.resources.limits.cpu	Expressed in CPU units, it represents the maximum processing power that the UCT-C controller pod is allowed to use.	100m
uctcController.resources.limits.memory	Expressed in Gibibytes, it represents the maximum amount of RAM that the UCT-C controller pod is allowed to consume.	256Mi
uctcController.resources.requests.cpu	Expressed in CPU units, it represents the minimum processing power that the UCT-C controller pod needs.	100m
uctcController.resources.requests.memory	Expressed in Gibibytes, it represents the minimum amount of RAM that the UCT-C controller pod needs.	256Mi
uctcController.nodeSelector	Node labels can be specified to target specific nodes in the Kubernetes cluster where the UCT-C Controller pod should be scheduled.	
uctcController.tolerations	Toleration's can be specified to tolerate specific taints on nodes.	
uctcController.affinity	Affinity rules can be specified to place UCT-C Controller pod on nodes based on complex rules.	

UCT-C Tap configuration

Parameter	Description	Default Value
uctcTap.image.repository	UCT-C Tap Docker image repository.	gigamon/gigamon-gigavue-uctc-tap
uctcTap.image.tag	UCT-C Tap Docker image tag.	XXX_IMAGE_TAG_XXX
uctcTap.nameOverride	This value will override the default resource's name generated by the chart's templates.	
uctcTap.fullnameOverride	This value will combine with the default resource's name.	
uctcTap.podAnnotations	Annotations to be added to the pod.	
uctcTap.podSecurityContext	Security context to be added to the pod.	
uctcTap.uctc_tap_ip_config	User's preferred IP stack for communication between the UCT-C Controller and UCT-C Tap.	IPv4
uctcTap.uctc_tap_fallback_config	If enabled as 'True,' the uctc_tap_ip_config will utilize the alternative IP stack if the preferred one is unavailable.	True
uctcTap.uctc_cntlr_inventory_batch_	This configuration is used to configure UCT-C	25

Parameter	Description	Default Value
sz	inventory batch size in terms of the number of pods.	
uctcTap.resources.limits.cpu	Expressed in CPU units, it represents the maximum processing power that the UCT-C Tap pod is allowed to use.	1
uctcTap.resources.limits.memory	Expressed in mebibytes, it represents the maximum amount of RAM that the UCT-C Tap pod is allowed to consume.	1Gi
uctcTap.resources.requests.cpu	Expressed in CPU units, it represents the minimum processing power that the UCT-C Tap pod needs.	1
uctcTap.resources.requests.memory	Expressed in mebibytes, it represents the minimum amount of RAM that the UCT-C Tap pod needs.	1Gi
uctcTap.cri_socket_path	Key-in the container run-time specific socket path for, for example for cri-o -> "/run/crio/crio.sock" containerd -> /run/containerd/containerd.sock and docker -> /var/run/docker.sock.	
uctcTap.nodeSelector	Node labels can be specified to target specific nodes in the Kubernetes cluster where the UCT-C Tap pod should be scheduled.	
uctcTap.tolerations	Toleration's can be specified to tolerate specific taints on nodes.	
uctcTap.affinity	Affinity rules can be specified to place UCT-C Tap pod on nodes based on complex rules.	

Common Configuration

Parameter	Description	Default Value
namespace	Namespace in which the UCT-C components are to be deployed.	uct
serviceAccount.create	Specifies whether a service account should be created.	FALSE
serviceAccount.annotations	Annotations to add to the service account.	
serviceAccount.name	Name of the service account if uctcTap.serviceAccount.create is set to true.	gigamon
imagePullSecrets	List of image pull secrets for private registries.	gigamon (customize as needed)
ingress.enabled	Specify if an Ingress controller is already installed. Setting this to 'true' will create an	TRUE

Parameter	Description	Default Value
	ingress resource.	
ingress.annotations	Annotations to be added to the ingress.	
ingress.annotations.kubernetes.io/ingress.class	Class name of the ingress controller to be used.	nginx
debugmode	Specified in the hex form of 0x00[aaaa][b][c] where aaaa is the number of pcap messages to maintain before rollover, b can have the value 0 or 1 where 0=do not create pcap or 1=create pcap, and c can have the value between 1 to 4 where 1=fatal, 2=error, 3=info, 4=debug.	0x0A000003
securityContextConstraints.create	If the platform is OpenShift, set securityContextConstraints.create to true. This will create a custom Security Context Constraint (SCC) with the required permissions for the UCT-C solution.	FALSE
securityContextConstraints.name	Name of the Security context constraints (SCC) to be created.	gigamon
validation	Setting this to true will deploy a pre-validator pod before and a post-validator pod after the deployment of the UCT-C solution.	TRUE

You should specify each parameter using the --set key=value argument with the helm install command. You can customize the Helm chart by modifying the values in the values.yaml file or by using the --set flag with the helm install command.

Red Hat OpenShift Platform using OpenShift UI

You can deploy the UCT-C Controller and Taps in the Red Hat OpenShift Platform using Helm Charts. Refer to the following sections for detailed information.

- Prerequisites
- Deployment of UCT-C Controller and Taps

Prerequisites

- To deploy, you should have Developer access in Red Hat OpenShift Platform.
- To validate the deployment, you should have Administrator access.

Deployment of UCT-C Controller and Taps

To deploy UCT-C Controller and Taps, follow the below-listed steps:

1. Log in to the Red Hat OpenShift online platform using your Red Hat login credentials.

2. Switch to Developer access in the drop-down on the top of the page, navigate to the **Helm** section, and click **Create > Helm Release**. The Helm Charts screen appears.
3. Browse and select Gigamon from the **All items** search menu.
4. On the Gigamon-UCT-C landing page, click **Create**. The Create **Helm Release** page appears.

   NOTE: The **README** content on the Gigamon-UCT-C landing page provides information on how to deploy the UCT-C Controller and Tap on a Kubernetes cluster using Helm Chart.

5. To create Helm Release, specify the Helm Release name.
6. Select the appropriate release version from the drop-down menu. By default, the latest uploaded version of the release will be displayed.
7. Select between Form view and YAML view for configuration and specify the created secret name in imagePullSecrets field.
8. In **uctcTap** section:
   a. Specify the socket location details in resources - crisocketvolume filed.

   NOTE: The socket location for commonly used CRIs are as follows:
   docker - /var/run/docker.sock
   containerd - /var/run/containerd/containerd.sock
   cri-o - /var/run/crio/crio.sock

   b. Specify the namespace.
   c. Specify the following details in the ingress section:
      i. **enabled** - Click the check box to enable.
      ii. **annotations** - Specify the annotations details (kubernetes.io/ingress.class and nginx.ingress.kubernetes.io/backend-protocol).
   d. Enable the Create option and specify the serviceAccount name.
9. In **uctcController** section:
   a. Specify the port value.
   b. Specify the following details in certs field:
      i. **ext_load_balancer** - The external load balancer IP/DNS value to allow GigaVUE-FM to communicate with the UCT-C Controller within Kubernetes.
      ii. **k8s_cluster_url** - Kubernetes cluster endpoint for GigaVUE-FM to access the control plane.
      Example: https://<kubernetesapiserverurl>:6443

c. Specify the service label name. For example: uctc-cntrl-service.

d. Specify the repository and pullPolicy details in the image field.

e. Update the namespace and fm_ip details.

10. Click **Create** to deploy the UCT-C solution.
11. To validate the deployment, switch to **Administrator** view and navigate to:
    a. **DaemonSets** option to validate the UCT-C-Tap deployment.
    b. **Deployment** option to validate the UCT-C-Controller deployment.

Post Deployment

Refer to the sections listed below for post deployment verification checks after deploying UCT-C Solution in Kubernetes.

- Verify UCT-C deployment using CLI
- Verify UCT-C deployment using GigaVUE-FM

Verify UCT-C deployment using CLI

You can verify the UCT-C deployment using the following CLI commands to ensure that the UCT-C components are up and running as expected within the specified namespace.

1. Check for the Controller Deployment:
   ```
   kubectl get deployment -n <namespace>
   ```
   This command verifies that the UCT-C controller deployment exists and is running.
2. Verify the TAP DaemonSet Status:
   ```
   kubectl get daemonset -n <namespace>
   ```
   This command checks the status of the TAP DaemonSet, ensuring that it is properly deployed.
3. Verify if the Controller and TAP Pods are Up and Running:
   ```
   kubectl get pods -n <namespace>
   ```
   This command checks that the controller and TAP pods are running and available.
4. Confirm the Controller Service is Present and Accessible:
   ```
   kubectl get svc -n <namespace>
   ```
   Run this command to confirm that the controller service is correctly deployed and accessible within the cluster.
5. Check the Ingress Resource (if applicable):
   ```
   kubectl get ingress -n <namespace>
   ```

If using the Ingress resource for routing external traffic, use this command to verify the Ingress configuration for the controller.

6. Check the Route Resource (if applicable):

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

   If you are using a Route resource to direct external traffic to the appropriate service in the cluster, use this command to confirm the route configuration.

7. Verify Helm Chart Deployment (if using Helm):

   ```
   helm list
   ```

   This command lists the Helm releases deployed in your cluster.

Verify UCT-C deployment using GigaVUE-FM

You can verify the following interactions between UCT-C and GigaVUE-FM:

1. Register UCT-C Controller and TAP
2. Check UCT-C Controller and TAP Status
3. Check UCT-C Controller Connectivity
4. Verify the Cluster Inventory

Register UCT-C Controller and TAP

When a UCT-C Controller and TAP come up in the Kubernetes environment, it registers itself with GigaVUE-FM.

Check the network requirements for the registration to be successful. Refer to Network Ports Requirements.

UCT-C supports IPv4 and IPv6 protocols. Refer to Deploy UCT-C Solution in Kubernetes.

When UCT-C Controller and TAP is terminated normally, it sends the deregistration message to GigaVUE-FM. If UCT-C Controller and TAP goes down abnormally and GigaVUE-FM fails to receive a couple of heartbeats, it will get disconnected.

Check UCT-C Controller and TAP Status

GigaVUE-FM marks the heartbeat status of UCT-C Controller and TAP as **Connected** when it gets registered. After successful registration, UCT-C Controller and TAP sends heartbeats to GigaVUE-FM every 30 seconds. GigaVUE-FM scans the last received heartbeat of the registered UCT-C Controller and TAP pods and marks the heartbeat status periodically (1 minute). The following are the various scenarios where the heartbeat status changes:

- If 2 consecutive heartbeats are missed, GigaVUE-FM marks the status as **Pending**.
- If 3 consecutive heartbeats are missed, GigaVUE-FM marks the status as **Disconnected**.
- GigaVUE-FM automatically purges disconnected or terminated UCT-C Controllers and TAPs after a retention period of 7 days.

NOTE: GigaVUE-FM generates an alarm for the disconnected UCT-C when three consecutive heartbeats are missed. Refer to "Alarms" topic in the *GigaVUE Administration Guide* .

Check UCT-C Controller Connectivity

After successful registration of UCT-C Controller, GigaVUE-FM periodically checks connectivity to Controller. The following are the various scenarios where the UCT-C Controller connectivity changes:

- If GigaVUE-FM can connect with the UCT-C Controller, the connectivity status will be marked as **Reachable**.
- If GigaVUE-FM cannot connect with the UCT-C Controller, the connectivity status will be marked as **Unreachable**.

NOTE: You should ensure that the UCT-C Controller connectivity is Reachable before doing any configurations. If the connectivity shown for a UCT-C controller in a cluster is not reachable, the deployment for that cluster will not go through.

Verify the Cluster Inventory

You can verify the UCT-C cluster inventory in GigaVUE-FM by navigating to **Inventory > CONTAINER > Universal Cloud Tap - Container**. In the Clusters, Nodes, and Pods sections you can verify that all clusters are listed, operational, and running. You can also check the cluster's health status, connectivity and resource usage.

Verify UCT-C deployment using CLI

You can verify the UCT-C deployment using the following CLI commands to ensure that the UCT-C components are up and running as expected within the specified namespace.

1. Check for the Controller Deployment:

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

   This command verifies that the UCT-C controller deployment exists and is running.

2. Verify the TAP DaemonSet Status:

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

   Use this command to check the status of the TAP DaemonSet, ensuring that it is properly deployed.

3. Verify if the Controller and TAP Pods are Up and Running:

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

   This command checks that the controller and TAP pods are running and available.

4. Confirm the Controller Service is Present and Accessible:

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

   Run this command to confirm that the controller service is correctly deployed and accessible within the cluster.

5. Check the Ingress Resource (if applicable):

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

   If using the Ingress resource for routing external traffic, use this command to verify the Ingress configuration for the controller.

6. Check the Route Resource (if applicable):

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

   If you are using a Route resource to direct external traffic to the appropriate service in the cluster, use this command to confirm the route configuration.

7. Verify Helm Chart Deployment (if using Helm):

   ```
   helm list
   ```

   This command lists the Helm releases deployed in your cluster.

Verify UCT-C deployment using GigaVUE-FM

You can verify the following interactions between UCT-C and GigaVUE-FM:

- Register UCT-C Controller and TAP
- Check UCT-C Controller and TAP Status
- Check UCT-C Controller Connectivity
- Verify the Cluster Inventory

Register UCT-C Controller and TAP

When a UCT-C Controller and TAP come up in the Kubernetes environment, it registers itself with GigaVUE-FM.

Check the network requirements for the registration to be successful. For more information, refer to Network Firewall Requirements.

UCT-C supports IPv4 and IPv6 protocols. For more information, refer to Deploy UCT-C Controller and TAP in Kubernetes.

When UCT-C Controller and TAP is terminated normally, it sends the deregistration message to GigaVUE-FM. If UCT-C Controller and TAP goes down abnormally and GigaVUE-FM fails to receive a couple of heartbeats, it will get disconnected.

Check UCT-C Controller and TAP Status

GigaVUE-FM marks the heartbeat status of UCT-C Controller and TAP as **Connected** when it gets registered. After successful registration, UCT-C Controller and TAP sends heartbeats to GigaVUE-FM every 30 seconds. GigaVUE-FM scans the last received heartbeat of the registered UCT-C Controller and TAP pods and marks the heartbeat status periodically (1 minute). The following are the various scenarios where the heartbeat status changes:

- If 2 consecutive heartbeats are missed, GigaVUE-FM marks the status as **Pending**.
- If 3 consecutive heartbeats are missed, GigaVUE-FM marks the status as **Disconnected**.
- GigaVUE-FM purges disconnected or terminated UCT-C Controllers and TAPs after 7 days.

NOTE: GigaVUE-FM generates an alarm for the disconnected UCT-C when three consecutive heartbeats are missed. Refer to "Alarms" topic in the *GigaVUE Administration Guide* for detailed information on Alarms.

Check UCT-C Controller Connectivity

After successful registration of UCT-C Controller, GigaVUE-FM periodically checks connectivity to Controller. The following are the various scenarios where the UCT-C Controller connectivity changes:

- If GigaVUE-FM can connect with the UCT-C Controller, the connectivity status will be marked as **Reachable**.
- If GigaVUE-FM cannot connect with the UCT-C Controller, the connectivity status will be marked as **Unreachable**.

NOTE: You should ensure that the UCT-C Controller connectivity is Reachable before doing any configurations. If the connectivity shown for a UCT-C controller in a cluster is not reachable, the deployment for that cluster will not go through.

Verify the Cluster Inventory

You can verify the UCT-C cluster inventory in GigaVUE-FM by navigating to the **Inventory > CONTAINER > Universal Cloud Tap - Container**. In the Clusters, Nodes, and Pods sections you can verify that all clusters are listed, operational, and running. You can also check the cluster's health status, connectivity and resource usage.

Configure UCT-C Solution using GigaVUE-FM

This section describes how to configure UCT-C through GigaVUE-FM. Refer to the following section for details.

- Universal Cloud Tap - Container Inventory
- Create Monitoring Domain
- Create Source Selectors
- Create Tunnel Specifications
- Configure Traffic Policy
- View Policy Configurations
- View Traffic Policy Statistics

Universal Cloud Tap - Container Inventory

In GigaVUE-FM, on the left navigation pane, go to **Inventory > CONTAINER > Universal Cloud Tap - Container**. You can view the following tabs on the Universal Cloud Tap - Container launch page:

Tabs	Description
Monitoring Domains	Displays the Monitoring Domain details and the connectivity status from GigaVUE-FM to Cluster. The count of reachable and unreachable clusters per Monitoring Domain.
Clusters	Displays Kubernetes Clusters, along with UCT-C Controller information and the total number of nodes per Cluster. Also displays GigaVUE-FM to Controller connectivity and Heartbeats status. Heartbeat status is from UCT-C Controller to GigaVUE-FM. **NOTE:** You can delete a cluster that is not associated to any Monitoring Domain from the Clusters page. Ensure that you delete the cluster only after stopping the UCT-C Controller and the UCT-C TAP.
Nodes	Displays the Nodes from all Kubernetes Clusters along with UCT-C TAP information and Total Pods per Node. UCT-C TAP status should be Connected for deployments for respective Worker Nodes to go through. If the UCT-C TAP status for a Worker Node is not shown as Connected, the deployment for that Worker Node will not go through.
Pods	Displays the list of Pods from all Kubernetes Clusters. For each Pod, all metadata - Pod Name, Labels, IPs, Namespace, Service Name, Service IPs, Node Name, Containers, and Host Network information is displayed.
Settings	Displays the general settings which include disconnected UCT-C Controller or TAP Purge Interval days, and the maximum number of Clusters allowed in GigaVUE-FM.

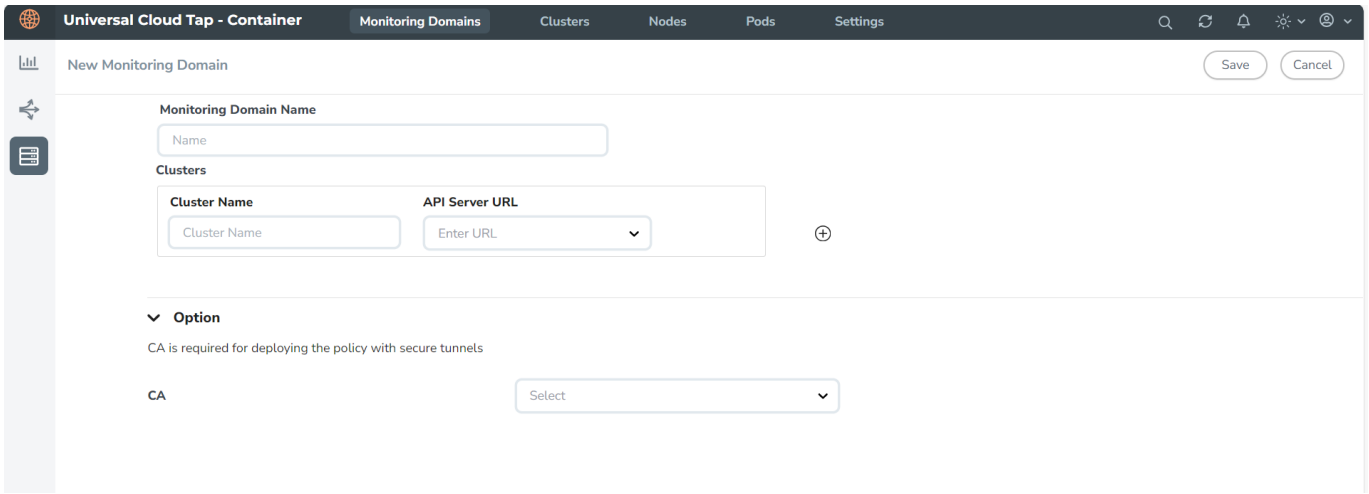
To view and filter the list of Monitoring Domains, Cluster, and Node details, click the filter button on the left side of any of the above-listed tabs. You can also create a new Monitoring Domain, edit, and delete the existing Monitoring Domains.

On Clusters, Nodes, and Pods screens, you can click the **Filter** button on the right side to filter the details on that particular screen.

Create Monitoring Domain

To create a monitoring domain in GigaVUE-FM:

1. Go to **Inventory > CONTAINER > Universal Cloud Tap - Container > Monitoring Domains**.
2. In the **Monitoring Domains** page, click **New**. The **New Monitoring Domain** wizard appears.



3. Enter the name of the Monitoring Domain and the Cluster Name.
4. Enter or select the URL of the API server.
5. Select the required CA name from the drop-down menu.

   NOTE: CA is required to deploy the policy with Secure Tunnels.

6. Click ⊕ to add another cluster and click ⊖ to remove an existing cluster.
7. Click **Save**.

You can view the Monitoring Domain created in the list view. The list view shows the following information for UCT-C and controllers:

- Monitoring Domain - Shows the list of Monitoring Domains created.

- Cluster - Displays the status of GigaVUE-FM to UCT-C Controller connectivity. You can click the number link next to (connected) or the (disconnected) icons to view the cluster details for the selected Monitoring Domain.

Use the following buttons to manage your Monitoring Domain:

  - **New**: Use to create a new Monitoring Domain.
  - **Actions**: Use to edit or delete the Monitoring Domain(s).
  - **Refresh Inventory**: Triggers Inventory Refresh on all Clusters in the Monitoring Domain.

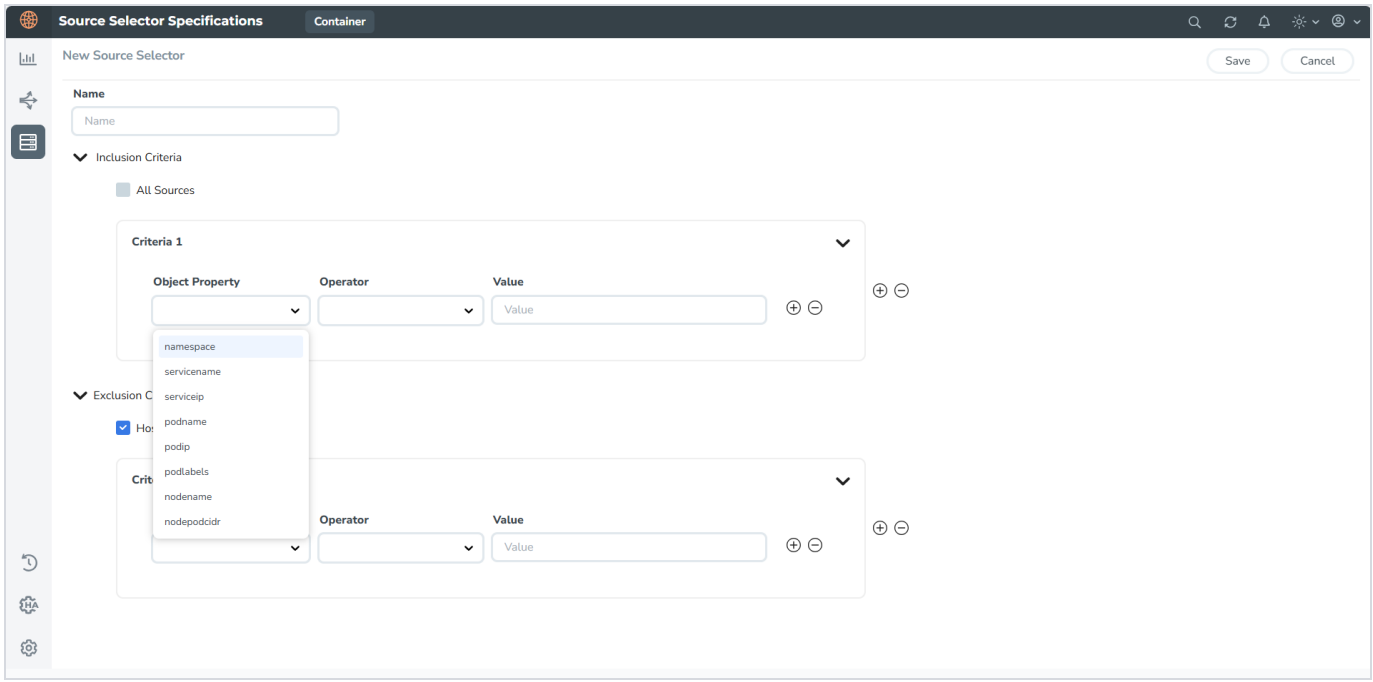
Create Source Selectors

When setting up a traffic flow, it is important to define the selection criteria for the sources of traffic. You should configure the sources of the traffic to be monitored.

DefaultExclusion: It is a default source selector which will be applied to all policies. It cannot be deleted but can be modified. After modifying the DefaultExclusion source selector, policies need to be deployed again for the changes to take effect. DefaultExclusion appears by default on the **Source Selector Specifications** page.

- To exclude the pods from monitoring, you can add criteria to DefaultExclusion. From the drop-down list, select any of the following Object Property to exclude them from the monitoring, and provide the value for the property selected in the value field:
  - servicename
  - serviceip
  - podname
  - podip
  - podlabels
  - nodename
  - namespace
  - nodepodcidr

  By default, pods in kube-system namespace, and metallb-system namespace are excluded from monitoring.
- You can add criteria to DefaultExclusion to exclude nodenames where UCT-C TAP is not launched. If Master node(s) does not have UCT-C TAP, add master node names to DefaultExclusion.
- During the upgrade to Version 6.10, GigaVUE-FM will automatically remove the Host Network criteria from the DefaultExclusion Source Selector and will add the Host Network Enabled **Exclusion Criteria** by default to all the existing policies.



To configure the Source Selectors:

1. Go to **Inventory** > **Resources> Source Selectors> Container**.
2. On the **Source Selector Specifications** page, navigate to the **Container** tab.
3. On the **Source Selector Specifications** page for Containers, click **Create**. The **New Source Selector** wizard appears.
4. Enter the name of the source.
5. In the **Inclusion Criteria**, select any from the following options:
   a. All Sources - Select this option to acquire traffic from all namespaces and pods within the selected cluster(s). The traffic volume may be large, depending on the cluster(s) size.
   b. Criteria1 - You must enter the following options:
      i. Select an object property to filter the traffic source.
      ii. Select the operator and enter the values for the filter (values are case-sensitive).
      iii. On the Criteria, click ⊕ to add another Object and click ⊖ to remove an existing Object.
6. In the **Exclusion Criteria**, select or enter the following options:
   a. Select an object property to filter the traffic source.
   b. Select the operator and enter the values for the filter (values are case-sensitive).

c. Select the Host Network Enabled to view the configuration in your policies. The UCT Container TAP introduces support for tapping Host Network Enabled pods. Refer to the Exclusion Criteria for more information.

7. On the Inclusion or Exclusion Criteria sections, click ⊕ to add another Criteria and click ⊖ to remove an existing Criteria.
8. Click **Save**.

Notes: You can create multiple criteria. Within each criteria, you can configure multiple objects.

- If you have configured multiple objects in a criteria, the traffic will be filtered only if all the object rules are true (AND condition).
- If you have configured multiple criteria, then the traffic will be filtered even if one of the criteria is true (OR condition).

Exclusion Criteria

Host Network Enabled - The UCT-C introduces support for tapping Host Network Enabled pods. By default, this check box is selected (i.e.) you are excluding the host network enabled pods.

When you want to monitor the pod, clear the **Host Network Enabled** checkbox. A warning message appears and requires your confirmation to proceed with tapping pods with Host Network Enabled.

Notes:

- Worker Node must have cgroup version 2 to support the Host Network Enabled feature.
- If the Worker Node has cgroup version 1, the policy deployment status for pod will show an error message.
- When tapping Host Network enabled pods, tapped traffic is sent to user space for tunneling. It uses performance buffers, requiring more memory. To accommodate this, increase the memory request/limit to atleast 1GB for UCT-C taps.

Identify the cgroup Version on the Worker Node

To check which cgroup version your distribution uses, there are two ways:

1. Run the **stat -fc %T /sys/fs/cgroup/** command on the worker node:
   - For cgroup v2, the output is **cgroup2fs**.
   - For cgroup v1, the output is **tmpfs**.

2. Check if **/sys/fs/cgroup/cgroup.controllers** is present, then it is cgroup v2.

Identify the cgroup Version for Worker Pod

To check which cgroup version your worker pod uses:

1. Login to the worker pod and check file **/proc/$$/cgroup**.
2. If the file has **net_cls**, then it's cgroup v1 otherwise it is cgroup v2.

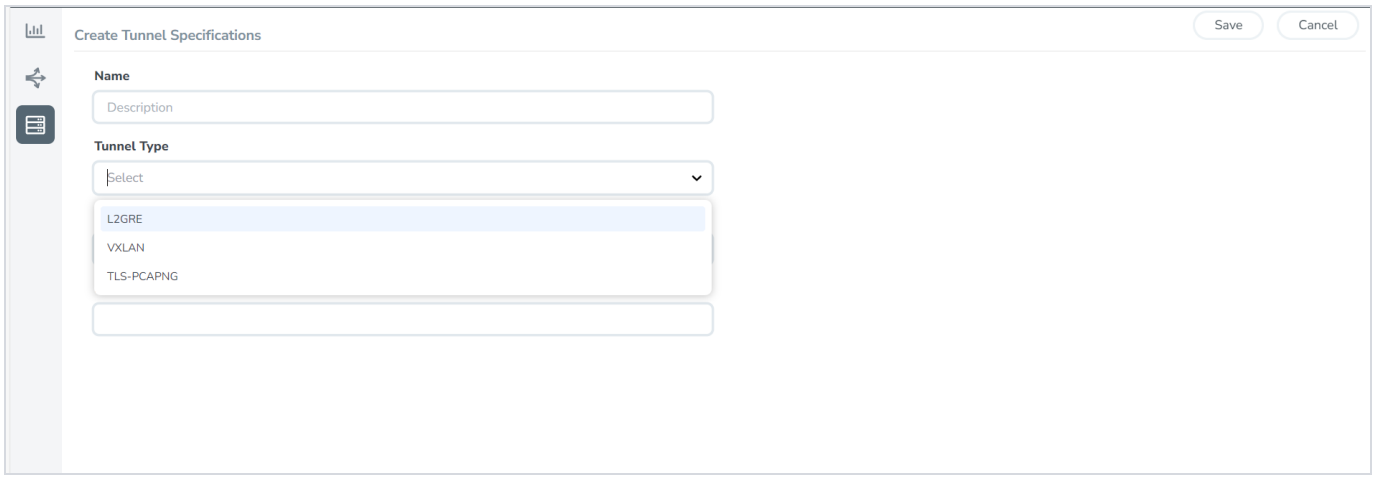
Create Tunnel Specifications

A tunnel of type L2GRE, VXLAN, or TLS-PCAPNG can be created. The tunnel is an egress tunnel. For more information on creating a tunnel of type TLS-PCAPNG, refer to Secure Tunnels.

NOTE: The L2GRE tunnel is not supported on the Azure platform.

To configure the tunnels:

1. Go to **Inventory** > **Resources > Tunnel Specifications**.
2. On the **Tunnel Specifications** page, navigate to the **Container** tab and click **Create**. The **Create Tunnel Specification** wizard appears.



3. Enter the name of the tunnel endpoint.

   NOTE: Do not enter spaces in the alias name.

4. Select **L2GRE**, **VXLAN**, or **TLS-PCAPNG** tunnel type to create a tunnel.
5. Enter the IP address of the destination endpoint.
6. Enter a value for the tunnel key (applicable when the selected tunnel type is L2GRE).
7. Enter the identifier key for the VXLAN network (applicable when the selected tunnel type is VXLAN).

8. Specify the destination port value. Enter a value between 1 and 65535 (applicable when the selected tunnel type is VXLAN or TLS-PCAPNG).
9. Click **Save**.

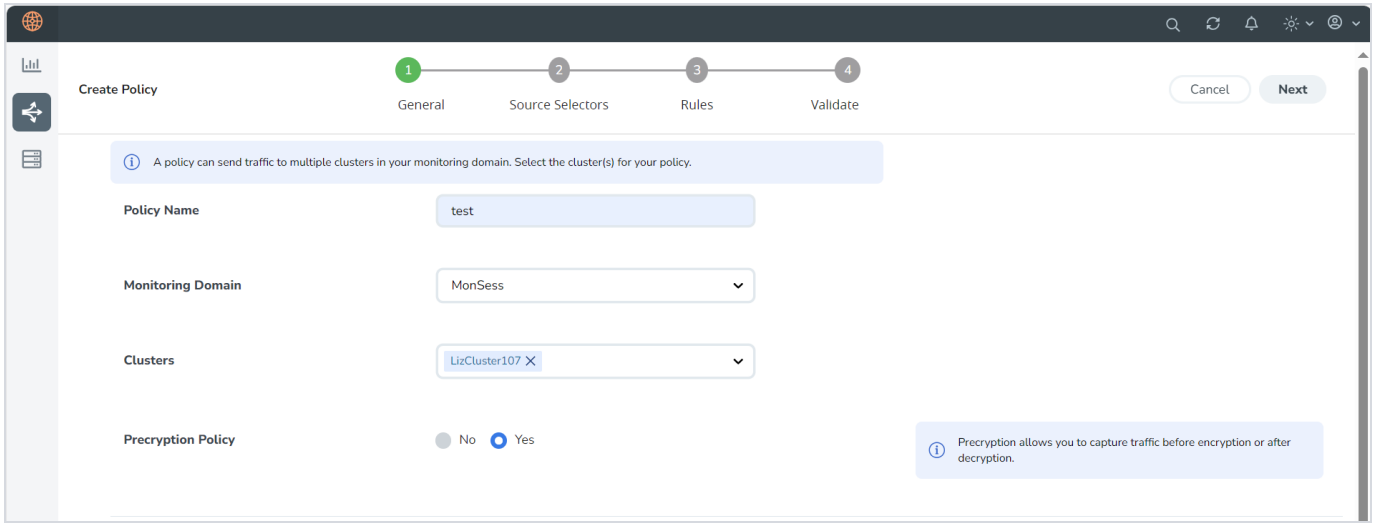
Configure Traffic Policy

The traffic from the workload pods is processed based on the Traffic Policy configuration. The UCT-C TAP routes the traffic to the tunnel destination IP addresses specified in the Traffic Policy rules.

You can refer to the GigaVUE API Reference for detailed information on the REST APIs of UCT-C.

To create a UCT-C Traffic Policy in GigaVUE-FM, follow these steps:

1. Go to **Traffic > CONTAINER > Universal Cloud Tap - Container**. The **Policies** page appears.



2. In the **Policies** page, click **New**. The Create Policy wizard appears.

   NOTE: You can deploy a maximum of eight policies per Monitoring Domain.

3. In the **General** tab, enter a unique name for the Traffic Policy.
4. Select an existing Monitoring Domain. To create a new monitoring domain, refer to Create Monitoring Domain.
5. Select the required cluster from the drop-down menu.
6. Click the radio button **Yes**, to enable the Precryption rules for the policy. Click radio button **No** to enable the Mirroring. Refer to Configure Precryption in UCT-C.

> **NOTE:** Once the policy is deployed, you cannot change the Precryption Policy setting.

7. Click **Next** to switch to the **Source Selectors** tab, select an existing source selector and click **Add** or select **Create New** to create a new source selector, refer to Create Source Selectors. You can configure a maximum of eight source selectors per policy.

8. In the **Source Selectors** page, click  to expand the **Default Exclusion** section. DefaultExclusion Source Selector is applied automatically for all policies.

   > **NOTE:** You can edit the values across the Monitoring Domain in **Inventory** > **Resources > Source Selectors** section. On the **Source Selectors Specifications** page, navigate to **Container > Default Exclusion**. In the **Edit Source Selector** wizard, you can edit the values in the **Exclusion Criteria** section.

9. Click **Next** to switch to the **Rules** tab, enter or select the required information for the **Ingress Rules** and the **Egress Rules** as described in the following steps. You must select CA in the Monitoring Domain page to use secure tunnels in rules:

   a. Select an existing tunnel or select **Create New** to create a new tunnel, refer to Create Tunnel Specifications. For Precryption, only one Tunnel Specification field will be displayed at the top for all the rules. For Mirroring, Tunnel Specification can be configured for every individual rule.

   b. On the Ingress or Egress rules, click  to add another rule and click  to remove an existing rule. You must select CA in the Monitoring Domain page to use secure tunnels in rules.

   c. Enter a unique name for the rule.

   > **NOTE:** Rule names ending with __I, __E, __RI, __RE are not recommended as the names are invalid in policy rules. Rule names like passall, ingress-passall, and egress-passall are restricted.

   d. Select **On** to enable the passall rule or select **Off** to disable the passall rule. Refer to Enable Selective Precryption to add the filters when you choose to disable the passall rule.

   e. Select **Pass** to allow the packets or select **Drop** to block the packets based on the filters.

   f. Select the direction from the following options:

      - Bidirectional - Taps the traffic in both directions. Each bidirectional rule will add 2 ingress rules and 2 egress rules

> **NOTE:** When you apply filters to two pods on the same worker node to capture traffic in both directions, only one copy of the packet will be tunneled out for each packet traveling from one pod to the other.

- Ingress- Taps the ingress traffic
- Egress - Taps the egress traffic
- Ingress Pass All - Taps all the ingress traffic
- Egress Pass All - Taps all the egress traffic

> **NOTE:** The maximum number of rules supported per direction is 32.

g. Enter a priority value to specify the order of rule execution on the selected Pod. Unique Priority is enforced in a policy within ingress and egress space. Bidirectional rules get expanded in ingress and egress space. Priority is not applicable for Drop Rules. Drop Rules are executed first, followed by passall rules, and then filter rules based on specified priority values.

10. Click **Next** to switch to the **Validate** tab. Click **Save** to the policy or click **Deploy**, and the selected traffic policy rules get deployed to the required UCT-C TAPs present on the nodes corresponding to the source pods selected for monitoring.

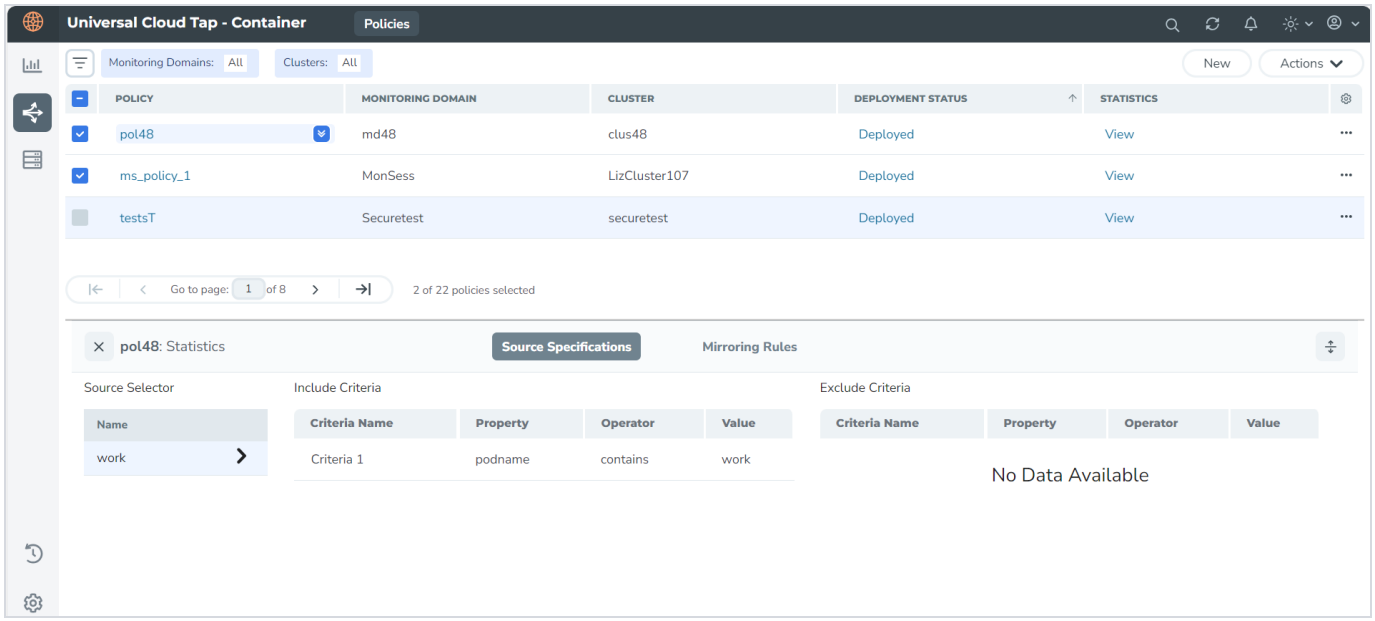
> **NOTE:** If policy deployment for pods fails with a **Duplicate Filter Rules** error, you need to decide which policy the pod should be monitored under and make changes to the failed policies by removing the overlapping sources from the failed policy.

View Policy Configurations

To view the Policy Configurations of the traffic policy configured in the GigaVUE-FM, click the policy name. The configurations appear on the bottom of the **Policies** page.
You can view the following tabs along with the policy name:

- Source Specifications
- Mirroring Rules

You can scroll each of the tables to view more columns. The fields and description for the tab that appears when you click the tabs are described in the topics respectively.



Source Specifications

You can view the criteria based on which a pod is selected for tapping.

The fields and descriptions of the source specifications tab are described in the following table:

Table 1: Source Specifications

Tab- Source Specifications	Field	Description
Source Selector		
	Name	Specifies the name of the Source selector.
Include Criteria		
	Criteria Name	Specifies the include criteria for the source selector. Pod that matches the include criteria is part of the source for the given traffic policy.
	Property	Specifies the attributes of the pod. The available attributes are: • namespace • servicename • serviceip • podname • podip • podlables • nodename • nodepodcidr

Tab- Source Specifications	Field	Description
	Operator	Specifies the operator used in the criteria.
	Value	Specifies the value for the attributes in the criteria.
Exclude Criteria		
	Criteria Name	Specifies the exclude criteria for the source selector. Pod that matches the exclude criteria will be excluded from the source for the given traffic policy.
	Property	Specifies the property in the exclude criteria based on which the pod associated with the source is excluded.
	Operator	Specifies the operator involved in the exclude criteria in tapping the traffic in the pod.
	Value	Specifies the value in the criteria based on which traffic in the pod is excluded.

Mirroring Rules

You can view the aggregate value of all the rules configured for the policy on the node in the UCT-C TAP present in a cluster. The fields and their descriptions in the Mirroring Rules tab are detailed in the following table:

Table 2: Mirroring Rules

Tab-Rules Rules	Field	Description
Mirroring Rules		
	Rule	Specifies the name of the rules in which the traffic is filtered in the pod. Click on the Rule name to view the filters.
	Tunnel	Specifies the tunnel details which is associated with the rules to send the traffic out. When you hover over the tunnel specification value, you can view the details of the tunnel in a message box.
	Priority	Specifies the priority assigned for the rule.
	Action	Specifies whether to pass or drop the rule.
	Direction	Specifies whether the traffic flow direction is ingress, egress, or bidirectional (both directions).
Filter		
	Type	Specifies the filter type.
	Filter	Specifies the name for the filter.
	Value	Specifies the value of the filter.

View Traffic Policy Statistics

Traffic Policy Statistics in the GigaVUE-FM provides the visibility of the policies within a Monitoring Domain and displays the information of the policies and its rules statistics in the dashboard.

Rules are configured in the UCT-C to either forward the traffic to a Tunnel or drop the flow of the traffic.

The activities of the rules are reflected by the statistics counters. The statistics counters show how the policy statistics are directly co-related to the policy and its rules being configured through the GigaVUE-FM.

To view the statistics of the traffic policy configured in the GigaVUE-FM, click **View** status link in the **Statistics** column of a policy. The Policy Statistics and the Mirroring Statistics appear on the bottom of the **Policies** page:

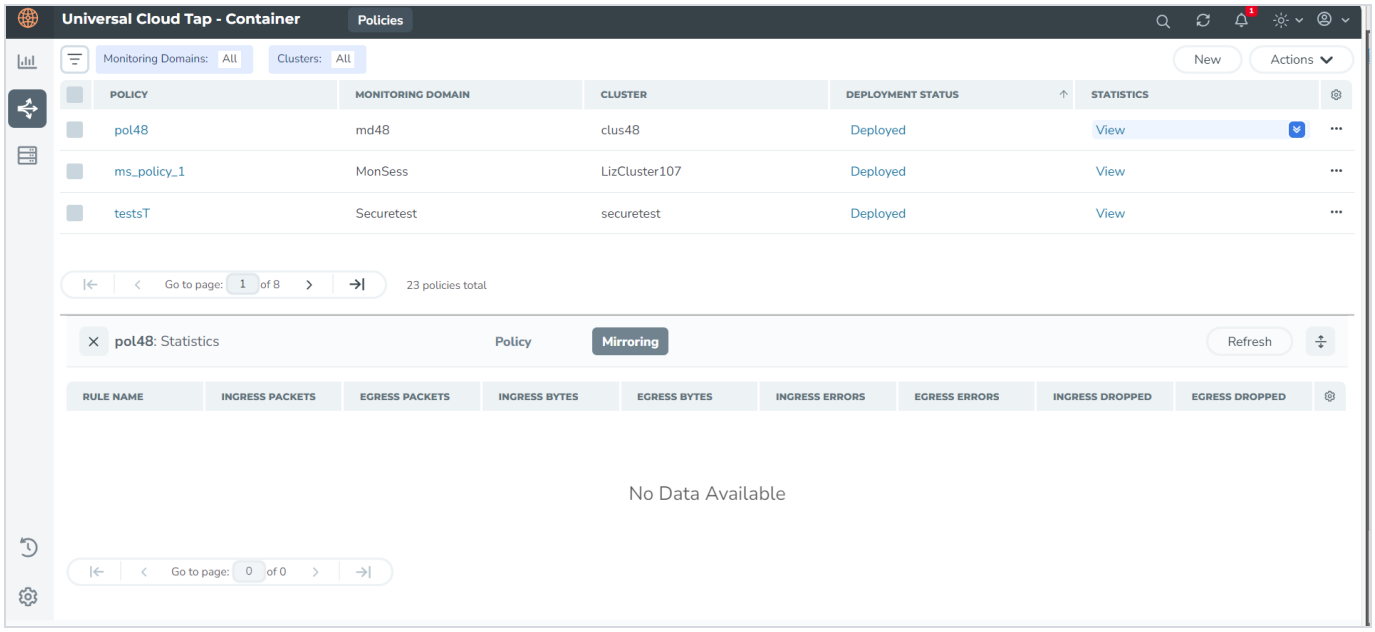


Table 3: Policy Statistics

Fields	Description
Policy Name	Name of the policy.
Ingress packets	Total aggregate value of the ingress packets associated with the policy.
Egress packets	Total aggregate value of the egress packets associated with the policy.
Ingress Bytes	Total aggregate value of the ingress bytes associated with the policy.
Egress Bytes	Total aggregate value of the egress bytes associated with the policy.
Ingress Errors	Total aggregate value of the ingress errors associated with the policy.

Fields	Description
Egress Errors	Total aggregate value of the egress errors associated with the policy.
Ingress Dropped	The total aggregate value of the ingress packets dropped associated with the policy.
Egress Dropped	Total aggregate value of the egress packets dropped associated with the policy.

Table 4: Mirroring Statistics

Fields	Description
Rule Name	Name of the individual rule.
Ingress packets	Total aggregate value of the ingress packets associated with the rule.
Egress packets	Total aggregate value of the egress packets associated with the rule.
Ingress Bytes	Total aggregate value of the ingress bytes associated with the rule.
Egress Bytes	Total aggregate value of the egress bytes associated with the rule.
Ingress Errors	Total aggregate value of the ingress errors associated with the rule.
Egress Errors	Total aggregate value of the egress errors associated with the rule.
Ingress Dropped	Total aggregate value of the ingress packets dropped associated with the rule.
Egress Dropped	Total aggregate value of the egress packets dropped associated with the rule.

Configure UCT-C Features

Refer to the following sections for more detailed information:

- Configure Precryption in UCT-C
- Configure Secure Tunnels in UCT-C

Configure Precryption in UCT-C

GigaVUE-FM allows you to enable or disable the Precryption feature.

Rules and Notes

The following are the memory limits to be applied to UCT-C:

- The memory limit changes depending on the number of vCPUs in the worker node. For example, if the worker node has 16 vCPUs, the Precryption feature consumes around 1GB of memory (16 * 64 MB).
- When you deploy secure tunnels, it requires additional (16 *64 MB) memory. Hence, the total memory that you must allocate for the TAP is 1 GB.
- You can always configure the memory allocation using PRECRYPTION_RING_BUFFER_MEMORY_MB in YAML file.
- Protocol version IPv4 and IPv6 are supported.
- If you wish to use IPv6 tunnels, your GigaVUE-FM and the fabric components version must be 6.6.00 or above.

The YAML configuration option allows you to choose the amount of buffer size.

To configure the Precryption feature in UCT-C, follow the steps listed in Configure Traffic Policy.

After enabling the Precryption, configure the Create Source Selectors, and the **Rules**.

Selective Precryption

GigaVUE-FM allows you to filter packets during the Precryption in the Data Acquisition at the UCT-C level. This filtering is done based on L3/L4 5 tuple information (5-tuple filtering) running on the containers.

Refer to Enable Selective Precryption for information on how to configure Selective Precryption when configuring the **Rules**.

Enable Selective Precryption

If you wish to use selective Precryption, follow the steps given below:

1. Disable the **Enable** toggle button to turn off the default passall rule.
2. Click ⊕ to add another rule.
3. Enter the name of the rule and choose Pass (passes the traffic) or Drop (drops the traffic) in Action menu.

   NOTE: In the absence of a Precryption rule, traffic is implicitly allowed. However, once rules are defined, they include an implicit pass all rule. If the traffic is not conformed to any of the specified rules, it will be passed.
4. Select the direction from the below options:
   - Bi-directional -Allows the traffic in both directions of the flow. A single Bi-direction rule should consist of 1 Ingress and 1 Egress rule.
   - Ingress- Filters the traffic that flows in.
   - Egress - Filters the traffic that flows out.
5. Select the value of the priority based on which the rules must be prioritized for filtering. Select the value as 1 to pass or drop a rule in top priority. Similarly, you can select the value as 2, 3, 4 to 8, where 8 can be used for setting a rule with the least priority. Drop rules are added based on the priority and, then pass rules are added.
6. Select the required **Filter Type** (L3 or L4).
   a. For L3:

i. Select the required **Filter Name**. The available options are IPv4 Source, IPv4 Destination, IPv6 Source, IPv6 Destination, and Protocol (common for both IPv4 and IPv6).

ii. Enter or select the Filter Value based on the selected Filter Name.

> **NOTE:** When using **Protocol** as the **Filter Name**, select **TCP** from the drop-down menu.

b. For L4:

i. Select the required **Filter Name**. The available options are Source Port and Destination Port.

ii. Select the **Filter Relation**. The available options are Not Equal to and Equal to.

iii. Enter the source or destination port value.

Configure Secure Tunnels in UCT-C

Secure tunnel can be configured on:

- Precryption Traffic
- Mirrored Traffic

Precryption Traffic

You can send the Precryption traffic through secure tunnel. When secure tunnel for Precryption is enabled, packets are framed and sent to the TLS socket. PCAPng format is used to send the packet.

When you enable the secure tunnel option for both regular and Precryption packets, two TLS secure tunnel sessions will be created.

It is recommended to always enable secure tunnels for Precryption traffic to securely transfer the sensitive information.

Mirrored Traffic

You can enable the Secure Tunnel for mirrored traffic. By default, Secure Tunnel is disabled.

Prerequisites

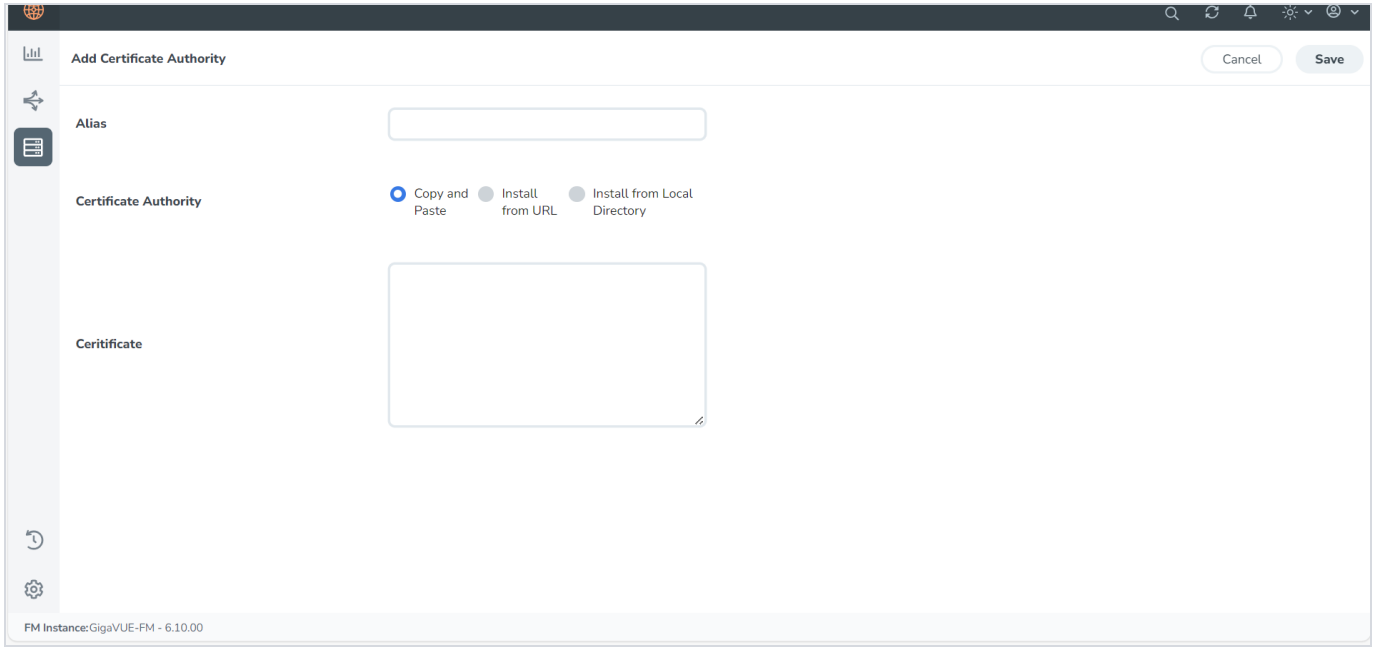
While creating Secure Tunnel, you must provide the following details:

- SSH key pair
- CA certificate

Configure Secure Tunnels from UCT-C Container to GigaVUE V Series Node

To configure a secure tunnel in a UCT-C Container, you must configure one end of the tunnel to the UCT-C and the other end to a GigaVUE Cloud Suite V Series node. You must configure CA certificates in UCT Container, and the private keys and SSL certificates in the GigaVUE Cloud Suite V Series Node. Refer to the following steps for configuration:

1. Upload a Custom Certificate:
   a. You must upload a CA to UCT-C Container for establishing a connection with the GigaVUE Cloud Suite V Series node.
   b. Go to **Inventory > Resources > Security > CA List**.
   c. Click **Add**. The **Add Certificate Authority** page appears.
   d. Enter the Alias and choose the certificate from the desired location.
   e. Click **Save**. For more information, refer to Adding Certificate Authority.



2. Upload an SSL Key - To add an SSL Key to GigaVUE Cloud Suite V Series node, follow the steps in the section *SSL Decrypt* in GigaVUE V Series Applications Guide.
3. Select the SSL Key when you create a Monitoring Domain and configure the fabric components in GigaVUE-FM.
4. Select the CA certificate when you create a monitoring domain and configuring the fabric components in GigaVUE-FM. To select the CA certificate, follow the steps in the section Create Monitoring Domain.
5. Create and add the secure tunnel when you configure the traffic policy. Refer to Configure Traffic Policy.

Adding Certificate Authority

The Certificate Authority (CA) List page allows you to add the root CA for the devices.

To upload the CA using GigaVUE-FM follow the steps given below:

1. Go to **Inventory > Resources > Security > CA List**.
2. Click **Add**, to add a new Custom Authority. The **Add Certificate Authority** page appears.
3. In the **Alias** field, enter the alias name of the Certificate Authority.
4. Use one of the following options to enter the Certificate Authority:
   - **Copy and Paste**: In the **Certificate** field, enter the certificate.
   - **Install from URL**: In the **Path** field, enter the URL in the format: <protocol>://<username>@<hostname/IP address>/<file path>/<file name>. In the **Password** field, enter the password.
   - **Install from Local Directory**: Click **Choose File** to browse and select a certificate from the local directory.
5. Click **Save**.

Configure UCT-C Settings

You can configure the following UCT-C settings in GigaVUE-FM:

- UCT-C General Settings
- UCT-C Log Level Settings

UCT-C General Settings

In GigaVUE-FM, you can control the number of permitted clusters and purge time intervals of the UCT-C solution. You can specify the purge interval to automatically remove the UCT-Cs that are disconnected for a long duration.

To edit the UCT-C general settings:

1. In GigaVUE-FM, navigate to **Inventory > CONTAINER > Universal Cloud Tap - Container > Settings**, and the **Settings** page will appear with the existing General settings.
2. On the **General** section, click **Edit**.
3. Edit the following required values in the **General Settings** section.
   a. **Maximum number of clusters allowed** - Enter the maximum number of clusters allowed in the UCT-C solution. Enter a value between 1 and 64. The default value is 50.

b. **Disconnected UCT-C Purge Interval (days)** - Enter a value for the purge time interval for the disconnected UCT-Cs in days. Enter a value between 7 and 180. The default value is 30.

4. Click **Save** to save the updates made on the General Settings

UCT-C Log Level Settings

In GigaVUE-FM, you can control the level of logs created at each individual UCT-C TAP for troubleshooting. The default UCT-C log file name format is **uctc_tap.log**.

To view or edit the UCT-C log level settings:

1. In GigaVUE-FM, navigate to **Inventory > CONTAINER > Universal Cloud Tap - Container > Nodes**, the **Nodes** page appears with the existing information about the node, cluster, pod, and UCT-C details.
2. On the **UCT-C TAP** section, on any Monitoring Domain, click on the UCT-C TAP link, which is in connected status. The Universal Cloud Tap - Container Settings quick view appears.
3. Edit the following required UCT-C log values in the **LOGGING** section.
   a. Select the **Log Level** from the following options:
      - DEBUG—fine-grained log information for application debugging.
      - INFO—coarse-grained log information for highlighting application progress.
      - WARN—log information of potentially harmful situations.
      - ERROR—log information of the error events that allows the application to run continuously.
      - FATAL—log information of severe error events that presumably lead the application to abort.

   b. Enter a value for the number of lines in the UCT-C log file size field.
4. On any of the above fields, click **Reset** to reset the value to default.

Upgrade UCT-C

To upgrade UCT-C, you must perform the following steps:

1. **Upgrade to GigaVUE-FM 6.10**: Before upgrading GigaVUE-FM from earlier versions less than 6.8, delete the UCT-C Monitoring Domain, Policy, UCT-C Controller, and UCT-C TAP. Refer to GigaVUE-FM Installation and Upgrade guide for platform-specific upgrade instructions.

> **NOTE:** GigaVUE-FM 6.10 is not compatible with older versions of UCT-C (less than 6.8). During the upgrade, GigaVUE-FM will delete all UCT-C Inventory and configurations.

2. **Upgrade to GigaVUE-FM (6.8+ to 6.10)**: When upgrading GigaVUE-FM from version 6.8 and above to 6.10, all the UCT-C configurations will be retained in GigaVUE-FM allowing for a direct upgrade.
3. **Upgrade to UCT-C 6.10**: Before upgrading UCT-C from earlier versions less than 6.8, you must delete the older versions and deploy UCT-C 6.10 version. To deploy UCT-C, refer to Steps to Delete and Redeploy the UCT-C Solution. Post UCT-C upgrade, verify that the controller version is 6.10.00.
4. **Upgrade to UCT-C (6.8+ to 6.10)**: When upgrading UCT-C from version 6.8 and above to 6.10, you should upgrade the UCT-C Controller first and then upgrade UCT-C TAP.

> **NOTE:** Changing the configuration from an IPv4 stack to an IPv6 stack or vice versa in the Helm chart is not supported through upgrades. If you need to switch between the IPv4 and IPv6 stacks, you must uninstall and then reinstall the Helm chart.

Post Upgrade Checklist

After upgrading, ensure that you verify the following interactions:

- Controller connectivity should be **Reachable**.
- Controller Heartbeat status should be **Connected**.
- TAP Heartbeat status should be **Connected**.
- Tools or V Series Node should receive traffic from monitored worker pods.

Steps to Delete and Redeploy the UCT-C Solution

Using YAML files

1. To delete the UCT-C Controller or TAP using YAML files, run the below-listed commands:
   a. To delete UCT-C Controller

```
kubectl delete -f uctc-controller.yaml
```

   b. To delete UCT-C TAP

```
kubectl delete -f uctc-tap.yaml -n <namespace>
```

2. To redeploy the UCT-C Controller or TAP using YAML files, download the UCT-C Controller and TAP yaml files, edit the values (imagePullSecrets, FM IP address, External IP, Kubernetes API URL, VolumeMounts) in those yaml files, and run the below-listed commands:
   a. To install UCT-C Controller

```
kubectl create -f uctc-controller.yaml
```

   b. To install UCT-C TAP

```
kubectl create -f uctc-tap.yaml -n <namespace>
```

For more details on deployment of UCT-C Controller and TAPs using YAML files, refer to YAML Files.

Using Helm Charts

1. To uninstall the UCT-C Controller or TAP using Helm Charts, run the below command in the location where the UCT-C directory is present.

```
helm uninstall uctc -n <namespace>
```

2. To redeploy the UCT-C Controller or TAP using Helm Charts, edit the values (imagePullSecrets, namespace, GigaVUE-FM IP, external load balancer IP and Kubernetes API URL) in values.yaml file and run the below command in the location where the UCT-C directory is present.

```
helm install uctc <path_to_new_hemchart> -n <namespace>
```

3. You can alternatively run the following command to upgrade the UCT-C solution directly without deleting and redeploying.

```
helm upgrade uctc <path_to_new_hemchart> -n <namespace>
```

For more details on the deployment of UCT-C Controller and TAPs using Helm Charts, refer to Helm Charts.

Troubleshooting

- For analyzing the issues, log into the UCT-C Controller /UCT-C TAP pod using the following command and verify the logs present in pod-data folder.

  ```
  kubectl -it exec <<pod name>> -n <<namespace>> -- bash
  ```

- **GigaVUE-FM to UCT-C Controller Connectivity Issues** - When UCT-C Controller connectivity is unreachable, verify whether **503 Service Temporarily Unavailable** error messages are observed in GigaVUE-FM's vmm.log (refer to the log messages below). If the error messages are available, check and update the UCT-C Controller service name or port number (as shown in the nginx.yaml) used in the ingress resource.

  nginx.yaml

  ```
  apiVersion: networking.k8s.io/v1
  kind: Ingress
  metadata:
  annotations:
   kubernetes.io/ingress.allow-http: "false"
   kubernetes.io/ingress.class: nginx-uct
   nginx.ingress.kubernetes.io/backend-protocol: HTTPS
   nginx.ingress.kubernetes.io/configuration-snippet: proxy_set_header
  Authorization
   $http_authorization;
   nginx.ingress.kubernetes.io/rewrite-target: /
   nginx.ingress.kubernetes.io/secure-backends: "true"
   nginx.ingress.kubernetes.io/ssl-passthrough: "true"
  name: uct-cntlr-ingress
  namespace: uct
  spec:
   rules:
    - http:
       paths:
         - backend:
             service:
               name: gigamon-uctc-cntlr-service
                port:
                  number: 8443
  path: /
  pathType: ImplementationSpecific

  Log-Snippet
  ```

```
2024-08-16 08:00:35,527 INFO [uctcControllerConnectivity-585] UctcControllerRestClientImpl -
isAlive : connectivitUrl GET: https://<ExternalIP>:<ExternalPort>/api/v1.3/controller
2024-08-16 08:00:35,527 INFO [uctcControllerConnectivity-585] UctcRestClientBase - REQUEST GET
https://<ExternalIP>:<ExternalPort>/api/v1.3/controller null
2024-08-16 08:00:36,566 INFO [uctcControllerConnectivity-585] UctcRestTemplateResponseErrorHandler
- $$$ UCT-C 5XX Rest Error 503 Service Temporarily Unavailable
2024-08-16 08:00:36,566 ERROR [uctcControllerConnectivity-585] UctcRestClientBase - Request GET
https://<ExternalIP>:<ExternalPort>/api/v1.3/controller UctcRestException::
com.gigamon.cloud.uctc.rest.client.UctcRestException: UCTC SERVER
ERROR:: 503 SERVICE_UNAVAILABLE Service Temporarily Unavailable
```

- **UCT-C Controller not discovered by GigaVUE-FM** - When the UCT-C Controller is not discovered by GigaVUE-FM, check whether the Kubernetes cluster URL is updated properly in the yaml file.
- **UCT-C TAP not discovered by GigaVUE-FM**- When UCT-C tap is not discovered by GigaVUE-FM, verify whether the namespace in uctc-tap yaml file (as shown in the following uctc-tap.yaml) is same as that of UCT-C Controller yaml file.

  uct-tap.yaml

  ```
  # Value need to match me        tadata used for gcb-cntlr
  #  value: "<UCT-CNTLR-SVC-NAME.UCT-CNTLR-NAMESPACE>.svc.cluster.local"
  - name: UCTC_CNTLR_SVC_DNS
  value: gigamon-uctc-cntlr-service.<<namespace>>.svc.cluster.local  ===> This
  should be same as that of the namespace in which uctc-controller is deployed.
  ```

- **Policy Rules stuck in deploying status for nodes where UCT-C TAP pod is not present** - If Policy Source Selection Criteria matches Pods on the node where TAP is not launched, Rule status for those Pods will be 'deploying' until a UCT-C TAP pod gets launched on respective nodes. If Master Nodes in Cluster do not have UCT-C TAP, add nodename in the DefaultExclusion Source Selector.
  If you miss adding the node names, the policy rules on pods will be stuck in Undeploying status when you try to undeploy them. It is recommended that you delete the policy.

Additional Sources of Information

This appendix provides additional sources of information. Refer to the following sections for details:

- Documentation
- Documentation Feedback
- Contact Technical Support
- Contact Sales
- The VÜE Community

Documentation

This table lists all the guides provided for GigaVUE Cloud Suite software and hardware. The first row provides an All-Documents Zip file that contains all the guides in the set for the release.

NOTE: In the online documentation, view What's New to access quick links to topics for each of the new features in this Release; view Documentation Downloads to download all PDFs.

Table 1: Documentation Set for Gigamon Products

GigaVUE Cloud Suite 6.10 Hardware and Software Guides
DID YOU KNOW? If you keep all PDFs for a release in common folder, you can easily search across the doc set by opening one of the files in Acrobat and choosing **Edit > Advanced Search** from the menu. This opens an interface that allows you to select a directory and search across all PDFs in a folder.
Hardware how to unpack, assemble, rackmount, connect, and initially configure ports the respective GigaVUE Cloud Suite devices; reference information and specifications for the respective GigaVUE Cloud Suite devices
GigaVUE-HC1 Hardware Installation Guide
GigaVUE-HC3 Hardware Installation Guide
GigaVUE-HC1-Plus Hardware Installation Guide
GigaVUE-HCT Hardware Installation Guide
GigaVUE-TA25 Hardware Installation Guide
GigaVUE-TA25E Hardware Installation Guide
GigaVUE-TA100 Hardware Installation Guide

GigaVUE Cloud Suite 6.10 Hardware and Software Guides
GigaVUE-TA200 Hardware Installation Guide
GigaVUE-TA200E Hardware Installation Guide
GigaVUE-TA400 Hardware Installation Guide
GigaVUE-OS Installation Guide for DELL S4112F-ON
G-TAP A Series 2 Installation Guide
GigaVUE M Series Hardware Installation Guide
GigaVUE-FM Hardware Appliances Guide
Software Installation and Upgrade Guides
GigaVUE-FM Installation, Migration, and Upgrade Guide
GigaVUE-OS Upgrade Guide
GigaVUE V Series Migration Guide
Fabric Management and Administration Guides
GigaVUE Administration Guide covers both GigaVUE-OS and GigaVUE-FM
GigaVUE Fabric Management Guide how to install, deploy, and operate GigaVUE-FM; how to configure GigaSMART operations; covers both GigaVUE-FM and GigaVUE-OS features
GigaVUE Application Intelligence Solutions Guide
Cloud Guides how to configure the GigaVUE Cloud Suite components and set up traffic monitoring sessions for the cloud platforms
GigaVUE V Series Applications Guide
GigaVUE Cloud Suite Deployment Guide - AWS
GigaVUE Cloud Suite Deployment Guide - Azure
GigaVUE Cloud Suite Deployment Guide - OpenStack
GigaVUE Cloud Suite Deployment Guide - Nutanix
GigaVUE Cloud Suite Deployment Guide - VMware (ESXi)
GigaVUE Cloud Suite Deployment Guide - VMware (NSX-T)
GigaVUE Cloud Suite Deployment Guide - Third Party Orchestration
GigaVUE Cloud Suite Deployment Guide Universal Cloud Tap - Container

GigaVUE Cloud Suite 6.10 Hardware and Software Guides
Gigamon Containerized Broker Deployment Guide
GigaVUE Cloud Suite Deployment Guide - AWS Secret Regions
GigaVUE Cloud Suite Deployment Guide - Azure Secret Regions
Reference Guides
GigaVUE-OS CLI Reference Guide library of GigaVUE-OS CLI (Command Line Interface) commands used to configure and operate GigaVUE HC Series and GigaVUE TA Series devices
GigaVUE-OS Security Hardening Guide
GigaVUE Firewall and Security Guide
GigaVUE Licensing Guide
GigaVUE-OS Cabling Quick Reference Guide guidelines for the different types of cables used to connect Gigamon devices
GigaVUE-OS Compatibility and Interoperability Matrix compatibility information and interoperability requirements for Gigamon devices
GigaVUE-FM REST API Reference in GigaVUE-FM User's Guide samples uses of the GigaVUE-FM Application Program Interfaces (APIs)
Factory Reset Guidelines for GigaVUE-FM and GigaVUE-OS Devices Sanitization guidelines for GigaVUE Fabric Management Guide and GigavUE-OS devices.
Release Notes
GigaVUE-OS, GigaVUE-FM, GigaVUE-VM, G-TAP A Series, and GigaVUE Cloud Suite Release Notes new features, resolved issues, and known issues in this release ; important notes regarding installing and upgrading to this release **NOTE:** Release Notes are not included in the online documentation. **NOTE:** Registered Customers can log in to My Gigamon to download the Software and Release Notes from the Software and Docs page on to My Gigamon. Refer to How to Download Software and Release Notes from My Gigamon.
In-Product Help
GigaVUE-FM Online Help how to install, deploy, and operate GigaVUE-FM.

How to Download Software and Release Notes from My Gigamon

Registered Customers can download software and corresponding Release Notes documents from the **Software & Release Notes** page on to My Gigamon. Use the My Gigamon Software & Docs page to download:

- Gigamon Software installation and upgrade images,
- Release Notes for Gigamon Software, or
- Older versions of PDFs (pre-v5.7).

To download release-specific software, release notes, or older PDFs:

1. Log in to My Gigamon.
2. Click on the **Software & Release Notes** link.
3. Use the **Product** and **Release** filters to find documentation for the current release. For example, select Product: "GigaVUE-FM" and Release: "5.6," enter "pdf" in the search box, and then click **GO** to view all PDF documentation for GigaVUE-FM 5.6.xx.

NOTE: My Gigamon is available to registered customers only. Newer documentation PDFs, with the exception of release notes, are all available through the publicly available online documentation.

Documentation Feedback

We are continuously improving our documentation to make it more accessible while maintaining accuracy and ease of use. Your feedback helps us to improve. To provide feedback and report issues in our documentation, send an email to: documentationfeedback@gigamon.com

Please provide the following information in the email to help us identify and resolve the issue. Copy and paste this form into your email, complete it as able, and send. We will respond as soon as possible.

Documentation Feedback Form		
About You	**Your Name**	
	Your Role	
	Your Company	

For Online Topics	**Online doc link**	*(URL for where the issue is)*
	Topic Heading	*(if it's a long topic, please provide the heading of the section where the issue is)*
For PDF Topics	**Document Title**	*(shown on the cover page or in page header)*
	Product Version	*(shown on the cover page)*
	Document Version	*(shown on the cover page)*
	Chapter Heading	*(shown in footer)*
	PDF page #	*(shown in footer)*
How can we improve?	**Describe the issue**	*Describe the error or issue in the documentation.* *(If it helps, attach an image to show the issue.)*
	How can we improve the content? **Be as specific as possible.**	
	Any other comments?	

Contact Technical Support

For information about Technical Support: Go to **Settings**  **> Support > Contact Support** in GigaVUE-FM.

You can also refer to https://www.gigamon.com/support-and-services/contact-support for Technical Support hours and contact information.

Email Technical Support at support@gigamon.com.

Contact Sales

Use the following information to contact Gigamon channel partner or Gigamon sales representatives.

Telephone: +1.408.831.4025

Sales: inside.sales@gigamon.com

Partners: www.gigamon.com/partners.html

Premium Support

Email Gigamon at inside.sales@gigamon.com for information on purchasing 24x7 Premium Support. Premium Support entitles you to round-the-clock phone support with a dedicated Support Engineer every day of the week.

The VÜE Community

The VÜE Community is a technical site where Gigamon users, partners, security and network professionals and Gigamon employees come together to share knowledge and expertise, ask questions, build their network and learn about best practices for Gigamon products.

Visit the VÜE Community site to:

- Find knowledge base articles and documentation
- Ask and answer questions and learn best practices from other members.
- Join special-interest groups to have focused collaboration around a technology, use-case, vertical market or beta release
- Take online learning lessons and tutorials to broaden your knowledge of Gigamon products.
- Open support tickets (Customers only)
- Download the latest product updates and documentation (Customers only)

The VÜE Community is a great way to get answers fast, learn from experts and collaborate directly with other members around your areas of interest.

Register today at community.gigamon.com

Questions? Contact our Community team at community@gigamon.com.

Glossary

D

decrypt list

need to decrypt (formerly blacklist)

decryptlist

need to decrypt - CLI Command (formerly blacklist)

drop list

selective forwarding - drop (formerly blacklist)

F

forward list

selective forwarding - forward (formerly whitelist)

L

leader

leader in clustering node relationship (formerly master)

M

member node

follower in clustering node relationship (formerly slave or non-master)

N

no-decrypt list

no need to decrypt (formerly whitelist)

nodecryptlist

no need to decrypt- CLI Command (formerly whitelist)

P

primary source

root timing; transmits sync info to clocks in its network segment (formerly grandmaster)

R

receiver

follower in a bidirectional clock relationship (formerly slave)

S

source

leader in a bidirectional clock relationship (formerly master)