Gigamon®

GigaVUE-FM Ansible User Guide

Product Version: 6.14

Document Version: 1.0

(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
6.14	1.0	07/10/2026	The original release of this document with 6.14.00 GA.

Contents

GigaVUE-FM Ansible

A GigaVUE-FM Ansible is a reusable, standalone unit of code that performs a specific task, forming a collection of building blocks that Ansible calls to configure, manage, and automate systems.

The GigaVUE-FM Ansible integration allows:

- One Interface for Everything- Automate infrastructure, traffic, and subscriber in one place — no need for custom scripts.
- Secure and Consistent- Shared client with token or basic authentication, TLS validation, and connection reuse.
- Safe and Predictable Automation-Modules apply changes only when needed. In check mode, modules compare the current configuration in GigaVUE-FM with the desired payload and show a comparison list before applying any changes.
- Faster Onboarding- Sample playbooks and inventories give you a head start, making it easy to adapt and reuse.
- Aligned with REST APIs- Parameters and payloads match GigaVUE-FM's REST API models, so behavior is familiar and predictable.

The Ansible package has the following :

- Ansible Collection File- A versioned tarball **`(gigamon-fm_ansible-<version>.tar.gz)`** that installs easily through Ansible Galaxy.
- Modules for key resources such as:
  - Device and fabric modules: manage ports, port groups, GigaStream, IP interfaces, nodes, and maps.
  - Traffic modules, including maps, unified Traffic Policy, GTP whitelist, and GigaSMART modules such as gsop, gsgroup, and gsparams.
  - Subscriber modules: support App Exporter, configure GPFCP profiles, and orchestration playbooks.
- Sample content – Ready-to-use playbooks and inventories for common use cases such as Device Infrastructure configuration, Traffic Intelligence, and Subscriber Intelligence solutions, including a sample orchestrated CPN–UPN mobility playbook.
- Documentation and Support Files- READ ME , CHANGELOG files module references, and dependency lists to help you get up and running fast.

License

The GigavUE-FM Ansible Collection does not require any additional licensing.However, a valid GigaVUE-FM license and the corresponding feature licenses must be procured to enable functionality.

For detailed information on license types, acquisition, and usage, refer to the Gigamon Licensing Guide.

Supportability

This section describes the supported operating systems, Ansible Core and Python versions, and GigaVUE-FM versions for this collection; for this 6.14 release, the collection is qualified with GigaVUE-FM 6.14.

Supported OS for the Ansible Control Node

The GigaVUE-FM Ansible Collection runs on the Ansible control node and inherits operating system support directly from ansible-core.

Use a Linux-based control node that meets the required Ansible Core and Python versions. The GigaVUE-FM Ansible Collection does not introduce additional operating system requirements beyond those defined by Ansible itself.

Supported Python and Ansible Versions

The GigaVUE-FM Ansible collection supports the following Ansible Core and Python versions:

Ansible	Python
2.18	3.11
2.19	3.11
2.20	3.12 and 3.13

Supported Ansible Modules

The supported modules include:

Traffic Intelligence Modules

The Traffic Intelligence modules included in this package are as follows:

Module	Ansible module name(s)	Description
VPORT	gigamon_vport	Configure virtual ports
GSOP	gigamon_gsop	Configure GigaSMART Operations
GSGROUP	gigamon_gsgroup	Configure GigaSMART Groups
Map	gigamon_map	Configure maps
Port	gigamon_port	Configure ports
GigaStream	gigamon_gigastream	Configure GigaStream
PortGroup	gigamon_portgroup	Configure port groups
IP Interface	gigamon_ip_interface	Configure IP interface solution.
Manage Node	gigamon_manage_node	Manage device onboarding in GigaVUE-FM.
GTP Whitelist	gigamon_config_gtp_whitelist, gigamon_config_gtp_whitelist_db, gigamon_bulk_config_gtp_whitelist	Configure GTP whitelists and DB
Legacy Map Groups	gigamon_map_group	Configure legacy map groups
Legacy Map Chaining	gigamon_map_chain	Configure legacy map chains
Fabric Maps	gigamon_map (solutionType FABRIC/CLUSTER)	Configure fabric/cluster maps
Fabric Port Group	gigamon_fabric_port_group	Used to configure fabric port group.
Traffic Policy	gigamon_traffic_policy	Configure unified Traffic Policies
Traffic Policy Map Groups	gigamon_traffic_policy_map_group	Configure Traffic Policy map groups
Traffic Policy Map Chaining	gigamon_traffic_policy_map_chain	Configure Traffic Policy map chaining
IP Interface Solution	gigamon_ipinterface_solution	To configure an IP Interface for a Subscriber Intelligence solution.

Subscriber Intelligence Modules

The Subscriber Intelligence Modules included in this package are as follows:

Module	Ansible module name (s)	Description
App Exporter	gigamon_app_exporter	Configure App Exporters for Subscriber Intelligence solution.
GPFCP Profile	gigamon_gpfcp_profile	Configure GPFCP profiles for Subscriber Intelligence solution.

NOTE: To know more about the modules, refer to the documentation using the **ansible-doc -t <module-name>** command.

Sample Playbooks

Module	Ansible module name(s)	Description
CPN–UPN Communication Orchestration	Custom orchestrated playbooks using multiple modules.	Brownfield CPN–UPN communication for existing Subscriber Intelligence solution

GigaVUE-FM Device Infrastructure Modules

The GigaVUE-FM Device Infrastructure Modules included in this package are as follows:

Module	Ansible module name(s)	Description
AAA	gigamon_aaa	Used to configure the device AAA settings.
TACACS	gigamon_tacacs	Used to configure the TACACS services on device.
LDAP	gigamon_ldap	Used to configure the LDAP on one or more devices.
RADIUS	gigamon_radius	Used to configure the RADIUS on device or cluster.
Backup	gigamon_backup	Used to backup the configuration of the cluster or the device.
Clear Configuration	gigamon_clear_config	Used to clear the configuration of the given device or cluster.
Save Configuration	gigamon_save_config	Used to save the configuration on a device or cluster.
Running Configuration	gigamon_running_ config	Used to generate the running configuration from the device or cluster.
SNMP	gigamon_snmp	Used to configure SNMP traps on one or more devices.
Upload and Apply Configuration	gigamon_upload_and_ apply_config	Used to upload and apply configuration from text file to the given device or cluster.
Card	gigamon_card	Used to manage the device card (single device).
Card All	gigamon_card_all	Used to manage the card of all devices (bulk).
Chassis	gigamon_chassis	Used to manage the device chassis operations.
Banners	gigamon_banners	Used to set the login message for device or cluster.
Device Hostname	gigamon_device_ hostname	Used to set the hostname for the given device.
Email	gigamon_email	Used to configure the email services on devices.
NTP Keys	gigamon_ntp_keys	Used to manage NTP Keys.
NTP	gigamon_ntp	Used to configure NTP services.
DNS	gigamon_dns	Used to set DNS settings for a device or cluster.

Module	Ansible module name(s)	Description
Join Cluster	gigamon_join_cluster	Used to add the device to the cluster.
Leave Cluster	gigamon_leave_cluster	Used to remove device from the cluster.
Spine Leaf	gigamon_spine_leaf	Used to manage the GigaStream operations on spine leaf cluster.
Spine Link	gigamon_spinelink	Used to manage the spine link configuration.
Stack Link	gigamon_stacklink	Used to configure the stacklinks on clusters.
Image Upload	gigamon_image_ upload	Used to manage the images on the GigaVUE-FM server.
Software	gigamon_device_ upgrade	Used to upgrade the device or cluster.
Device Users	gigamon_device_users	Used to configure Device Users.
Device Roles	gigamon_device_roles	Used to configure Device Roles.
Cluster creation	gigamon_cluster	Used for creating a normal cluster.

Rules and Notes

The following list are limitations, dependencies, behaviors, and rules that apply when using these modules.

Dependencies

- Each module operates on a single resource; parent–child links. For example, to use a GigaStream in a map, you must first create the GigaStream and then reference it in the map configuration. This dependency handling is done in the playbook by creating the GigaStream before creating the map.
- For any module, the payload structure for the entity follows the corresponding Swagger/API schema for that GigaVUE-FM resource.
- Module documentation instructs users to refer to Swagger for full schema details.

Rules

- Modules update only when the current state differs from the desired state.
- Check mode allows preview of changes without applying them.

Notes

- GigaVUE-FM Ansible distribution includes sample playbooks for each supported module. Users can write their own playbooks using these modules and the sample content as starting points.

Limitations

- Application Intelligence components are not available as standalone Ansible modules.
- TLS/SSL deployment in Traffic Policy is not supported.

- Global tunneling configuration is not supported.

Pre-requisites

Before you use the GigaVUE-FM Ansible collection, ensure the following requirements are met.

- Ansible Core and Python - Ansible Core and Python must be installed on the Ansible control node. For the exact supported versions, see Supportability
- GigaVUE-FM version -GigaVUE-FM must run a version that is compatible with this collection. For this release, the collection is qualified with GigaVUE-FM 6.14.
- An account associated with a user or token that has the **fm_admin role**, or a custom role with equivalent write privileges, is required. The role must provide access to the resource categories used by the modules you intend to run (for example, ports, maps/traffic policies, infrastructure, and mobility objects). If you use basic authentication, ensure that basic authentication is enabled in GigaVUE-FM.
- The Ansible host must have HTTPS connectivity to GigaVUE-FM.

Install the GigaVUE-FM Ansible collection

This section describes the steps required to obtain the GigaVUE-FM Ansible collection, review its contents, and complete the installation using **ansible-galaxy** and the associated Python dependencies.

Download the tarball

Follow the steps below to download and move the Ansible collection tarball to your control node:

1. Sign in to your Gigamon Community Portal.
2. Download the Ansible collection tarball for your release, for example:

   gigamon-fm_ansible-6.14.0-<build_id>.tar.gz
3. Copy the tarball to the Ansible control node (for example, into /tmp or your home directory).

Install the GigaVUE-FM Ansible Collection

Follow the steps below to install the Ansible collection and its Python dependencies:

1. On the Ansible control node, open a terminal and create a folder for your Ansible work, for example:

   mkdir -p ~/fm-ansible
   cd ~/fm-ansible

2. Copy the tarball into this working directory, for example:

   cp /path/to/gigamon-fm_ansible-6.14.0-<build_id>.tar.gz ~/fm-ansible/

3. Install the collection using ansible-galaxy:

   ansible-galaxy collection install gigamon-fm_ansible-6.14.0-<build_id>.tar.gz

4. By default, the collection installs to:

   ~/.ansible/collections/ansible_collections/gigamon/fm_ansible/

5. To use a custom path instead, run:

   ansible-galaxy collection install gigamon-fm_ansible-<version>.tar.gz -p /path/to/<custom>/collections

6. Verify the installation:

   ansible-galaxy collection list | grep gigamon.fm_ansible

7. Move into the collection root directory:

   cd ~/.ansible/collections/ansible_collections/gigamon/fm_ansible

   The main elements here are:
   - plugins/modules/ – all gigamon.fm_ansible.* modules
   - playbooks/ – sample playbooks and inventories per feature/solution
   - README – quick-start and requirements
   - CHANGELOG – release notes
   - requirements.txt – Python dependencies

8. Install the Python dependencies:

   pip install -r requirements.txt

NOTE: Use virtual environments to keep dependencies isolated, especially when working with multiple python and Ansible versions.

Configure Subscriber Intelligence Solution with GigaVUE-FM Ansible

The Subscriber Intelligence solution allows you to automate communication between existing CPN and UPN nodes. This solution already knows how to configure:

- Create GPFCP profiles with the desired configuration on UPN and CPN nodes.

- Attach the corresponding GPFCP profiles to GSParams on each UPN and CPN node.
- Create App Exporters with the desired configuration on UPN and CPN nodes.
- Create first-level maps for IP interfaces and vPorts on each UPN and CPN node.

The following example shows how to use the CPN–UPN playbooks with an abstract intent file and an inventory.

Example: CPN–UPN Subscriber Solution

The GigaVUE-FM Ansible collection includes a pre-built CPN–UPN subscriber solution, under its **playbooks** directory.

To run the CPN–UPN subscriber solution using the playbooks provided in the collection follow the below steps:

1. Copy the **abstract_input.yml** template from the collection (under **playbooks/upn-cpn/vars**) into your own Ansible project, or edit it in place within the collection directory. This file defines the abstract intent for the CPN and UPN nodes and acts as the source of truth for the orchestration. This template will have the following fields:
   - **alias** – Logical name for the CPN or UPN node used by the orchestration. Use the same alias for later updates or deletions.
   - **comment** – The description of the policy or node (for example, site or purpose).
   - **file_path** – Path (relative to the collection root) of the detailed input file for that node (for example, **playbooks/upn-cpn/vars/UPN/upn1.yml**).
   - **state** – Operation for that node in the CPN–UPN communication:
     - present: Create or update configuration for this CPN/UPN pair (GPFCP profile, App Exporter, first-level maps, and GSParams attachment).
     - absent: Remove configuration for this CPN/UPN pair (delete GPFCP profile, App Exporter, first-level maps, and detach from GSParams).
2. Prepare the input files used by the CPN–UPN playbooks. The input files reside in the collection under **playbooks/upn-cpn/vars**. Per-CPN and per-UPN input files under **vars/CPN** and **vars/UPN**. Each CPN and UPN input file must define at least the following fields:
   - **alias** – logical name identifying the CPN or UPN node (for example, cpn1, upn1).
   - **cluster_id** – GigaVUE-FM cluster ID where this node's configuration is created.
   - **gs_group** – GigaSMART group alias to be used for CPN–UPN communication.
   - **vport** – vPort alias to be used for CPN–UPN communication.
   - **port_numbers.file_path** – path to a YAML file (for example, **common_vars/cpn_upn_port_numbers.yml**) that defines source and destination port ranges for each CPN–UPN pair.

- **ip_interface.alias** – IP interface alias to be used for CPN–UPN communication.
- **ip_addresses.file_path** – path to a YAML file (for example, **CPN/cpn1_ips.yml** or **UPN/upn1_ips.yml**) that lists the IP addresses or ranges for this node.
- **vnf_ips.file_path** – path to a YAML file that defines the VNF IP addresses or ranges associated with this CPN or UPN node.
- **tags** – any required tags (such as site or alias tags) with keys and values used by the mobility solution.

> **NOTE:** You can either edit these vars files manually under **playbooks/upn-cpn/vars**, or use the **playbooks/upn-cpn/utils/generate_upn_cpn_vars** Python utility with a single intent file **(upn_cpn_com_intent.yml)** to generate all required vars and then copy them into **playbooks/upn-cpn/vars** before running the CPN–UPN playbook.

> **NOTE:** When you add more nodes, append new entries at the end of the existing UPN/CPN lists to preserve ordering.

3. Create the inventory for GigaVUE-FM: Create an inventory file (for example, **inventory.yml.template.**) that defines:
   - The GigaVUE-FM address (**fm_address**).
   - The API token or username/password.
   - Paths to your abstract intent file and any generated vars, if required.
4. Run the parent playbook using the following commands:

   ansible-playbook -i upn_cpn_inventory.yml create_upn_cpn.yml --check
   ansible-playbook -i upn_cpn_inventory.yml create_upn_cpn.yml

   The parent playbook reads your abstract intent, prepares the required configuration for each CPN and UPN node, and calls internal playbooks to configure GPFCP profiles, App Exporters, IP interfaces, and first-level maps.
5. Verify in GigaVUE-FM that the following exists:
   - GPFCP profiles on the correct CPN and UPN nodes.
   - App Exporters and first-level maps exist on the expected nodes.
   - If you need changes, update the abstract intent file and run the parent playbook again.

The GigaVUE-FM Ansible modules allow you to rerun the parent playbook after updating the intent file, ensuring the configuration safely adjusts to align with your new intent.

Best Practices

The following are the best practices to keep in mind:

- Use token-based authentication for safety.
- Start runs in check mode to review changes before applying them.
- Keep custom files separate so they aren't lost during updates.
- Reuse the provided sample files as templates.
- Keep tasks small and focused for clarity and easier management.

Troubleshooting

This section lists common issues you may see when running GigaVUE-FM Ansible modules and explains how to identify the cause and fix the problem.

Version mismatch

Symptom: The system reports errors that show the GigaVUE-FM version is not in the supported range.

Action: Check the GigaVUE-FM and collection versions against the support matrix. Upgrade the GigaVUE-FM system or the collection so the versions match the supported range.

Missing Resources

Symptom: You see 404 or 409 errors for resources such as ports, GSOP, or groups.

Action:

- Create the needed resources first using modules like **gigamon_port, gigamon_vport, gigamon_gsop**, and **gigamon_gsgroup**.
- Run the playbook again after the missing items are in place.

Invalid payload

Symptom: The system returns 4xx errors due to an invalid payload structure.

Action:

- Compare the payload with the Swagger schema to make sure the structure is valid.
- Start with examples from ansible-doc and adjust the payload step by step.

Authentication or TLS issues

Symptom: Authentication fails or TLS validation errors appear.

Action:

- Create a new API token (fm_token) in the GigaVUE-FM.
- Check fm_username and fm_password if you are using basic authentication.
- Verify **fm_ca_path** and **fm_verify_certs** settings.

Logging and verification

From Ansible

- Use -v or -vv to see more detailed logs.
- Review the registered results from each task.
- Use **--check** to view diff information without making changes.

From GigaVUE-FM

- Review GigaVUE-FM logs and audit trails to inspect REST request activity.
- Check the GigaVUE-FM to confirm updates to ports, maps, Traffic Policies, or Subscriber Intelligence solutions.

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)