# NetFlow Data Collection

This section contains the following topics:



# NetFlow Data Collection

WAE can collect and aggregate exported NetFlow and related flow measurements. These measurements can be used to construct accurate demand traffic data for WAE Design. Flow collection provides an alternative to the estimation of demand traffic from interfaces, LSPs, and other statistics using Demand Deduction. NetFlow gathers information about the traffic flow and helps to build traffic and demand matrix. Importing flow measurements is particularly useful when there is full or nearly full flow coverage of a network's edge routers. Additionally, it is beneficial when accuracy of individual demands between external autonomous systems (ASes) is of interest.

Network data collected separately by NIMOs, including topology, BGP neighbors, and interface statistics, is combined with the flow measurements to scale flows and provide a complete demand mesh between both external autonomous systems and internal nodes.

WAE gathers the following types of data to build a network model with flows and their traffic measurements aggregated over time:

- Flow traffic using NetFlow, JFlow, CFlowd, IPFIX, and Netstream flows
- Interface traffic and BGP peers over SNMP
- BGP path attributes over peering sessions

# NetFlow Collection Architectures

There are two types of flow collection architectures:

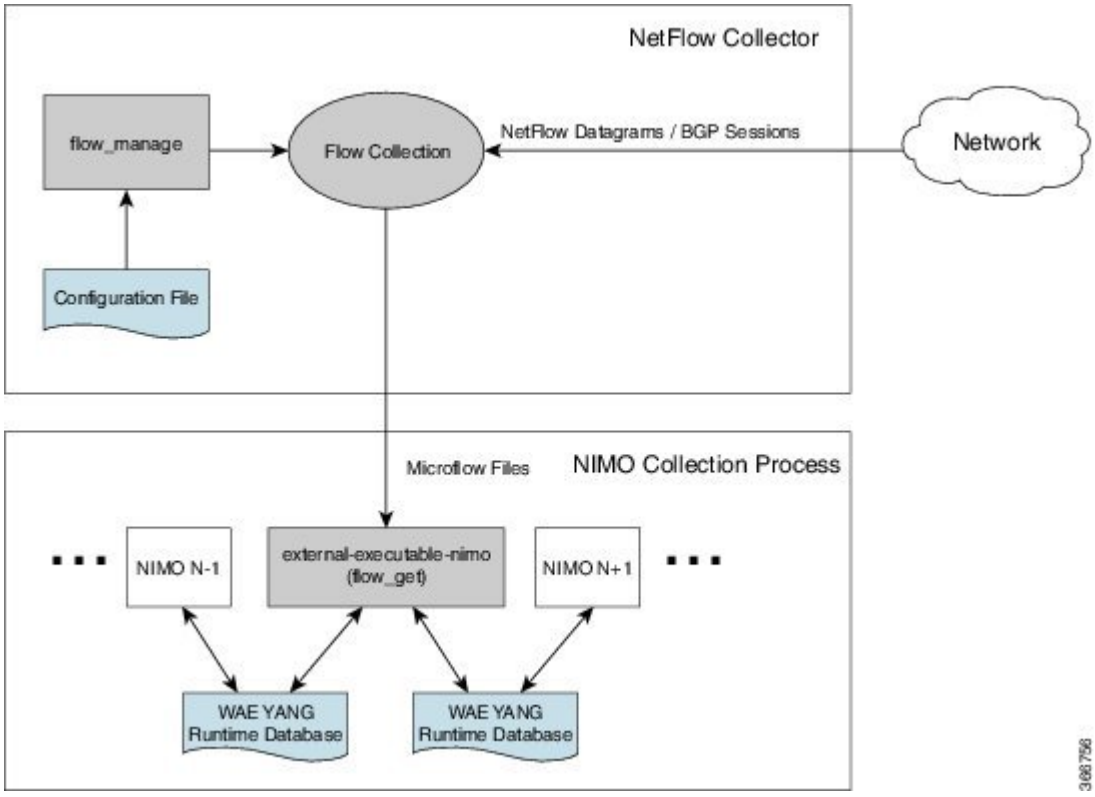
**Note** The collection architecture to deploy depends on the measured or estimated rate of NetFlow traffic export from the network in Mbps or fps.

- Centralized NetFlow (CNF)—Typically used for small to medium networks. This is a single-server architecture.
- Distributed NetFlow (DNF)—Typically used for larger networks. This architecture consists of a JMS broker, master, and agents.

# CNF Collection

The following figure shows the workflow for collecting and computing flow data in CNF. The WAE Collector CLI tools, `flow_manage` and `flow_get`, integrate with an external configuration file and the NIMO collection process, respectively. Flow-based demands and demand traffic are passed to the WAE YANG run-time system.

***Figure 1: Centralized Collection and Demand Creation***




- **`flow_manage`**—This CLI tool configures network connectivity and manages the collection server, including starting, stopping and configuring the flow collection process. It uses input from the <NodeFlowConfigs> table from a configuration file to generate configuration information, which it then sends to the flow collection server.

- **Flow collection server**—This background process receives configuration information from `flow_manage`, which it uses to configure the collection server and receive flow data and BGP attributes. The collection server then aggregates this data and forwards the microflows file to the `flow_get` tool.
- **`flow_get`**—This CLI tool is configured inside the `nimo_flow_get.sh script` and is executed within the external-executable-nimo. It reads flow data (microflows file) from the collection server, produces NetFlow demands and demand traffic data, and inserts this data into the WAE YANG run-time database. In addition to producing demand and traffic data, `flow_get` also produces inter-AS (IAS) flow files.

**Note** In production networks, do not use -log-level=INFO | DEBUG | TRACE for `flow_get`.

# DNF Collection

The following figures show the DNF architecture and the DNF workflow. In this architecture, each set of network devices exports flow data to a corresponding collection server. The DNF cluster performs flow computation so that each agent is responsible for the flow computation of its corresponding flow collection server that runs the flow collector. The master node aggregates this information and passes it back to `flow_collector_ias`.

*Figure 2: DNF Architecture*

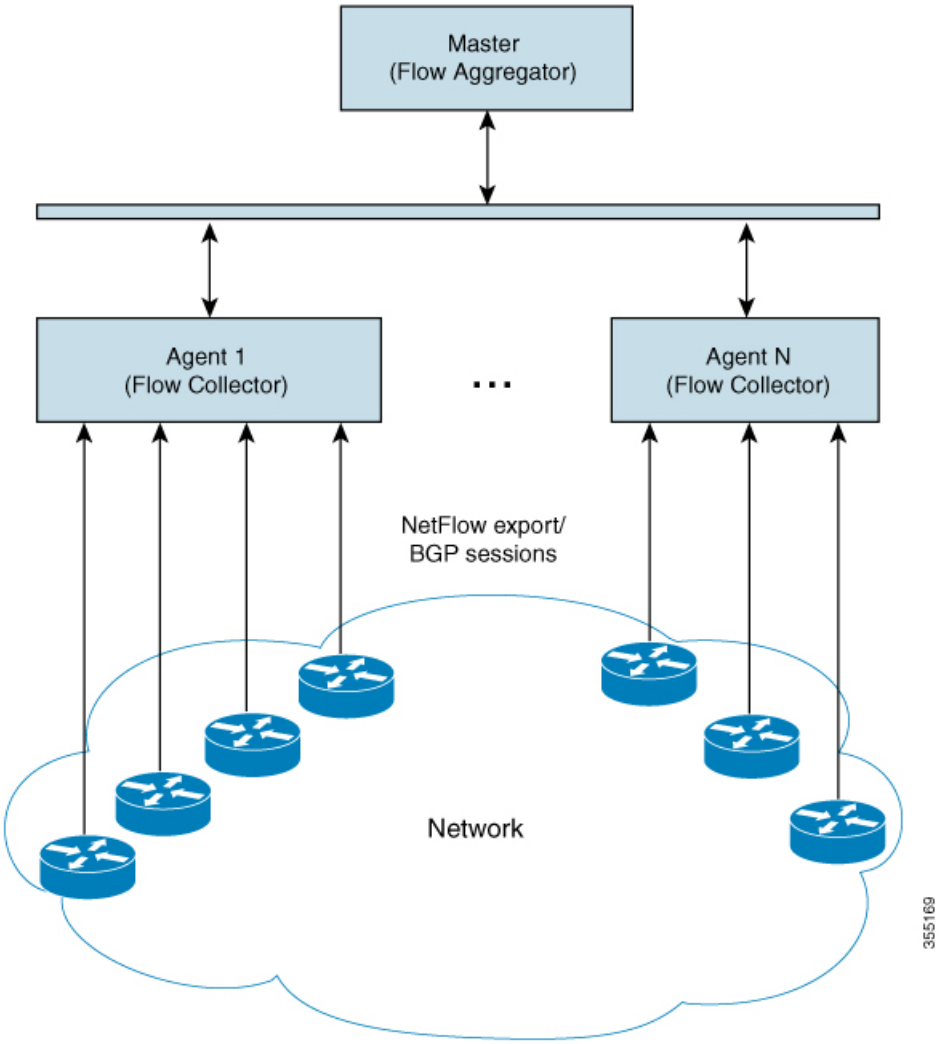
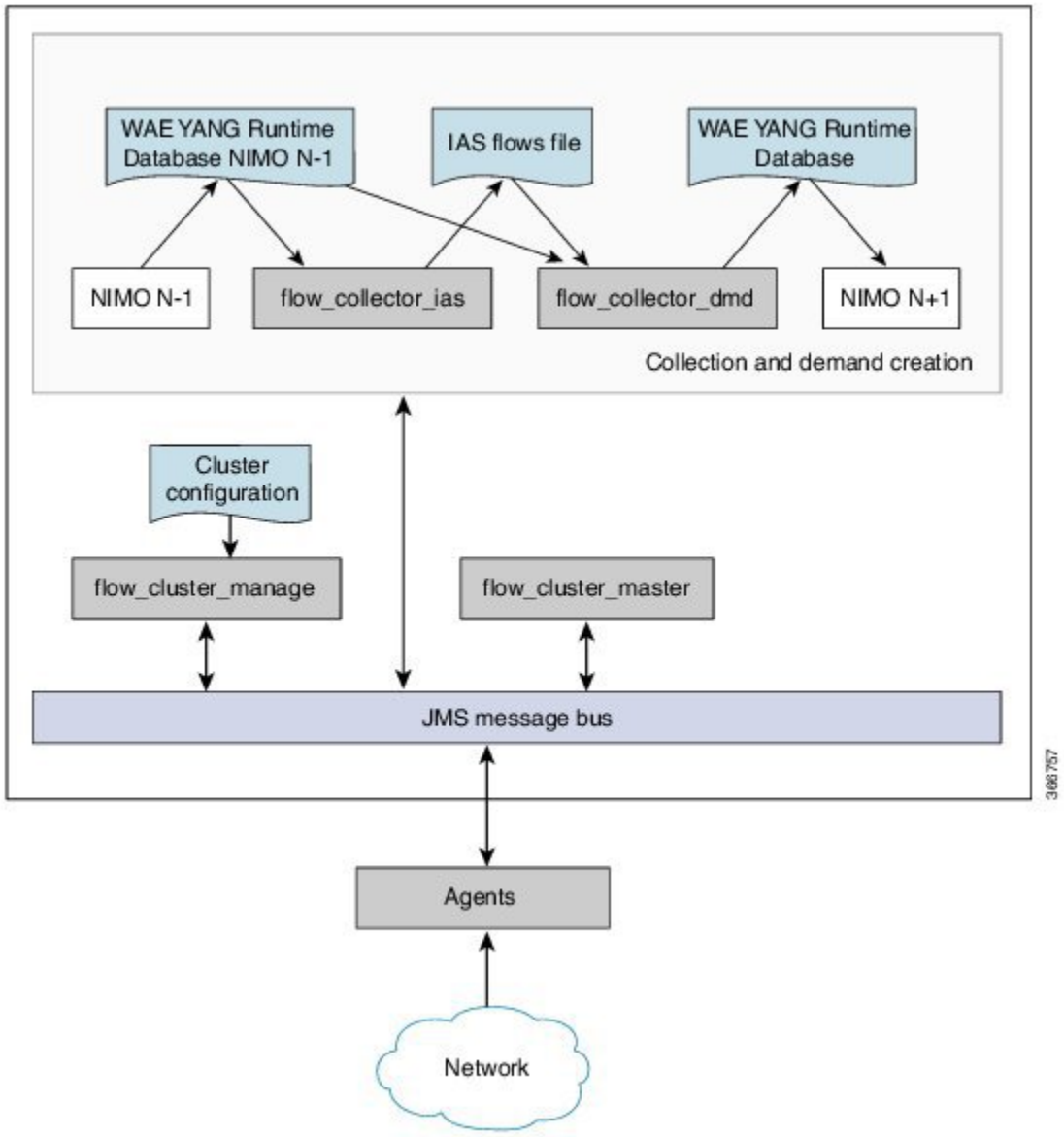


*Figure 3: DNF Collection Workflow*




- **`flow_cluster_manage`**—This CLI tool is used to configure and get status from the cluster. It takes a cluster configuration file and sends the configuration to the cluster. For more information, see Use the DNF Configuration File (Run flow_cluster_manage), on page 20.

  A REST API is also available to configure and request status from the cluster as an alternative to using `flow_cluster_manage`. For more information, see the API documentation from one of the following locations:

  - `<wae-installation-directory>docs/api/netflow/distributed-netflow-rest-api.html`
  - http://<*master-IP-address*>:9090/api-doc For example, to get the cluster configuration:

    For example, to get the cluster configuration:

```
curl -X GET http://localhost:9090/cluster-config > config-file-1
```

For example, to set the cluster configuration:

```
curl -X PUT http://localhost:9090/cluster-config @config-file-2
```

For example, to get the cluster status:

```
curl -X GET http://localhost:9090/cluster-status > config-file-1
```

- **flow_cluster_master**—The master service collects all flow data results from all the agents and aggregates the data, which is sent back to `flow_collector_ias`. For more information, see Master and Agents, on page 12.
- **flow_cluster_agent**—The agent service manages and tracks the status of the associated flow collector. Each agent receives and computes the flow data from its corresponding collection server.
- **flow_cluster_broker**—(not shown in diagram) The JMS broker service allows communication between all components within the architecture, including master and agents. For more information, see Java Message Server (JMS) Broker, on page 12.
- **flow_collector_ias**—This CLI tool, which is configured inside the `nimo_flow_collector_ias_and_dmd.sh` file and is executed within the external-executable-nimo, receives the flow data from the master and produces the IAS flows file. For more information, see Configure flow_collector_ias and flow_collector_dmd, on page 22.
- **flow_collector_dmd**—This CLI tool sends NetFlow demands and demand traffic to the WAE YANG run-time database. This is configured inside the `nimo_flow_collector_ias_and_dmd.sh` file and is executed within the external-executable-nimo.

**Note** In production networks, do not use -log-level=INFO | DEBUG | TRACE for `flow_collector_ias` or `flow_collector_dmd`.

# Centralized NetFlow Configuration Workflow

To configure CNF and start collection:

**Note** Unless stated otherwise, do not change permissions on files that were deployed during WAE installation.

**Step 1** Confirm that the CNF NetFlow Requirements , on page 7 are met.

**Step 2** Prepare the Operating System for CNF, on page 7

**Step 3** Create the CNF Configuration File, on page 8

**Step 4** Use the CNF Configuration File (Run flow_manage), on page 9

**Step 5** Configure CNF Collection, on page 9

a) Configure flow_get, on page 9

b) Configure the external-executable-nimo for CNF, on page 10

# CNF NetFlow Requirements

For system requirements, see the *Cisco WAE System Requirements* document.

## Licensing

Confirm with your Cisco WAE representative that you have the correct licenses for getting flow and flow demands when using the `flow_manage` and `flow_get` tools.

# Prepare the Operating System for CNF

To prepare the OS for CNF, run the following `flow_manage` command from the WAE CLI:

```
sudo -E ./flow_manage -action prepare-os-for-netflow
```

The `prepare-os-for-netflow` option does the following:

- Uses the `setcap` command to allow non-root users limited access to privileged ports (0-1023). This is necessary when configuring the flow collector to use a port under 1024 to listen to BGP messages.
- Configures the OS instance to reserve up to 15,000 of file descriptors to account for the large number of temporary files that may be produced by `flow_get` in a CNF architecture.

**Note** After executing this command, you must reboot the server.

# NetFlow Collection Configuration

The flow collection process supports IPv4 and IPv6 flows captured and exported by routers in the ingress direction. It also supports IPv4 and IPv6 iBGP peering.

Routers must be configured to export flows to and establish BGP peering with the flow collection server. Note the following recommendations:

- NetFlow v5, v9, and IPFIX datagram export to the UDP port number of the flow collection server, which has a default setting of 2100. Export of IPv6 flows requires NetFlow v9 or IPFIX.
- Configure the flow collection server on the routers as an iBGP route reflector client so that it can send BGP routes to edge or border routers. If this is not feasible, configure a router or route server that has a complete view of all relevant routing tables.
- Configure the source IPv4 address of flow export data grams to be the same as the source IPv4 address of iBGP messages if they are in the same network address space.
- Explicitly configure the BGP router ID.
- Configure static routing.

- If receiving BGP routes, the maximum length of the BGP **AS_path** attribute is limited to three hops. The reason is to prevent excessive server memory consumption, considering that the total length of BGP attributes, including **AS_path**, attached to a single IP prefix can be very large (up to 64 KB).

# Create the CNF Configuration File

The <NodeFlowConfigs> table contains basic node configuration information used by the `flow_manage` tool when generating configuration information that it passes to the flow collection server. Thus, prior to executing `flow_manage`, you must construct this table as follows:

- Use a tab or comma delimited format.
- Include one row per node (router) from which you are collecting flow data.
- Enter contents described in the following table for each of these nodes. The BGP columns are required only if collecting BGP information.

***Table 1: <NodeFlowConfigs> Table Columns***

| Column | Description |
|---|---|
| Name | Node name |
| SamplingRate | Sampling rate of the packets in exported flows from the node. For example, if the value is 1,024, then one packet out of 1,024 is selected in a deterministic or random manner. |
| FlowSourceIP | IPv4 source address of flow export packets. |
| BGPSourceIP | IPv4 or IPv6 source address of iBGP update messages.<br>This column is needed if the `flow_manage -bgp` option is true. |
| BGPPassword | BGP peering password for MD5 authentication.<br>Use this column if the `flow_manage -bgp` option is true and if BGPSourceIP has a value. |

The following is a <NodeFlowConfigs> Table example:

| Name | SamplingRate | FlowSourceIP | BGPSourceIP | BGPPassword |
|---|---|---|---|---|
| paris-er1-fr | 1024 | 192.168.75.10 | 69.127.75.10 | ag5Xh0tGbd7 |
| chicago-cr2-us | 1024 | 192.168.75.15 | 69.127.75.15 | ag5Xh0tGbd7 |
| chicago-cr2-us | 1024 | 192.168.75.15 | 2001:db9:8:4::2 | ag5Xh0tGbd7 |
| tokyo-br1-jp | 1024 | 192.168.75.25 | 69.127.75.25 | ag5Xh0tGbd7 |
| brazilia-er1-bra | 1024 | 192.168.75.30 | 2001:db8:8:4::2 | ag5Xh0tGbd7 |

# Use the CNF Configuration File (Run flow_manage)

The `flow_manage` tool starts and stops the flow collection process (pmacct), as well as reloads the configuration information stored in the <NodeFlowConfigs> table when you change it. As such, you must run it before executing the CNF collection process:

```
flow_manage -server-ip 198.51.100.1 -action start -node-flow-configs-table flowconfigs.txt
```

We recommend that you configure your operating system to automatically start and stop `flow_manage` at system start or shutdown.

The following command reloads the <NodeFlowConfigs> table in the flowconfigs.txt file to a flow collection server with an IP address of 192.168.1.3.

```
flow_manage -server-ip 198.51.100.1 -action reload -node-flow-configs-table flowconfigs.txt
```

Sample Configuration File:

```
<NodeFlowConfigs>
Name,BGPSourceIP,FlowSourceIP,BGPPassword,SamplingRate
ar1.dus.lab.test.com,1.2.3.4,1.2.3.5,bgp-secret,666
ar1.ham.lab.test.com,1.2.3.41,1.2.3.52,bgp-secret-2,667
cr1.ams.lab.test.com,1.2.3.51,1.2.3.53,bgp-secret-3,8000
<IPPrefixFiltering>
NetworkAddress
198.51.100.1/24
198.51.100.1/23
198.51.100.1/22
198.51.100.1/21
```

For more information on `flow_manage` options, navigate to *`wae-installation-directory`*`/bin` and enter **`flow_manage -help`**.

# Configure CNF Collection

## Configure flow_get

This CLI tool is configured inside the `<WAE_installation_directory>/etc/netflow/ansible/bash/nimo_flow_get.sh` script and is executed within the external-executable-nimo. The tool combines the data from topology NIMO network models and the flow collection server.

Before editing, change the permissions on this file:

```
chmod +x nimo_flow_get.sh
```

Edit the `nimo_flow_get.sh` as follows:

- **CUSTOMER_ASN**—Enter the ASN.
- **SPLIT_AS_FLOWS_ON_INGRESS**—When multiple external ASNs are connected to an IXP switch, it determines whether to aggregate traffic from all ASNs or to distribute it proportionally to MAC accounting ingress traffic. The default value is aggregate. The other value is mac-distribute.
- **ADDRESS_FAMILY**—Enter list of protocol versions to include (comma-separated entries). The default is ipv4,ipv6.

`nimo_flow_get.sh` example:

```
#!/bin/bash
# modify as needed - BEGIN
CUSTOMER_ASN=4103291
SPLIT_AS_FLOWS_ON_INGRESS=aggregate
ADDRESS_FAMILY=ipv4,ipv6
# modify as needed - END
```

For more information on `flow_get` options, see https://www.cisco.com/c/en/us/td/docs/net_mgmt/wae/6-4/platform/configuration/guide/WAE_Platform_Configuration_Guide/wp_netflow.html#pgfId-1082437 or navigate to *`wae-installation-directory`*`/bin` and enter **`flow_get -help`**.

## Configure the external-executable-nimo for CNF

The external-executable-nimo runs the `nimo_flow_get.sh` script against a selected network model. In this case, you take an existing model created in WAE and append information from `nimo_flow_get.sh` to create a final network model that contains the flow data you want.

### Before you begin

- You must have a source network model. This is the final network model which includes topology collection and any other NIMO collections you want to include.
- Confirm that you have already completed the preliminary tasks in Centralized NetFlow Configuration Workflow, on page 6.

**Step 1** From the Expert Mode, navigate to **/wae:networks**.

**Step 2** Click the plus (+) sign and enter a network model name. We recommend a unique name that is easily identifiable; for example, networkABC_CNF_flow_get.

**Step 3** Click the **nimo** tab.

**Step 4** From the **Choice - nimo-type** drop-down list, choose **external-executable-nimo**.

**Step 5** Click **external-executable-nimo** and select the source network.

**Step 6** Click the **advanced** tab and enter the following:

- **input-file-version**—Enter **7.1**.
- **input-file-format**—Select **.pln** as the plan file format of the source network model.
- **argv**—Enter ***<directory_path>*`/nimo_flow_get.sh $$input $$output`**.

**Step 7** To verify configuration, click **run** from the external-executable-nimo tab.

### Example

If using the WAE CLI (in config mode), enter:

```
networks network <network-model-name> nimo external-executable-nimo source-network
<source-network> advanced argv nimo_flow_get.sh $$input $$output ]
admin@wae(config-network-<network-model-name>)# commit
Commit complete.
admin@wae(config-network-<network-model-name>)# exit
```

```
admin@wae(config)# exit

admin@wae# networks network <network-model-name> nimo external-executable-nimo run
```

### What to do next

Once the external-executable-nimo is configured, you can schedule it to run or access the data from WAE Design.

# DNF NetFlow Configuration Workflow

To configure DNF and start collection:

**Note** Unless stated otherwise, do not change permissions on files that were deployed during WAE installation.

**Step 1** Confirm that the Distributed NetFlow Requirements, on page 11 are met.

**Step 2** Set Up the DNF Cluster, on page 13

a) Modify the DNF Configuration Files, on page 13

b) Deploy DNF Cluster, on page 16

**Step 3** Configure DNF Cluster, on page 17

a) Create the DNF Cluster Configuration File, on page 17

b) Use the DNF Configuration File (Run flow_cluster_manage), on page 20

**Step 4** Configure DNF Collection, on page 22

a) Configure flow_collector_ias and flow_collector_dmd, on page 22

b) Configure the external-executable-nimo for DNF, on page 23

# Distributed NetFlow Requirements

For system requirements, see the *Cisco WAE System Requirements* document.

In addition, the following are required for all cluster elements (master, agents, JMS Broker):

- Ansible 2.1 or later.
- Java virtual machine (JVM) has the same installation path for all elements. The java executable should be in the path readable for all users.
- A sudo SSH user with the same name in each server dedicated for the cluster (broker, master, and all the agents) must exist. Make a note of this user name because it is used in the group_vars/all Ansible file (discussed later in this section).

  WAE Planning software must be installed on a server (installation server) with the appropriate license file.
- Agent system requirements meet the same requirements needed for WAE installation.

- The flow collection process supports IPv4 and IPv6 flows captured and exported by routers in the ingress direction. It also supports IPv4 and IPv6 iBGP peering. Routers must be configured to export flows to and establish BGP peering with the flow collection server. For more information, see NetFlow Collection Configuration, on page 7

## Licensing

Confirm with your Cisco WAE representative that you have the correct licenses for getting flow and flow demands when using the `flow_cluster_master`, `flow_collector_ias`, and `flow_collector_dmd` tools.

## Java Message Server (JMS) Broker

Each distributed flow collection setup must have a single JMS broker instance in order for the master, agents, and client within a cluster to exchange information. All information is interchanged through the broker and enables all the components to communicate with each other. DNF supports a dedicated JMS broker.

The broker must have the following features enabled in order for all JMS clients (master, agents, and `flow_collector_ias` instances) to work:

- Out of band file messaging
- Support of obfuscated passwords in configuration files

## Master and Agents

Ansible files are used to install and run DNF configuration on the JMS broker, master, and agent servers.

**Master**

The master node provides the following services in the cluster:

- Monitors and tracks agent status.
- Monitors and tracks the status of the last completed IAS computation.
- Aggregates IAS flow data coming from all agents back to the client.
- Handles configuration and status requests from the cluster.

**Agents**

Only one agent per server is supported. Agents cannot be on the WAE installation or data collection server. Each agent receives and computes flow data from its corresponding collection server.

**Note** You have the option to deploy only one agent in the cluster. This is an alternative to CNF for networks that are expected to expand in size or grow in traffic.

# Set Up the DNF Cluster

## Modify the DNF Configuration Files

If you use default WAE installation options, there are only a few mandatory parameters that must be changed. These will be noted in the applicable configuration topics. The topics described in this section assume the following:

- The master server (installation server) is where the WAE planning software has been installed and default directories are used. In particular, the configuration files used for DNF on the installation server are located in `<wae_installation_directory>/etc/netflow/ansible`.
- A dedicated JMS broker will be used in DNF configuration.
- In configuration examples, the following values are used:
  - Master and JMS broker IP address—198.51.100.10
  - Agent 1 IP address—198.51.100.1
  - Agent 2 IP address—198.51.100.2
  - Agent 3 IP address—198.51.100.3

### group_vars/all

The file is located in `<WAE_installation_directory>/etc/netflow/ansible/group_vars/all`. This file is the Ansible file that contains the variable definitions that are used in the playbook files.

Edit the following options:

| Option | Description |
|---|---|
| LOCAL_WAE_INSTALLATION_DIR_NAME | The local path that contains the WAE installation file. |
| WAE_INSTALLATION_FILE_NAME | The filename of the WAE installation file. |
| TARGET_JDK_OR_JRE_HOME | The full path and filename of the Oracle JRE file. All machines in the cluster (broker, master, and all the agents) should have the JRE previously installed under this variable. |
| LOCAL_LICENSE_FILE_PATH | The full path to the license file. |
| SSH_USER_NAME | The SSH user name created or used when SSH was enabled on each machine.<br>This sudo user is used by Ansible to deploy the cluster over SSH. |

For example (comments removed):

```
LOCAL_WAE_INSTALLATION_DIR_NAME: "/wae/wae-installation"
WAE_INSTALLATION_FILE_NAME: "wae-linux-v16.4.8-1396-g6114ffa.rpm"
TARGET_JDK_OR_JRE_HOME: "/usr/lib/jvm/java-1.8.0-openjdk-1.8.0_45"
LOCAL_LICENSE_FILE_PATH: "/home/user1/.cariden/etc/MATE_Floating.lic"
TARGET_SSH_USER: ssh_user
```

### hosts

The file is located in *<WAE_installation_directory>*/etc/netflow/ansible/hosts. This file is the Ansible inventory file and it includes a list of all the servers in the cluster.

Only edit the corresponding IP addresses for the broker, master, and all agents. Do not edit any of the other variables. If applicable, add more agents.

For example:

```
[dnf-broker]
198.51.100.10 ansible_ssh_user={{SSH_USER_NAME}}
[dnf-master]
198.51.100.10 ansible_ssh_user={{SSH_USER_NAME}}
[dnf-agent-1]
198.51.100.1 ansible_ssh_user={{SSH_USER_NAME}}
[dnf-agent-2]
198.51.100.2 ansible_ssh_user={{SSH_USER_NAME}}
[dnf-agent-3]
198.51.100.3 ansible_ssh_user={{SSH_USER_NAME}}
```

### prepare-agents.yml

This file does not need to be edited and provides the following to all specified agents:

- Allows non-root users limited access to privileged ports (0-1023). This is necessary when configuring the flow collector to use a port under 1024 to listen to BGP messages.
- Configures the OS instance to reserve up to 15,000 of file descriptors to account for the large number of temporary files that may be produced.
- Reboots all the agents.

The file is located in *<WAE_installation_directory>*/etc/netflow/ansible/prepare-agents.yml.

### startup.yml

The file is located in *<WAE_installation_directory>*/etc/netflow/ansible/startup.yml.

This file is used to automatically start the broker, master, and agents. If you have more than two agents, edit this file to add more.

For example:

```
- hosts: all
roles:
- check-ansible-version
- hosts: dnf-broker
roles:
- start-broker
- hosts: dnf-master
roles:
- start-master
- hosts: dnf-agent-1
roles:
- {role: start-agent, instance: instance-1}
- hosts: dnf-agent-2
roles:
- {role: start-agent, instance: instance-2}
- hosts: dnf-agent-3
roles:
- {role: start-agent, instance: instance-3}
```

### service_conf

The file is located in *<WAE_installation_directory>*/etc/netflow/ansible/bash/service.conf.

This file provides the common configuration options that are used by the broker, master, and agents.

Edit the following options:

| Option | Description |
|---|---|
| jms-broker-server-name-or-ip-address | IP address of the broker. |
| jms-broker-jms-port | JMS port number being used for the broker. |
| jms-broker-http-port | HTTP port number being used for the broker. |
| jms-broker-username | This is used internally and does not need to be changed. |
| jms-broker-password | We recommend generating and using an obfuscated password. For example:<br>`# ./flow_cluster_manage -action print-obfuscation`<br>`type in the clear text > password-0`<br>`obfuscated text: ENC(h4rWRpG54WgVZRTE90Zb/JszY4dd4CGc)` |
| obfuscated text | From example above:<br>**`ENC(h4rWRpG54WgVZRTE90Zb/JszY4dd4CGc)`** |
| jms-broker-use-tls | To encrypt all data communication in the DFC cluster, then enter true. If set to true, there will be some performance degradation. |
| append-to-log-file | If appending information to the local log file, enter true. |
| use-flume | If using a flume server, enter true. |
| flume-server | Enter the IP address of the server running the flume agent. If using the flume server that is automatically installed during WAE server installation, enter the installation server IP address. |
| log-level | Enter logging level type:<br>• off<br>• activity<br>• fatal<br>• error<br>• warn<br>• notice<br>• info<br>• debug<br>• trace |

For example :

```
# jms
jms-broker-server-name-or-ip-address=198.51.100.10
jms-broker-jms-port=61616
jms-broker-http-port=8161
jms-broker-username=user-0
jms-broker-password=ENC(ctrG7GGRJm983M0AsPGnabwh)
jms-broker-use-tls=false

# local logging
append-to-log-file=false

# distributed logging
use-flume=true
flume-server=198.51.100.10

# default for all commands, will be superseded if specified locally in each .sh
log-level=info
```

## Deploy DNF Cluster

To deploy the DNF cluster:

**Step 1** Install the broker, master and agents:

```
# ansible-playbook -i hosts install.yml
```

**Note** The `uninstall.yml` playbook file uninstalls the files and removes the TARGET_WAE_ROOT directory, which is defined in the all file.

**Step 2** Prepare and reboot the agents for DNF:

```
# ansible-playbook -i hosts prepare-agents
```

**Step 3** Start the master, broker, and agents.:

```
# ansible-playbook -i hosts startup.yml
```

**Note** The `shutdown.yml` playbook file shuts down the master, broker, and agents.

**Step 4** Confirm that the master, broker, and agents are running:

```
# ansible-playbook -i hosts list.yml
```

**Step 5** After the machines reboot, you can verify if all the agents are up by executing the following command:

```
# flow_cluster_manage -active request-cluster-status
```

A successful result should list running details of the master and all agents. At the end of the result, the CLUSTER SUMMARY should look similar to the following:

```
CLUSTER SUMMARY - BEGIN
cluster all OK: false
configured size: 0
agents up: 2
daemons up: 0
agents w/wrong IDs: []
agents w/low ulimit IDs: []
computation mode: ias-in-the-background
last result time: n/a
last no-result time: n/a
```

```
max diff time: 2 ms
max diff time OK: true
CLUSTER SUMMARY - END
```

**Note** In the preceding example, the `agents up` lists two running agents. The `cluster all OK` field is false because the cluster has not been configured yet. This status should change after configuring the cluster.

# Configure DNF Cluster

## Create the DNF Cluster Configuration File

To more easily create the cluster configuration file for `flow_manage_cluster`, you can use the CNF configuration file produced from `flow_manage` as a template for the cluster configuration file.

For example:

**Step 1** Produce the template configuration file:

```
${CARIDEN_HOME}/flow_manage \
-action produce-config-file \
-node-flow-configs-table <input-path> \
-cluster-config-file <output-path> \
-interval 120 \
-bgp true \
-bgp-port 10179 \
-port 12100 \
-flow-size lab \
-server-ip ::
```

where *<input-path>* is the path of the node configuration .txt file used in CNF (see Configure and Run the Collector Server for more information on creating this file) and *<output-path>* is the path where you want the resulting seed cluster configuration file to reside. Verify that the output of the seed cluster configuration file is similar to the following:

```
{
    "agentConfigMapInfo": {
        "cluster_1::instance_1":
        {
            "flowManageConfiguration":
            {
                "maxBgpdPeers": 150,
                "bgpTcpPort": 179,
                "flowType": "Netflow",
                "useBgpPeering": true,
                "outfileProductionIntervalInSecs": 900,
                "networkDeploymentSize": "medium",
                "netflowUdpPort": 2100,
                "keepDaemonFilesOnStartStop": true,
                "purgeOutputFilesToKeep": 3,
                "daemonOutputFileMaskSuffix": "%Y.%m.%d.%H.%M.%s",
                "daemonOutputDirPath":
"<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
                "daemonOutputFileMaskPrefix": "out_matrix_",
                "daemonOutputSoftLinkName": "flow_matrix_file-latest",
                "extraAggregation": [],
                "routerConfigList":
```

```
                    [
                        {
                            "name": "ar1.dus.lab.cariden.com",
                            "bGPSourceIP": "1.2.3.4",
                            "flowSourceIP": "1.2.3.5",
                            "bGPPassword": "bgp-secret",
                            "samplingRate": "666"
                        },
                        {
                            "name": "cr1.ams.lab.cariden.com",
                            "bGPSourceIP": "1.2.3.51",
                            "flowSourceIP": "1.2.3.53",
                            "bGPPassword": "bgp-secret-3",
                            "samplingRate": "8000"
                        }
                    ],
                "appendedProperties":
                {
                    "key1": "value1",
                    "key2": "value2"
                }
            }
        },
}
```

**Step 2** Edit the file to include each agent configuration. Copy, paste, and edit each section as it applies to each agent in the cluster. This example shows two agents:

```
{
    "agentConfigMapInfo": {
        "cluster_1::instance_1":
        {
            "flowManageConfiguration":
            {
                "maxBgpdPeers": 150,
                "bgpTcpPort": 179,
                "flowType": "Netflow",
                "useBgpPeering": true,
                "outfileProductionIntervalInSecs": 900,
                "networkDeploymentSize": "medium",
                "netflowUdpPort": 2100,
                "keepDaemonFilesOnStartStop": true,
                "purgeOutputFilesToKeep": 3,
                "daemonOutputFileMaskSuffix": "%Y.%m.%d.%H.%M.%s",
                "daemonOutputDirPath":
"<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
                "daemonOutputFileMaskPrefix": "out_matrix_",
                "daemonOutputSoftLinkName": "flow_matrix_file-latest",
                "extraAggregation": [],
                "routerConfigList":
                    [
                        {
                            "name": "ar1.dus.lab.anyname.com",
                            "bGPSourceIP": "1.2.3.4",
                            "flowSourceIP": "1.2.3.5",
                            "bGPPassword": "bgp-secret",
                            "samplingRate": "666"
                        },
                        {
                            "name": "cr1.ams.lab.anyname.com",
                            "bGPSourceIP": "1.2.3.51",
                            "flowSourceIP": "1.2.3.53",
                            "bGPPassword": "bgp-secret-3",
                            "samplingRate": "8000"
```

```
                        }
                    ],
                "appendedProperties":
                {
                    "key1": "value1",
                    "key2": "value2"
            }
        }
},
```

The information for the second agent starts here:

```
         "cluster_1::instance_2":
         {
             "flowManageConfiguration":
             {
                 "maxBgpdPeers": 150,
                 "bgpTcpPort": 179,
                 "flowType": "Netflow",
                 "useBgpPeering": true,
                 "outfileProductionIntervalInSecs": 900,
                 "networkDeploymentSize": "medium",
                 "netflowUdpPort": 2100,
                 "keepDaemonFilesOnStartStop": true,
                 "purgeOutputFilesToKeep": 3,
                 "daemonOutputFileMaskSuffix": "%Y.%m.%d.%H.%M.%s",
                 "daemonOutputDirPath":
"<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
                 "daemonOutputFileMaskPrefix": "out_matrix_",
                 "daemonOutputSoftLinkName": "flow_matrix_file-latest",
                 "extraAggregation": [],
                 "routerConfigList":
                     [
                         {
                             "name": "ar1.dus.lab.anyname.com",
                             "bGPSourceIP": "5.6.7.8",
                             "flowSourceIP": "5.6.7.9",
                             "bGPPassword": "bgp-secret-2",
                             "samplingRate": "666"
                         },
                         {
                             "name": "cr1.ams.lab.anyname.com",
                             "bGPSourceIP": "5.6.7.81",
                             "flowSourceIP": "5.6.7.83",
                             "bGPPassword": "bgp-secret-4",
                             "samplingRate": "8000"
                         }
                     ],
                 "appendedProperties":
                 {
                     "key1": "value1",
                     "key2": "value2"
             }
         }
},
```

# Use the DNF Configuration File (Run flow_cluster_manage)

The `flow_cluster_manage` tool diagnoses and controls the distributed NetFlow collection cluster. After creating the configuration file, use `flow_cluster_manage` to send the cluster configuration file to the cluster (**`flow_cluster_manage -send-cluster-configuration`**). All flow collection processes in all agents will reload the configuration information stored in that configuration file.

**Note** We recommend that you configure your system to automatically start and stop `flow_cluster_master`, `flow_cluster_agent`, and `flow_cluster_broker` at system start or shutdown.

You can also use the `flow_cluster_manage` tool to retrieve cluster status. For example:

```
# flow_cluster_manage -action request-cluster-status
```

**Note** The cluster will take approximately a minute to take the configuration.

Sample result of cluster status:

```
CLUSTER STATUS - BEGIN

    AGENT NODE - BEGIN
        cluster ID:            cluster_1
        instance ID:           instance_1
        process ID:            15292
        start time:            2017-07-10.09:19:43.000-0700
        up time:               00d 00h 00m 40s 824ms
        unique ID:
    bc.30.5b.df.8e.b5-15292-1729199940-1499703582925-1a23cb00-ed76-4861-94f5-461dcd5b2070
        last HB received:      2017-07-10.09:20:24.004-0700
        last HB age:           00d 00h 00m 04s 779ms
        skew time:             00d 00h 00m 00s 010ms computation sequence   0
        computational model    ias-in-the-background computing IAS:         false
        ip addresses:          [128.107.147.112, 172.17.0.1,
2001:420:30d:1320:24a8:5435:2ed5:29ae, 2001:420:30d:1320:be30:5bff:fedf:8eb5,
    2001:420:30d:1320:cd72:ec61:aac8:2e72,2001:420:30d:1320:dc55:a772:de80:a73f]
       mac address:          bc.30.5b.df.8e.b5 jvm memory utilization: 4116Mb/4116Mb/3643Mb
 max opened files:      15000
        processors:            8
        daemon period:         00d 00h 15m 00s 000ms
        daemon out dir:
   /media/1TB/user1/sandboxes/git/netflow-flexible/package/linux-release/lib/ext/pmacct/insta

    nces/flow_cluster_agent_cluster_1::instance_1
        daemon process ID: 15344
        daemon is: running
        bgp port: 179
        bgp port status: up
        netflow port: 2100
        netflow port status: up
    AGENT NODE - END

    AGENT NODE - BEGIN
        cluster ID: cluster_1
        instance ID: instance_2
        process ID: 15352
```

```
        start time: 2017-07-10.09:19:49.000-0700
        up time: 00d 00h 00m 30s 748ms
        unique ID:
      bc.30.5b.df.8e.b5-15352-1729199940-1499703589727-12989336-b314-4f85-9978-242882dd16da
        last HB received: 2017-07-10.09:20:20.746-0700
        last HB age: 00d 00h 00m 08s 037ms
        skew time: 00d 00h 00m 00s 014ms
        computation sequence 0
        computational model ias-in-the-background
        computing IAS: false
        ip addresses: [128.107.147.112, 172.17.0.1,
      2001:420:30d:1320:24a8:5435:2ed5:29ae, 2001:420:30d:1320:be30:5bff:fedf:8eb5,
      2001:420:30d:1320:cd72:ec61:aac8:2e72, 2001:420:30d:1320:dc55:a772:de80:a73f]
        mac address: bc.30.5b.df.8e.b5
        jvm memory utilization: 4116Mb/4116Mb/3643Mb
        max opened files: 15000
        processors: 8
        daemon period: 00d 00h 15m 00s 000ms
        daemon out dir:
     /media/1TB/user1/sandboxes/git/netflow-flexible/package/linux-release/lib/ext/pmacct/insta

      nces/flow_cluster_agent_cluster_1::instance_2
        daemon process ID: 15414
        daemon is: running
        bgp port: 10179
        bgp port status: up
        netflow port: 12100
        netflow port status: up
      AGENT NODE - END

      MASTER NODE - BEGIN
        cluster ID: cluster_1
        instance ID: instance_id_master_unique
        process ID: 15243
        start time: 2017-07-10.09:19:34.000-0700
        up time: 00d 00h 00m 50s 782ms
        unique ID:
      bc.30.5b.df.8e.b5-15243-415138788-1499703574719-cd420a81-f74c-49d4-a216-ffeb7cde31d5
        last HB received: 2017-07-10.09:20:25.563-0700
        last HB age: 00d 00h 00m 03s 220ms
        ip addresses: [128.107.147.112, 172.17.0.1,
      2001:420:30d:1320:24a8:5435:2ed5:29ae, 2001:420:30d:1320:be30:5bff:fedf:8eb5,
      2001:420:30d:1320:cd72:ec61:aac8:2e72, 2001:420:30d:1320:dc55:a772:de80:a73f]
        mac address: bc.30.5b.df.8e.b5
        jvm memory utilization: 2058Mb/2058Mb/1735Mb
        processors: 8
      MASTER NODE - END

      CLUSTER SUMMARY - BEGIN
        cluster all OK: true
        configured size: 2
        agents up: 2
        daemons up: 2
        agents w/wrong IDs: []
        agents w/low ulimit IDs: []
        computation mode: ias-in-the-background
        last result time: n/a
        last no-result time: n/a
        max diff time: 4 ms
        max diff time OK: true
      CLUSTER SUMMARY - END

CLUSTER STATUS - END
```

The CLUSTER SUMMARY entry at the end of the result gives you a quick summary of whether or not your cluster configuration is operational. You should confirm that `cluster all OK` is true and that the `configured size`, `agents up`, and `daemons up` match the number of agents you configured. There should be no value in `agents w/wrong IDs` and `agents w/low ulimit IDs`. The `max diff` time OK should also be set to true. If this is not the case, look into the agent and master details for troubleshooting information.

For more information on `flow_manage_cluster` options, navigate to *`wae-installation-directory`*`/bin` and enter **`flow_manage_cluster -help`**.

# Configure DNF Collection

## Configure flow_collector_ias and flow_collector_dmd

These CLI tools are configured inside the *`<WAE_installation_directory>`*`/etc/netflow/ansible/bash/nimo_flow_collector_ias_dmd.sh` script and is executed within the external-executable-nimo. The `flow_collector_ias` and `flow_collector_dmd` tools generate demands and demand traffic with NetFlow data received from the cluster. Edit the as follows:

Before editing, change the permissions on this file:

```
chmod +x nimo_flow_collector_ias_dmd.sh
```

- **CUSTOMER_ASN**—Enter ASN.
- **SPLIT_AS_FLOWS_ON_INGRESS**—When multiple external ASNs are connected to an IXP switch, it determines whether to aggregate traffic from all ASNs or to distribute it proportionally to MAC accounting ingress traffic. The default value is aggregate. The other value is mac-distribute.
- **ADDRESS_FAMILY**—Enter list of protocol versions to include (comma-separated entries). The default is ipv4,ipv6.
- **WAIT_ON_CLUSTER_TIMEOUT_SEC**—Enter the number of seconds to wait before for timing out when delegating the computation of the IAS flows into the distributed cluster. The default is 60 seconds.

`nimo_flow_collector_ias_dmd.sh` example:

```
#!/bin/bash

# this script should be called from NSO's 'external executable NIMO' configuration window
# in this way:
# /path-to/nimo_flow_collector_ias_and_dmd.sh $$input $$output

# modify as needed - BEGIN
CUSTOMER_ASN=142313
SPLIT_AS_FLOWS_ON_INGRESS=aggregate
ADDRESS_FAMILY=ipv4,ipv6
WAIT_ON_CLUSTER_TIMEOUT_SEC=60
# modify as needed - END
```

For more information on `flow_collector_ias` or `flow_collector_dmd`options, navigate to *`wae-installation-directory`*`/bin` and enter **`flow_collector_ias -help`** or **`flow_collector_dmd -help`**.

# Configure the external-executable-nimo for DNF

The external-executable-nimo runs the `nimo_flow_collector_ias_dmd.sh` script against a selected network model. In this case, you take an existing model created in WAE and append information from `nimo_flow_collector_ias_dmd.sh` to create a final network model that contains the flow data you want.

### Before you begin

- You must have a source network model. This is the final network model which includes topology collection and any other NIMO collections you want to include.
- Confirm that you have already completed the preliminary tasks in DNF NetFlow Configuration Workflow, on page 11.

**Step 1** From the Expert Mode, navigate to **/wae:networks**.

**Step 2** Click the plus (+) sign and enter a network model name. We recommend a unique name that is easily identifiable; for example, networkABC_DNF_flow_ias_dmd

**Step 3** Click the **nimo** tab.

**Step 4** From the **Choice - nimo-type** drop-down list, choose **external-executable-nimo**.

**Step 5** Click **external-executable-nimo** and select the source network.

**Step 6** Click the **advanced** tab and enter the following:

- **input-file-version**—Enter **7.1**.
- **input-file-format**—Select **.pln** as the plan file format of the source network model.
- **argv**—Enter ***<directory_path>*/nimo_flow_collector_ias_dmd.sh $$input $$output**.

**Step 7** To verify configuration, click **run** from the external-executable-nimo tab.

### Example

If using the WAE CLI (in config mode), enter:

```
networks network <network-model-name> nimo external-executable-nimo source-network
<source-network> advanced argv nimo_flow_collector_ias_dmd.sh $$input $$output ]
admin@wae(config-network-<network-model-name>)# commit
Commit complete.
admin@wae(config-network-<network-model-name>)# exit
admin@wae(config)# exit

admin@wae# networks network <network-model-name> nimo external-executable-nimo run
```

### What to do next

Once the external-executable-nimo is configured, you can schedule it to run or access the data from WAE Design.