



## Enhancements to VM Image Packaging

The Cisco Enterprise NFVIS VM image packaging tool, `nfvpt.py`, is enhanced to support functionality required for Cisco SD-WAN Cloud OnRamp for Colocation solution.

- [NFVIS Specific Enhancements, on page 1](#)
- [Cisco SD-WAN Cloud OnRamp for Colocation Packaging Enhancements, on page 2](#)
- [VM Packaging Parameters, on page 3](#)
- [VM Packaging Utility Usage Examples, on page 4](#)
- [Packaging a VM, on page 4](#)

## NFVIS Specific Enhancements



**Note** Use `pack_dir` option if the `*.tar.gz` already exists and you want to modify the bootstrap configuration file or `image_properties.xml` manually.

The following parameters are added as part of the NFVIS specific enhancements:

```
--pack_dir <DIR> PACK
                                package all files in directory

Resources:

--vnic_names VNIC_NAMES
                                list of vnic number to name mapping in format
                                number:name example --vnic_names
                                1:GigabitEthernet2,2:GigabitEthernet4
```

### Usage

Follow the steps to change a single line in `day-0` configuration file or add a single option in `image_properties.xml`:

1. Get the working VM packaging image - `isrv*.tar.gz`.

2. Extract the contents - tar -xvf isrv\*.tar.gz.
3. Modify the file contents as required.
4. nfvpt.py --pack\_dir current-working-dir-with-files -i isrv.qcow2 -o isrv.tar.gz

## Cisco SD-WAN Cloud OnRamp for Colocation Packaging Enhancements

The following parameters are the enhancements specific to SD-WAN:

```
--json JSON           Provide JSON input for bootstrap variables; mutually
                       exclusive with custom and bootstrap configs

--multi_use           Add options for use in multiple use-cases

--app_vendor APP_VENDOR
                       Application Vendor e.g. Cisco, Juniper etc

--bootstrap BOOTSTRAP
                       Every bootstrap file should be a different option HA
                       packaging format: --bootstrap mount_point:<value>,file
                       :<file2mount>[,<attrib>:<value>] mount_point:<value>
                       and file:<file2mount> are mandatory followed by one or
                       more attributes in the format <attrib>:<value> Legacy
                       format: --bootstrap file1,file2... See usage.txt for
                       more details

--ha_package          enable HA packaging

--mgmt_vnic MGMT_VNIC
                       VM management interface identifier

HA options:

--ha_capable

--ha_vnic HA_VNIC     VM HA vnic CSV list

Custom Properties:

--custom CUSTOM       custom properties format: --custom ["propattr_"<attr>:
                       <value>],key:<value>,[keyattr_<attr>:<value>],type:<va
```

`lue>,val<N>:<value>,[val<N>attr_<attr>:<value>]` Allows  
specification of custom properties: 0 or more  
`propattr_<attr>:<value>` pairs - 'propattr' is a  
keyword and used to specify property attributes  
`key:<value>` pairs 0 or more `keyattr_<attr>:value` pairs  
- 'keyattr' is a keyword and is used to specify key  
attributes `type:<value>` pair - type of value  
`valN:<value>` pair - `val1:value,val2:value` etc 0 or  
more `valNattr_<attr>:<value>` pairs - 'val<N>attr' is  
an attribute for `val<N>` See `usage_examples.txt`

## VM Packaging Parameters

The table lists the new parameters that can be passed to the `nfvpt.py` command.

Parameter	Mandatory/Optional	Description
<code>json</code>	Optional	Provide JSON input for bootstrap variables. It's mutually exclusive with custom and bootstrap configs
<code>multi_use</code>	Optional	option for use in multiple use-cases
<code>ha_package</code>	Optional	enable HA packaging
<code>mgmt_vnic</code>	Optional	VM management interface identifier
<code>pack_dir</code>	Optional	package all files in directory
<code>app_vendor</code>	Required	Application Vendor e.g. Cisco, Juniper
<code>ha_capable</code>	Optional	For HA capability
<code>vnic_names</code>	Optional	list of vnic number to name mapping in format <code>number:name</code> <code>--vnic_names</code> <code>1:GigabitEthernet2,2:GigabitEthernet4</code>

## VM Packaging Utility Usage Examples

Given below are the contents of the file *nfvis\_vm\_packaging\_utility\_examples.txt*:

### Example 1: Usage for Palo Alto Firewall

```
nfvpt.py -o PA_L3_HA -i PA-VM-KVM-8.0.5.qcow2 --json d.json -t firewall -n "PA FIREWALL"
-r 8.0.5 --app_vendor PA --monitor true --ha_package
```

### Example 1: Usage for Asav

```
nfvpt.py -i foo.qcow2 -o asav.tar.gz --json pal.json --app_vendor cisco -t firewall -r 10
--optimize true -n asav --monitored true --ha_package -ha_capable
```

### Example 1: Usage for csr

```
nfvpt.py --ha_package --pack_dir /data/intdatastore -i csr1000v-universalk9.16.09.01.qcow2
-o csr1000v-universalk9.16.09.01-ha.tar.gz
```

## Packaging a VM

The following steps shows how to package a bundled VM image, bootstrap files and metadata into an archive:

1. Create a json file using `gen_json.py` tool. The `gen_json.py` needs a pattern that matches bootstrap files as an option. `gen_json.py --help` shows all the details about the options. Redirect the output of `gen_json.py` into a json file.

```
gen_json.py --g "boot*,ios*" --ha > temp.json
```

Include `--ha` option if the packaging is for HA. Include `--multi_use` option if the VM is a part of a service chain.

2. The `temp.json` file has two arrays - `Userinput` and `SysGen`. `Userinput` and `SysGen` are variables from bootstrap files which were tokenized. By default all the variables are included in `Userinput` array. The system generated variables should be moved to `SysGen` array. `vManage` generates some of these variables like `MGMT` and `DATA` IP addresses from the pool provided in the cluster creation on `vManage`. All other variables like `DNS_SERVER`, `VM password` etc. are user inputs at the VM/servicechain provisioning.

Example:

```
interface G0/1
ip address ${MGMT_PRIM} <-- variable
```

3. After making changes to the json file you can package the VM with the script - `nfvpt.py`.

```
nfvpt.py -i <qcow file> -o <tar file name> --json <json file> --app_vendor cisco -t
firewall -r 10 --optimize true -n asav --monitored true --ha_package -ha_capable
```

The tool creates a `.tar.gz` file with the name you have provided.