HITACHI
Inspire the Next

# Hitachi Storage Modules for Red Hat® Ansible

**v02.0.0**

## User Guide

**MK-92ADPTR149-00**
**June 2020**

# Contents

# Preface

## Intended audience

This document is intended for IT and data center administrators in order to automate and manage the configuration of the following Hitachi storage systems:

- Virtual Storage Platform F1500, VSP G1000, VSP G1500
- Virtual Storage Platform G200, G400, G600, G800
- Virtual Storage Platform F400, F600, F800
- Virtual Storage Platform Gx00 with NAS Module (G400, G600, G800)
- Virtual Storage Platform G130, G150, G350, G370, G700, G900
- Virtual Storage Platform F350, F370, F700, F900
- Virtual Storage Platform 5100, 5500, 5100H, 5500H
- Hitachi NAS Platform
- Virtual Storage Platform E990

## Release notes

Read the release notes before installing and using this product. They may contain requirements or restrictions that are not fully described in this document or updates or corrections to this document. Release notes are available on Hitachi Vantara Support Connect: https://knowledge.hitachivantara.com/Documents.

## Referenced documents

The following documents are referenced in this guide:

- *Hitachi Unified Storage Command Control Interface Installation and Configuration Guide*, MK-91DF8306
- *Hitachi Command Control Interface Command Reference*, MK-90RD7009
- *Hitachi Command Control Interface User and Reference Guide*, MK-90RD7010
- *Hitachi VSP G Series Hitachi Universal Volume Manager User Guide*, MK-92RD8024

# Document conventions

This document uses the following typographic conventions:

| Convention | Description |
|---|---|
| **Bold** | ▪ Indicates text in a window, including window titles, menus, menu options, buttons, fields, and labels. Example:<br>Click **OK**.<br>▪ Indicates emphasized words in list items. |
| *Italic* | ▪ Indicates a document title or emphasized words in text.<br>▪ Indicates a variable, which is a placeholder for actual text provided by the user or for output by the system. Example:<br>`pairdisplay -g group`<br>(For exceptions to this convention for variables, see the entry for angle brackets.) |
| `Monospace` | Indicates text that is displayed on screen or entered by the user. Example: `pairdisplay -g oradb` |
| < > angle brackets | Indicates variables in the following scenarios:<br>▪ Variables are not clearly separated from the surrounding text or from other variables. Example:<br>`Status_<report-name><file-version>.csv`<br>▪ Variables in headings. |
| [ ] square brackets | Indicates optional values. Example: [ a \| b ] indicates that you can choose a, b, or nothing. |
| { } braces | Indicates required or expected values. Example: { a \| b } indicates that you must choose either a or b. |
| \| vertical bar | Indicates that you have a choice between two or more options or arguments. Examples:<br>[ a \| b ] indicates that you can choose a, b, or nothing.<br>{ a \| b } indicates that you must choose either a or b. |

This document uses the following icons to draw attention to information:

| Icon | Label | Description |
|---|---|---|
|  | Note | Calls attention to important or additional information. |
|  | Tip | Provides helpful information, guidelines, or suggestions for performing tasks more effectively. |
|  | Caution | Warns the user of adverse conditions and/or consequences (for example, disruptive operations, data loss, or a system crash). |
|  | WARNING | Warns the user of a hazardous situation which, if not avoided, could result in death or serious injury. |

## Conventions for storage capacity values

Physical storage capacity values (for example, disk drive capacity) are calculated based on the following values:

| Physical capacity unit | Value |
|---|---|
| 1 kilobyte (KB) | 1,000 ($10^3$) bytes |
| 1 megabyte (MB) | 1,000 KB or $1,000^2$ bytes |
| 1 gigabyte (GB) | 1,000 MB or $1,000^3$ bytes |
| 1 terabyte (TB) | 1,000 GB or $1,000^4$ bytes |
| 1 petabyte (PB) | 1,000 TB or $1,000^5$ bytes |
| 1 exabyte (EB) | 1,000 PB or $1,000^6$ bytes |

Logical capacity values (for example, logical device capacity, cache memory capacity) are calculated based on the following values:

| Logical capacity unit | Value |
|---|---|
| 1 block | 512 bytes |
| 1 cylinder | Mainframe: 870 KB |

| Logical capacity unit | Value |
|---|---|
| | Open-systems:<br>▪ OPEN-V: 960 KB<br>▪ Others: 720 KB |
| 1 KB | 1,024 ($2^{10}$) bytes |
| 1 MB | 1,024 KB or $1{,}024^2$ bytes |
| 1 GB | 1,024 MB or $1{,}024^3$ bytes |
| 1 TB | 1,024 GB or $1{,}024^4$ bytes |
| 1 PB | 1,024 TB or $1{,}024^5$ bytes |
| 1 EB | 1,024 PB or $1{,}024^6$ bytes |

## Accessing product documentation

Product user documentation is available on Hitachi Vantara Support Connect: https://knowledge.hitachivantara.com/Documents. Check this site for the most current documentation, including important updates that may have been made after the release of the product.

## Getting help

Hitachi Vantara Support Connect is the destination for technical support of products and solutions sold by Hitachi Vantara. To contact technical support, log on to Hitachi Vantara Support Connect for contact information: https://support.hitachivantara.com/en_us/contact-us.html.

Hitachi Vantara Community is a global online community for Hitachi Vantara customers, partners, independent software vendors, employees, and prospects. It is the destination to get answers, discover insights, and make connections. **Join the conversation today!** Go to community.hitachivantara.com, register, and complete your profile.

## Comments

Please send us your comments on this document to doc.comments@hitachivantara.com. Include the document title and number, including the revision level (for example, -07), and refer to specific sections and paragraphs whenever possible. All comments become the property of Hitachi Vantara LLC.

**Thank you!**

# Chapter 1: Introduction

## Overview

Hitachi Storage Modules for Red Hat® Ansible® allows IT and data center administrators to automate and manage the configuration of Hitachi storage systems.

With Hitachi Storage Modules for Red Hat® Ansible®, the administrator can create playbooks together with logic and the other ansible modules to automate complex tasks. Administrators can filter, sort and group the information by piping the output from one module to the other.

Hitachi Storage Modules for Red Hat® Ansible® tasks can be executed by running simple playbooks written in yaml syntax and are idempotent. For example, create LUN tasks can be used to create new LUN or update the LUN if already exists.

The storage modules are supported in environments with ansible open source community version (ansible.com/community) and Red Hat Ansible Engine versions.

## Main components

The following are the main components of the Hitachi Storage Modules for Red Hat Ansible:

### Hitachi API Gateway Service

This includes Hitachi Storage Ansible modules along with Hitachi API Gateway Service installed on the Linux VM. It communicates storage operation requests from the Ansible modules to the web service component.

### Hitachi Peer Service

The Hitachi Peer Service performs storage operations requested by the Ansible modules. It communicates with the Hitachi API Gateway Service using HTTPS.

# Chapter 2: System requirements

## Hardware requirements

| Hitachi Storage Device | Microcode version |
|---|---|
| Virtual Storage Platform F1500, VSP G1000, VSP G1500 | 80-06-75-00/00 |
| Virtual Storage Platform G200, VSP G400, VSP G600, VSP G800 | 83-05-34-x0/00 |
| Virtual Storage Platform F400, VSP F600, VSP F800 | 83-05-34-x0/00 |
| Virtual Storage Platform Gx00 with NAS Module (G400/ G600/ G800) | 83-05-34-x0/00 |
| Virtual Storage Platform G130, VSP G150, VSP G350, VSP G370, VSP G700, VSP G900 | 88-06-01 (SVOS 9.4.0) |
| Virtual Storage Platform F350, VSP F370, VSP F700, VSP F900 | 88-06-01 (SVOS 9.4.0) |
| Virtual Storage Platform 5100, 5500, 5100H, 5500H | 90-04-01_00/00 (SVOS 9.4.0 Q-code) |
| VSP E990 | 93-02-01-02_60-00 (SVOS 9.4.0 Q-code) |
| Hitachi NAS Platform | 13.5 |

## Port requirements

| Port | Protocol | Application / Storage |
|---|---|---|
| 22 | TCP | SSH |
| 8444 | HTTPS | Hitachi Peer Service |
| 2023 | HTTPS | Hitachi API Gateway Service |

# Software requirements

| Software | Supported version |
|---|---|
| Oracle Java | java-11-openjdk |
| Microsoft dotnet-sdk | 2.2.402 or higher |
| Python scp library | 0.13.2 or higher |
| Python requests library | 2.22 or higher |
| Ansible | 2.9 or higher |
| Extra Packages for Enterprise Linux (epel-release) | 7-11 or higher |
| Python pip | 19.3 or higher |

**Supported operating system**

CentOS 7.6, RHEL 7.6 and RHEL 7.7

**Note:** Hitachi recommends that you update OS with the latest software packages.

# Distribution package

The distribution package contains the following files:

- Installer file: `HV_Storage_Ansible-02.0.0.tar.gz`
- User Guide: `Storage_Modules_for_Red_Hat_Ansible_User_Guide_MK-92ADPTR149-00.pdf`
- Release Notes: `Storage_Modules_for_Red_Hat_Ansible_Release_Notes_RN-92ADPTR150-00.pdf`
- EULA: `hiAdapterLicense.pdf`

# Chapter 3: Installation and configuration

## Installing Hitachi Storage Modules for Red Hat Ansible

Install the Hitachi Storage Modules for Red Hat Ansible as follows:

### Procedure

1. Extract the following files from the distribution media kit (ISO) installation zip file: `HV_Storage_Ansible-02.0.0.tar.gz`
2. Upload `HV_Storage_Ansible-02.0.0.tar.gz` to the linux host that installs the Hitachi Storage Modules for Red Hat Ansible.
3. Run the following command in the linux server to extract the installation files: `tar -zxvf HV_Storage_Ansible-02.0.0.tar.gz`
   The following files are extracted to the `HV_Storage_Ansible` folder:
   - `install.sh`
   - `uninstall.sh`
   - `HV_Storage_Ansible-02.0.0-1.el7.x86_64.rpm`
4. During installation, three components are installed: Hitachi Storage Ansible Modules, Hitachi API Gateway Service and Hitachi Peer Service.
5. Installation can be done in one of three ways:
   a. Full Installation - Installs Hitachi Storage Ansible modules, Hitachi API Gateway Service, and Hitachi Peer Service.
   ```
   [root@localhost HV_Ansible]# source install.sh
   ```
   b. Hitachi Storage Ansible modules only - Installs Hitachi Storage Ansible modules only.
   ```
   [root@localhost HV_Ansible]# source install.sh --
   ansible_modules_only
   ```
   c. Hitachi Storage Gateway Server only - Installs Hitachi Peer Service only.
   ```
   [root@localhost HV_Ansible]# source install.sh --
   gateway_server_only
   ```
6. Post installation, Hitachi Storage Ansible modules are available in the following locations:
   a. Block modules: `/opt/hitachi/ansible/modules/block`
   b. File modules: `/opt/hitachi/ansible/modules/file`

7. Post installation, Hitachi Storage Ansible sample playbooks are available in the following locations:
   a. Block playbooks: `/opt/hitachi/ansible/playbooks/block`
   b. File playbooks: `/opt/hitachi/ansible/playbooks/file`

# Uninstallation

To remove the Hitachi Storage Modules for Red Hat Ansible from the CentOS host, follow these steps:

### Procedure

1. Open the command line console.
2. Navigate to the folder `<Installation_directory>/HV_Storage_Ansible` containing the `uninstall.sh` file that is extracted from `HV_Storage_Ansible-02.0.0.tar.gz`.
3. Execute the `./uninstall.sh` script to uninstall the service.

```
[root@localhost HV_Storage_Ansible]# ./uninstall.sh
Uninstalling Hitachi API Gateway Service...
Preparing uninstall...
Perform some post-tasks for uninstallation...
Uninstalling Hitachi Peer Service...
Preparing uninstall...
Perform some post-tasks for uninstallation...
```

4. Delete the remaining files from the following locations:
   - `<Installation_directory>/HV_Storage_Ansible`
   - `/opt/hitachi/ansible/`
   - `/var/log/hitachi/ansible/`

# Hitachi Storage Ansible modules

**To view the list of all available Hitachi Storage Ansible modules**

- Run the following command on Hitachi API Gateway Service VM:

```
ansible-doc -l | grep "hv"
```

**To view the documentation for a module**

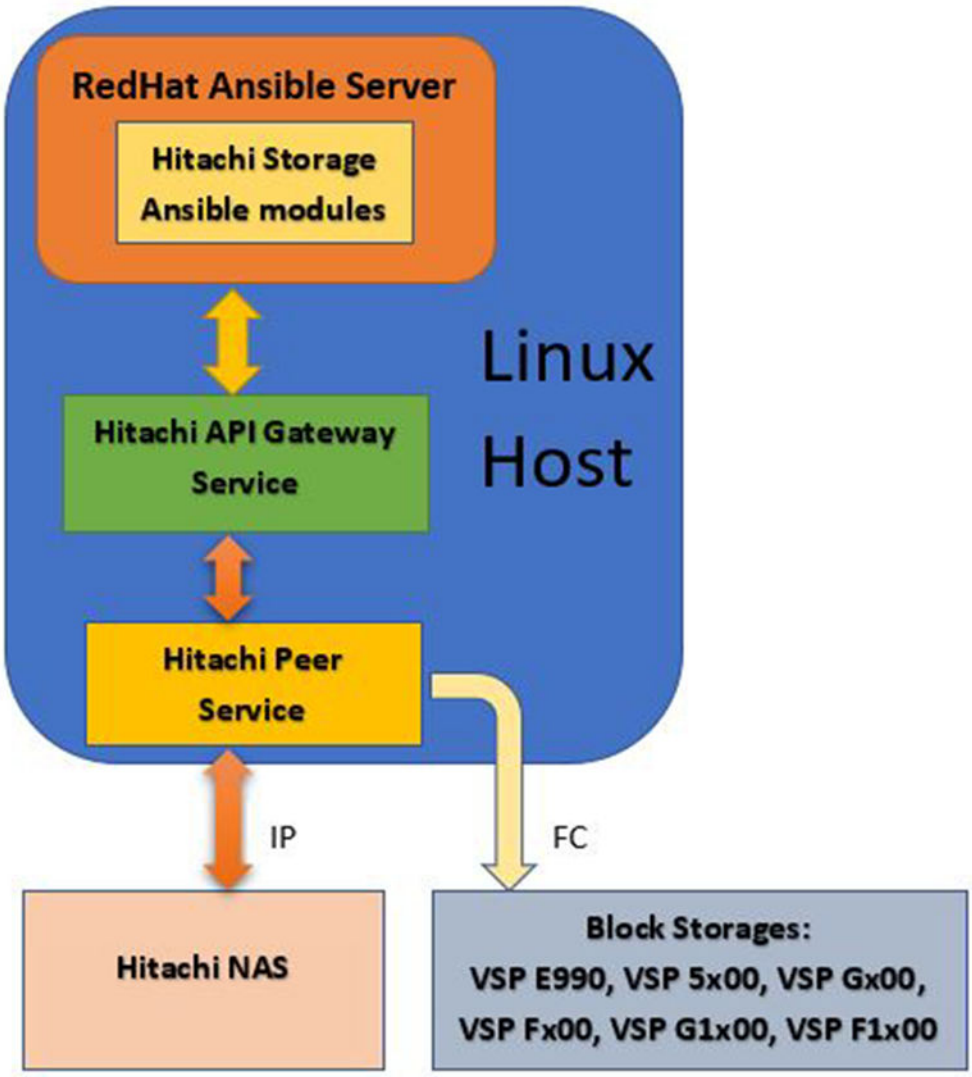
- Run the following command on Hitachi API Gateway Service VM:

```
ansible-doc <module-name>
```

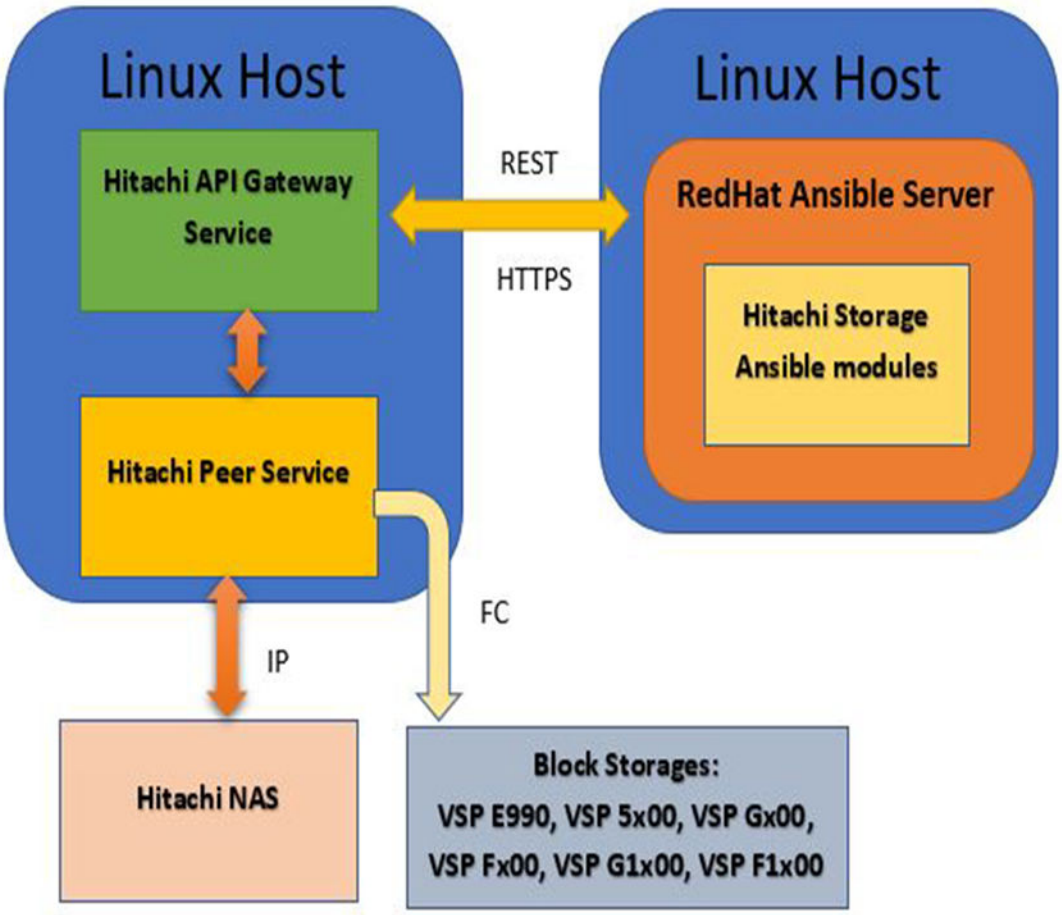
# Supported configurations

The following figures illustrate the supported configurations:

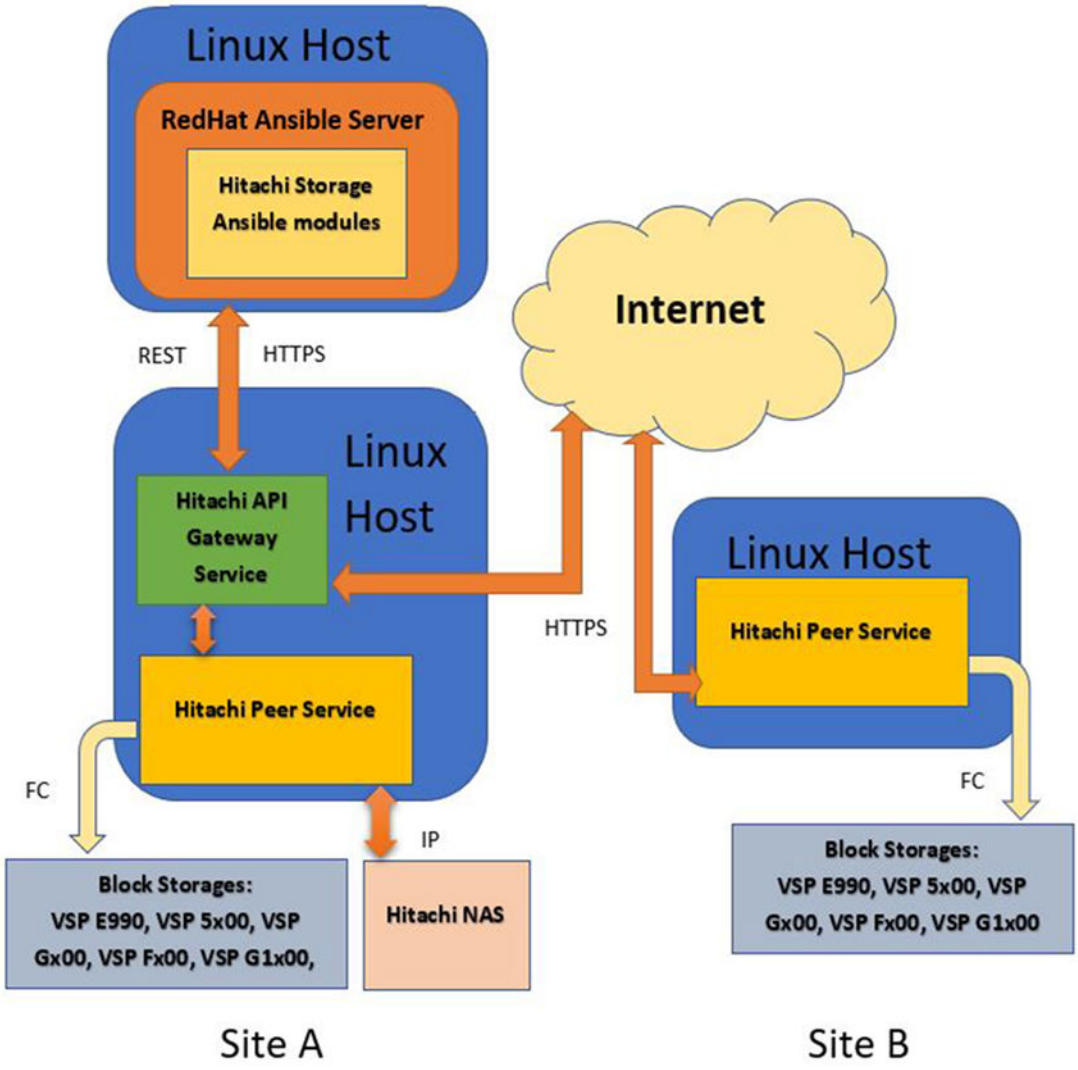
1) Hitachi Storage Ansible modules, Hitachi API Gateway Service, and Hitachi Peer Service in the same VM.




2) Hitachi Storage Ansible modules on one VM, and the Hitachi API Gateway Service and Hitachi Peer Service on another VM.




3) Hitachi Peer Service in multiple sites:




# List of sample playbooks

*** = Supported**

Block Storage sample playbooks

| Playbook name | Out-Of-Band Support |
|---|---|
| add_storagesystems.yml | * |
| clone_lun_in_dp_playbook.yml | |
| create_cmd_devs.yml | * |

| | | |
|---|---|---|
| create_gad_pair_vbox_multi.yml | | |
| create_gad_pair_vbox.yml | | |
| create_hg.yml | | * |
| create_hti_pair.yml | | |
| create_lun_parity.yml | | * |
| create_lun.yml | | * |
| create_multi_luns_with_count.yml | | * |
| create_multi_luns.yml | | |
| create_tc_pair.yml | | |
| create_ur_pair.yml | | |
| create_vsm.yml | | |
| delete_consistency_group.yml | | |
| delete_gad_pair.yml | | |
| delete_hg.yml | | * |
| delete_hti_pair.yml | | |
| delete_lun.yml | | * |
| delete_snapshot_group.yml | | |
| delete_tc_pair.yml | | |
| delete_ur_pair.yml | | |
| delete_vsm.yml | | |
| expand_lun.yml | | * |
| get_ctg_pair_group.yml | | |
| get_gad_pairs.yml | | |
| get_hg.yml | | * |
| get_hti_pairs.yml | | |
| get_lun_playbook.yml | | * |
| get_snapshot_group.yml | | |
| get_storagesystem.yml without any query | | * |
| get_storagesystem.yml with query | pools | * |

| | | |
|---|---|---|
| | ports | * |
| | quorumdisks | |
| | journalPool | |
| | nextFreeGADConsistencyGroupId | |
| | nextFreeHTIConsistencyGroupId | |
| | nextFreeTCConsistencyGroupId | |
| | nextFreeURConsistencyGroupId | |
| | freeLogicalUnitList | * |
| get_tc_pairs.yml | | |
| get_ur_pairs.yml | | |
| get_vsm.yml | | |
| present_lun.yml | | * |
| restore_hti_pair.yml | | |
| restore_snapshot_group.yml | | |
| resync_consistency_group.yml | | |
| resync_gad_pair.yml | | |
| resync_hti_pair.yml | | |
| resync_snapshot_group.yml | | |
| resync_tc_pair.yml | | |
| resync_ur_pair.yml | | |
| split_consistency_group.yml | | |
| split_gad_pair_multi.yml | | |
| split_gad_pair.yml | | |
| split_hti_pair.yml | | |
| split_snapshot_group.yml | | |
| split_tc_pair.yml | | |
| split_ur_pair.yml | | |
| unpresent_lun.yml | | * |

File Storage sample playbooks

| |
|---|
| create_nfs_share.yml |
| delete_nfs_share.yml |
| get_evs.yml |
| get_file_server.yml |
| get_file_systems.yml |
| get_nfs_shares.yml |

# Chapter 4: Storage registration and command device provisioning

Summary of steps

1. Add storage details to the `storage.json` storage connection configuration file.
2. Save credentials of all the storage systems in encrypted form in the `storage.json` file.
3. Optionally, create command devices for the storage systems in case of an in-band connection.
4. Register storage systems mentioned in the `storage.json` file.

## Storage connection configuration file

The storage connection configuration file is available in the location `/opt/hitachi/ansible/` post installation. To register any storage or perform command device addition operations, the `storage.json` configuration file needs to be updated.

This file includes configuration of the following devices:

1. Hitachi API Gateway Service
2. VMware vCenter® details for out-of-band command device creation
3. Optional Hitachi Peer API Gateway Service
4. Hitachi storage (block and file) system connection

**storage.json parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| vCenterIP | String | vCenter IP address | No | Required when the `hv_cmddev.py` module is called and "isPhysical" variable value is set to "False" in `create_cmd_devs.yml` file. |
| vCenterUser | String | vCenter user name | No | |
| vCenterVMName | String | VM name for mapping command device | No | |
| vCenterVMHostIpAddress | String | VMware ESXi™ host IP address where the VM is created | No | |
| vCenterCluster | String | vCenter cluster name | No | |
| hitachiAPIGatewayService | String | Linux VM IP address where the hitachiAPIGateway Service installation is performed | Yes | |
| hitachiAPIGatewayServicePort | Integer | Linux VM API Gateway Service port | Yes | Default: 2023 |
| useOutOfBandConnection | Boolean | If True, SAN storages mentioned in "sanStorageSystems" section would be added using Out-Of-Band command device.<br>If False, SAN storages would be added using InBand/FC command device | Yes | Default: False |
| nasStorageSystems | | | | |
| fileServerIP | String | HNAS FileServer IP address | Yes | |
| sanStorageSystems | | | | |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| useOutOfBand Connection | Boolean | If True, SAN storage would be added using Out-Of-Band command device.<br>If False, SAN storage would be added using InBand/FC command device | No | This `useOutOfBand Connection` is optional and overwrites the top level `useOutOfBand Connection` |
| serialNumber | Integer | Storage system serial number | Yes | |
| svpIP | String | Storage SVP IP address | Yes | |
| controllerIP | String | Storage controller IP address | No | Required for command device creation for Gx00/Fx00 storage systems |
| poolId | Integer | Dynamic pool ID | No | Dynamic pool where the command device is created. Default: Parity group with the most free space. |
| hitachiPeerServ ice | String | Peer Linux VM IP address where the hitachiAPIGateway Service is installed | No | If Hitachi storage command device is mapped to a VM other than the local hitachiAPIGate wayService VM |

# Credentials encryption

This section describes the process of passing user credentials for all the entities listed in the `storage.json` file.

The following entities in the `storage.json` file have a username and password that can be encrypted:

1. vCenter
2. Hitachi API Gateway Service Server
3. File Server
4. Block Storage

### Procedure

1. Update the `/opt/hitachi/ansible/storage.json` file with required IP addresses and then make a copy for your own backup.
2. Navigate to the `/opt/hitachi/ansible/bin` folder.
3. If all the SAN storages in the `storage.json` file have same user credentials, enter the following command to encrypt user credentials for all SAN storages at once:

   **addLoginToConfigurations -all_san_storages -user <storage_username> -password <storage_password> [-storage_profile <storage.json_file_path>] -display**

   For example:

   **addLoginToConfigurations -all_san_storages -user administrator -password Hitachi -storage_profile /opt/hitachi/ansible/storage.json -display**

   Output:

   ```
   [root@localhost bin]# addLoginToConfigurations -all_san_storages -user
   vmware -password Hitachi -storage_profile ../storage.json -display
   10.15.40.50 :
   5fd3c8e1dacdc6179c92e9dd947152cef19d21da038384e629bd087b74886d19e7cf25a
   c
   10.15.47.20 :
   d3b0659a8b7f4328f46755a46180685341570911f0ddd51132c4252cf5a01cd863eebbc
   8
   ```

4. If all the NAS storages in the `storage.json` file have the same user credentials, enter the following command to encrypt the user credentials for all NAS storages at once:

   **addLoginToConfigurations -all_nas_storages -user <storage_username> -password <storage_password> [-storage_profile <storage.json_file_path>] -display**

5. If all the SAN and NAS storages in the `storage.json` file have the same user credentials, enter the following command to encrypt the user credentials for all storages at once:

   **`addLoginToConfigurations -all_san_storages -all_nas_storages -user <storage_username> -password <storage_password> [-storage_profile <storage.json_file_path>] -display`**

6. For cases other than the ones mentioned above, add one entry for each ipaddress using the following command :

   **`addLoginToConfigurations -ip <SVP-IP> -user <storage_username> -password <storage_password> [-storage_profile <storage.json_file_path>] -display`**

7. For entities other than storages, add one entry for each ipaddress using the following command :

   **`addLoginToConfigurations -ip <vCenter-IP> -user <username> -password <password> [-storage_profile <storage.json_file_path>] -display`**

   **`addLoginToConfigurations -ip <Hitachi-API-Gateway-Service-Server> -user <username> -password <password>[-storage_profile <storage.json_file_path>] -display`**

8. If the `storage.json` file is in the current directory, then the `storage_profile` parameter is optional:

   **`addLoginToConfigurations -all_san_storages -user administrator -password Hitachi`**

   - The original `storage.json` file is modified with the above mentioned user details.
   - Navigate to the `/opt/hitachi/ansible/playbooks/block` folder and open the `add_storagesystems.yml` playbook in EDIT mode.
   - Update the path of the `storage.json` file in the `storage_profile` variable section as below:

     ```
     [root@localhost block]# cat add_storagesystems.yml
     - name: Adding Storage System from storage.json
     hosts: localhost
     gather_facts: false
     vars:
     - storage_profile: /opt/hitachi/ansible/storage.json
     tasks:
     - hv_storagesystem:
     storage_profile: '{{ storage_profile | default(omit) }}'
     register: result
     - debug: var=result
     ```

     ```
     The path can be absolute path or relative path from the current folder.
     ```

9. Now register the storage systems using the `add_storagesystems.yml` playbook:
   **`ansible-playbook add_storagesystems.yml`**

```
[root@localhost block]# ansible-playbook add_storagesystems.yml
[WARNING]: provided hosts list is empty, only localhost is available.
Note that the implicit localhost does not match 'all'
PLAY [Adding Storage System from storage.json]
**********************************************************************
*****************
TASK [hv_storagesystem]
**********************************************************************
**************************************
ok: [localhost]TASK [debug]
**********************************************************************
**************************************
ok: [localhost] => {
  "result": {
    "changed": false,
    "failed": false,
    "sessionIds": {
      "10.15.20.58": "087f5ee1-89b1-4287-a1f1-e1591da10be0"
    },
    "storageSystems": [
      444044
    ]
  }
}
PLAY RECAP
**********************************************************************
**************************************
localhost : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0
ignored=0
```

10. To encrypt another password for a different IP address, edit the `storage.json` file and run the **`addLoginToConfigurations`** command again.
11. If the path of the `storage.json` file in the `storage_profile` variable section is incorrect, the following exception is encountered:

```
[root@localhost block]# ansible-playbook add_storagesystems.yml
[WARNING]: provided hosts list is empty, only localhost is available.
Note that the implicit localhost does not match 'all'
PLAY [Adding Storage System from storage.json]
**********************************************************************
*****************
TASK [hv_storagesystem]
**********************************************************************
**************************************
fatal: [localhost]: FAILED! => {"changed": false, "msg": "The storage
profile ./storage.json does not exist.", "type": "Exception"}
PLAY RECAP
```

```
**********************************************************************
**************************************
localhost : ok=0 changed=0 unreachable=0 failed=1 skipped=0 rescued=0
ignored=0
```

12. To verify if all user names are set and passwords are encryted in the `storage.json` configuration file, run the following command in the `/opt/hitachi/ansible/bin` folder:

    **`[root@localhost bin]# checkLoginConfigurations.sh /opt/hitachi/ansible/storage.json`**

    After executing this, it prints to the console output the list of IP addresses and whether the user name is set or empty, and the password is encrypted, not-encrypted, or empty.

```
VCenter : ip=10.25.11.169 username=SET password=NOT-ENCRYPTED
StorageMgmtServer : ip=10.25.11.167 username=EMPTY password=EMPTY
SanStorageSystem : ip=10.25.11.168 username=SET password=ENCRYPTED
StorageGatewayServer : ip=10.25.11.166 username=SET password=ENCRYPTED
SanStorageSystem : ip=10.25.11.165 username=SET password=ENCRYPTED
StorageGatewayServer : ip=10.25.11.164 username=SET password=ENCRYPTED
NasStorageSystem : ip=10.25.11.163 username=SET password=ENCRYPTED
```

# Create command device

**Module: hv_cmddev**

Use the `hv_cmddev.py` module to create a command device and map it to the VM in the `storage.json` file.

This module is used to map a new SAN storage system command device from the SAN storage systems in the `storage.json` file to the VMware ESXi™ VM or RHEL physical server that hosts Hitachi Storage Modules for Red Hat Ansible (Hitachi API Gateway Service). This module does not require a command device as a prerequisite.

There is a boolean parameter `isPhysical` in the `create_cmd_devs.yml` file for mapping newly created command devices to either ESXi VM or the physical RHEL server.

If the parameter `isPhysical` value is set to "False", the module validates the vCenter details provided in the `storage.json` file. If the details are valid and the ESXi host is reachable, the HBA WWNs of the ESXi host are queried. After all the WWNs for the ESXi host are available, the WWN is searched in all the controller ports of the storage in the `storage.json` file. The result is based on the use cases listed in the following section.

Note: This module can be used to create a command device on one or more SAN storage systems mentioned in the `storage.json` file and the operation may take around 7 mins per SAN storage (mentioned in the `storage.json` file) for creating a command device and mapping it to the given VM.

For more info, run **`ansible-doc hv_cmddev`** to view the documentation for the module.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| isPhysical | Boolean | Physical RHEL host or ESXi virtual machine | No | Whether the command device is to be mapped to the ESXi virtual machine or to the physical RHEL host.<br>Default: False |
| cmdDevicePhysicalServerIp | String | Physical RHEL host IP address | No | Required if the `isPhysical` variable value is True<br>**Note:** For security purposes, remove the password after usage |
| cmdDevicePhysicalServerUser | String | Physical RHEL host login user | No | |
| cmdDevicePhysicalServerPassword | String | Physical RHEL host login password | No | |
| vCenterIP | String | vCenter IP address | No | vCenter IP address where the VM is available |
| vCenterUser | String | vCenter user name | No | vCenter user name |
| vCenterPassword | String | vCenter password | No | vCenter password |
| vCenterCluster | String | vCenter cluster name | No | vCenter cluster name |
| vCenterVMHostIpAddress | String | ESXi host IP address | No | ESXi host IP address hosted in the given vCenter |
| vCenterVMName | String | VM name | No | VM name available on the ESXi host |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| sanStorageSystems | Array | Storage details | Yes | Storage details on which the command device is to be created |
| hostPorts | String | Storage port where the host group should be created | No | Required when the ESXi host WWN is not found on the storage system |
| hostGroup | String | Host group name | No | Required when the host group for the ESXi host is not available on the storage system. Default: HITACHI_ANSIBLE_HG |
| hostMode | String | Host group host mode | No | Required when the host group for the ESXi host is not available on the storage system. Default: Linux |

**Use cases when the `isPhysical` variable is set to "False":**

**Use case 1**

If the WWN is available on any storage port, there is an additional search for the host group with that WWN on that storage port. If the host group exists, a command device is created with the name CMD-AUTO-CREATED and the size is 50 MB on the given HDP pool. After the LUN is created, the command device is mapped to the host group. The ESXi host HBA is rescanned for the new LUN. After it is available, the LUN is mapped to the VM mentioned in the `storage.json` file.

The Hitachi Peer Service needs to be restarted for the command device to work.

```
[root@localhost ~]# puma_adm -re
Restarting SDI Gateway.
```

The command device status can be verified using the following command:

```
[root@localhost ~]# puma_adm -ck
SDI Gateway is currently running on port 8444, using ip address 0.0.0.0.
● puma.service - SDI Gateway
   Loaded: loaded (/etc/systemd/system/puma.service; enabled; vendor
preset: disabled)
   Active: active (running) since Fri 2019-12-20 13:45:16 EST; 16s ago
  Process: 3071 ExecStart=/opt/hitachi/puma/puma start (code=exited,
status=0/SUCCESS)
 Main PID: 5149 (RESTServer)
   CGroup: /system.slice/puma.service
           ├─3084 /opt/hitachi/puma/utils/db/redis-server 127.0.0.1:6379
           ├─5130 horcmd_0560
           └─5149 /opt/hitachi/puma/RESTServer --daemon --umask=127 --
pidfile=/var/run/puma.pid

Dec 20 13:45:11 localhost.localdomain systemd[1]: Starting SDI Gateway...
Dec 20 13:45:16 localhost.localdomain puma[3071]: Starting puma:
Dec 20 13:45:16 localhost.localdomain systemd[1]: Started SDI Gateway.

SDI Gateway RPM Install Check           [ PASS ]
SDI Gateway Process Check               [ PASS ]
SDI Gateway cURL Check                  [ PASS ]

All command devices that exist on this machine:

        LDEV ID   Device   Storage Serial

          290    /dev/sdb    415022

Command devices used by SDI Gateway:

        Device: /dev/sdb
        LDEV ID: 290
        Storage Array Serial Number: 415022
        SDI Gateway HORCM Instance Number: 560
        HORCM Process State: Running

System disk space status:
1K-blocks    Used   Avail Use% File
 14034944 4247092 9787852  31% /
```

**Use case 2**

If the WWN is available on any storage port, there is an additional search for the host group with that WWN on that storage port. If the host group is not available/created, the host group name for the new host group is searched in the `storage.json` file. If the host group name is available, the host group is created with the given host group name and the WWN is added to the new host group. A command device is created with name CMD-AUTO-CREATED and the size is 50 MB on the given HDP pool. After the LUN is created, the command device is mapped to the new host group. Because this is the first LUN on the ESXi host, the new LUN must be manually discovered on the ESXi host and mapped to the VM.

**Use case 3**

If the WWN is available on any storage port, there is an additional search for the host group with that WWN on that storage port. If the host group is not available or not created, the host group name for the new host group is searched in the `storage.json` file. If the host group name is not available, the host group is created with the default name HITACHI_ANSIBLE_HG and the WWN is added to the new host group. A command device is created with the name CMD-AUTO-CREATED and the size is 50 MB on the given Dynamic Provisioning pool. After the LUN is created, the command device is mapped to the new host group. Because this is the first LUN on the ESXi host, the new LUN must be manually discovered on the ESXi host and mapped to the VM.

**Use case 4**

If the WWN is not available on any of the storage ports, the host group details (host group name, host mode, and storage port) for creating the new host group are queried in the `storage.json` file. If the details are available, the host group is created with the given information and the WWN is added to the new host group. A command device is created with the name CMD-AUTO-CREATED and the size is 50 MB on the given Dynamic Provisioning pool. After the LUN is created, the command device is mapped to the new host group. Because this is the first LUN on the ESXi host, the new LUN needs to be manually discovered on the ESXi host and mapped to the VM.

If the host group details are not available in `storage.json`, the operation is cancelled.

**Use case when the `isPhysical` variable is set to "True"**

If the `isPhysical` parameter value is set to "True", the physical RHEL server details need to be added in `create_cmd_devs.yml` file. The physical RHEL server should have Hitachi Peer Service installed prior to running this operation. After the operation is initiated, the HBA WWNs of the physical host are queried. After all the WWNs for the physical host are available, the WWN is searched in all the controller ports of the storage in the `storage.json` file. After the hostgroup is found, a command device is created with the name CMD-AUTO-CREATED and the size is 50 MB on the given Dynamic Provisioning pool. After the LUN is created, the command device is mapped to the host group. The RHEL host HBA is rescanned for the new LUN.

The Hitachi Peer Service needs to be restarted for the command device to work.

```
[root@localhost ~]# puma_adm -re
```

**Output**

```
[root@localhost playbooks]# ansible-playbook create_cmd_devs.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [create storage command device for each of the storage systems
defined in storages.json file.]
*********************************************************************
*****************************

TASK [hv_cmddev]
*********************************************************************
*********************************************************************
************************************
ok: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
****************************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "cmd_output": [
            [
                "Creating an in-band command device using the following
out-of-band command device:",
                "HORCM Configuration File: /HORCM/configs/horcm1992.conf",
                "HORCM Instance Number: 1992",
                "Storage Array IP/Hostname: 172.25.47.58",
                "This HORCM instance is stopped.",
                "Serial Number: 310076",
                "Checking initiators...",
                "Checking pools...",
                "Checking target port CL2-A...",
                "Finding a hostgroup that already contains initiators in
port CL2-A...",
                "Found a hostgroup that contains the intitiators (tng-dev-
esxi-20.77)...",
                "Getting command device name...",
                "Getting available LUN...",
                "Creating LUN 740 and adding to:",
                "Hostgroup tng-dev-esxi-20.77 on port CL2-A",
                "Creating LU...",
                "Formatting LU...",
                "Making LU a command device...",
                "Adding command device to hostgroup...",
                "",
                "In-band command device creation successful:",
```

```
                "",
                "LUN (decimal): 740",
                "LUN (hexadecimal): 02E4",
                "Host Group: tng-dev-esxi-20.77 ",
                "Port: CL2-A ",
                "Storage Array IP/Hostname: 172.25.47.58",
                "DP Pool ID: 28",
                "",
                "Please rescan the HBA to use this command device.",
                ""
            ]
        ],
        "failed": false,
        "serials": [
            10076
        ]
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*****************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

The logs for the `hv_cmddev.py` module can be found in the following location: `/var/log/hitachi/ansible/hv_storage_modules.log`

# Register storage systems

**Module: hv_storagesystem**

The `hv_storagesystem.py` module registers storage systems listed in the `storage.json` file (available in `/opt/hitachi/ansible/`), into a Red Hat Ansible session.

For more info, run **`ansible-doc hv_storagesystem`** to view the documentation for the module.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage.json | json file | JSON file that contains connection information of the physical storage systems and the Hitachi Peer Service. | Yes | |

**Output**

```
[root@172 block]# ansible-playbook add_storagesystems.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Adding Storage System from storage.json]
*********************************************************************
*********************************************************************
*************************************

TASK [hv_storagesystem]
*********************************************************************
*********************************************************************
************************************************************
ok: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "details": {
            "444044": {
                "APIGatewayServiceIP": "10.15.20.12",
                "APIGatewayServicePort": "2023",
                "Controller0": "",
                "Controller1": "",
                "FreeCapacityInMB": 86039867,
                "IsHUVMCapable": true,
                "IsUnifiedStorage": false,
                "IsVirtual": false,
                "MgmtIPAddress": "10.15.25.25",
                "MicroCodeVersion": "80-06-73-00/00",
                "Name": "VSP G1500 (444044)",
                "PeerServiceIP": "10.15.20.12",
                "SerialNumber": 444044,
                "StorageDeviceModel": "VSP_G1500",
                "StorageDeviceType": "VSP_G1500",
                "TotalCapacityInMB": 105761816,
                "Vendor": "HITACHI"
            },
    "failed": false,
    "sessionIds": {
      "10.15.20.12": "545b5323-7b89-4653-a1ef-fd3fb36eab8d"
    },
    "storageSystems": [
      444044
    ]
```

```
  }
}
```

If the Hitachi Peer Service VM is different than the Hitachi API Gateway Service VM, you see the following error when adding storage:

```
TASK [hv_storagesystem]
*********************************************************************************
***********************************
fatal: [localhost]: FAILED! => {"changed": false, "msg": {"ErrorCode": 0,
"ErrorMessage": "Error occurred while registering RAID Subsystem. One or
more errors occurred. (An error occurred while sending the request.)",
"MessageID": "", "OriginalExceptionType": "HiException"}, "type":
"Exception"}
```

To correct this error, run **`generate-remote-gw-cert.sh`** (available in `<extracted dir>/HV_Storage_Ansible/`) with the remote Hitachi Peer Service VM IP as shown:

```
[root@localhost HV_Ansible]# ./generate-remote-gw-cert.sh 10.15.59.13
depth=0 C = US, ST = CA, L = Santa Clara, O = Hitachi Vantara, OU = SIE,
CN = 10.15.59.13
verify return:1
depth=0 C = US, ST = CA, L = Santa Clara, O = Hitachi Vantara, OU = SIE,
CN = 10.15.59.13
verify return:1

And then run add_storagesystems.yml again
```

# Chapter 5: Block storage modules

The block storage modules are located in the `/opt/hitachi/ansible/modules/block` directory.

## Provide storage system information

**Module: hv_storagesystem_facts**

The `hv_storagesystem_facts.py` module provides information about the added Hitachi storage system.

For more info, run **`ansible-doc hv_storagesystem_facts`** to view the documentation for the module.

### Get storage system

**Module: hv_storagesystem_facts**

The `get_storagesystem.yml` playbook provides information about the storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| query | string | Optional input parameter<br>List of string value with valid input value:<br>▪ pools<br>▪ ports<br>▪ quorumdisks<br>▪ journalPool<br>▪ nextFreeGADConsistencyGroupId<br>▪ nextFreeHTIConsistencyGroupId<br>▪ nextFreeTCConsistencyGroupId | No | |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| | | ▪ nextFreeURConsistencyGroupId<br>▪ freeLogicalUnitList | | |

**Example: Testing Get storage system**

```
name: Testing Get Storage System
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 415056
    - query:
      - nextFreeHTIConsistencyGroupId
      - quorumdisks
  tasks:
    - hv_storagesystem_facts:
        storage_serial: '{{ storage_serial }}'
        query: '{{ query | default(omit) }}'
      register: result
    - debug: var=result.storageSystem
```

**Output**

```
[root@localhost playbooks]# ap get_storagesystem.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [Testing Get Storage System]
**********************************************************************
**********************************************************************
**********************

TASK [hv_storagesystem_facts]
**********************************************************************
**********************************************************************
*************************
ok: [localhost]

TASK [debug]
**********************************************************************
**********************************************************************
*****************************************
ok: [localhost] => {
    "result.storageSystem": {
        "DeviceID": "415056",
        "DeviceType": "VSP_GX00",
```

```
        "MgmtIpAddress": "172.25.43.250",
        "Name": "VSP_G700 (415056)",
        "QuorumDisks": [
            {
                "DeviceId": 415022,
                "DeviceType": 0,
                "LogicalUnitId": -1,
                "QuorumDiskId": 1,
                "Status": 0,
                "Timeout": 40
            },
            {
                "DeviceId": 415022,
                "DeviceType": 0,
                "LogicalUnitId": 416,
                "QuorumDiskId": 2,
                "Status": 0,
                "Timeout": 40
            }
        ],
        "SerialNumber": 415056,
        "StorageDeviceModel": 8,
        "StorageDeviceType": "VSP_GX00",
        "Vendor": "HITACHI",
        "nextFreeHTIConsistencyGroupId": 43
    }
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*****************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Manage host groups

**Module: hv_hg**

The `hv_hg.py` module manages host groups on Hitachi storage systems.

For more info, run **`ansible-doc hv_hg`** and **`ansible-doc hv_hg_facts`** to view the documentation for the module.

## Get host group

**Module: hv_hg_facts**

The `get_hg.yml` playbook provides host group information about the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | storage serial number | Yes | If only the storage system serial number is given as an input parameter, then all host group information is retrieved according to the storage system serial number. |
| hgName | string | Host group name | No | Optional. If specified, returns host groups with the input name. |
| ports | string | Storage Fibre Channel port | No | Optional. If specified, returns all host groups of the input Fibre Channel ports. |
| lun | integer | LDEV ID in decimal or HEX | No | Optional. If specified, returns all host groups of the input LUN. |
| query | string | Valid values:<br>▪ wwns<br>▪ luns | No | Optional. If specified, returns host WWNs and LUNs mapped to the hostgroup |

**Example: Testing get host group**

```
name: Testing Get Host Group
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - host_group_name: testhg23
    - ports:
      - CL1-A
    - query:
      - luns
      - wwns
  tasks:
    - hv_hg_facts:
        storage_serial: '{{ storage_serial }}'
```

```
        data:
          query: '{{ query | default(None) }}'
          lun: '{{ lun | default(None) }}'
          name: '{{ host_group_name | default(None) }}'
          ports: '{{ ports | default(None) }}'
      register: result
    - debug: var=result.hostGroups
```

**Output**

```
[root@localhost block]# ansible-playbook get_hg.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Get Host Group]
*********************************************************************
*********************************************************************
*************************************************

TASK [hv_hg_facts]
*********************************************************************
*********************************************************************
*****************************************************************
ok: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result.hostGroups": [
        {
            "HgName": "testhg23",
            "HostMode": "VMWARE_EXTENSION",
            "HostModeOptions": [
                68
            ],
            "HostgroupID": 105,
            "Paths": [
                {
                    "hexLdevId": "00:05:92",
                    "hostGroupLunId": 0,
                    "ldevId": 1426
                }
            ],
            "Port": "CL1-A",
            "ResourceGroupId": 0,
            "WWNS": [
                "7890790000000019"
            ]
```

```
        }
    ]
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*********************************************************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create host group

**Module: hv_hg**

The `create_hg.yml` playbook creates a host group in the provided storage system.

**Note:** This module can also be used for updating the host group. It will update the host group, if the same already exists in the storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set the state to present under module for creating and updating a host group.<br>Set the state to present under module and absent under vars to remove items from host group.<br>Set the state to absent under module for deleting a host group. The LUNs should be unmapped if any before deleting the host group. |
| host_group_name | string | Name of the host group | Yes | N/A |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| host_mode | string | Host mode of the host group | No | Host mode string value. The default value is “LINUX”. Valid input values:<br>▪ LINUX<br>▪ VMWARE<br>▪ HP<br>▪ OPEN_VMS<br>▪ TRU64<br>▪ SOLARIS<br>▪ NETWARE<br>▪ WINDOWS<br>▪ HI_UX<br>▪ AIX<br>▪ VMWARE_EXTENSION<br>▪ WINDOWS_EXTENSION<br>▪ UVM<br>▪ HP_XP<br>▪ DYNIX |
| host_mode_options | string | Host mode options of the host group | No | The following host mode options are supported:<br>▪ 0 [RESERVED]<br>▪ 2 [VERITAS_DB_EDITION_ADV_CLUSTER]<br>▪ 6 [TPRLO]<br>▪ 7 [AUTO_LUN_RECOGNITION<br>▪ 12 [NO_DISPLAY_FOR_GHOST_LUN] |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| | | | | ▪ 13 [SIM_REPORT_AT_LINK_FAILURE]<br>▪ 14 [HP_TRUECLUSTER_WITH_TRUECOPY]<br>▪ 15 [RAID_HACMP]<br>▪ 22 [VERITAS_CLUSTER_SERVER]<br>▪ 23 [REC_COMMAND_SUPPORT]<br>▪ 33 [SET_REPORT_DEVICE_ID_ENABLE]<br>▪ 39 [CHANGE_NEXUS_SPECIFIED_IN_SCSI_TARGET_RESET]<br>▪ 40 [VVOL_EXPANSION]<br>▪ 41 [PRIORITIZED_DEVICE_RECOGNITION]<br>▪ 42 [PREVENT_OHUB_PCI_RETRY]<br>▪ 43 [QUEUE_FULL_RESPONSE]<br>▪ 48 [HAM_SVOL_READ]<br>▪ 49 [BB_CREDIT_SETUP_1]<br>▪ 50 [BB_CREDIT_SETUP_2] |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| | | | | ▪ 51 [ROUND_TRIP_SETUP]<br>▪ 52 [HAM_AND_CLUSTER_SW_FOR_SCSI_2]<br>▪ 54 [EXTENDED_COPY]<br>▪ 57 [HAM_RESPONSE_CHANGE]<br>▪ 60 [LUN0_CHANGE_GUARD]<br>▪ 61 [EXPANDED_PERSISTENT_RESERVE_KEY]<br>▪ 63 [VSTORAGE_APIS_ON_T10_STANDARDS]<br>▪ 65 [ROUND_TRIP_EXTENDED_SETUP]<br>▪ 67 [CHANGE_OF_ED_TOV_VALUE]<br>▪ 68 [PAGE_RECLAMATION_LINUX]<br>▪ 69 [ONLINE_LUSE_EXPANSION]<br>▪ 71 [CHANGE_UNIT_ATTENTION_FOR_BLOCKED_POOL_VOLS]<br>▪ 72 [AIX_GPFS]<br>▪ 73 [WS2012]<br>▪ 78 [NON_PREFERRED_PATH] |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| | | | | ▪ 95 [CHANGE_SCSI_LU_RESET_NEXUS_VSP_HUS_VM]<br>▪ 96 [CHANGE_SCSI_LU_RESET_NEXUS]<br>▪ 97 [PROPRIETARY_ANCHOR_COMMAND_SUPPORT]<br>▪ 100 [HITACHI_HBA_EMULATION_CONNECTION_OPTION]<br>▪ 102 [GAD_STANDARD_INQURY_EXPANSION_HCS]<br>▪ 105 [TASK_SET_FULL_RESPONSE_FOR_IO_OVERLOAD] |
| wwns | array | List of host WWNs | No | N/A |
| ports | array | List of Fibre Channel ports | Yes | N/A |
| luns | array | List of LDEV IDs in decimal or HEX | No | N/A |

**Example: Creating a host group**

```
name: Create Host Group
  hosts: localhost
  gather_facts: false
  vars:
    - host_group_name: testhg23
    - ports:
      - CL1-A
    - host_mode: VMWARE_EXTENSION
```

```
    - host_mode_options:
      - 68
    - storage_serial: 123456
    - wwns:
      - 7890790000000019
    - luns:
      - 1426
  tasks:
    - hv_hg:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          name: '{{ host_group_name }}'
          ports: '{{ ports }}'
          host_mode: '{{ host_mode | default(None)}}'
          host_mode_options: '{{ host_mode_options | default(None)}}'
          wwns: '{{ wwns | default(None) }}'
          luns: '{{ luns | default(None) }}'
          state: '{{ state | default(None) }}'
      register: result
    - debug: var=result.hostGroups
```

**Output**

```
[root@localhost block]# ansible-playbook create_hg.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Create Host Group]
*********************************************************************
*********************************************************************
*************************************************

TASK [hv_hg]
*********************************************************************
*********************************************************************
*****************************************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*****************************************************************
ok: [localhost] => {
    "result.hostGroups": [
        {
            "HgName": "testhg23",
            "HostMode": "VMWARE_EXTENSION",
            "HostModeOptions": [
                68
```

```
            ],
            "HostgroupID": 105,
            "Paths": [
                {
                    "hexldevId": "00:05:92",
                    "hostGroupLunId": 0,
                    "ldevId": 1426
                }
            ],
            "Port": "CL1-A",
            "ResourceGroupId": 0,
            "WWNS": [
                "7890790000000019"
            ]
        }
    ]
}

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example: Testing Creating a host group when LUN ID given in Hexadecimal form**

```
name: Create Host Group
hosts: localhost
gather_facts: false
vars:
  - host_group_name: testhg18
  - ports:
  - CL1-A
  - host_mode: VMWARE_EXTENSION
  - host_mode_options:
  - 63
  - storage_serial: 123456
  - luns:
  - 00:05:B3
tasks:
  - hv_hg:
  state: present
  storage_serial: '{{ storage_serial }}'
  data:
  name: '{{ host_group_name }}'
  ports: '{{ ports }}'
  host_mode: '{{ host_mode | default(None)}}'
  host_mode_options: '{{ host_mode_options | default(None)}}'
  wwns: '{{ wwns | default(None) }}'
```

```
  luns: '{{ luns | default(None) }}'
  state: '{{ state | default(None) }}'
register: result
  - debug: var=result.hostGroups
```

**Output**

```
[root@localhost block]# ansible-playbook create_hg.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
PLAY [Create Host Group]
*********************************************************************
Create host group
Chapter 5: Block storage modules
Hitachi Storage Modules for Red Hat Ansible User Guide 50
*********************************************************************
***********************************
TASK [hv_hg]
*********************************************************************
*********************************************************************
*********************************************
changed: [localhost]
TASK [debug]
*********************************************************************
*********************************************************************
*********************************************
ok: [localhost] => {
"result.hostGroups": [
{
  "HgName": "testhg18",
  "HostMode": "VMWARE_EXTENSION",
  "HostModeOptions": [
    63
  ],
  "HostgroupID": 91,
  "Paths": [
  {
    "hexldevId": "00:05:B3",
    "hostGroupLunId": 0,
    "ldevId": 1459
  }
  ],
  "Port": "CL1-A",
  "ResourceGroupId": 0,
  "WWNS": []
  }
 ]
}
PLAY RECAP
*********************************************************************
```

```
*******************************************************************************
************************************************
localhost : ok=2 changed=1 unreachable=0
failed=0 skipped=0 rescued=0 ignored=0
```

## Delete host group

**Module: hv_hg**

The `delete_hg.yml` playbook deletes a host group in the provided storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | default: absent | Yes | Set the state to absent for deleting a host group. |
| host_group_name | string | Name of the host group to be deleted | Yes | N/A |
| ports | array | controller ports | No | One or more controller ports from where the host group needs to be deleted.<br>If not specified, the host group will be deleted from all applicable ports if there is no LUN path available in any of the ports. |

**Example: Testing delete host group**

```
name: Testing Delete Host Group
  hosts: localhost
  gather_facts: false
  vars:
```

```
    - host_group_name: testhg23
    - ports:
      - CL1-A
    - storage_serial: 123456
  tasks:
    - hv_hg:
        state: absent
        storage_serial: '{{ storage_serial }}'
        data:
          name: '{{ host_group_name }}'
          ports: '{{ ports | default(None) }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook delete_hg.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Delete Host Group]
*********************************************************************
*********************************************************************
*************************************************

TASK [hv_hg]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false,
        "hostGroups": [
            {
                "HgName": "testhg23",
                "HostMode": "VMWARE_EXTENSION",
                "HostModeOptions": [
                    63
                ],
                "HostgroupID": 105,
                "Paths": [],
                "Port": "CL1-A",
                "ResourceGroupId": 0,
```

```
                "WWNS": []
            }
        ]
    }
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*********************************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Present LUN

**Module: hv_hg**

The `present_lun.yml` playbook presents LUNs to the host group on the provided storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| host_group_name | string | Name of the host group present on the storage | Yes | N/A |
| ports | array | List of Fibre Channel ports | No | N/A |
| luns | array | LUN ID in decimal or HEX of the LUN that you want to present | Yes | N/A |
| state | string | For module:- default: present<br>For data:- the value can be present or absent.<br>default : present | Yes | If this parameter is set to absent in data, it will unpresent the LUN from the host group. |

**Testing Present LUN**

```
name: Testing Present LUN
  hosts: localhost
  gather_facts: false
  vars:
```

```
    - host_group_name: testhg23
    - luns:
      - 1426
    - storage_serial: 123456
  tasks:
    - hv_hg:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          state: '{{ state | default(None) }}'
          name: '{{ host_group_name }}'
          ports: '{{ ports | default(None) }}'
          luns: '{{ luns | default(None) }}'
      register: result
```

**Output**

```
[root@localhost block]# ansible-playbook present_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Present LUN]
*********************************************************************
*********************************************************************
*************************************************

TASK [hv_hg]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost]

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=1    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Unpresent LUN

**Module: hv_hg**

The `unpresent_lun.yml` playbook unpresents a LUN from the host group on the provided storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | For module:- default: present<br>For data:- the value can be present or absent.<br>default : absent | Yes | If this parameter is set to present in data, it will present the LUNs to the host group. |
| host_group_name | string | Name of the host group present on the storage | Yes | N/A |
| ports | array | List of Fibre Channel ports | No | N/A |
| luns | array | LUN ID in decimal or HEX of the LUN that you want to unpresent | Yes | N/A |

**Example: Testing Upresent LUN**

```
name: Testing Unpresent LUN
  hosts: localhost
  gather_facts: false
  vars:
    - host_group_name: testhg23
    - luns:
      - 1426
    - storage_serial: 123456
    - state: absent
  tasks:
    - hv_hg:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          state: '{{ state | default(None) }}'
          name: '{{ host_group_name }}'
          ports: '{{ ports | default(None) }}'
```

```
          luns: '{{ luns }}'
      register: result
```

**Output**

```
[root@localhost block]# ansible-playbook unpresent_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Unpresent LUN]
*********************************************************************
*********************************************************************
*****************************************************

TASK [hv_hg]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost]

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=1    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Manage LUN

**Module: hv_lun**

This module manages LUN on Hitachi Storage System.

For more info, run ***ansible-doc hv_lun*** and ***ansible-doc hv_lun_facts*** to view the documentation for the module.

## Get LUN

**Module: hv_lun_facts**

The `get_lun_playbook.yml` playbook provides LUN information on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | If the storage system serial number parameter is given, it returns all LUN information for that storage system. |
| lun | integer | LDEV ID in decimal or HEX in storage or ESXi host LUN ID in canonical form | No | If the LDEV ID is provided in decimal or HEX format, it returns information for the specified LUN on the given storage serial.<br>If the ESXi host LUN ID is given in canonical form, the LUN value must be prefixed with 'naa'. Storage serial would be ignored and LUN details would be retrieved from any storage added to the current session. |
| name | string | LUN name | No | N/A |
| max_count | integer | Number of luns to be displayed | No | This works along with the variable `lun`. When specified, the next `max_count` number of LUNs are displayed starting from the LDEV ID specified by the variable `lun`. |

**Example 1: Testing Get LUN with LUN ID**

```
name: Testing Get LUN
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - lun: 300
    - max_count: 1
  tasks:
    - hv_lun_facts:
        storage_serial: '{{ storage_serial | default(None)}}'
        data:
          lun: '{{ lun | default(None) }}'
          name: '{{ name | default(None) }}'
          max_count: '{{ max_count | default(None) }}'
      register: result
    - debug: var=result.luns
```

**Output**

```
[root@localhost block]# ansible-playbook get_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Get LUN]
*********************************************************************
*********************************************************************
*************************************************************

TASK [hv_lun_facts]
*********************************************************************
*********************************************************************
*************************************************************
ok: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*************************************************************
ok: [localhost] => {
    "result.luns": [
        {
            "CanonicalName": "naa.60060e80123ad00050403ad00000012c",
            "DedupCompressionMode": "DISABLED",
            "DedupCompressionProgress": "",
            "DedupCompressionStatus": "DISABLED",
            "EmulationType": "OPEN-V-CVS",
            "FormatOrShredRate": 100,
            "HexLunNumber": "00:01:2C",
            "IsALU": false,
            "IsALUA": false,
            "IsBusy": false,
            "IsCommandDevice": false,
            "IsDMLU": false,
            "IsDuplicated": false,
            "IsDynamicPoolVolume": false,
            "IsInGADPair": false,
            "IsJournalPoolVolume": false,
            "IsSLU": false,
            "IsVVOL": false,
            "Lun": 300,
            "MetaResourceSerialNumber": 123456,
            "Name": "",
            "ParityGroup": "",
            "PathCount": 1,
            "PhysicalLunNumber": 300,
            "PoolId": "5",
            "PoolType": "HDP",
            "RG": 0,
```

```
            "RaidLevel": 0,
            "ResourceGroupId": 0,
            "Status": "NORMAL",
            "TotalCapacity": "3.0GB",
            "UsedCapacity": "798.0MB",
            "VirtualLun": 300,
            "VirtualStorageDeviceId": 123456,
            "stripeSize": 512
        }
    ]
}

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example 2: Testing Get LUN when ESXi host LUN ID is given in canonical form**

```
name: Testing Get LUN
  hosts: localhost
  gather_facts: false
  vars:
    - lun: naa.60060e80123ad00050403ad00000012c
    - max_count: 1
  tasks:
    - hv_lun_facts:
        storage_serial: '{{ storage_serial | default(None)}}'
        data:
          lun: '{{ lun | default(None) }}'
          name: '{{ name | default(None) }}'
          max_count: '{{ max_count | default(None) }}'
      register: result
    - debug: var=result.luns
```

**Output**

```
[root@localhost block]# ansible-playbook get_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Get LUN]
*********************************************************************
*********************************************************************
*************************************************************

TASK [hv_lun_facts]
*********************************************************************
```

```
*******************************************************************************
************************************************************
ok: [localhost]

TASK [debug]
*******************************************************************************
*******************************************************************************
*********************************************************************
ok: [localhost] => {
    "result.luns": [
        {
            "CanonicalName": "naa.60060e80123ad00050403ad00000012c",
            "DedupCompressionMode": "DISABLED",
            "DedupCompressionProgress": "",
            "DedupCompressionStatus": "DISABLED",
            "EmulationType": "OPEN-V-CVS",
            "FormatOrShredRate": 100,
            "HexLunNumber": "00:01:2C",
            "IsALU": false,
            "IsALUA": false,
            "IsBusy": false,
            "IsCommandDevice": false,
            "IsDMLU": false,
            "IsDuplicated": false,
            "IsDynamicPoolVolume": false,
            "IsInGADPair": false,
            "IsJournalPoolVolume": false,
            "IsSLU": false,
            "IsVVOL": false,
            "Lun": 300,
            "MetaResourceSerialNumber": 123456,
            "Name": "",
            "ParityGroup": "",
            "PathCount": 1,
            "PhysicalLunNumber": 300,
            "PoolId": "5",
            "PoolType": "HDP",
            "RG": 0,
            "RaidLevel": 0,
            "ResourceGroupId": 0,
            "Status": "NORMAL",
            "TotalCapacity": "3.0GB",
            "UsedCapacity": "798.0MB",
            "VirtualLun": 300,
            "VirtualStorageDeviceId": 123456,
            "stripeSize": 512
        }
    ]
}

PLAY RECAP
```

```
**********************************************************************
**********************************************************************
**********************************************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example 3: Testing Get LUN with max_count parameter**

```
name: Testing Get LUN
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - lun: 00:01:2C
    - max_count: 2
  tasks:
    - hv_lun_facts:
        storage_serial: '{{ storage_serial | default(None) }}'
        data:
          lun: '{{ lun | default(None) }}'
          name: '{{ name | default(None) }}'
          max_count: '{{ max_count | default(None) }}'
      register: result
    - debug: var=result.luns
```

**Output**

```
[root@localhost block]# ansible-playbook get_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Get LUN]
**********************************************************************
**********************************************************************
*****************************************************

TASK [hv_lun_facts]
**********************************************************************
**********************************************************************
*****************************************************
ok: [localhost]

TASK [debug]
**********************************************************************
**********************************************************************
**********************************************************************
ok: [localhost] => {
    "result.luns": [
        {
            "CanonicalName": "naa.60060e80123ad00050403ad00000012c",
```

```
            "DedupCompressionMode": "DISABLED",
            "DedupCompressionProgress": "",
            "DedupCompressionStatus": "DISABLED",
            "EmulationType": "OPEN-V-CVS",
            "FormatOrShredRate": 100,
            "HexLunNumber": "00:01:2C",
            "IsALU": false,
            "IsALUA": false,
            "IsBusy": false,
            "IsCommandDevice": false,
            "IsDMLU": false,
            "IsDuplicated": false,
            "IsDynamicPoolVolume": false,
            "IsInGADPair": false,
            "IsJournalPoolVolume": false,
            "IsSLU": false,
            "IsVVOL": false,
            "Lun": 300,
            "MetaResourceSerialNumber": 123456,
            "Name": "",
            "ParityGroup": "",
            "PathCount": 1,
            "PhysicalLunNumber": 300,
            "PoolId": "5",
            "PoolType": "HDP",
            "RG": 0,
            "RaidLevel": 0,
            "ResourceGroupId": 0,
            "Status": "NORMAL",
            "TotalCapacity": "3.0GB",
            "UsedCapacity": "798.0MB",
            "VirtualLun": 300,
            "VirtualStorageDeviceId": 123456,
            "stripeSize": 512
        },
        {
            "CanonicalName": "naa.60060e80123ad00050403ad00000012d",
            "DedupCompressionMode": "DISABLED",
            "DedupCompressionProgress": "",
            "DedupCompressionStatus": "DISABLED",
            "EmulationType": "OPEN-V-CVS",
            "FormatOrShredRate": 100,
            "HexLunNumber": "00:01:2D",
            "IsALU": false,
            "IsALUA": false,
            "IsBusy": false,
            "IsCommandDevice": false,
            "IsDMLU": false,
            "IsDuplicated": false,
            "IsDynamicPoolVolume": false,
            "IsInGADPair": false,
```

```
            "IsJournalPoolVolume": false,
            "IsSLU": false,
            "IsVVOL": false,
            "Lun": 301,
            "MetaResourceSerialNumber": 123456,
            "Name": "",
            "ParityGroup": "",
            "PathCount": 1,
            "PhysicalLunNumber": 301,
            "PoolId": "5",
            "PoolType": "HDP",
            "RG": 0,
            "RaidLevel": 0,
            "ResourceGroupId": 0,
            "Status": "NORMAL",
            "TotalCapacity": "3.0GB",
            "UsedCapacity": "0",
            "VirtualLun": 301,
            "VirtualStorageDeviceId": 123456,
            "stripeSize": 512
        }
    ]
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*********************************************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create LUN

**Module: hv_lun**

The `create_lun.yml` playbook creates a LUN on the dynamic pool of the provided storage system.

**Note:**

This module can also be used for updating the LUN. It will update the LUN, if the same already exists on the storage system.

Different `pool_id`, `lun` and less `size` provided in the playbook to update LUN, will result an error.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set the state to present for creating and updating LUN.<br>Set the state to absent for deleting LUN. |
| pool_id | Integer | Dynamic pool ID present on the storage | Yes | N/A |
| size | string | LUN size that you want to create | Yes | Example: 10GB, 5TB |
| name | string | Name of the new LUN to be created | No | N/A |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| lun | integer | LUN ID in decimal or HEX of the newly created LUN | No | Free LDEV ID, which will be used for the LUN creation.<br>If not provided, the first free LDEV ID will automatically be selected. |
| cap_saving | string | Valid values:<br>▪ compression<br>▪ deduplication<br>▪ disable (default) | No | N/A |

**Example: Testing Create LUN In DP**

```
name: Testing Create LUN In DP
  hosts: localhost
  gather_facts: false
  vars:
    - pool_id: 27
    - size: 0.07GB
    - storage_serial: 123456
    - lun: 1416
    - name: AnsiJUNE
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
```

```
          storage_pool:
            id: '{{ pool_id  | default(None) }}'
          name: '{{ name  | default(None) }}'
          size: '{{ size | default(None)}}'
          lun: '{{ lun | default(None) }}'
          cap_saving: '{{ cap_saving | default(None) }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook create_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
[WARNING]: Found variable using reserved name: name

PLAY [Testing Create LUN In DP]
*********************************************************************
*********************************************************************
************************************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false,
        "lun": {
            "CanonicalName": "naa.60060e80123ad00050403ad000000588",
            "DedupCompressionMode": "DISABLED",
            "DedupCompressionProgress": "",
            "DedupCompressionStatus": "DISABLED",
            "EmulationType": "OPEN-V-CVS",
            "FormatOrShredRate": 100,
            "HexLunNumber": "00:05:88",
            "IsALU": false,
            "IsALUA": false,
            "IsBusy": false,
            "IsCommandDevice": false,
            "IsDMLU": false,
            "IsDuplicated": false,
            "IsDynamicPoolVolume": false,
```

```
            "IsInGADPair": false,
            "IsJournalPoolVolume": false,
            "IsSLU": false,
            "IsVVOL": false,
            "Lun": 1416,
            "MetaResourceSerialNumber": 123456,
            "Name": "AnsiJUNE",
            "ParityGroup": "",
            "PathCount": 0,
            "PhysicalLunNumber": 1416,
            "PoolId": "6",
            "PoolType": "HDP",
            "RG": 0,
            "RaidLevel": 0,
            "ResourceGroupId": 0,
            "Status": "NORMAL",
            "TotalCapacity": "71.0MB",
            "UsedCapacity": "0",
            "VirtualLun": 1416,
            "VirtualStorageDeviceId": 123456,
            "stripeSize": 512
        }
    }
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example: Create LUN In DP when LUN ID given in Hexadecimal form**

```
name: Testing Create LUN In DP when LUN ID given in Hexadecimal form
  hosts: localhost
  gather_facts: false
  vars:
    - pool_id: 27
    - size: 0.07GB
    - storage_serial: 123456
    - lun: 00:06:40
    - name: TestJune16
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          storage_pool:
            id: '{{ pool_id  | default(None) }}'
```

```
          name: '{{ name  | default(None) }}'
          size: '{{ size | default(None)}}'
          lun: '{{ lun | default(None) }}'
          cap_saving: '{{ cap_saving | default(None) }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook create_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
[WARNING]: Found variable using reserved name: name

PLAY [Testing Create LUN In DP]
*********************************************************************
*********************************************************************
**************************************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false,
        "lun": {
            "CanonicalName": "naa.60060e80123ad00050403ad000000640",
            "DedupCompressionMode": "DISABLED",
            "DedupCompressionProgress": "",
            "DedupCompressionStatus": "DISABLED",
            "EmulationType": "OPEN-V-CVS",
            "FormatOrShredRate": 100,
            "HexLunNumber": "00:06:40",
            "IsALU": false,
            "IsALUA": false,
            "IsBusy": false,
            "IsCommandDevice": false,
            "IsDMLU": false,
            "IsDuplicated": false,
            "IsDynamicPoolVolume": false,
            "IsInGADPair": false,
            "IsJournalPoolVolume": false,
```

```
            "IsSLU": false,
            "IsVVOL": false,
            "Lun": 1600,
            "MetaResourceSerialNumber": 123456,
            "Name": "TestJune16",
            "ParityGroup": "",
            "PathCount": 0,
            "PhysicalLunNumber": 1600,
            "PoolId": "6",
            "PoolType": "HDP",
            "RG": 0,
            "RaidLevel": 0,
            "ResourceGroupId": 0,
            "Status": "NORMAL",
            "TotalCapacity": "71.0MB",
            "UsedCapacity": "0",
            "VirtualLun": 1600,
            "VirtualStorageDeviceId": 123456,
            "stripeSize": 512
        }
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create LUN in a parity group

**Module: hv_lun**

The `create_lun_parity.yml` playbook creates a LUN in the parity group of the provided storage system.

**Note:**

This module can also be used for updating the LUN in the parity group. It will update the LUN, if the same already exists on the storage system.

Different `parity_group` and less `size` provided in the playbook to update the LUN, will result an error.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible choices:<br>▪ present<br>▪ absent<br>default: present | Yes | Set the state to present for creating and updating the LUN.<br>Set the state to absent for deleting the LUN. |
| parity_group | string | Parity group present on the storage | Yes | N/A |
| size | string | LUN size that you want to create | Yes | Example: 10GB, 5TB |
| name | string | Name of the new LUN to be created | No | N/A |
| lun | integer | LUN ID in decimal or HEX of the newly created LUN | No | Free LDEV ID that will be used for LUN creation.<br>If not provided, the first free LDEV ID will automatically be selected. |

**Example: Creating LUN In PG**

```
name: Create LUN In PG
  hosts: localhost
  gather_facts: false
  vars:
    - parity_group: 1-1
    - size: 1GB
    - lun: 6271
    - storage_serial: 123456
    - name: ansidec26
```

```
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          parity_group: '{{ parity_group | default(None)}}'
          size: '{{ size | default(None) }}'
          name: '{{ name  | default(None) }}'
          lun: '{{ lun | default(None) }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook create_lun_parity.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Create LUN In PG]
*********************************************************************
*********************************************************************
*************************************************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false,
        "lun": {
            "CanonicalName": "naa.60060e80123ad00050403ad00000058b",
            "DedupCompressionMode": "UNKNOWN",
            "DedupCompressionProgress": "",
            "DedupCompressionStatus": "UNKNOWN",
            "EmulationType": "OPEN-V-CVS",
            "FormatOrShredRate": 100,
            "HexLunNumber": "00:05:8B",
            "IsALU": false,
            "IsALUA": false,
            "IsBusy": false,
            "IsCommandDevice": false,
            "IsDMLU": false,
```

```
            "IsDuplicated": false,
            "IsDynamicPoolVolume": false,
            "IsInGADPair": false,
            "IsJournalPoolVolume": false,
            "IsSLU": false,
            "IsVVOL": false,
            "Lun": 1419,
            "MetaResourceSerialNumber": 123456,
            "Name": "",
            "ParityGroup": "1-4",
            "PathCount": 0,
            "PhysicalLunNumber": 1419,
            "PoolId": "1-4",
            "PoolType": "PARITYGROUP",
            "RG": 0,
            "RaidLevel": 0,
            "ResourceGroupId": 0,
            "Status": "QUICKFORMAT",
            "TotalCapacity": "1.0GB",
            "UsedCapacity": "1.0GB",
            "VirtualLun": 1419,
            "VirtualStorageDeviceId": 123456,
            "stripeSize": 512
        }
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create multiple LUNs on a dynamic pool

**Module: hv_lun**

The `create_multi_luns.yml` playbook creates multiple LUNs on the dynamic pool of the provided storage system.

**Note:**

This module can also be used for updating the LUNs. It will update the LUNs, if the same already exists on the storage system.

Different `pool_id`, `luns` and less `size` provided in the playbook to update the LUN, will result an error

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set the state to present for creating and updating LUNs.<br>Set the state to absent for deleting LUNs. |
| pool_id | Integer | Dynamic pool ID present on the storage | Yes | N/A |
| size | string | LUN size that you want to create | Yes | Example: 10GB, 5TB |
| luns | array | LUN ID in decimal or HEX of the newly created LUN | Yes | Free LDEV ID that will be used for the LUN creation |
| cap_saving | string | Valid values:<br>▪ compression<br>▪ deduplication<br>▪ disable (default) | No | N/A |

**Example: Testing Create Multiple LUNs in DP**

```
name: Testing Create Multiple LUN with ID In DP
  hosts: localhost
```

```
  gather_facts: false
  vars:
    - pool_id: 6
    - size: 0.08GB
    - storage_serial: 123456
    - luns:
      - 1577
      - 1578
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          storage_pool:
          id: '{{ pool_id | default(None)}}'
          luns: '{{ item }}'
          size: '{{ size | default(None)}}'
          name: '{{ name | default(None) }}'
          cap_saving: '{{ cap_saving | default(None) }}'
      with_items: '{{ luns }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook create_multi_luns.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Create Multiple LUN with ID In DP]
*********************************************************************
*********************************************************************
**********************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost] => (item=1577)
changed: [localhost] => (item=1578)

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "msg": "All items completed",
        "results": [
```

```
            {
                "ansible_loop_var": "item",
                "changed": true,
                "failed": false,
                "invocation": {
                    "module_args": {
                        "data": "{\"luns\": 1577, \"storage_pool\": {\"id
\": \"6\"}, \"name\": \"\", \"size\": \"0.08GB\"}",
                        "state": "present",
                        "storage_serial": 123456
                    }
                },
                "item": 1577,
                "lun": {
                    "CanonicalName":
"naa.60060e80123ad00050403ad000000629",
                    "DedupCompressionMode": "DISABLED",
                    "DedupCompressionProgress": "",
                    "DedupCompressionStatus": "DISABLED",
                    "EmulationType": "OPEN-V-CVS",
                    "FormatOrShredRate": 100,
                    "HexLunNumber": "00:06:29",
                    "IsALU": false,
                    "IsALUA": false,
                    "IsBusy": false,
                    "IsCommandDevice": false,
                    "IsDMLU": false,
                    "IsDuplicated": false,
                    "IsDynamicPoolVolume": false,
                    "IsInGADPair": false,
                    "IsJournalPoolVolume": false,
                    "IsSLU": false,
                    "IsVVOL": false,
                    "Lun": 1577,
                    "MetaResourceSerialNumber": 123456,
                    "Name": "",
                    "ParityGroup": "",
                    "PathCount": 0,
                    "PhysicalLunNumber": 1577,
                    "PoolId": "6",
                    "PoolType": "HDP",
                    "RG": 0,
                    "RaidLevel": 0,
                    "ResourceGroupId": 0,
                    "Status": "NORMAL",
                    "TotalCapacity": "81.0MB",
                    "UsedCapacity": "0",
                    "VirtualLun": 1577,
                    "VirtualStorageDeviceId": 123456,
                    "stripeSize": 512
                }
```

```
            },
            {
                "ansible_loop_var": "item",
                "changed": true,
                "failed": false,
                "invocation": {
                    "module_args": {
                        "data": "{\"luns\": 1578, \"storage_pool\": {\"id
\": \"6\"}, \"name\": \"\", \"size\": \"0.08GB\"}",
                        "state": "present",
                        "storage_serial": 123456
                    }
                },
                "item": 1578,
                "lun": {
                    "CanonicalName":
"naa.60060e80123ad00050403ad00000062a",
                    "DedupCompressionMode": "DISABLED",
                    "DedupCompressionProgress": "",
                    "DedupCompressionStatus": "DISABLED",
                    "EmulationType": "OPEN-V-CVS",
                    "FormatOrShredRate": 100,
                    "HexLunNumber": "00:06:2A",
                    "IsALU": false,
                    "IsALUA": false,
                    "IsBusy": false,
                    "IsCommandDevice": false,
                    "IsDMLU": false,
                    "IsDuplicated": false,
                    "IsDynamicPoolVolume": false,
                    "IsInGADPair": false,
                    "IsJournalPoolVolume": false,
                    "IsSLU": false,
                    "IsVVOL": false,
                    "Lun": 1578,
                    "MetaResourceSerialNumber": 123456,
                    "Name": "",
                    "ParityGroup": "",
                    "PathCount": 0,
                    "PhysicalLunNumber": 1578,
                    "PoolId": "6",
                    "PoolType": "HDP",
                    "RG": 0,
                    "RaidLevel": 0,
                    "ResourceGroupId": 0,
                    "Status": "NORMAL",
                    "TotalCapacity": "81.0MB",
                    "UsedCapacity": "0",
                    "VirtualLun": 1578,
                    "VirtualStorageDeviceId": 123456,
                    "stripeSize": 512
```

```
                }
            }
        ]
    }
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*********************************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete LUN

**Module: hv_lun**

The `delete_lun_playbook.yml` playbook deletes one or more LUN from the provided storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | default: absent | Yes | Set the state to absent for deleting LUNS. |
| luns | array | LUN ID in decimal or HEX present on the storage | Yes | Can provide multiple LUNs for deleting. |
| name | string | Name of the LUN to be deleted | No | Ignored if `luns` value is provided. Operation is aborted if more than 1 LUN with matching name is found. |

**Example 1: Testing delete LUN with LUN name**

```
name: Testing Delete LUN
  hosts: localhost
```

```
  gather_facts: false
  vars:
    - luns:
    - name:  delete_lun_byName
    - storage_serial: 123456
  tasks:
    - hv_lun:
        state: absent
        storage_serial: '{{ storage_serial }}'
        data:
          luns: '{{ item }}'
          name: '{{ name | default(None) }}'
      with_items: '{{ luns }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook delete_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
[WARNING]: Found variable using reserved name: register
[WARNING]: Found variable using reserved name: name

PLAY [Testing Delete LUN]
*********************************************************************
*********************************************************************
*************************************************

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
```

## Create multiple LUNs on dynamic pool with count

**Module: hv_lun**

The `create_multi_luns_with_count.yml` playbook creates the number of LUNs provided by the count on the playbook, on the dynamic pool of the provided storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present | Yes | Set the state to present for creating and updating LUNS. |
| pool_id | Integer | Dynamic pool ID present on the storage | Yes | N/A |
| size | string | LUN size that you want to create | Yes | Example:<br>10 GB or 5 TB |
| count | integer | Number of LUNs to be created | Yes | N/A |
| cap_saving | string | Valid values:<br>▪ compression<br>▪ deduplication<br>▪ disable (default) | No | N/A |

**Example: Testing create multiple LUNs with Count in DP**

```
name: Testing Create Multiple LUN with count In DP
  hosts: localhost
  gather_facts: false
  vars:
    - pool_id: 6
    - size: 0.09GB
    - storage_serial: 123456
    - count: 2
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          storage_pool:
            id: '{{ pool_id }}'
```

```
          size: '{{ size }}'
          cap_saving: '{{ cap_saving | default(None) }}'
          with_sequence:  end={{count}} start=1
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook create_multi_luns_with_count.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Create Multiple LUN With Count In DP]
*********************************************************************
*********************************************************************
*******************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost] => (item=1)
changed: [localhost] => (item=2)

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "msg": "All items completed",
        "results": [
            {
                "ansible_loop_var": "item",
                "changed": true,
                "failed": false,
                "invocation": {
                    "module_args": {
                        "data": "{\"cap_saving\": \"\", \"storage_pool\":
{\"id\": 6}, \"size\": \"0.09GB\"}",
                        "state": "present",
                        "storage_serial": 123456
                    }
                },
                "item": "1",
                "lun": {
                    "CanonicalName":
"naa.60060e80123ad00050403ad000000589",
                    "DedupCompressionMode": "DISABLED",
```

```
                    "DedupCompressionProgress": "",
                    "DedupCompressionStatus": "DISABLED",
                    "EmulationType": "OPEN-V-CVS",
                    "FormatOrShredRate": 100,
                    "HexLunNumber": "00:05:89",
                    "IsALU": false,
                    "IsALUA": false,
                    "IsBusy": false,
                    "IsCommandDevice": false,
                    "IsDMLU": false,
                    "IsDuplicated": false,
                    "IsDynamicPoolVolume": false,
                    "IsInGADPair": false,
                    "IsJournalPoolVolume": false,
                    "IsSLU": false,
                    "IsVVOL": false,
                    "Lun": 1417,
                    "MetaResourceSerialNumber": 123456,
                    "Name": "",
                    "ParityGroup": "",
                    "PathCount": 0,
                    "PhysicalLunNumber": 1417,
                    "PoolId": "6",
                    "PoolType": "HDP",
                    "RG": 0,
                    "RaidLevel": 0,
                    "ResourceGroupId": 0,
                    "Status": "NORMAL",
                    "TotalCapacity": "92.0MB",
                    "UsedCapacity": "0",
                    "VirtualLun": 1417,
                    "VirtualStorageDeviceId": 123456,
                    "stripeSize": 512
                }
            },
            {
                "ansible_loop_var": "item",
                "changed": true,
                "failed": false,
                "invocation": {
                    "module_args": {
                        "data": "{\"cap_saving\": \"\", \"storage_pool\":
{\"id\": 6}, \"size\": \"0.09GB\"}",
                        "state": "present",
                        "storage_serial": 123456
                    }
                },
                "item": "2",
                "lun": {
                    "CanonicalName":
"naa.60060e80123ad00050403ad00000058a",
```

```
                    "DedupCompressionMode": "DISABLED",
                    "DedupCompressionProgress": "",
                    "DedupCompressionStatus": "DISABLED",
                    "EmulationType": "OPEN-V-CVS",
                    "FormatOrShredRate": 100,
                    "HexLunNumber": "00:05:8A",
                    "IsALU": false,
                    "IsALUA": false,
                    "IsBusy": false,
                    "IsCommandDevice": false,
                    "IsDMLU": false,
                    "IsDuplicated": false,
                    "IsDynamicPoolVolume": false,
                    "IsInGADPair": false,
                    "IsJournalPoolVolume": false,
                    "IsSLU": false,
                    "IsVVOL": false,
                    "Lun": 1418,
                    "MetaResourceSerialNumber": 123456,
                    "Name": "",
                    "ParityGroup": "",
                    "PathCount": 0,
                    "PhysicalLunNumber": 1418,
                    "PoolId": "6",
                    "PoolType": "HDP",
                    "RG": 0,
                    "RaidLevel": 0,
                    "ResourceGroupId": 0,
                    "Status": "NORMAL",
                    "TotalCapacity": "92.0MB",
                    "UsedCapacity": "0",
                    "VirtualLun": 1418,
                    "VirtualStorageDeviceId": 123456,
                    "stripeSize": 512
                }
            }
        ]
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Expand LUN

**Module: hv_lun**

The `expand_lun_playbook.yml` playbook expands the LUN size with additional given size on the dynamic pool of the provided storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set the state to present for updating the LUN.<br>Set the state to absent for deleting the LUN. |
| luns | Integer | LUN ID in decimal or HEX of the LUN to be expanded | Yes | N/A |
| size_gb | Integer | Additional size of the LUN to be expanded | Yes | N/A |
| cap_saving | string | Valid values:<br>▪ compression<br>▪ deduplication<br>▪ disable (default) | No | N/A |

**Example: Testing expand LUN**

```
name: Testing Expand LUN
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - luns: 836
```

```
    - size_gb: 3
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          luns: '{{ luns }}'
          size: '{{size_gb}}'
          cap_saving: '{{ cap_saving | default(None) }}'
      register: result
    - debug: var=result.lun
```

**Output**

```
[root@localhost block]# ansible-playbook expand_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Expand LUN]
*********************************************************************
*********************************************************************
*************************************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result.lun": {
        "CanonicalName": "naa.60060e80123ad00050403ad000000344",
        "DedupCompressionMode": "DISABLED",
        "DedupCompressionProgress": "",
        "DedupCompressionStatus": "DISABLED",
        "EmulationType": "OPEN-V-CVS",
        "FormatOrShredRate": 100,
        "HexLunNumber": "00:03:44",
        "IsALU": false,
        "IsALUA": false,
        "IsBusy": false,
        "IsCommandDevice": false,
        "IsDMLU": false,
        "IsDuplicated": false,
        "IsDynamicPoolVolume": false,
        "IsInGADPair": false,
```

```
        "IsJournalPoolVolume": false,
        "IsSLU": false,
        "IsVVOL": false,
        "Lun": 836,
        "MetaResourceSerialNumber": 123456,
        "Name": "",
        "ParityGroup": "",
        "PathCount": 0,
        "PhysicalLunNumber": 836,
        "PoolId": "26",
        "PoolType": "HDP",
        "RG": 0,
        "RaidLevel": 0,
        "ResourceGroupId": 0,
        "Status": "NORMAL",
        "TotalCapacity": "10.0GB",
        "UsedCapacity": "0",
        "VirtualLun": 836,
        "VirtualStorageDeviceId": 123456,
        "stripeSize": 512
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example: Testing Expand LUN when LUN ID given in Hexadecimal form**

```
name: Testing Expand LUN when LUN ID given in Hexadecimal form
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - luns: 00:05:87
    - size_gb: 7
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          luns: '{{ luns }}'
          size: '{{size_gb}}'
          cap_saving: '{{ cap_saving | default(None) }}'
      register: result
    - debug: var=result.lun
```

**Output**

```
[root@localhost block]# ansible-playbook expand_lun.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Expand LUN]
*********************************************************************
*********************************************************************
*************************************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
*****************************************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*****************************************************************
ok: [localhost] => {
    "result.lun": {
        "CanonicalName": "naa.60060e80123ad00050403ad000000587",
        "DedupCompressionMode": "DISABLED",
        "DedupCompressionProgress": "",
        "DedupCompressionStatus": "DISABLED",
        "EmulationType": "OPEN-V-CVS",
        "FormatOrShredRate": 100,
        "HexLunNumber": "00:05:87",
        "IsALU": false,
        "IsALUA": false,
        "IsBusy": false,
        "IsCommandDevice": false,
        "IsDMLU": false,
        "IsDuplicated": false,
        "IsDynamicPoolVolume": false,
        "IsInGADPair": false,
        "IsJournalPoolVolume": false,
        "IsSLU": false,
        "IsVVOL": false,
        "Lun": 1415,
        "MetaResourceSerialNumber": 123456,
        "Name": "",
        "ParityGroup": "",
        "PathCount": 0,
        "PhysicalLunNumber": 1415,
        "PoolId": "6",
        "PoolType": "HDP",
        "RG": 0,
        "RaidLevel": 0,
```

```
        "ResourceGroupId": 0,
        "Status": "NORMAL",
        "TotalCapacity": "13.08984375GB",
        "UsedCapacity": "42.0MB",
        "VirtualLun": 1415,
        "VirtualStorageDeviceId": 123456,
        "stripeSize": 512
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Clone LUN

**Module: hv_lun**

The `clone_lun_in_dp_playbook.yml` playbook clones the LUN on the dynamic pool of the provided storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | No | Set the state to present for creating and updating.<br>Set the state to absent for deleting. |
| source_lun | Integer | LUN ID in decimal or HEX of the LUN to be cloned | Yes | N/A |
| cloned_lun_name | string | Name of the cloned LUN | Yes | N/A |
| pool_id | Integer | Dynamic pool ID | Yes | N/A |

**Example: Testing Clone LUN**

```
name: Testing Clone LUN
  hosts: localhost
  gather_facts: false
  vars:
    - pool_id: 31
    - source_lun: 245
    - cloned_lun_name: 'clone245'
    - storage_serial: 123456
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          luns: '{{ source_lun }}'
          cloned_lun_name: '{{ cloned_lun_name }}'
          storage_pool:
            id: '{{ pool_id }}'
      register: result
    - debug: var=result.lun
```

**Output**

```
[root@localhost playbooks]# ansible-playbook clone_lun_in_dp_playbook.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [Testing Clone LUN]
*********************************************************************
*********************************************************************
*******************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
****************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
****************************************
ok: [localhost] => {
    "result.lun": {
        "CanonicalName": "naa.60060e80123ad00050403ad000000183",
        "DedupCompressionMode": "DISABLED",
        "DedupCompressionProgress": "",
        "DedupCompressionStatus": "DISABLED",
```

```
        "EmulationType": "OPEN-V-CVS",
        "FormatOrShredRate": 100,
        "HexLunNumber": "00:01:83",
        "IsALU": false,
        "IsALUA": false,
        "IsBusy": false,
        "IsCommandDevice": false,
        "IsDMLU": false,
        "IsDuplicated": false,
        "IsDynamicPoolVolume": false,
        "IsInGADPair": false,
        "IsJournalPoolVolume": false,
        "IsSLU": false,
        "IsVVOL": false,
        "Lun": 387,
        "MetaResourceSerialNumber": 123456,
        "Name": "clone245",
        "ParityGroup": "",
        "PathCount": 0,
        "PhysicalLunNumber": 387,
        "PoolId": "31",
        "PoolType": "HDP",
        "RG": 0,
        "RaidLevel": 0,
        "ResourceGroupId": 0,
        "Status": "NORMAL",
        "TotalCapacity": "2048MB",
        "UsedCapacity": "0",
        "VirtualLun": -1,
        "VirtualStorageDeviceId": 123456,
        "stripeSize": 512
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example: Testing Clone LUN when LUN ID given in Hexadecimal form**

```
name: Testing Clone LUN when LUN ID given in Hexadecimal form
  hosts: localhost
  gather_facts: false
  vars:
    - pool_id: 6
    - source_lun: 00:05:B3
    - cloned_lun_name: 'clonehex'
```

```
    - storage_serial: 123456
  tasks:
    - hv_lun:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          luns: '{{ source_lun }}'
          cloned_lun_name: '{{ cloned_lun_name }}'
          storage_pool:
            id: '{{ pool_id }}'
      register: result
    - debug: var=result.lun
```

**Output**

```
[root@localhost block]# ansible-playbook clone_lun_in_dp.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Clone Lun]
*********************************************************************
*********************************************************************
***********************************

TASK [hv_lun]
*********************************************************************
*********************************************************************
*********************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*********************************************
ok: [localhost] => {
    "result.lun": {
        "CanonicalName": "naa.60060e80123ad00050403ad000000570",
        "DedupCompressionMode": "DISABLED",
        "DedupCompressionProgress": "",
        "DedupCompressionStatus": "DISABLED",
        "EmulationType": "OPEN-V-CVS",
        "FormatOrShredRate": 100,
        "HexLunNumber": "00:05:70",
        "IsALU": false,
        "IsALUA": false,
        "IsBusy": false,
        "IsCommandDevice": false,
        "IsDMLU": false,
        "IsDuplicated": false,
        "IsDynamicPoolVolume": false,
```

```
        "IsInGADPair": false,
        "IsJournalPoolVolume": false,
        "IsSLU": false,
        "IsVVOL": false,
        "Lun": 1392,
        "MetaResourceSerialNumber": 123456,
        "Name": "clonehex",
        "ParityGroup": "",
        "PathCount": 0,
        "PhysicalLunNumber": 1392,
        "PoolId": "6",
        "PoolType": "HDP",
        "RG": 0,
        "RaidLevel": 0,
        "ResourceGroupId": 0,
        "Status": "NORMAL",
        "TotalCapacity": "71.0MB",
        "UsedCapacity": "0",
        "VirtualLun": 1392,
        "VirtualStorageDeviceId": 123456,
        "stripeSize": 512
    }
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Manage local and remote replication pairs

**Module: hv_rep**

The `hv_rep` module manages Hitachi Thin Image (HTI), Hitachi TrueCopy® (TC), Hitachi Universal Replicator (UR), global-active device (GAD), and volume clone on a Hitachi storage system.

This module provides the following HTI management operations:

1. create HTI pair
2. split HTI pair
3. resync HTI pair
4. restore HTI pair
5. delete HTI pair

This module provides the following TrueCopy management operations:

1. create TrueCopy pair
2. split TrueCopy pair
3. resync TrueCopy pair
4. delete TrueCopy pair

This module provides the following UR management operations:

1. create UR pair
2. split UR pair
3. resync UR pair
4. delete UR pair

This module provides the following global-active device management operations:

1. create global-active device pair
2. split global-active device pair
3. resync global-active device pair
4. delete global-active device pair

For more info, run **`ansible-doc hv_rep`** and **`ansible-doc hv_rep_facts`** to view the documentation for the module.

## Provide replication pair information

**Module: hv_rep_facts**

The `hv_rep_facts` module provides HTI, TrueCopy, Hitachi Universal Replicator, and global-active device replication pair information on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | Integer | Storage system serial number | Yes | If the storage serial number parameter is given, it returns replication pairs depending upon the storage serial number and replication type. |
| lun | integer | Primary volume in decimal or HEX | No | If the primary volume parameter is provided, it returns the replication pair on the basis of the specified primary volume. |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| copy_type | String | Replication type. Possible values:<br>▪ ti<br>▪ tc<br>▪ ur<br>▪ gad | Yes | For HTI: the value must be ti.<br>For TrueCopy: the value must be tc.<br>For Hitachi Universal Replicator: the value must be ur.<br>For global-active device: the value must be gad. |

**Example: Getting HTI Pair Information**

```
name: Getting HTI pairs info
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - lun: 1299
  tasks:
    - hv_rep_facts:
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: ti
          lun: '{{lun | default(None)}}'
      register: result
    - debug: var=result.pairs
```

**Output**

```
[root@172 block]# ansible-playbook get_hti_pairs.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Getting HTI pairs info]
**********************************************************************
**************

TASK [hv_rep_facts]
**********************************************************************
***********************
ok: [localhost]

TASK [debug]
**********************************************************************
*****************************
```

```
ok: [localhost] => {
    "result.pairs": [
        {
            "ConsistencyGroupId": 139,
            "DataPoolId": 7,
            "GroupName": "",
            "HexPVOL": "00:05:13",
            "HexSVOL": "00:05:58",
            "MirrorId": 3,
            "PVol": 1299,
            "SVol": 1368,
            "Serial": 12345,
            "Status": "PAIR",
            "Type": "THIN_IMAGE"
        },
        {
            "ConsistencyGroupId": 136,
            "DataPoolId": 7,
            "GroupName": "snappyhappy1",
            "HexPVOL": "00:05:13",
            "HexSVOL": "00:05:5A",
            "MirrorId": 4,
            "PVol": 1299,
            "SVol": 1370,
            "Serial": 12345,
            "Status": "PAIR",
            "Type": "THIN_IMAGE"
        },
        {
            "DataPoolId": 7,
            "GroupName": "",
            "HexPVOL": "00:05:13",
            "HexSVOL": "00:05:63",
            "MirrorId": 5,
            "PVol": 1299,
            "SVol": 1379,
            "Serial": 12345,
            "Status": "PAIR",
            "Type": "THIN_IMAGE"
        },
        {
            "DataPoolId": 7,
            "GroupName": "",
            "HexPVOL": "00:05:13",
            "HexSVOL": "00:05:64",
            "MirrorId": 6,
            "PVol": 1299,
            "SVol": 1380,
            "Serial": 12345,
            "Status": "PAIR",
            "Type": "THIN_IMAGE"
```

```
        }
    ]
}

PLAY RECAP
*********************************************************************************
*******************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example: Getting TC pair information**

```
name: Getting TC Pairs Info
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
  tasks:
    - hv_rep_facts:
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: tc
          lun: '{{lun | default(None)}}'
      register: result
```

**Output**

```
[root@172 block]# ansible-playbook get_tc_pairs.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match  'all'
PLAY [Getting TC Pairs Info]
*********************************************************************************
***************
TASK [hv_rep_facts]
*********************************************************************************
***********************
ok: [localhost]

TASK [debug]
*********************************************************************************
*****************************
ok: [localhost] => {
    "result.pairs": [
        {
            "FenceLevel": "NEVER",
            "HexPVOL": "00:03:DD",
            "HexSVOL": "00:01:7E",
            "PVol": 989,
            "PVolSerial": 12345,
            "SVol": 382,
```

```
            "SVolSerial": 123456,
            "Status": "PAIR",
            "Type": "TRUE_COPY"
        },
        {
            "FenceLevel": "NEVER",
            "HexPVOL": "00:03:E5",
            "HexSVOL": "00:07:61",
            "PVol": 997,
            "PVolSerial": 12345,
            "SVol": 1889,
            "SVolSerial": 123456,
            "Status": "PSUS",
            "Type": "TRUE_COPY"
            }
    ]
}

PLAY RECAP
*********************************************************************
*******************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example: Getting UR pair information**

```
name: Getting Universal Replicator Pairs Info
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
  tasks:
    - hv_rep_facts:
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: ur
          lun: '{{lun | default(None)}}'
      register: result
    - debug: var=result.pairs
```

**Output**

```
[root@172 block]# ansible-playbook get_ur_pairs.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Getting Universal Replicator Pairs Info]
*********************************************************************
```

```
TASK [hv_rep_facts]
*********************************************************************************
***********************
ok: [localhost]

TASK [debug]
*********************************************************************************
*******************************
ok: [localhost] => {
    "result.pairs": [
        {
            "ConsistencyGroupId": 10,
            "HexPVOL": "00:00:AA",
            "HexSVOL": "00:02:AB",
            "MirrorId": 1,
            "PVol": 170,
            "PVolSerial": 12345,
            "PrimaryJournalPoolId": 10,
            "SVol": 683,
            "SVolSerial": 123456,
            "Status": "PAIR",
            "Type": "UR"
        },
        {
            "ConsistencyGroupId": 2,
            "HexPVOL": "00:FE:06",
            "HexSVOL": "00:BF:06",
            "MirrorId": 1,
            "PVol": 65030,
            "PVolSerial": 12345,
            "PrimaryJournalPoolId": 11,
            "SVol": 48902,
            "SVolSerial": 123456,
            "Status": "PAIR",
            "Type": "UR"
        }
    ]
}

PLAY RECAP
*********************************************************************************
*********************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

**Example: Getting global-active device pair information**

```
name: Getting GAD pairs Info
  hosts: localhost
  gather_facts: false
```

```
  vars:
    - storage_serial: 12345
  tasks:
    - hv_rep_facts:
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: gad
          lun: '{{lun | default(None)}}'
      register: result
    - debug: var=result.pairs
```

**Output**

```
[root@172 block]# ansible-playbook get_gad_pairs.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Getting GAD pairs Info]
*****************************************************************************
****************************

TASK [hv_rep_facts]
*****************************************************************************
************************************
ok: [localhost]

TASK [debug]
*****************************************************************************
*******************************************
ok: [localhost] => {
    "result.pairs": [
        {
            "ConsistencyGroupId": -1,
            "HexPVOL": "00:0F:A2",
            "HexSVOL": "00:0F:A2",
            "PVol": 4002,
            "SVol": 4002,
            "SVolSerial": 123456,
            "Serial": 12345,
            "Status": "PAIR",
            "Type": "GAD",
            "VSMSerial": 415099,
            "VirtualLogicalUnitId": 4002
        },
        {
            "ConsistencyGroupId": -1,
            "HexPVOL": "00:13:88",
            "HexSVOL": "00:17:70",
            "PVol": 5000,
```

```
            "SVol": 6000,
            "SVolSerial": 123456,
            "Serial": 12345,
            "Status": "PAIR",
            "Type": "GAD",
            "VSMSerial": 415099,
            "VirtualLogicalUnitId": 5555
        }
    ]
}

PLAY RECAP
*********************************************************************************
*********************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create HTI pair

**Module: hv_rep**

The `create_hti_pair.yml` playbook creates an HTI pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | 1.Set state to present for creating an HTI pair.<br>2.Set state to absent for deleting an HTI pair. |
| copy_type | string | Replication type | Yes | copy_type=ti |
| lun | integer | Primary volume in decimal or HEX | Yes | The `lun` should be presented to a host group |
| target_pool | integer | HTI Pool ID | Yes | N/A |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| auto_split | boolean | Automatically split the created HTI pair | No | Possible values:<br>▪ true<br>▪ false |
| snapshot_grp | string | Snapshot group name | No | N/A |
| consistency_group_id | integer | Consistency group ID | No | 1. Consistency group to which the newly created snapshot pair needs to be added.<br>2. Current supported values are 0 - 255.<br>3. When the snapshot group is given, the consistency group ID is not considered |
| target_lun | integer | HTI VVOL in decimal or HEX | No | If the value is not specified, `hv_rep` finds the free LDEV, creates the VVOL for the primary volume and maps the VVOL to the HITACHI_HG host group.<br>The HITACHI_HG host group is automatically created on port CL1-A if not available. |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| allocate_consistency_group | boolean | A Boolean value (true or false) to be used in conjunction with the snapshot group name | No | Set to true if the snapshot group should be created in a consistency group |

**Example: Create HTI pair**

```
name: Create HTI Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - lun: 1299
    - target_pool: 7
    - target_lun: 1387
    - consistency_group_id: 134
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: ti
          lun: '{{lun}}'
          data_pool: '{{target_pool}}'
          auto_split: '{{auto_split | default(None)}}'
          snapshot_grp: '{{snapshot_grp | default(None)}}'
          consistency_group_id: '{{consistency_group_id | default(None)}}'
          allocate_consistency_group: '{{allocate_consistency_group |
default(None)}}'
          target_lun: '{{target_lun | default(None)}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook create_hti_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Create HTI Pair]
*********************************************************************
```

```
*********************

TASK [hv_rep]
*****************************************************************************
****************************
ok: [localhost]

TASK [debug]
*****************************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false,
        "pair": {
            "ConsistencyGroupId": 134,
            "DataPoolId": 7,
            "GroupName": "",
            "HexPVOL": "00:05:13",
            "HexSVOL": "00:05:6B",
            "MirrorId": 8,
            "PVol": 1299,
            "SVol": 1387,
            "Serial": 12345,
            "Status": "PAIR",
            "Type": "THIN_IMAGE"
        }
    }
}

PLAY RECAP
*****************************************************************************
*******************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Split HTI pair

**Module: hv_rep**

The `split_hti_pair.yml` playbook splits the HTI pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication Type | Yes | copy_type=ti |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to split HTI pair.<br>Set `state` to absent to delete HTI pair. |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | HTI VVOL in decimal or HEX | Yes | Either input `target_lun` or `mirror_id` is required |
| mirror_id | integer | Mirror ID | Yes | Either input `target_lun` or `mirror_id` is required |
| enable_quick_mode | boolean | Determines whether the pair is split in quick. Possible values:<br>▪ true<br>▪ value | No | Default value: true |
| subtask | string | subtask: split | Yes | N/A |

**Example: Split HTI pair**

```
name: Split HTI Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - lun: 1299
    - target_lun: 1387
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
```

```
        data:
          copy_type: ti
          subtask: split
          lun: '{{lun}}'
          target_lun: '{{target_lun  | default(None)}}'
          mirror_id: '{{mirror_id | default(None)}}'
          enable_quick_mode: '{{enable_quick_mode | default(None)}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook split_hti_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Split HTI Pair]
**********************************************************************
*********************

TASK [hv_rep]
**********************************************************************
*****************************
ok: [localhost]

TASK [debug]
**********************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false
    }
}

PLAY RECAP
**********************************************************************
*********************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Restore HTI pair

**Module: hv_rep**

The `restore_hti_pair.yml` playbook splits the HTI pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication Type | Yes | copy_type=ti |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present for restoring the HTI pair<br>Set `state` to absent for deleting the HTI pair |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | HTI VVOL in decimal or HEX | Yes | Either `target_lun` or `mirror_id` input parameter is required |
| mirror_id | integer | Mirror id | Yes | Either `target_lun` or `mirror_id` input parameter is required |
| enable_quick_mode | boolean | Determines whether the pair is split in quick mode. Possible values:<br>▪ true<br>▪ false | No | Default value: true |
| subtask | string | subtask: restore | Yes | N/A |

**Example: Restore HTI pair**

```
name: Restore HTI Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - lun: 1299
    - target_lun: 1387
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
```

```
          copy_type: ti
          lun: '{{lun}}'
          target_lun: '{{target_lun  | default(None)}}'
          mirror_id: '{{mirror_id | default(None)}}'
          enable_quick_mode: '{{enable_quick_mode | default(None)}}'
          subtask: restore
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook restore_hti_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Restore HTI Pair]
*********************************************************************
********************

TASK [hv_rep]
*********************************************************************
*****************************
ok: [localhost]

TASK [debug]
*********************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************
*******************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Resync HTI pair

**Module: hv_rep**

The `resync_hti_pair.yml` playbook re-synchronizes the HTI pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication Type | Yes | copy_type=ti |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to resync HTI pair.<br>Set state to absent to delete HTI pair. |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | HTI VVOL in decimal or HEX | Yes | Either `target_lun` or `mirror_id` input parameter is required |
| mirror_id | integer | Mirror ID | Yes | Either `target_lun` or `mirror_id` input parameter is required |
| enable_quick_mode | boolean | Determines whether the pair is split in quick mode | No | Default value: true |
| subtask | string | subtask: resync | Yes | N/A |

**Example: Resync HTI Pair**

```
name: Resync HTI Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - lun: 1299
    - target_lun: 1387
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: ti
          lun: '{{lun}}'
          target_lun: '{{target_lun  | default(None)}}'
          mirror_id: '{{mirror_id | default(None)}}'
          enable_quick_mode: '{{enable_quick_mode | default(None)}}'
```

```
          subtask: resync
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook resync_hti_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Resync HTI Pair]
**********************************************************************
********************

TASK [hv_rep]
**********************************************************************
*****************************
ok: [localhost]

TASK [debug]
**********************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false
    }
}

PLAY RECAP
**********************************************************************
*********************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete HTI pair

**Module: hv_rep**

The `delete_hti_pair.yml` playbook deletes the specified HTI pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication Type | Yes | copy_type=ti |
| state | string | Default:absent | Yes | N/A |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | HTI VVOL in decimal or HEX | Yes | Either the `target_lun` or `mirror_id` input parameter is required |
| mirror_id | integer | Mirror ID | Yes | Either the `target_lun` or `mirror_id` input parameter is required |
| delete_target_lun | boolean | Delete the target LUN in decimal or HEX | No | Default value: false |
| subtask | string | subtask: delete | Yes | N/A |

**Example: Delete HTI Pair**

```
name: Delete HTI Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - lun: 1299
    - mirror_id: 8
  tasks:
    - hv_rep:
        state: absent
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: ti
          lun: '{{lun}}'
          target_lun: '{{target_lun  | default(None)}}'
          mirror_id: '{{mirror_id | default(None)}}'
          delete_target_lun: '{{delete_target_lun | default(None)}}'
          subtask: delete
```

```
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook delete_hti_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Delete HTI Pair]
**********************************************************************
********************

TASK [hv_rep]
**********************************************************************
*****************************
ok: [localhost]

TASK [debug]
**********************************************************************
*******************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false,
        "pair": {
            "DataPoolId": 7,
            "GroupName": "",
            "HexPVOL": "00:05:13",
            "HexSVOL": "00:05:6B",
            "MirrorId": 8,
            "PVol": 1299,
            "SVol": 1387,
            "Serial": 415056,
            "Status": "PAIR",
            "Type": "THIN_IMAGE"
        }
    }
}

PLAY RECAP
**********************************************************************
*******************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create TC pair

**Module: hv_rep**

The `create_tc_pair.yml` playbook creates a TrueCopy (TC) pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to create a TC pair<br>Set `state` to absent to delete a TC pair |
| copy_type | string | Replication type | Yes | copy_type=tc |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_pool_id | integer | Dynamic Pool ID | Yes | Either `target_pool_id` or `target_lun` input is required<br>If `target_pool_id` is specified, new SVOL of size PVOL would be created and mapped to HITACHI_HG host group.<br>HITACHI_HG host group is automatically created on port CL1-A if not available. |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| target_storage_serial | integer | Remote storage system serial number | Yes | N/A |
| target_lun | integer | Secondary volume in decimal or HEX | Yes | Either `target_pool_id` or `target_lun` input is required |
| copy_pace | integer | Specifies the number of tracks on a RAID system that can be run simultaneously before the primary system accepts another request.<br>Valid input: 1 to 15. | No | Default value: 3 |
| consistency_group_id | integer | Consistency group ID | No | Consistency group to which the newly created replication pair needs to be added |
| fence_level | string | Fence level<br>Valid input:<br>▪ NEVER<br>▪ DATA<br>▪ STATUS | No | Default value is NEVER |
| allocate_consistency_group | boolean | This parameter only takes effect if the consistency group ID is not specified.<br>Valid input:<br>▪ true<br>▪ false | No | Default value: false |

**Example: Create True Copy Pair using target LUN**

```
name: Create True Copy Pair using target lun
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - target_storage_serial: 123456
    - lun: 989
    - target_lun: 382
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: tc
          lun: '{{lun}}'
          target_lun: '{{target_lun  | default(None)}}'
          target_pool_id: '{{target_pool_id  | default(None)}}'
          target_storage_serial: '{{target_storage_serial}}'
          fence_level: '{{fence_level  | default(None)}}'
          consistency_group_id: '{{consistency_group_id  |
default(None) }}'
          allocate_consistency_group: '{{allocate_consistency_group |
default(None)}}'
          copy_pace: '{{copy_pace | default(None)}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook create_tc_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Create True Copy Pair using target lun]
*********************************************************************

TASK [hv_rep]
*********************************************************************
*****************************
ok: [localhost]

TASK [debug]
*********************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
```

```
        "failed": false,
        "pair": {
            "FenceLevel": "NEVER",
            "HexPVOL": "00:03:DD",
            "HexSVOL": "00:01:7E",
            "PVol": 989,
            "PVolSerial": 12345,
            "SVol": 382,
            "SVolSerial": 123456,
            "Status": "COPY",
            "Type": "TRUE_COPY"
        }
    }
}

PLAY RECAP
*********************************************************************************
*******************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Split TC pair

**Module: hv_rep**

The `split_tc_pair.yml` playbook splits the TC pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication type | Yes | copy_type=tc |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to split a TC pair<br>Set `state` to absent to delete a TC pair |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | Secondary volume in decimal or HEX | No | N/A |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| target_storage_serial | integer | Remote storage system serial number | Yes | N/A |
| subtask | string | subtask: split | Yes | N/A |
| enable_read_write | boolean | Set the remote S-VOL to read-write mode | No | Default value: false |

**Example: Split True Copy Pair**

```
name: Split True Copy Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - target_storage_serial: 123456
    - lun: 989
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: tc
          lun: '{{lun}}'
          subtask: split
          target_lun: '{{target_lun | default(None)}}'
          target_storage_serial: '{{target_storage_serial}}'
          enable_read_write: '{{enable_read_write | default(None)}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook split_tc_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Split True Copy Pair]
****************************************************************************
***************

TASK [hv_rep]
****************************************************************************
*****************************
```

```
ok: [localhost]

TASK [debug]
*******************************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false
    }
}

PLAY RECAP
*******************************************************************************
*******************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Resync TC pair

**Module: hv_rep**

The `resync_tc_pair.yml` playbook re-synchronizes the TC pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication Type | Yes | copy_type=tc |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to resync a TC pair<br>Set `state` to absent to delete TC pair |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | Secondary volume in decimal or HEX | No | N/A |
| target_storage_serial | integer | Remote storage system serial number | Yes | N/A |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| copy_pace | integer | Specifies the number of tracks on a RAID system that can be run simultaneously before the primary system accepts another request.<br>Valid input: 1 to 15 | No | Default value: 3 |
| subtask | string | subtask: resync | Yes | N/A |

**Example: Resync True Copy Pair**

```
name: Resync True Copy Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - target_storage_serial: 123456
    - lun: 989
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: tc
          subtask: resync
          lun: '{{lun}}'
          target_lun: '{{target_lun | default(None)}}'
          target_storage_serial: '{{target_storage_serial}}'
          copy_pace: '{{copy_pace | default(None)}}'
      register: result
    - debug: var=result
```

**Output**

```
root@172 block]# ansible-playbook resync_tc_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Resync True Copy Pair]
**********************************************************************
**************

TASK [hv_rep]
```

```
*********************************************************************************
*****************************
ok: [localhost]

TASK [debug]
*********************************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************************
*******************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete TC pair

**Module: hv_rep**

The `delete_tc_pair.yml` playbook deletes the TC pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication type | Yes | copy_type=tc |
| state | string | default: absent | Yes | N/A |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | Secondary volume LDEV ID in decimal or HEX | No | N/A |
| target_storage_serial | integer | Remote storage system serial number | Yes | N/A |

**Example: Delete True Copy Pair**

```
name: Delete True Copy Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - target_storage_serial: 123456
    - lun: 989
  tasks:
    - hv_rep:
        state: absent
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: tc
          lun: '{{lun}}'
          target_lun: '{{target_lun | default(None)}}'
          target_storage_serial: '{{target_storage_serial}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook delete_tc_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Delete True Copy Pair]
*********************************************************************
***************

TASK [hv_rep]
*********************************************************************
*****************************
ok: [localhost]

TASK [debug]
*********************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************
*********************************
```

```
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create UR pair

**Module: hv_rep**

The `create_ur_pair.yml` playbook creates the Universal Replicator (UR) pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to create a UR pair.<br>Set `state` to present to delete a UR pair. |
| copy_type | string | Replication Type | Yes | copy_type=ur |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| lun_journal_id | integer | Primary volume Journal ID in decimal or HEX | Yes | N/A |
| target_storage_serial | integer | Remote storage system serial number | Yes | N/A |
| target_lun | integer | Secondary volume in decimal or HEX | Yes | Either `target_pool_id` or `target_lun` input is required |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| target_pool_id | integer | Dynamic pool ID | Yes | Either `target_pool_id` or `target_lun` input is required<br>If `target_pool_id` is specified, a new SVOL of size PVOL would be created and mapped to the HITACHI_HG host group.<br>The HITACHI_HG host group is automatically created on port CL1-A if not available. |
| target_lun_journal_id | integer | Secondary volume Journal ID in decimal or HEX | Yes | N/A |
| consistency_group_id | integer | Consistency group ID | No | Consistency group to which the newly created UR pair needs to be added |
| copy_pace | integer | Specifies the number of tracks on a RAID system that can be run simultaneously before the primary system accepts another request.<br>Valid input: 1 to 15. | No | Default value: 3 |
| enable_delta_resync | boolean | If true, creates a delta-resync HUR pair | No | Default value: false |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| allocate_consistency_group | boolean | This parameter only takes effect if the consistency Group ID is not specified.<br>Valid input:<br>▪ true<br>▪ false | No | If true, get a free consistency group ID and assign a newly created remote clone to it.<br>Default value: true |

**Example: Create Universal Replicator Pair**

```
name: Create Universal Replicator Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - target_storage_serial: 123456
    - lun: 170
    - lun_journal_id: 10
    - target_lun: 683
    - target_lun_journal_id: 10
    - consistency_group_id:10
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: ur
          lun: '{{lun}}'
          lun_journal_id: '{{lun_journal_id}}'
          target_lun_journal_id: '{{target_lun_journal_id}}'
          target_lun: '{{target_lun  | default(None)}}'
          target_pool_id: '{{target_pool_id  | default(None)}}'
          target_storage_serial: '{{target_storage_serial}}'
          consistency_group_id: '{{consistency_group_id  |
default(None) }}'
          allocate_consistency_group: '{{allocate_consistency_group |
default(None)}}'
          copy_pace: '{{copy_pace | default(None)}}'
          enable_delta_resync: '{{enable_delta_resync | default(None)}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 playbooks]# ansible-playbook create_ur_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
PLAY [Create Universal Replicator Pair]
**********************************************************************
**********************************************************************
*********************************
TASK [hv_rep]
**********************************************************************
**********************************************************************
****************************************
ok: [localhost]
TASK [debug]
**********************************************************************
**********************************************************************
****************************************
ok: [localhost] => {
  "result": {
    "changed": false,
    "failed": false,
    "pair": {
      "ConsistencyGroupId": 10,
      "HexPVOL": "00:00:AA",
      "HexSVOL": "00:02:AB",
      "MirrorId": 1,
      "PVol": 170,
      "PVolSerial": 12345,
      "PrimaryJournalPoolId": 10,
      "SVol": 683,
      "SVolSerial": 123456,
      "Status": "PAIR",
      "Type": "UR"
   }
  }
}
PLAY RECAP
**********************************************************************
**********************************************************************
*********************************************
localhost : ok=2 changed=0 unreachable=0
failed=0 skipped=0 rescued=0 ignored=0
```

## Split UR pair

**Module: hv_rep**

The `split_ur_pair.yml` playbook splits the UR pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication Type | Yes | copy_type=ur |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to split a UR pair<br>Set `state` to absent to delete a UR pair |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | Secondary volume in decimal or HEX | Yes | N/A |
| target_storage_serial | integer | Remote storage system serial number | Yes | N/A |
| enable_read_write | boolean | Set the remote S-VOL to read-write mode. | No | Default value: false |
| subtask | string | subtask: split | Yes | N/A |

**Example: Split Universal Replicator Pair**

```
name: Split Universal Replicator Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - target_storage_serial: 123456
    - lun: 170
    - target_lun: 683
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: ur
          lun: '{{lun}}'
          subtask: split
          target_lun: '{{target_lun}}'
          target_storage_serial: '{{target_storage_serial}}'
```

```
          enable_read_write: '{{enable_read_write | default(None)}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook split_ur_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Split Universal Replicator Pair]
*********************************************************************
*****

TASK [hv_rep]
*********************************************************************
*****************************
changed: [localhost]

TASK [debug]
*********************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************
*******************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Resync UR pair

**Module: hv_rep**

The `resync_ur_pair.yml` playbook re-synchronizes the UR pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication type | Yes | copy_type=ur |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to resync a UR pair.<br>Set `state` to absent to delete a UR pair. |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | Secondary volume in decimal or HEX | Yes | N/A |
| target_storage_serial | integer | Remote storage system serial number | Yes | N/A |
| copy_pace | integer | Specifies the number of tracks on a RAID system that can be run simultaneously before the primary system accepts another request.<br>Valid input: 1 to 15. | No | Default value: 3 |
| subtask | string | subtask: resync | Yes | N/A |

**Example: Resync Universal Replicator Pair**

```
name: Resync Universal Replicator Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - target_storage_serial: 123456
    - lun: 170
    - target_lun: 683
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
```

```
          copy_type: ur
          subtask: resync
          lun: '{{lun}}'
          target_lun: '{{target_lun}}'
          target_storage_serial: '{{target_storage_serial}}'
          copy_pace: '{{copy_pace | default(None)}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook resync_ur_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Resync Universal Replicator Pair]
****************************************************************************
****

TASK [hv_rep]
****************************************************************************
*****************************
changed: [localhost]

TASK [debug]
****************************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
****************************************************************************
*********************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete UR pair

**Module: hv_rep**

The `delete_ur_pair.yml` playbook deletes the UR pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| copy_type | string | Replication type | Yes | copy_type=ur |
| state | string | default: absent | Yes | N/A |
| lun | integer | Primary volume in decimal or HEX | Yes | N/A |
| target_lun | integer | Secondary volume in decimal or HEX | Yes | N/A |
| target_storage_serial | integer | Remote storage system serial number | Yes | N/A |

**Example: Delete Universal Replicator Pair**

```
name: Delete Universal Replicator Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - target_storage_serial: 123456
    - lun: 170
    - target_lun: 683
  tasks:
    - hv_rep:
        state: absent
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: ur
          lun: '{{lun}}'
          target_lun: '{{target_lun}}'
          target_storage_serial: '{{target_storage_serial}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook delete_ur_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match
'all'

PLAY [Delete Universal Replicator Pair]
```

```
*****************************************************************************
****

TASK [hv_rep]
*****************************************************************************
*****************************
ok: [localhost]

TASK [debug]
*****************************************************************************
*****************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false
    }
}

PLAY RECAP
*****************************************************************************
*********************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create GAD pair

**Module: hv_rep**

The `create_gad_pair_vbox.yml` playbook creates a global-active device pair on the specified Hitachi storage system. The global-active device pair configurations supported are Physical to Virtual Storage Machine (VSM) and VSM to VSM. For Physical to VSM global-active device pairs, the SVOL physical LDEV ID in the VSM is queried the same as a PVOL LDEV ID. If available, the same LUN is used as the SVOL for global-active device pairs. If the physical LDEV ID is not available, any free LUN available in the secondary storage VSM is used as the SVOL. For VSM to VSM global-active device pairs, the SVOL virtual LDEV ID in the VSM is queried the same as a PVOL virtual LDEV ID. If available, the same LUN is used as the SVOL for global-active device pairs. If the LDEV ID is not available, any free LUN available in the secondary storage VSM is used as the SVOL.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | Physical or VSM serial |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| lun | integer | Primary volume of the local storage in decimal or HEX | Yes | Physical LDEV ID in the case of a physical storage serial.<br>Virtual LDEV ID in the case of a VSM serial. |
| target_pool_id | integer | Dynamic Pool ID of the remote storage | Yes | N/A |
| quorum_disk_id | integer | Quorum disk. It can be in a third and external storage system or in an iSCSI-attached host server. It is used to monitor the global-active device pair volumes. | Yes | N/A |
| copy_type | string | Replication type | Yes | The default type is gad for global-active device-related operations |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to create and update.<br>Set `state` to absent to delete |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| target_host_group | string | Host group in the remote storage | No | If not provided, the new SVOL would be mapped to the existing host group HITACHI_HG on the target resource group. If the default host group HITACHI_HG is not available, then the operation is aborted. |
| target_resource_group_id | integer | Resource group in the remote storage | No | If not provided, first resource group with PVOL serial is selected. |
| consistency_group_id | Integer | Consistency group to which the newly-created snapshot pair needs to be added. | No | N/A |
| allocate_consistency_group | string | Possible values:<br>▪ true<br>▪ false<br>The consistency group to which the newly-created snapshot pair needs to be added. | No | If true, get a free consistency group ID and assign the newly-created remote clone to it. |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| copy_pace | string | Specifies the number of tracks on a RAID system that can be run simultaneously before the primary system accepts another request. | No | Default value: 3<br>Valid input: 1 to 15. |
| target_storage_serial | integer | Storage system serial number of the remote storage | No | N/A |

**Example: Create global-active device pair with target pool ID**

```
  name: Create GAD Pair with target pool id
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - lun: 1052
    - target_pool_id: 26
    - quorum_disk_id: 20
    - target_resource_group_id: 4
    - consistency_group_id: 7
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: gad

          lun: '{{lun}}'
          target_pool_id: '{{target_pool_id}}'
          quorum_disk_id: '{{ quorum_disk_id }}'

          target_host_group: '{{ target_host_group | default(None)}}'
          target_storage_serial: '{{ target_storage_serial |
default(None)}}'
          target_resource_group_id: '{{ target_resource_group_id |
default(None)}}'
          consistency_group_id: '{{consistency_group_id  |
default(None) }}'
          allocate_consistency_group: '{{allocate_consistency_group |
default(None)}}'
          copy_pace: '{{ copy_pace | default(None)}}'

      register: result
    - debug: var=result
```

**Output**

```
[root@localhost playbooks]# ansible-playbook create_gad_pair_vbox.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Create GAD Pair with target pool id]
*********************************************************************
*********************************************************************
*************************************


TASK [hv_rep]
*********************************************************************
*********************************************************************
*********************************************************************
changed: [localhost]


TASK [debug]
*********************************************************************
*********************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false,
        "pair": {
            "ConsistencyGroupId": 7,
            "HexPVOL": "00:04:1C",
            "HexSVOL": "00:03:48",
            "PVol": 1052,
            "SVol": 840,
            "SVolSerial": 10075,
            "Serial": 123456,
            "Status": "COPY",
            "Type": "GAD",
            "VSMSerial": 123456,
            "VirtualLogicalUnitId": 1052
        }
    }
}


PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
```

```
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create multiple GAD pairs

**Module: hv_rep**

The `create_gad_pair_vbox_multi.yml` playbook creates a global-active device pair with a target pool ID on the specified Hitachi storage system.

**Input Parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| luns | array | Primary volumes in decimal or HEX of the local storage | Yes | N/A |
| target_pool_id | integer | Pool ID of the remote storage | Yes | N/A |
| quorum_disk_id | integer | Quorum disk, can be in a third and external storage system or in an iSCSI-attached host server. It is used to monitor the global-active device pair volumes. | Yes | N/A |
| copy_type | string | Replication type | Yes | Value: gad |
| state | string | values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to create and update<br>Set `state` to absent to delete |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| target_host_group | string | Host group in the remote storage | No | If not provided, the new SVOL is mapped to the existing host group HITACHI_HG on the target resource group. If the default host group HITACHI_HG is not available then the operation is cancelled. |
| target_resource_group_id | integer | Resource group in the remote storage | No | N/A |
| consistency_group_id | integer | Consistency group to which the newly created snapshot pair needs to be added. | No | N/A |
| allocate_consistency_group | string | Consistency group to which the newly created snapshot pair needs to be added. | No | If true, get a Free Consistency Group ID and assign a new created Remote Clone to it. |
| copy_pace | string | Specifies the number of tracks on a RAID system that can be run simultaneously before the primary system accepts another request. | No | Default value: 3<br>Valid input: 1 to 15. |
| target_storage_serial | integer | Storage serial of the remote storage | No | N/A |

**Example: Create multiple global-active device pairs with target pool ID**

```
name: Create multiple GAD Pairs with target pool id
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - luns:
      - 8014
      - 1052
    - target_pool_id: 26
    - quorum_disk_id: 20
    - target_resource_group_id: 4
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: gad

          lun: '{{ item }}'
          target_pool_id: '{{target_pool_id}}'
          quorum_disk_id: '{{ quorum_disk_id }}'

          target_host_group: '{{ target_host_group | default(None)}}'
          target_storage_serial: '{{ target_storage_serial |
default(None)}}'
          target_resource_group_id: '{{ target_resource_group_id |
default(None)}}'
          consistency_group_id: '{{consistency_group_id  |
default(None) }}'
          allocate_consistency_group: '{{allocate_consistency_group |
default(None)}}'
          copy_pace: '{{ copy_pace | default(None)}}'

      with_items: '{{ luns }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@172 block]# ansible-playbook create_gad_pair_vbox_multi.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'



PLAY [Create multiple GAD Pairs with target pool id]
```

```
*********************************************************************************
*********************************************************************************
*****************************


TASK [hv_rep]
*********************************************************************************
*********************************************************************************
*********************************************************************
changed: [localhost] => (item=8014)
changed: [localhost] => (item=1052)



TASK [debug]
*********************************************************************************
*********************************************************************************
*********************************************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "msg": "All items completed",
        "results": [
            {
                "ansible_loop_var": "item",
                "changed": true,
                "failed": false,
                "invocation": {
                    "module_args": {
                        "data": "{\"target_storage_serial\": \"\",
\"target_host_group\": \"\", \"quorum_disk_id\": 20, \"copy_pace\": \"\",
\"consistency_group_id\": \"\", \"target_pool_id\": 26,
\"target_resource_group_id\": \"4\", \"copy_type\": \"gad\",
\"allocate_consistency_group\": \"\", \"lun\": 8014}",
                        "state": "present",
                        "storage_serial": 123456
                    }
                },
                "item": 8014,
                "pair": {
                    "ConsistencyGroupId": -1,
                    "HexPVOL": "00:1F:4E",
                    "HexSVOL": "00:03:4F",
                    "PVol": 8014,
                    "SVol": 847,
                    "SVolSerial": 123456,
                    "Serial": 123456,
                    "Status": "COPY",
                    "Type": "GAD",
                    "VSMSerial": 123456,
```

```
                    "VirtualLogicalUnitId": 8014
                }
            },
            {
                "ansible_loop_var": "item",
                "changed": true,
                "failed": false,
                "invocation": {
                    "module_args": {
                        "data": "{\"target_storage_serial\": \"\",
\"target_host_group\": \"\", \"quorum_disk_id\": 20, \"copy_pace\": \"\",
\"consistency_group_id\": \"\", \"target_pool_id\": 26,
\"target_resource_group_id\": \"4\", \"copy_type\": \"gad\",
\"allocate_consistency_group\": \"\", \"lun\": 1052}",
                        "state": "present",
                        "storage_serial": 123456
                    }
                },
                "item": 1052,
                "pair": {
                    "ConsistencyGroupId": -1,
                    "HexPVOL": "00:04:1C",
                    "HexSVOL": "00:03:5B",
                    "PVol": 1052,
                    "SVol": 859,
                    "SVolSerial": 123456,
                    "Serial": 123456,
                    "Status": "COPY",
                    "Type": "GAD",
                    "VSMSerial": 123456,
                    "VirtualLogicalUnitId": 1052
                }
            }
        ]
    }
}



PLAY RECAP
*********************************************************************
*********************************************************************
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Split GAD pair

**Module: hv_rep**

The `split_gad_pair.yml` playbook splits a global-active device pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | |
| lun | integer | Primary volume in decimal or HEX of the local storage | Yes | |
| copy_type | string | Replication type | Yes | Default type is gad for global-active device-related operations |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to create and update.<br>Set `state` to absent to delete. |

**Example: Split global-active device pair**

```
name: Split GAD Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - lun: 69
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: gad
          subtask: split
          lun: '{{lun}}'
```

```
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost playbooks]# ap split_gad_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
PLAY [Split GAD Pair]
*********************************************************************
*********************************************************************
*********************************
TASK [hv_rep]
*********************************************************************
*********************************************************************
****************************************
changed: [localhost]
TASK [debug]
*********************************************************************
*********************************************************************
****************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
****************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Resync GAD pair

**Module: hv_rep**

The `resync_gad_pair.yml` playbook resyncs a global-active device pair on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| lun | integer | Primary volume in decimal or HEX of the local storage | Yes | |
| copy_type | string | Replication type | Yes | Default type is gad for global-active device-related operations |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to create and update.<br>Set `state` to absent to delete. |

**Example: Resync global-active device pair**

```
name: Resync GAD Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - lun: 69
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: gad
          subtask: resync
          lun: '{{lun}}'
      register: result
    - debug: var=result
```

**Output**

```
 [root@localhost playbooks]# ap resync_gad_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
PLAY [Resync GAD Pair]
*********************************************************************
```

```
*********************************************************************************
*********************************
TASK [hv_rep]
*********************************************************************************
*********************************************************************************
****************************************
changed: [localhost]
TASK [debug]
*********************************************************************************
*********************************************************************************
****************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}
PLAY RECAP
*********************************************************************************
*********************************************************************************
*********************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete GAD pair

**Module: hv_rep**

The `delete_gad_pair.yml` playbook deletes a global-active device pair on the specified Hitachi storage system. Post global-active device pair deletion, the SVOL should be unmapped and deleted manually if required.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| lun | integer | Primary volume in decimal or HEX of the local storage | Yes | N/A |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| copy_type | string | Replication type | Yes | Default type is gad for global-active device-related operation |
| state | string | default: absent | Yes | N/A |

**Example: Delete global-active device pair**

```
name: Delete GAD Pair
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 123456
    - lun: 69
  tasks:
    - hv_rep:
        state: absent
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: gad
          lun: '{{lun}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost playbooks]# ap delete_gad_pair.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
PLAY [Delete GAD Pair]
**********************************************************************
**********************************************************************
*********************************
TASK [hv_rep]
**********************************************************************
**********************************************************************
*****************************************
changed: [localhost]
TASK [debug]
**********************************************************************
**********************************************************************
*****************************************
ok: [localhost] => {
    "result": {
        "PVOL": 69,
```

```
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*****************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Manage snapshot group pairs

## Get snapshot group

**Module: hv_rep_facts**

The `get_snapshot_group.yml` playbook provides HTI snapshot group pairs information on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| snapshot_grp | string | Snapshot group name | No | If provided, specific group information is displayed. Else all the snapshot group on the specified storage will be listed |
| grouping | string | Group type | Yes | grouping: snapshot group |

**Sample get_snapshot_group.yml playbook**

```
name: Getting Snapshot Group info
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - snapshot_grp: snewar_snap3
  tasks:
    - hv_rep_facts:
```

```
        storage_serial: '{{ storage_serial }}'
        data:
          grouping: snapshot group
          snapshot_grp: '{{snapshot_grp | default(None)}}'
      register: result
    - debug: var=result.pairs
```

**Output**

```
[root@localhost block]# ansible-playbook get_snapshot_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [Getting Snapshot Group info]
*********************************************************************
**************************

TASK [hv_rep_facts]
*********************************************************************
****************************************
ok: [localhost]

TASK [debug]
*********************************************************************
************************************************
ok: [localhost] => {
    "result.pairs": [
        {
            "IsInConsistencyGroup": true,
            "SnapshotGroupName": "snewar_snap3",
            "SnapshotPairs": [
                {
                    "SnapShotGroupName": "snewar_snap3",
                    "consistencyGroupId": 77,
                    "copyRate": 100,
                    "mirrorUnitId": 3,
          "poolId": 92,
                    "primaryHexVolumeId": "00:05:E0",
                    "primaryVolumeId": 1504,
                    "primaryVolumeStorageId": 12345,
                    "secondaryHexVolumeId":
                    "secondaryVolumeId":
                    "secondaryVolumeStorageId": 12345,
                    "splitDateTime": "2020-06-03 11:31:47",
                    "status": "PSUS",
                    "svolAccessMode": "READWRITE",
                    "type": "THIN_IMAGE"
                },
                {
```

```
                    "SnapShotGroupName": "snewar_snap3",
                    "consistencyGroupId": 77,
                    "copyRate": 100,
                    "mirrorUnitId": 3,
                    "poolId": 92,
                    "primaryHexVolumeId": "00:05:E2",
                    "primaryVolumeId": 1506,
                    "primaryVolumeStorageId": 12345,
                    "secondaryHexVolumeId":
                    "secondaryVolumeId":
                    "secondaryVolumeStorageId": 12345,
                    "splitDateTime": "2020-06-03 11:31:47",
                    "status": "PSUS",
                    "svolAccessMode": "READWRITE",
                    "type": "THIN_IMAGE"
}
            ]
        }
```


## Split snapshot group

**Module: hv_rep**

The `split_snapshot_group.yml` playbook splits HTI pairs on the specified snapshot group.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| snapshot_grp | string | Snapshot group name | Yes | N/A |
| grouping | string | Group type | Yes | grouping: snapshot group |
| subtask | string | Subtask type | Yes | subtask: split |
| State | String | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | 1.Set state to present for split snapshot group.<br>2.Set state to absent for deleting snapshot group. |

**Sample split_snapshot_group.yml playbook**

```
  name: Split Snapshot Group
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - snapshot_grp: snewar_snap3
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          grouping: snapshot group
          snapshot_grp: '{{ snapshot_grp }}'
          subtask: split
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook split_snapshot_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [Split Snapshot Group]
**********************************************************************
**********************************

TASK [hv_rep]
**********************************************************************
*********************************************
changed: [localhost]

TASK [debug]
**********************************************************************
*********************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
**********************************************************************
*************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Resync snapshot group

**Module: hv_rep**

The `resync_snapshot_group.yml` playbook resyncs HTI pairs on the specified snapshot group.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| snapshot_grp | string | Snapshot group name | Yes | N/A |
| grouping | string | Group type | Yes | grouping: snapshot group |
| subtask | string | Subtask type | Yes | subtask: resync |
| State | String | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | 1.Set state to present for resync snapshot group.<br>2.Set state to absent for deleting snapshot group. |

**Sample resync_snapshot_group.yml playbook**

```
name: Resync Snapshot Group
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - snapshot_grp: snewar_snap3
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          grouping: snapshot group
          snapshot_grp: '{{ snapshot_grp }}'
          subtask: resync
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook resync_snapshot_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
```

```
that the implicit localhost does not match 'all'


PLAY [Resync Snapshot Group]
*********************************************************************************
*********************************

TASK [hv_rep]
*********************************************************************************
***********************************************
changed: [localhost]

TASK [debug]
*********************************************************************************
***********************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************************
***************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Restore snapshot group

**Module: hv_rep**

The `restore_snapshot_group.yml` playbook restores HTI pairs on the specified snapshot group.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| snapshot_grp | string | Snapshot group name | Yes | N/A |
| grouping | string | Group type | Yes | grouping: snapshot group |
| subtask | string | Subtask type | Yes | subtask: restore |

| | | | | |
|---|---|---|---|---|
| State | String | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | 1.Set state to present for restore snapshot group.<br>2.Set state to absent for deleting snapshot group. |

**Sample restore_snapshot_group.yml playbook**

```
name: Restore Snapshot Group
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - snapshot_grp: snewar_snap3
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          grouping: snapshot group
          snapshot_grp: '{{ snapshot_grp }}'
          subtask: restore
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook restore_snapshot_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [Restore Snapshot Group]
**********************************************************************
*********************************

TASK [hv_rep]
**********************************************************************
*********************************************
changed: [localhost]

TASK [debug]
**********************************************************************
*********************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
```

```
    }
}

PLAY RECAP
*********************************************************************************
************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete snapshot group

**Module: hv_rep**

The `delete_snapshot_group.yml` playbook deletes HTI pairs on the specified snapshot group.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| snapshot_grp | string | Snapshot group name | Yes | N/A |
| grouping | string | Group type | Yes | grouping: snapshot group |
| state | string | Operation type | Yes | State: absent |

**Sample delete_snapshot_group.yml playbook**

```
name: Delete Snapshot Group
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - snapshot_grp: snewar_snap3
  tasks:
    - hv_rep:
        state: absent
        storage_serial: '{{ storage_serial }}'
        data:
          grouping: snapshot group
          snapshot_grp: '{{ snapshot_grp }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook delete_snapshot_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [Delete Snapshot Group]
**********************************************************************
**********************************

TASK [hv_rep]
**********************************************************************
**********************************************
changed: [localhost]

TASK [debug]
**********************************************************************
**********************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
**********************************************************************
**************************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Manage consistency group pairs

## Get consistency group

**Module: hv_rep_facts**

The `get_ctg_pair_group.yml` playbook retrieves replication pairs belonging to the given consistency group ID from the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |

| | | | | |
|---|---|---|---|---|
| consistency_group_id | integer | Consistency group ID | No | N/A |
| grouping | string | Group type | Yes | grouping: consistency group |

**Sample get_ctg_pair_group.yml playbook**

```
name: Getting Consistency Group info
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
  tasks:
    - hv_rep_facts:
        storage_serial: '{{ storage_serial }}'
        data:
          grouping: consistency group
          consistency_group_id: '{{consistency_group_id | default(None)}}'
      register: result
    - debug: var=result.pairs
```

**Output**

```
[root@172 block]# ansible-playbook get_ctg_pair_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Getting Consistency Group info]
*********************************************************************
*********************************************************************
******************

TASK [hv_rep_facts]
*********************************************************************
*********************************************************************
***********************************
ok: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
*****************************************
ok: [localhost] => {
    "result.pairs": [
        {
            "ConsistencyGroupId": 103,
            "HexPVOL": "00:00:AA",
            "HexSVOL": "00:02:AB",
```

```
            "PVol": 170,
            "PVolSerial": 55555,
            "SVol": 683,
            "SVolSerial": 44444,
            "Status": "PAIR",
            "Type": "UR"
        },
        {
            "ConsistencyGroupId": 0,
            "HexPVOL": "00:02:4B",
            "HexSVOL": "00:02:18",
            "PVol": 587,
            "PVolSerial": 415022,
            "SVol": 536,
            "SVolSerial": 415022,
            "Status": "PAIR",
            "Type": "THIN_IMAGE"
        },
        {
            "ConsistencyGroupId": 10,
            "FenceLevel": "NEVER",
            "HexPVOL": "00:FE:05",
            "HexSVOL": "00:BF:05",
            "PVol": 65029,
            "PVolSerial": 415022,
            "SVol": 48901,
            "SVolSerial": 415056,
            "Status": "PAIR",
            "Type": "TRUE_COPY"
        }
    ]
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*****************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Split consistency group

**Module: hv_rep**

The `split_consistency_group.yml` playbook splits replication pairs belonging to the given consistency group ID from the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| consistency_group_id | integer | Consistency group ID | Yes | N/A |
| copy_type | string | Replication type | Yes | copy_type: gad/tc/ur |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | 1. Set state to present for splitting a consistency group.<br>2. Set state to absent for deleting a consistency group. |
| subtask | string | subtask: split | Yes | N/A |

**Sample split_consistency_group.yml playbook**

```
name: Split Consistency Group
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - consistency_group_id: 3
    - copy_type: gad
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: '{{ copy_type }}'
          subtask: split
          grouping: consistency group
          consistency_group_id: '{{ consistency_group_id }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook split_consistency_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Split Consistency Group]
*********************************************************************
```

```
*********************************************************************************
*************************

TASK [hv_rep]
*********************************************************************************
*********************************************************************************
****************************************
changed: [localhost]

TASK [debug]
*********************************************************************************
*********************************************************************************
****************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
********************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Resync consistency group

**Module: hv_rep**

The `resync_consistency_group.yml` playbook resynchronizes replication pairs belonging to the given consistency group ID from the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| consistency_group_id | integer | Consistency group ID | Yes | N/A |
| copy_type | string | Replication type | Yes | copy_type: gad/tc/ur |
| state | string | Possible values:<br>▪ present<br>▪ absent | Yes | 1. Set state to present for resyncing a consistency group. |

| | | | | |
|---|---|---|---|---|
| | | default: present | | 2. Set state to absent for deleting a consistency group. |
| subtask | string | subtask: resync | Yes | N/A |

**Sample resync_consistency_group.yml playbook**

```
name: Resync Consistency Group
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - consistency_group_id: 3
    - copy_type: gad
  tasks:
    - hv_rep:
        state: present
        storage_serial: '{{ storage_serial }}'
        data:
          copy_type: '{{ copy_type }}'
          subtask: resync
          grouping: consistency group
          consistency_group_id: '{{ consistency_group_id }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook resync_consistency_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Resync Consistency Group]
*********************************************************************
*********************************************************************
*************************

TASK [hv_rep]
*********************************************************************
*********************************************************************
****************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
****************************************
ok: [localhost] => {
```

```
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*******************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete consistency group

**Module: hv_rep**

The `delete_consistency_group.yml` playbook deletes replication pairs belonging to the given consistency group ID from the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Storage system serial number | Yes | N/A |
| consistency_group_id | integer | Consistency group ID | Yes | N/A |
| copy_type | string | Replication type | Yes | copy_type: gad/tc/ur |
| state | string | default: absent | Yes | N/A |
| grouping | string | Group type | Yes | grouping:consistency group |

**Sample delete_consistency_group.yml playbook**

```
name: Delete Consistency Group
  hosts: localhost
  gather_facts: false
  vars:
    - storage_serial: 12345
    - consistency_group_id: 3
    - copy_type: gad
  tasks:
    - hv_rep:
        state: absent
        storage_serial: '{{ storage_serial }}'
```

```
        data:
          copy_type: '{{ copy_type }}'
          grouping: consistency group
          consistency_group_id: '{{ consistency_group_id }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost block]# ansible-playbook delete_consistency_group.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Delete Consistency Group]
************************************************************************
************************************************************************
*************************

TASK [hv_rep]
************************************************************************
************************************************************************
*****************************************
changed: [localhost]

TASK [debug]
************************************************************************
************************************************************************
*****************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false
    }
}

PLAY RECAP
************************************************************************
************************************************************************
*****************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Manage virtual storage machines

**Module: hv_vsm**

This module manages Virtual Storage Machines(VSM) on the specified Hitachi storage system

For more info, run **`ansible-doc hv_vsm`** and **`ansible-doc hv_vsm_facts`** to view the documentation for the module.

## Get VSM

**Module: hv_vsm_facts**

The `get_vsm.yml` playbook provides details about the virtual storage machines (VSM) on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| storage_serial | integer | Physical Storage system serial number | No | |
| virtual_storage_serial | integer | Serial number of the VSM | Yes | |

**Example: Get VSM Info**

```
name: Get VSM Info
  hosts: localhost
  gather_facts: false
  vars:
    - virtual_storage_serial: 411223
  tasks:
    - hv_vsm_facts:
        storage_serial: '{{ storage_serial  | default(omit) }}'
        data:
          virtual_storage_serial: '{{virtual_storage_serial}}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost playbooks]# ap get_vsm.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
PLAY [Get VSM Info]
*********************************************************************
*********************************************************************
***********************************
TASK [hv_vsm_facts]
*********************************************************************
*********************************************************************
***********************************
```

```
ok: [localhost]

TASK [debug]
*******************************************************************************
*******************************************************************************
*****************************************
ok: [localhost] => {
    "result": {
        "changed": false,
        "failed": false,
        "vsm": {
            "HasMetaResource": false,
            "ResourceGroups": [
                {
                    "HostGroups": [
                        {
                            "HgName": "HITACHI_HG2",
                            "HostMode": 3,
                            "HostgroupID": 8,
                            "Paths": [],
                            "Port": "CL2-B",
                            "ResourceGroupId": 6,
                            "WWNS": [],
                            "hostModeOptions": []
                        },
                        {
                            "HgName": "HITACHI_HG3",
                            "HostMode": 3,
                            "HostgroupID": 9,
                            "Paths": [],
                            "Port": "CL2-B",
                            "ResourceGroupId": 6,
                            "WWNS": [],
                            "hostModeOptions": []
                        }
                    ],
                    "IsLocked": false,
                    "LogicalUnits": [
                        {
                            "DynamicPool": 6,
                            "EmulationType": "OPEN-V-CVS",
                            "FormatOrShredRate": 100,
                            "HexLunNumber": "00:02:AA",
                            "IsCommandDevice": false,
                            "IsDMLU": false,
                            "IsDuplicated": false,
                            "IsDynamicPoolVolume": false,
                            "IsInGADPair": false,
                            "IsJournalPoolVolume": false,
                            "IsVVOL": false,
                            "IsVirtualLogicalUnit": false,
```

```
                            "LunNumber": 682,
                            "MetaResourceSerialNumber": 415056,
                            "Name": "klau-clone-test",
                            "ParityGroup": "",
                            "PathCount": 0,
                            "PhysicalLunNumber": 682,
                            "PoolId": "6",
                            "PoolType": 1,
                            "RG": 0,
                            "RaidLevel": 0,
                            "ResourceGroupId": 6,
                            "Status": 1,
                            "TotalCapacity": 1024,
                            "UsedCapacity": 0,
                            "VirtualLunNumber": -1,
                            "VirtualStorageDeviceId": 411223,
                            "stripeSize": 512
                        }
                    ],
                    "MetaResourceSerial": 415056,
                    "ParityGroups": [],
                    "Ports": [],
                    "ResourceGroupId": 6,
                    "ResourceGroupName": "ansible-11223",
                    "UnDefinedLuns": [],
                    "VirtualDeviceId": 411223
                },
                {
                    "HostGroups": [],
                    "IsLocked": false,
                    "LogicalUnits": [],
                    "MetaResourceSerial": 415022,
                    "ParityGroups": [],
                    "Ports": [],
                    "ResourceGroupId": 1,
                    "ResourceGroupName": "ansible-11223",
                    "UnDefinedLuns": [],
                    "VirtualDeviceId": 411223
                }
            ],
            "Serial": 411223,
            "StorageModel": "VSP_F370",
            "StorageType": "VSP_FX00"
        }
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*****************************************
```

```
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Create VSM

**Module: hv_vsm**

The `create_vsm.yml` module creates a virtual storage machine (VSM) on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to create and update a VSM.<br>Set `state` to absent to delete a VSM. |
| virtual_storage_serial | integer | Serial number of the VSM | Yes | N/A |
| model | string | Model of the VSM to be created | Yes | Supported models:<br>▪ VSP_F350<br>▪ VSP_F370<br>▪ VSP_F400<br>▪ VSP_F600<br>▪ VSP_F700<br>▪ VSP_F800<br>▪ VSP_F900<br>▪ VSP_F1500<br>▪ VSP_G130<br>▪ VSP_G150<br>▪ VSP_G200<br>▪ VSP_G350<br>▪ VSP_G370<br>▪ VSP_G400<br>▪ VSP_G600 |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| | | | | ▪ VSP_G700<br>▪ VSP_G800<br>▪ VSP_G900<br>▪ VSP_G1000<br>▪ VSP_G1500<br>▪ VSP_5100<br>▪ VSP_5500 |
| meta_resources | | Valid variables:<br>▪ serial<br>▪ name<br>▪<br>▪ host_groups<br>◦ port<br>◦ host_group_names<br>▪ parity_groups<br>▪ ports<br>▪ luns | Yes | Variable details are given in next rows. |
| serial | integer | Serial of the physical storage where the VSM is created | Yes | N/A |
| name | string | Name of the new VSM | No | Default: Physical storage serial will be used as name. |
| port | string | Port on which the host groups are available | No | N/A |
| host_group_names | array | Host groups to be added to VSM | No | N/A |
| luns | array | LUN ID in decimal or HEX of the LUNs to be added to VSM | No | LUNs can be existing/ created LUNs or free LUN IDs |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| ports | array | Ports to be added to VSM | No | N/A |
| parity_groups | array | Parity groups to be added to VSM | No | N/A |

**Example: Testing Create VSM**

```
name: Testing Create VSM
  hosts: localhost
  gather_facts: false
  vars:
    - state: present
    - virtual_storage_serial: 411229
    - model: VSP_F400
    - meta_resources:
      -
        serial: 415056
        name: ansible-229
        host_groups:
          -
            port: CL2-B
            host_group_names:
              - HITACHI_HG1
        parity_groups:
          - 5-5
        ports:
          - CL8-B
        luns:
          - 679
          - 680

  tasks:
    - hv_vsm:
        state: present
        virtual_storage_serial: '{{ virtual_storage_serial }}'
        physical_storage_serial: '{{ physical_storage_serial |
default(omit) }}'
        data:
          model: '{{ model }}'
          state: '{{ state | default(omit) }}'
          meta_resources: '{{ meta_resources }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost playbooks]# ap create_vsm.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
PLAY [Testing Create VSM]
*********************************************************************
*********************************************************************
*****************************
TASK [hv_vsm]
*********************************************************************
*********************************************************************
****************************************
changed: [localhost]
TASK [debug]
*********************************************************************
*********************************************************************
****************************************
ok: [localhost] => {
    "result": {
        "changed": true,
        "failed": false,
        "storageSystem": {}
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
********************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete VSM

**Module: hv_vsm**

The `delete_vsm.yml` playbook deletes a virtual storage machine on the specified Hitachi storage system.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| physical_storage_serial | integer | Storage system serial number | No | N/A |
| virtual_storage_serial | integer | Serial number of the VSM | Yes | N/A |

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: absent | No | Set `state` to present to create and update.<br>Set `state` to absent to delete. |

**Example: Testing Delete VSM**

```
name: Testing Delete VSM
  hosts: localhost
  gather_facts: false
  vars:
    - virtual_storage_serial: 411229
  tasks:
    - hv_vsm:
        state: absent
        virtual_storage_serial: '{{ virtual_storage_serial }}'
        physical_storage_serial: '{{ physical_storage_serial |
default(omit) }}'
      register: result
    - debug: var=result
```

**Output**

```
[root@localhost playbooks]# ap delete_vsm.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'
PLAY [Testing Delete VSM]
*********************************************************************
*********************************************************************
*******************************
TASK [hv_vsm]
*********************************************************************
*********************************************************************
****************************************
changed: [localhost]

TASK [debug]
*********************************************************************
*********************************************************************
****************************************
ok: [localhost] => {
    "result": {
        "changed": true,
```

```
        "failed": false
    }
}

PLAY RECAP
*********************************************************************
*********************************************************************
*******************************************
localhost                  : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Chapter 6: File storage modules

The file storage modules are located in the `/opt/hitachi/ansible/modules/file` directory.

## Get HNAS node or cluster

### Module: hv_fs_facts

The `get_file_server.yml` playbook retrieves the HNAS node or cluster details.

For more info, run **`ansible-doc hv_fs_facts`** to view the documentation for the module.

#### Input parameters

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| fileServerIP | string | HNAS node or cluster IP address | Yes | N/A |

#### Example Testing Get HNAS file server

```
name: Testing Get HNAS Fileserver
  hosts: localhost
  gather_facts: false
  vars:
    - fileServerIP: 172.25.41.67
  tasks:
    - hv_fs_facts:
        data:
          fileServerIP: '{{ fileServerIP }}'
      register: result
    - debug: var=result.hnas
```

#### Output

```
[root@172 file]# ansible-playbook get_file_server.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Get HNAS Fileserver]
```

```
*********************************************************************************
***********************

TASK [hv_fs_facts]
*********************************************************************************
************************************
ok: [localhost]

TASK [debug]
*********************************************************************************
*****************************************
ok: [localhost] => {
    "result.hnas": {
        "ClusterHealth": "ROBUST",
        "FileServer": "172.25.41.67",
        "GroupIdentifier": "a51d0c60-f11a-41ec-aa40-ee6db8485e1b",
        "IsClustered": true,
        "Name": "G600-440726",
        "NodeCount": 2,
        "Nodes": [
            {
                "Firmware": "13.5.5527.02",
                "IPAddresses": [
                    "10.0.0.20",
                    "192.168.48.1"
                ],
                "Model": "G600",
                "Name": "G600-440726-1",
                "NodeId": 1,
                "Serial": "440726",
                "Status": "ONLINE"
            },
            {
                "Firmware": "13.5.5527.02",
                "IPAddresses": [
                    "10.0.0.21",
                    "192.168.52.1"
                ],
                "Model": "G600",
                "Name": "G600-440726-2",
                "NodeId": 2,
                "Serial": "440726",
                "Status": "ONLINE"
            }
        ]
    }
}

PLAY RECAP
*********************************************************************************
*********************************************
```

```
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Get a list of HNAS file systems

**Module: hv_file_systems_facts**

The `get_file_systems.yml` playbook retrieves a list of HNAS file systems.

For more info, run **`ansible-doc hv_file_systems_facts`** to view the documentation for the module.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| fileServerIP | string | HNAS node or cluster IP address | Yes | N/A |
| file_system | string | HNAS file system | No | N/A |

**Example: Testing Get HNAS file systems**

```
name: Testing Get HNAS Filesystems
  hosts: localhost
  gather_facts: false
  vars:
    - fileServerIP: 172.25.41.67
  tasks:
    - hv_file_systems_facts:
        data:
          fileServerIP: '{{ fileServerIP }}'
          file_system: '{{ file_system | default(None) }}'
      register: result
    - debug: var=result.hnas
```

**Output**

```
[root@172 file]# ansible-playbook get_file_systems.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Get HNAS Filesystems]
*********************************************************************
**********************

TASK [hv_file_systems_facts]
*********************************************************************
****************************
```

```
ok: [localhost]

TASK [debug]
*******************************************************************************
*****************************************
ok: [localhost] => {
    "result.hnas": [
        {
            "BlockSizeInKB": 0,
            "DedupeEnabled": "False",
            "DedupeSupported": "False",
            "EVSIP": "",
            "EVSName": "",
            "ExpansionLimitInGB": "10GB",
            "FileSystemLabel": "UCP_automation_sanjeev",
            "FileSystemStatus": "VOLUME_NOT_AVAILABLE_TO_BS",
            "FreeCapacityInGB": "0GB",
            "NDMPRecoveryTarget": "False",
            "NonStrictWORM": "",
            "ReadCache": "",
            "StoragePool": "",
            "SysLocked": "",
            "TotalCapacityInGB": "3GB",
            "UnlimitedExpansion": "",
            "UsedCapacityInGB": "0GB",
            "WORM": "False"
        },
        {
            "BlockSizeInKB": 0,
            "DedupeEnabled": "False",
            "DedupeSupported": "False",
            "EVSIP": "",
            "EVSName": "",
            "ExpansionLimitInGB": "17179869183GB",
            "FileSystemLabel": "new_fs_dra_01",
            "FileSystemStatus": "VOLUME_NOT_AVAILABLE_TO_BS",
            "FreeCapacityInGB": "0GB",
            "NDMPRecoveryTarget": "False",
            "NonStrictWORM": "",
            "ReadCache": "",
            "StoragePool": "",
            "SysLocked": "",
            "TotalCapacityInGB": "3GB",
            "UnlimitedExpansion": "",
            "UsedCapacityInGB": "0GB",
            "WORM": "False"
        },
        {
            "BlockSizeInKB": 4,
            "DedupeEnabled": "True",
            "DedupeSupported": "True",
```

```
            "EVSIP": "192.0.2.77",
            "EVSName": "zeinhvs",
            "ExpansionLimitInGB": "36GB",
            "FileSystemLabel": "UCS-FS-source",
            "FileSystemStatus": "MOUNTED",
            "FreeCapacityInGB": "15GB",
            "NDMPRecoveryTarget": "False",
            "NonStrictWORM": "",
            "ReadCache": "False",
            "StoragePool": "UCS-StoragePool-Source",
            "SysLocked": "False",
            "TotalCapacityInGB": "18GB",
            "UnlimitedExpansion": "",
            "UsedCapacityInGB": "2GB",
            "WORM": "False"
        }
    ]
}

PLAY RECAP
*********************************************************************************
*********************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Get a list of enterprise virtual servers

**Module: hv_evs_facts**

The `get_evs.yml` playbook retrieves a list of enterprise virtual servers (EVS).

For more info, run **`ansible-doc hv_evs_facts`** to view the documentation for the module.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| fileServerIP | string | HNAS node or cluster IP address | Yes | N/A |
| evs | string | Enterprise virtual servers | No | N/A |

**Example: Testing Get HNAS EVS**

```
name: Testing Get HNAS EVS
  hosts: localhost
  gather_facts: false
  vars:
```

```
    - fileServerIP: 172.25.41.67
    - evs: 200.200.200.200
  tasks:
    - hv_evs_facts:
        data:
          fileServerIP: '{{ fileServerIP }}'
          evs: '{{ evs  | default(None)  }}'
      register: result
    - debug: var=result.hnas
```

**Output**

```
[root@172 file]# ansible-playbook get_evs.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Get HNAS EVS]
*****************************************************************************
*****************************

TASK [hv_evs_facts]
*****************************************************************************
************************************
ok: [localhost]

TASK [debug]
*****************************************************************************
*******************************************
ok: [localhost] => {
    "result.hnas": [
        {
            "IPAddress": "200.200.200.200",
            "Name": "evs-sanjeev",
            "status": "ONLINE"
        }
    ]
}

PLAY RECAP
*****************************************************************************
*******************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Manage NFS exports

## Create an NFS export

**Module: hv_nfs_share**

The `create_nfs_share.yml` playbook creates an NFS export.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| fileServerIP | string | HNAS node or cluster IP address | Yes | N/A |
| evs | string | Enterprise virtual servers | Yes | N/A |
| file_system | string | HNAS file system | Yes | N/A |
| share_name | string | NFS export name | Yes | N/A |
| path | string | Path to export | Yes | For example: `/path1/path2`<br>**Note**: The virtual volume root directory cannot be the root directory of the file system. |
| access_config | string | Access config | Yes | N/A |
| state | string | Possible values:<br>▪ present<br>▪ absent<br>default: present | Yes | Set `state` to present to create an HNAS NFS export. Set `state` to absent to delete an HNAS NFS export. |

**Example: Testing Create HNAS NFS Export**

```
name: Testing Create HNAS NFS Export
  hosts: localhost
  gather_facts: false
  vars:
    - fileServerIP: 172.25.41.67
    - evs: 200.200.200.200
    - file_system: test
    - share_name: ansible-test-nfs-export2406-test2
    - path: /Sample52
    - access_config: "*"
  tasks:
```

```
    - hv_nfs_share:
        state: present
        data:
          fileServerIP: '{{ fileServerIP }}'
          evs: '{{ evs  }}'
          file_system: '{{ file_system }}'
          share_name: '{{ share_name }}'
          path: '{{ path }}'
          access_config: '{{ access_config }}'
      register: result
    - debug: var=result.hnas
```

**Output**

```
[root@172 file]# ansible-playbook create_nfs_share.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Create HNAS NFS Export]
**********************************************************************
********************

TASK [hv_nfs_share]
**********************************************************************
************************************
ok: [localhost]

TASK [debug]
**********************************************************************
*****************************************
ok: [localhost] => {
    "result.hnas": {
        "AccessConfig": "*",
        "Evs": "200.200.200.200",
        "FileSystemName": "test",
        "FileSystemPath": "/Sample52",
        "Name": "/ansible-test-nfs-export2406-test2",
        "ReadCacheOption": 0,
        "SnapshotOption": 0,
        "transferToReplication": 0
    }
}

PLAY RECAP
**********************************************************************
*****************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Delete an NFS export

**Module: hv_nfs_share**

The `delete_nfs_share.yml` playbook deletes an NFS export.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| fileServerIP | string | HNAS node or cluster IP address | Yes | N/A |
| evs | string | Enterprise virtual servers | Yes | N/A |
| state | string | Default: absent | Yes | N/A |
| share_name | string | NFS export name | Yes | N/A |

**Example: Testing Delete HNAS NFS Export**

```
name: Testing Delete HNAS NFS Export
  hosts: localhost
  gather_facts: false
  vars:
    - fileServerIP: 172.25.41.67
    - evs: 200.200.200.200
    - share_name: ansible-test-nfs-export2406-test2
  tasks:
    - hv_nfs_share:
        state: absent
        data:
          fileServerIP: '{{ fileServerIP }}'
          evs: '{{ evs  }}'
          share_name: '{{ share_name }}'
      register: result
    - debug: var=result.hnas
```

**Output**

```
[root@172 file]# ansible-playbook delete_nfs_share.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Delete HNAS NFS Export]
*********************************************************************
********************

TASK [hv_nfs_share]
*********************************************************************
************************************
```

```
ok: [localhost]

TASK [debug]
*********************************************************************************
*****************************************
ok: [localhost] => {
    "result.hnas": ""
}

PLAY RECAP
*********************************************************************************
*****************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

## Get an NFS export

**Module: hv_nfs_share_facts**

The `get_nfs_shares.yml` playbook gets NFS export information.

**Input parameters**

| Name | Type | Description | Required | Comments |
|---|---|---|---|---|
| fileServerIP | string | HNAS node or cluster IP address | Yes | N/A |
| evs | string | Enterprise virtual servers | No | N/A |
| share_name | string | NFS export name | No | N/A |

**Example: Testing Get HNAS NFS Exports**

```
name: Testing Get HNAS NFS Exports
  hosts: localhost
  gather_facts: false
  vars:
    - fileServerIP: 172.25.41.67
    - evs: 200.200.200.200
    - share_name: /ansible-test-nfs-export2406-test2
  tasks:
    - hv_nfs_share_facts:
        data:
          fileServerIP: '{{ fileServerIP }}'
          evs: '{{ evs  | default(None)  }}'
          share_name: '{{ share_name | default(None) }}'
      register: result
    - debug: var=result.hnas
```

**Output**

```
[root@172 file]# ansible-playbook get_nfs_shares.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Testing Get HNAS NFS Exports]
*********************************************************************************
**********************

TASK [hv_nfs_share_facts]
*********************************************************************************
*******************************
ok: [localhost]

TASK [debug]
*********************************************************************************
*******************************************
ok: [localhost] => {
    "result.hnas": [
        {
            "IPAddress": "200.200.200.200",
            "Name": "evs-sanjeev",
            "nfsExports": [
                {
                    "AccessConfig": "*",
                    "EVSIP": "200.200.200.200",
                    "FileSystemName": "test",
                    "FileSystemPath": "/Sample52",
                    "Name": "/ansible-test-nfs-export2406-test2",
                    "ReadCacheOption": "DISABLED",
                    "SnapshotOption": "SSOPTION_HIDE_AND_DISABLE_ACCESS",
                    "transferToReplication": "DONOTTRANSFER"
                }
            ],
            "status": "ONLINE"
        }
    ]
}

PLAY RECAP
*********************************************************************************
*******************************************
localhost                  : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

# Chapter 7: Logging and Troubleshooting

## Logging

### Hitachi API Gateway Service Logs

The log files reside at the following locations on the Hitachi API Gateway Service VM:

`/var/log/hitachi/ansible/storage-management-service/hv_management_service.log/var/log/hitachi/ansible/hv_storage_modules.log`

### Hitachi Peer Service Log

The log file resides at the following location on the Hitachi Peer Service VM:

`/var/log/puma.log`

### Hitachi Storage VI Service Log

The log file originates at the following location by default:

`/var/log/hitachi/hitachi_vi/vi_service on the CentOS host/VM` where Hitachi API Gateway Service is installed.

For troubleshooting, see the log file `VI-ServiceLog.log` at that location.

### Log Levels

The Hitachi API Gateway Service supports these log levels:

- ALL
- DEBUG
- INFO
- WARN
- ERROR
- OFF

The default log level is INFO

### Modifying the Hitachi API Gateway Service log level

To change the Hitachi API Gateway Service log level, follow these steps:

1. Open the `hilogger-ansible.config` file from `/opt/hitachi/ansible/storage_management_services/Configuration`

2. Modify the “level value” parameter in the <log4net> xml tag.
3. Restart the Hitachi API Gateway Service daemon for this to take effect by running the `/opt/hitachi/ansible/storage_management_services/hv-infra-gateway_service_restart.sh` script.

### Hitachi API Gateway Service Log Bundle Script Creation

This log collecting script collects all logs from the different services and all the relevant configuration files for further troubleshooting. The output of the script is an .tar.gz archive that contains this information.

On the Hitachi API Gateway Service Linux VM that contains the installer, the script is located at `/opt/hitachi/ansible/support`. The name of the playbook is: `get_support_logs.yml`.

```
[root@localhost block]# ansible-playbook get_support_logs.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'

PLAY [Collecting Hitachi API Gateway Service Support Log Bundle]
*********************************************************************
***********************************TASK [hv_troubleshooting_facts]
*********************************************************************
***********************************ok: [localhost]TASK [debug]
*********************************************************************
***********************************ok: [localhost] => {
"result": {
"changed": false,
"failed": false,
"filename": "/opt/hitachi/ansible/support/
hitachi_ansible_02.0.0_support_artifacts_2019_12_21_11_22_06.zip"
}
}
PLAY RECAP
*********************************************************************
****************************localhost : ok=2 changed=0 unreachable=0
failed=0 skipped=0 rescued=0 ignored=0
```

SSH needs to be enabled on the Linux VM.

After executing the script, the resulting .zip archive is named:

`hitachi_ansible_02.0.0_support_artifacts_yyyy_mm_dd_hh_mm_ss.zip`

This log bundle is created in the `/opt/hitachi/ansible/support/` directory.

The log bundle contains the following sub-directories:

**ansible_service**
: Contains ansible.log and `VI-ServiceLog.log`

**gateway_service**
Contains a folder for one or more folder with Hitachi API Gateway Service logs and Hitachi Peer Service logs.

**modules**
Contains copy of all the Hitachi Storage Ansible module, `modules.log` and `hv_storage_modules.log`

**playbooks**
Contains copies of all the playbooks.

**network_fact.txt**
Contains the network details for Hitachi API Gateway Service VM.

**puma_fact.txt**
Contains the Hitachi Peer Service status.

#### Hitachi Log Message Format

While the log messages can be verbose, it may be useful while troubleshooting to pay attention to these messages. All messages are subject to change. All storage provider error messages have the following format: EAC0AXXX

**E**
Denotes that the message is an error visually.

**AC**
Is the product code.

**0**
Is always set to 0.

**XXX**
Is 3 hexadecimal digits that describe the actual error.

For more info, run **`ansible-doc hv_troubleshooting_facts`** to view the documentation for the module.

# Known Issues

| Problem | Recommended Action |
|---|---|

| | |
|---|---|
| Running the **bash** command starts a new shell and playbook execution fails with the following error: `ERROR! couldn't resolve module/action '<module name>'. This often indicates a misspelling, missing collection, or incorrect module path.` | Defining the following exports should resolve the issue:<br>▪ **export ANSIBLE_LIBRARY=/opt/hitachi/ansible/modules/block:/opt/hitachi/ansible/modules/file**<br>▪ **export ANSIBLE_MODULE_UTILS=/opt/hitachi/ansible/module_utils/** |
| Hitachi Storage Modules for Red Hat Ansible installation might fail if installation is performed just after running **dotnet rpm uninstall** | Rebooting the VM after **dotnet uninstall** should fix the issue. |
| If the Hitachi Peer Service VM is different than the Hitachi API Gateway Service, the following error might be observed while adding storage:`"ErrorMessage": "Error occurred while registering RAID Subsystem. One or more errors occurred. (An error occurred while sending the request.)"` | To correct this error, run **generate-remote-gw-cert.sh** (available in `<installation dir>/HV_Ansible/`) with the remote Hitachi Peer Service VM IP as shown:<br>`[root@localhost HV_Ansible]# ./generate-remote-gw-cert.sh <remote_Hitachi_Peer_Service_IP>`<br>Then run `add_storagesystems.yml` again. |
| Create command device operation (using the `hv_cmddev.py` module) fails to map command devices to the given VM if the vCenter does not have the selected host in any cluster. | The command device is actually created and mapped to the ESXi host and the LUN details can be found in `/var/log/hitachi/ansible/hv_storage_modules.log`.<br>Either rescan the host, or search for the new LUN and attach it to the VM manually.<br>You can also create a new cluster on the vCenter, add the host to the new cluster, update the `storage.json` file with the new cluster name, and then run the create command device operation again. |

**Hitachi Vantara**

Corporate Headquarters
2535 Augustine Drive
Santa Clara, CA 95054 USA
HitachiVantara.com | community.HitachiVantara.com

Contact Information
USA: 1-800-446-0744
Global: 1-858-547-4526
HitachiVantara.com/contact