# **SD-WAN Device Status Playbook Documentation**

## **Overview**

The **get\_device\_status.yml** playbook is an Ansible automation script designed to check the operational status of devices in Cisco SD-WAN environments. This playbook connects to the vManage controller and retrieves device status information, saving the results as JSON files for analysis and monitoring purposes.

## **Use Case**

**Use Case 12: Get device status - Check device operational status**

This playbook addresses the need to:

* Check the operational status of all SD-WAN devices
* Monitor device health and connectivity
* Get device monitoring information for troubleshooting
* Export device status data for offline analysis
* Provide automated status checks for regular monitoring

## **Prerequisites**

### **Environment Variables**

The following environment variables must be set before running the playbook:

| **Variable** | **Description** | **Default Value** |
| --- | --- | --- |
| **VMANAGE\_HOST** | vManage controller hostname/IP | sandbox-sdwan-2.cisco.com |
| **VMANAGE\_USERNAME** | Username for vManage authentication | devnetuser |
| **VMANAGE\_PASSWORD** | Password for vManage authentication |  |

## **Playbook Structure**

### **Variables Configuration**

vars:

vmanage\_host: "{{ lookup('env', 'VMANAGE\_HOST') | default('vmanage-amfament-prod.sdwan.cisco.com') }}"

vmanage\_username: "{{ lookup('env', 'VMANAGE\_USERNAME') | default('automation') }}"

vmanage\_password: "{{ lookup('env', 'VMANAGE\_PASSWORD') | default('') }}"

vmanage\_port: "443"

generated\_dir: "{{ playbook\_dir }}/../generated"

### **Directory Structure**

The playbook creates the following directory structure:

playbook\_directory/

├── get\_device\_status.yml

└── generated/

├── device\_status.json

├── device\_control\_connections.json

├── device\_monitor\_status.json

└── device\_status\_summary.txt

## **Task Analysis**

#### **Task 1: Environment Variable Validation**

**Purpose:** Makes sure all required login information is available before starting

**What it does:**

* Checks that **VMANAGE\_HOST**, **VMANAGE\_USERNAME**, and **VMANAGE\_PASSWORD** are set
* Stops immediately if any required information is missing
* Prevents failures due to missing login details
* Shows clear error messages for troubleshooting

#### **Task 2: Display Connection Information**

**Purpose:** Shows connection details while hiding the password

**What it displays:**

* vManage host address
* Username being used
* Port number (443)
* Output directory location
* Hides password for security

#### **Task 3: Create Generated Directory**

**Purpose:** Creates the output folder for device status files

**What it does:**

* Creates the **generated** directory next to the playbook
* Sets proper file permissions (755)
* Makes sure the output location exists before saving files
* Creates parent directories if needed

#### **Task 4: vManage Connectivity Test**

**Purpose:** Checks if the vManage controller can be reached before getting device status

**What it does:**

* Makes a test API call to **/dataservice/system/device/controllers**
* Uses the provided username and password
* Sets **60-second timeout** for slow connections
* Ignores SSL certificate warnings for internal systems
* Saves test results for checking

#### **Task 5: Connectivity Validation**

**Purpose:** Stops the playbook if the connection test fails

**What it does:**

* Checks if the test returned **HTTP 200** (success)
* Stops with error message if vManage cannot be reached
* Prevents unnecessary operations when connection problems exist
* Shows clear failure message for troubleshooting

#### **Task 6: Get All Device Status Information**

**Purpose:** Gets the main device status data from vManage

**API endpoint used:** **/dataservice/device**

**What it does:**

* Connects to vManage using provided login information
* Gets complete device inventory and status
* Includes device health, connectivity, and operational state
* Stores response for saving to file

#### **Task 7: Get Device Control Connections Status**

**Purpose:** Gets control plane connection information

**API endpoint used:** **/dataservice/device/control/connections**

**What it does:**

* Gets control connection status between devices
* Shows how devices connect to controllers
* May fail in sandbox environments (this is normal)
* Stores response even if there are errors

#### **Task 8: Get Device Monitor Status**

**Purpose:** Gets device monitoring and performance data

**API endpoint used:** **/dataservice/device/monitor**

**What it does:**

* Gets device monitoring information
* Includes performance metrics and health data
* Shows device operational statistics
* Stores response for saving to file

#### **Task 9: Save Device Status to JSON File**

**Purpose:** Saves main device status data to a file

**Generated file:** **device\_status.json**

**What it does:**

* Converts device status data to readable JSON format
* Saves to the generated directory
* Only creates file if data was successfully retrieved
* Uses pretty formatting for easy reading

#### **Task 10: Save Device Control Connections to JSON File**

**Purpose:** Saves control connection data to a file

**Generated file:** **device\_control\_connections.json**

**What it does:**

* Saves control connection information (or error details)
* Helpful for debugging connection issues
* Only creates file if there is data to save
* Uses pretty formatting for easy reading

#### **Task 11: Save Device Monitor Status to JSON File**

**Purpose:** Saves monitoring data to a file

**Generated file:** **device\_monitor\_status.json**

**What it does:**

* Saves device monitoring and performance data
* Includes operational statistics and health metrics
* Only creates file if data was successfully retrieved
* Uses pretty formatting for easy reading

#### **Task 12: Create Device Status Summary**

**Purpose:** Creates a simple text summary of the results

**Generated file:** **device\_status\_summary.txt**

**What it includes:**

* Timestamp of when the check was run
* vManage host and username used
* Success/failure status for each API call
* Total number of devices found
* List of all files created

#### **Task 13: Display Completion Message**

**Purpose:** Shows that the playbook finished successfully

**What it displays:**

* Success message
* Location of the generated files
* Confirmation that device status check is complete

## **Generated Files**

The playbook creates these files in the **generated** directory:

* **device\_status.json** - Complete device inventory with operational status
* **device\_control\_connections.json** - Control plane connection information
* **device\_monitor\_status.json** - Device monitoring and performance data
* **device\_status\_summary.txt** - Simple text summary of results

## **Expected Results**

**Successful execution shows:**

* All environment variables validated
* Successful connection to vManage
* Device status data retrieved and saved
* 3-4 JSON files created in generated directory
* Summary report with device count and file locations

**Note:** Some API endpoints may return errors in sandbox environments due to permission restrictions. This is normal and does not affect the main functionality of checking device operational status.