Inventory Device Headless Operation

Endpoints Summary

Method	Path	Swagger
`GET`	`/inventory_device_headless_operation/`	Swagger ↗
`POST`	`/inventory_device_headless_operation/`	Swagger ↗

> The Inventory Device Headless Operation API enables automated device management operations on network devices through programmatic control. These endpoints allow you to schedule and execute device operations such as retrieving parameters, setting configurations, and managing objects without manual intervention, making them ideal for bulk device management and automated workflows.

Base URL: `https://control.zequenze.com/api/v1`

Authentication: All endpoints require a Bearer token:

```
Authorization: Bearer <your-api-token>
```

Overview

The Inventory Device Headless Operation API provides a powerful interface for managing network devices programmatically. This API category focuses on executing operations on devices that are part of your inventory without requiring direct user interaction.

Key Capabilities:

- **Get Operations**: Retrieve specific parameter values from devices (e.g., configuration settings, statistics, device information)

- **Set Operations**: Update device configurations by setting parameter values with specified data types
- **Add Object Operations**: Create new configuration objects on devices dynamically
- **Delete Object Operations**: Remove existing configuration objects from devices

Common Use Cases:

- Bulk configuration updates across multiple devices
- Automated device monitoring and parameter collection
- Scheduled maintenance operations
- Dynamic Wi-Fi network management
- Device compliance checking and reporting
- Remote troubleshooting and diagnostics

The operations are designed to work with TR-069/CWMP compatible devices and support standard device management protocols. All operations can be tracked and their status updated using the built-in helpers.

Endpoints

GET /inventory_device_headless_operation/

Description: Retrieves a list of scheduled headless operations for devices in your inventory. This endpoint allows you to monitor existing operations, check their configuration, and optionally update device status before returning the information. Use this to track ongoing operations or verify operation parameters before execution.

Use Cases:

- Monitor the status of scheduled device operations
- Verify operation parameters before execution
- Audit device management activities
- Retrieve specific operation details by ID

Full URL Example:

```
https://control.zequenze.com/api/v1/inventory_device_headless_operation/?id=12345&update_status=true
```

Parameters:

Parameter	Type	In	Required	Description

id	string	query	No	ID of the specific scheduled transaction to retrieve. If provided, returns only that operation
update_status	boolean	query	No	When true, uses configured helpers to update device status before returning information

cURL Example:

```
curl -X GET
"https://control.zequenze.com/api/v1/inventory_device_headless_operation/?update_status=true"
\
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json"
```

Example Response:


```
[
  {
    "id": 12345,
    "operation": "get",
    "variables": [
      {
        "variable_name": "Device.ManagementServer.PeriodicInformInterval"
      },
      {
        "variable_name": "Device.WiFi.SSID.1.SSID"
      },
      {
        "variable_name": "Device.DeviceInfo.SerialNumber"
      }
    ]
  },
  {
    "id": 12346,
    "operation": "set",
    "variables": [
      {
```

```
        "variable_name": "Device.ManagementServer.PeriodicInformInterval",
        "value": "300",
        "value_type": "integer"
      },
      {
        "variable_name": "Device.ManagementServer.PeriodicInformEnable",
        "value": "1",
        "value_type": "boolean"
      }
    ]
  }
]
```

Response Codes:

Status	Description
200	Success - Returns the list of scheduled operations
401	Unauthorized - Invalid or missing token
404	Not Found - Specific operation ID not found

POST /inventory_device_headless_operation/

Description: Creates a new headless operation for a device in your inventory. This endpoint schedules operations to be executed on network devices, including parameter retrieval, configuration updates, and object management. The operation will be queued and executed according to your system's scheduling configuration.

Use Cases:

- Schedule bulk configuration updates across multiple devices
- Set up automated parameter monitoring
- Create new Wi-Fi SSIDs or network configurations
- Remove outdated configuration objects
- Implement automated device maintenance routines

Full URL Example:

```
https://control.zequenze.com/api/v1/inventory_device_headless_operation/
```

Parameters:

Parameter	Type	In	Required	Description
data	string	body	Yes	JSON payload containing the operation details including device ID, operation type, and variables

Request Body Structure:

```
{
  "id": 12345,
  "operation": "get|set|add.obj|del.obj",
  "variables": [
    {
      "variable_name": "Device.Parameter.Name",
      "value": "parameter_value",
      "value_type": "string|integer|boolean"
    }
  ]
}
```

cURL Example - Get Operation:

```
curl -X POST "https://control.zequenze.com/api/v1/inventory_device_headless_operation/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "id": 12345,
    "operation": "get",
    "variables": [
      {"variable_name": "Device.ManagementServer.PeriodicInformInterval"},
      {"variable_name": "Device.WiFi.SSID.1.SSID"},
      {"variable_name": "Device.DeviceInfo.ModelName"}
    ]
  }'
```

cURL Example - Set Operation:

```
curl -X POST "https://control.zequenze.com/api/v1/inventory_device_headless_operation/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
```

```
  -d '{
    "id": 12345,
    "operation": "set",
    "variables": [
      {
        "variable_name": "Device.ManagementServer.PeriodicInformInterval",
        "value": "300",
        "value_type": "integer"
      },
      {
        "variable_name": "Device.WiFi.SSID.1.SSID",
        "value": "MyNewNetwork",
        "value_type": "string"
      }
    ]
  }'
```

Example Response:

```
{
  "id": 12345,
  "operation": "set",
  "variables": [
    {
      "variable_name": "Device.ManagementServer.PeriodicInformInterval",
      "value": "300",
      "value_type": "integer"
    },
    {
      "variable_name": "Device.WiFi.SSID.1.SSID",
      "value": "MyNewNetwork",
      "value_type": "string"
    }
  ]
}
```

Response Codes:

Status	Description
201	Created - Operation successfully scheduled

Status	Description
400	Bad Request - Invalid operation parameters
401	Unauthorized - Invalid or missing token
404	Not Found - Device ID not found in inventory

Common Use Cases

Use Case 1: Bulk Wi-Fi Configuration Update

Schedule configuration updates across multiple access points to change SSID names, passwords, or security settings. Use the `set` operation with Wi-Fi related parameters to update network configurations remotely.

Use Case 2: Device Health Monitoring

Set up automated parameter collection using `get` operations to monitor device status, performance metrics, and configuration drift. Schedule regular retrieval of key parameters like uptime, memory usage, and connection statistics.

Use Case 3: Dynamic Network Object Management

Use `add.obj` and `del.obj` operations to dynamically manage network configurations such as creating new SSID instances, adding firewall rules, or managing VLAN configurations based on business requirements.

Use Case 4: Compliance Auditing

Implement automated compliance checking by retrieving configuration parameters and comparing them against organizational standards. Use `get` operations to collect current settings for security auditing purposes.

Use Case 5: Remote Troubleshooting

Schedule diagnostic operations to collect device information and statistics when issues are reported. Use a combination of `get` operations to gather comprehensive device state information for troubleshooting.

Best Practices

- **Operation Planning**: Always test operations on a small subset of devices before scheduling bulk operations across your entire inventory
- **Parameter Validation**: Verify parameter names and paths are correct for your device models before creating operations, as incorrect parameters may cause operation failures
- **Status Monitoring**: Use the `update_status=true` parameter when retrieving operations to get the most current device status information
- **Error Handling**: Implement proper error handling for failed operations and consider retry mechanisms for transient failures
- **Security**: Limit the scope of operations and use appropriate authentication tokens with minimal required permissions
- **Scheduling**: Consider device load and network capacity when scheduling multiple operations, especially during business hours
- **Documentation**: Keep detailed records of scheduled operations for audit trails and troubleshooting purposes

---

Revision #4
Created 2026-02-04 05:04:31 UTC by ipena@zequenze.com
Updated 2026-02-11 02:51:16 UTC by ipena@zequenze.com