Inventory Device Name Headless Operation

Endpoints Summary

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

> The Inventory Device Name Headless Operation API enables automated remote management of network devices through headless operations. These endpoints allow you to schedule and execute device operations such as getting parameter values, setting configurations, adding objects, and deleting objects without manual intervention.

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

Authentication: All endpoints require a Bearer token:

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

Overview

The Inventory Device Name Headless Operation API provides programmatic control over device management operations in your network infrastructure. This API is designed for network administrators and developers who need to automate device configuration, monitoring, and maintenance tasks across multiple devices simultaneously.

Key Features:

- **Remote Device Operations**: Execute get, set, add object, and delete object operations on network devices
- **Batch Processing**: Manage multiple devices and operations through a single API interface
- **Flexible Variable Management**: Support for various data types including integers, booleans, and strings
- **Status Monitoring**: Optional real-time status updates using configured helpers

Common Use Cases:

- Automated device configuration deployment
- Periodic parameter monitoring and data collection
- Bulk device settings updates across your network
- Network maintenance and troubleshooting automation
- Integration with network management systems and monitoring tools

The API operates on TR-069/CWMP protocol standards, allowing you to interact with device parameters using standard device model paths (e.g., `Device.ManagementServer.PeriodicInformInterval`).

Endpoints

GET /inventory_device_name_headless_operation/

Description: Retrieves a list of scheduled headless operations for network devices. This endpoint allows you to monitor existing operations, check their status, and optionally update device status information before returning results.

Use Cases:

- Monitor the status of scheduled device operations
- Retrieve operation history for auditing and troubleshooting
- Check pending operations before scheduling new ones
- Update device status information for real-time monitoring

Full URL Example:

```
https://control.zequenze.com/api/v1/inventory_device_name_headless_operation/?id=txn_12345&update_status=true
```

Parameters:

Parameter	Type	In	Required	Description
id	string	query	No	ID of the specific scheduled transaction to retrieve
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_name_headless_operation/?update_status=t
rue" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json"
```

Example Response:


```
[
  {
    "name": "router-office-001",
    "operation": "get",
    "variables": [
      {
        "variable_name": "Device.ManagementServer.PeriodicInformInterval"
      },
      {
        "variable_name": "Device.WiFi.SSID.1.Stats.BytesReceived"
      }
    ]
  },
  {
    "name": "router-office-002",
    "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 list of scheduled operations
401	Unauthorized - Invalid or missing API token
404	Not Found - Specified transaction ID does not exist
500	Internal Server Error - Server processing error

POST /inventory_device_name_headless_operation/

Description: Creates a new scheduled headless operation for a network device. This endpoint allows you to define device operations including parameter retrieval, configuration changes, object management, and variable assignments.

Use Cases:

- Schedule automatic parameter retrieval from devices
- Deploy configuration changes to network devices
- Add new objects to device configurations
- Remove obsolete configuration objects
- Automate routine maintenance tasks

Full URL Example:

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

Parameters:

Parameter	Type	In	Required	Description
data	object	body	Yes	Complete operation definition including device name, operation type, and variables

Request Body Schema:

```
{
  "name": "string (required) - Name of the target device",
  "operation": "string (required) - Operation type: get, set, add.obj, del.obj",
  "variables": [
    {
      "variable_name": "string (required) - Device parameter path",
      "value": "string (optional) - Value for set operations",
      "value_type": "string (optional) - Data type: integer, boolean, string"
    }
  ]
}
```

cURL Example (GET Operation):

```
curl -X POST "https://control.zequenze.com/api/v1/inventory_device_name_headless_operation/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "router-branch-005",
    "operation": "get",
    "variables": [
      {"variable_name": "Device.ManagementServer.PeriodicInformInterval"},
      {"variable_name": "Device.WiFi.SSID.1.Enable"}
    ]
  }'
```

cURL Example (SET Operation):

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

```
  -d '{
    "name": "router-branch-005",
    "operation": "set",
    "variables": [
      {
        "variable_name": "Device.ManagementServer.PeriodicInformInterval",
        "value": "600",
        "value_type": "integer"
      },
      {
        "variable_name": "Device.WiFi.SSID.1.Enable",
        "value": "true",
        "value_type": "boolean"
      }
    ]
  }'
```

Example Response:

```
{
  "name": "router-branch-005",
  "operation": "set",
  "variables": [
    {
      "variable_name": "Device.ManagementServer.PeriodicInformInterval",
      "value": "600",
      "value_type": "integer"
    },
    {
      "variable_name": "Device.WiFi.SSID.1.Enable",
      "value": "true",
      "value_type": "boolean"
    }
  ]
}
```

Response Codes:

Status	Description
201	Created - Operation successfully scheduled

Status	Description
400	Bad Request - Invalid operation parameters or malformed JSON
401	Unauthorized - Invalid or missing API token
404	Not Found - Specified device does not exist
422	Unprocessable Entity - Valid JSON but invalid operation configuration

---

Common Use Cases

Use Case 1: Automated Device Monitoring

Regularly collect device statistics and performance metrics from multiple network devices for monitoring dashboards and alerting systems.

Endpoints: GET to check existing operations, POST to schedule new monitoring tasks

Use Case 2: Bulk Configuration Deployment

Deploy standardized configuration settings across multiple devices in a network, such as updating security parameters or Wi-Fi settings.

Endpoints: POST to create set operations with standardized variable values

Use Case 3: Network Maintenance Automation

Automate routine maintenance tasks like clearing statistics, updating inform intervals, or managing device objects during scheduled maintenance windows.

Endpoints: POST with add.obj and del.obj operations for object management

Use Case 4: Device Status Verification

Verify current device configurations and settings after deployment or troubleshooting to ensure consistency across the network.

Endpoints: GET with update_status=true to retrieve real-time device information

Use Case 5: Integration with Network Management Systems

Integrate with existing NMS platforms to provide automated device management capabilities through API calls.

Endpoints: Both GET and POST for comprehensive device operation management

---

Best Practices

Operation Types:

- Use `get` operations for non-intrusive parameter retrieval and monitoring
- Use `set` operations carefully as they modify device configurations
- Test `add.obj` and `del.obj` operations in development environments before production deployment
- Always validate device parameter paths before creating operations

Error Handling:

- Implement retry logic for network-related failures
- Check device connectivity status before scheduling operations
- Monitor operation results and implement appropriate error handling for failed operations
- Use the `update_status` parameter to get real-time device status when needed

Performance Optimization:

- Batch related operations together when possible to reduce API calls
- Use specific transaction IDs when retrieving operation status to improve query performance
- Schedule operations during maintenance windows for configuration changes
- Monitor API rate limits and implement appropriate throttling

Security Considerations:

- Store API tokens securely and rotate them regularly
- Validate all input parameters before sending requests
- Use HTTPS for all API communications
- Implement proper access controls for device management operations
- Log all configuration changes for audit trails

Data Type Guidelines:

- Always specify `value_type` for set operations to ensure proper parameter handling
- Use appropriate data types: "integer" for numeric values, "boolean" for true/false, "string" for text
- Validate parameter values against device specifications before submission

---

Revision #4
Created 2026-02-04 05:06:05 UTC by ipena@zequenze.com
Updated 2026-02-11 02:55:32 UTC by ipena@zequenze.com