Inventory Device Name Operation

Endpoints Summary

Method	Path	Swagger
PUT	/inventory_device_name_operation/{name}/	Swagger ↗

> The Inventory Device Name Operation API enables remote device management operations on specific devices in your inventory. This endpoint allows you to execute critical device operations such as rebooting, factory resets, configuration sync, and device reconfiguration by targeting devices using their unique names.

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 Operation API provides a powerful interface for performing remote management operations on devices within your inventory system. This API is specifically designed for network administrators, IT operations teams, and automated systems that need to execute device management commands remotely.

Key capabilities include:

- **Remote Device Control**: Execute operations on devices without physical access
- **Automated Device Management**: Integrate device operations into your automation workflows

- **Centralized Operations**: Manage multiple device operations from a single API interface
- **Real-time Execution**: Operations are executed immediately upon API call

This endpoint is particularly valuable in scenarios such as troubleshooting network issues, performing scheduled maintenance, deploying configuration changes, or recovering devices that have become unresponsive. The API supports various operation types including standard reboots, factory resets, configuration synchronization, and complete device reconfiguration.

The operations are executed asynchronously, allowing your applications to continue processing while device operations complete in the background. This makes it ideal for integration with monitoring systems, ticketing platforms, and automated incident response workflows.

Endpoints

PUT /inventory_device_name_operation/{name}/

Description: Executes a specified operation on a device identified by its name. This endpoint allows you to perform critical device management tasks remotely, including reboots, factory resets, configuration sync, and reconfiguration operations. The operation is executed immediately upon successful API call.

Use Cases:

- Remotely reboot unresponsive network devices during troubleshooting
- Perform factory resets on devices being reassigned to different locations
- Synchronize device configurations after policy updates
- Execute device reconfiguration as part of automated deployment workflows
- Recover devices that have lost connectivity or are in an error state

Full URL Example:

```
https://control.zequenze.com/api/v1/inventory_device_name_operation/office-router-01/
```

Parameters:

Parameter	Type	In	Required	Description
name	string	path	Yes	The unique name/identifier of the device to perform the operation on

Parameter	Type	In	Required	Description
data	string	body	Yes	JSON payload containing the operation details and parameters

Request Body Schema:

Field	Type	Required	Description	Allowed Values
operation	string	Yes	The operation to perform on the device	`reboot`, `factory`, `device_factory`, `sync`, `reconf`

Operation Types:

- `reboot`: Standard device reboot - restarts the device while preserving configuration
- `factory`: Factory reset - returns device to original factory settings
- `device_factory`: Device factory reset - complete reset including device-specific settings
- `sync`: Device config sync - synchronizes current configuration with central management
- `reconf`: Device reconfigure - applies new configuration settings to the device

cURL Example:

```
curl -X PUT "https://control.zequenze.com/api/v1/inventory_device_name_operation/office-router-01/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "operation": "reboot"
  }'
```

Example Request (Reboot Operation):

```
{
  "operation": "reboot"
}
```

Example Request (Configuration Sync):

```
{
  "operation": "sync"
}
```

Example Response:

```
{
  "id": 12345,
  "operation": "reboot",
  "device_name": "office-router-01",
  "status": "initiated",
  "timestamp": "2024-01-15T14:30:25Z",
  "execution_id": "op_67890abcdef",
  "estimated_completion": "2024-01-15T14:32:25Z"
}
```


Response Codes:

Status	Description
200	Success - Operation initiated successfully
400	Bad Request - Invalid operation type or malformed request
401	Unauthorized - Invalid or missing authentication token
404	Not Found - Device with specified name does not exist
409	Conflict - Device is currently offline or another operation is in progress
500	Internal Server Error - Failed to initiate operation

Common Use Cases

Use Case 1: Automated Incident Response

When monitoring systems detect device connectivity issues, automatically trigger a reboot operation using the `reboot` operation type. This can resolve many common network issues without manual intervention.

Use Case 2: Scheduled Maintenance Operations

Integrate with scheduling systems to perform configuration sync operations (`sync`) during maintenance windows, ensuring all devices have the latest approved configurations.

Use Case 3: Device Deployment Pipeline

Use the `reconf` operation as part of automated device deployment workflows when devices need to be reconfigured for new roles or locations.

Use Case 4: Security Incident Recovery

Execute `factory` or `device_factory` operations on compromised devices to ensure complete removal of potentially malicious configurations or software.

Use Case 5: Configuration Drift Remediation

Regularly use the `sync` operation to detect and correct configuration drift across your device inventory, maintaining compliance with organizational standards.

Best Practices

Operation Planning:

- Always verify device connectivity before initiating operations that may temporarily disconnect the device
- Use `sync` operations before more disruptive operations like `reboot` to ensure configurations are current
- Schedule factory reset operations during maintenance windows to minimize service disruption

Error Handling:

- Implement retry logic with exponential backoff for temporary failures (5xx errors)
- Monitor operation completion through separate status checking mechanisms
- Handle 409 conflicts by waiting for current operations to complete before retrying

Security Considerations:

- Restrict API access to authorized personnel and systems only
- Log all device operations for audit and compliance purposes
- Use factory reset operations judiciously as they require complete device reconfiguration

Performance Optimization:

- Batch similar operations during maintenance windows to reduce overall impact
- Allow adequate time between operations on the same device to prevent conflicts
- Monitor device response times and adjust operation scheduling accordingly

Monitoring Integration:

- Integrate operation results with your monitoring and alerting systems
- Track operation success rates to identify problematic devices or operation types
- Use operation logs for troubleshooting and capacity planning

---

Revision #4
Created 2026-02-04 05:06:27 UTC by ipena@zequenze.com
Updated 2026-02-11 02:56:27 UTC by ipena@zequenze.com