Inventory Device Name Diags

Endpoints Summary

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

> The Inventory Device Name Diagnostics API provides endpoints for scheduling and monitoring network diagnostic operations on specific network devices. These endpoints allow you to initiate various network tests such as ping, traceroute, speed tests, and WiFi diagnostics, as well as retrieve the status and results of previously scheduled diagnostic operations.

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 Diagnostics API enables network administrators and developers to perform comprehensive network diagnostics on managed devices remotely. This API category is essential for network monitoring, troubleshooting connectivity issues, and performing routine network health checks.

Key Features:

- **Remote Diagnostics**: Execute network tests on devices without physical access
- **Multiple Test Types**: Support for ping, traceroute, speed tests, DNS lookups, UDP echo tests, and WiFi neighbor diagnostics

- **Flexible Configuration**: Customize test parameters like packet size, timeout values, and hop limits
- **Status Monitoring**: Track the progress and results of diagnostic operations

Common Integration Scenarios:

- Network monitoring dashboards that need to verify device connectivity
- Automated troubleshooting workflows triggered by network alerts
- Scheduled health checks for critical network infrastructure
- Customer support tools for diagnosing connectivity issues

The API follows a simple pattern: use POST to schedule new diagnostic operations and GET to retrieve status updates and results. Each diagnostic operation is tied to a specific device name and can be configured with operation-specific parameters.

Endpoints

POST /inventory_device_name_diags/

Description: Schedules a new network diagnostic operation on a specified device. This endpoint allows you to initiate various types of network tests including connectivity checks, performance measurements, and WiFi diagnostics. The operation is queued and executed asynchronously on the target device.

Use Cases:

- Troubleshoot connectivity issues by running ping or traceroute tests
- Measure network performance with download/upload speed tests
- Verify DNS resolution functionality
- Analyze WiFi network environment and neighboring access points
- Perform routine network health checks as part of monitoring workflows

Full URL Example:

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

Request Body Schema:

Field	Type	Required	Description
name	string	Yes	Name of the target device to run diagnostics on

Field	Type	Required	Description
operation	string	Yes	Type of diagnostic operation (download, upload, ipping, udpecho, traceroute, dnslookup, wifi.neighbor)
target	string	Yes	Target for the operation (URL for speed tests, IP for ping/traceroute, hostname for DNS lookup)
upload_size	string	No	File size in MB for upload tests (e.g., "10")
count	integer	No	Number of packets/repetitions (default: 5 for ping, 1 for DNS)
size	integer	No	Packet size in bytes for ping operations (default: 64)
timeout	integer	No	Timeout in seconds for ping and DNS operations (default: 1)
max_hops	integer	No	Maximum hops for traceroute operations (default: 30)
interface	string	No	Network interface name to use for the operation

cURL Example (Ping Test):

```
curl -X POST "https://control.zequenze.com/api/v1/inventory_device_name_diags/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "router-main-office",
    "operation": "ipping",
    "target": "8.8.8.8",
    "count": 10,
    "size": 64,
    "timeout": 2
  }'
```

cURL Example (Speed Test):

```
curl -X POST "https://control.zequenze.com/api/v1/inventory_device_name_diags/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "cpe-customer-001",
    "operation": "download",
    "target": "http://speedtest.example.com/test-file.zip",
    "interface": "eth0"
  }'
```

Example Response:

```
{
  "name": "router-main-office",
  "operation": "ipping",
  "target": "8.8.8.8",
  "count": 10,
  "size": 64,
  "timeout": 2,
  "upload_size": null,
  "max_hops": null,
  "interface": null
}
```

Response Codes:

Status	Description
201	Created - Diagnostic operation scheduled successfully
400	Bad Request - Invalid operation type or missing required fields
401	Unauthorized - Invalid or missing authentication token
404	Not Found - Device name not found in inventory

GET /inventory_device_name_diags/

Description: Retrieves the status and results of scheduled diagnostic operations. This endpoint can return all diagnostic operations or filter by specific transaction ID. Use this to monitor the progress of tests and collect results once operations are complete.

Use Cases:

- Monitor the status of recently scheduled diagnostic operations
- Retrieve test results for analysis and reporting
- Check if devices are responding to diagnostic requests
- Update device connectivity status in monitoring systems

Full URL Example:

```
https://control.zequenze.com/api/v1/inventory_device_name_diags/?id=diag_12345&update_status=true
```

Parameters:

Parameter	Type	In	Required	Description
id	string	query	No	Filter results by specific diagnostic transaction ID
update_status	boolean	query	No	Refresh device status before returning results (may increase response time)

cURL Example (All Operations):

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

cURL Example (Specific Operation):

```
curl -X GET "https://control.zequenze.com/api/v1/inventory_device_name_diags/?id=diag_12345&update_status=true" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json"
```

Example Response:

```
[
  {
    "name": "router-main-office",
    "operation": "ipping",
```

```
    "target": "8.8.8.8",
    "count": 10,
    "size": 64,
    "timeout": 2,
    "upload_size": null,
    "max_hops": null,
    "interface": null,
    "status": "completed",
    "results": {
      "packets_sent": 10,
      "packets_received": 10,
      "packet_loss": "0%",
      "avg_rtt": "12.5ms",
      "min_rtt": "11.2ms",
      "max_rtt": "15.1ms"
    },
    "created_at": "2024-01-15T10:30:00Z",
    "completed_at": "2024-01-15T10:30:15Z"
  },
  {
    "name": "cpe-customer-001",
    "operation": "download",
    "target": "http://speedtest.example.com/test-file.zip",
    "upload_size": null,
    "count": null,
    "size": null,
    "timeout": null,
    "max_hops": null,
    "interface": "eth0",
    "status": "running",
    "results": null,
    "created_at": "2024-01-15T10:35:00Z",
    "completed_at": null
  }
]
```


Response Codes:

Status	Description
200	Success - Returns diagnostic operation data

Status	Description
401	Unauthorized - Invalid or missing authentication token
404	Not Found - Specified diagnostic ID not found

Common Use Cases

Use Case 1: Automated Connectivity Monitoring

Schedule periodic ping tests to critical infrastructure and monitor results to detect connectivity issues before they impact users.

```
# Schedule ping test
POST /inventory_device_name_diags/
{
  "name": "core-router-01",
  "operation": "ipping",
  "target": "8.8.8.8",
  "count": 5
}

# Check results
GET /inventory_device_name_diags/?id=<transaction_id>
```

Use Case 2: Customer Support Troubleshooting

When a customer reports slow internet, initiate speed tests and traceroute diagnostics to identify the source of performance issues.

```
# Run download speed test
POST /inventory_device_name_diags/
{
  "name": "customer-cpe-12345",
  "operation": "download",
  "target": "http://speedtest.local/100MB.bin"
}

# Run traceroute to identify routing issues
```

```
POST /inventory_device_name_diags/
{
  "name": "customer-cpe-12345",
  "operation": "traceroute",
  "target": "google.com"
}
```

Use Case 3: WiFi Environment Analysis

Analyze WiFi networks in the area to troubleshoot interference issues or optimize channel selection.

```
# Scan for neighboring WiFi networks
POST /inventory_device_name_diags/
{
  "name": "office-ap-01",
  "operation": "wifi.neighbor",
  "target": "2.4GHz"
}
```

Use Case 4: DNS Resolution Verification

Verify that devices can properly resolve domain names, which is critical for internet connectivity.

```
# Test DNS resolution
POST /inventory_device_name_diags/
{
  "name": "branch-router",
  "operation": "dnslookup",
  "target": "google.com",
  "count": 3,
  "timeout": 5
}
```

Use Case 5: Network Performance Baseline

Establish performance baselines by running regular upload/download tests during different times of day.

```
# Upload speed test
POST /inventory_device_name_diags/
{
  "name": "test-device",
  "operation": "upload",
  "target": "http://speedtest.local/upload",
  "upload_size": "50"
}
```

Best Practices

- **Operation Scheduling**: Avoid scheduling multiple intensive operations (like speed tests) simultaneously on the same device to prevent resource conflicts
- **Timeout Configuration**: Set appropriate timeout values based on network conditions - use longer timeouts for satellite or high-latency connections
- **Interface Selection**: Specify the interface parameter when testing specific network paths or when devices have multiple network interfaces
- **Result Polling**: When checking results with GET requests, implement reasonable polling intervals (30-60 seconds) to avoid overwhelming the API
- **Error Handling**: Always check the operation status before processing results - operations may fail due to device connectivity or configuration issues
- **Resource Management**: Speed tests can consume significant bandwidth - schedule them during maintenance windows when possible
- **Historical Data**: Store diagnostic results locally for trend analysis and historical performance comparisons
- **Device Status**: Use the `update_status=true` parameter sparingly as it may increase response times when querying device information

Revision #4
Created 2026-02-04 05:05:47 UTC by ipena@zequenze.com
Updated 2026-02-11 02:54:52 UTC by ipena@zequenze.com