Inventory Device Diags

Endpoints Summary

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

> The Inventory Device Diagnostics API enables you to create and monitor network diagnostic operations for devices in your inventory. These endpoints allow you to schedule various network tests like ping, traceroute, speed tests, and WiFi diagnostics, then retrieve results and status information for troubleshooting and network performance analysis.

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

Authentication: All endpoints require a Bearer token:

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

Overview

The Inventory Device Diagnostics API provides comprehensive network diagnostic capabilities for managed devices. This API category allows network administrators and developers to programmatically execute various diagnostic operations including connectivity tests, performance measurements, and network discovery tools.

Key Features:

- **Multiple Diagnostic Types:** Support for HTTP download/upload tests, ICMP ping, UDP echo, traceroute, DNS lookups, and WiFi neighbor discovery
- **Flexible Configuration:** Customizable parameters for packet size, timeout values, connection counts, and more

- **Real-time Monitoring:** Track diagnostic operation status and retrieve results as they become available
- **Interface-specific Testing:** Target specific network interfaces for precise diagnostics

Common Use Cases:

- Automated network health monitoring and alerting
- Troubleshooting connectivity issues remotely
- Performance benchmarking and capacity planning
- WiFi network analysis and optimization
- Compliance reporting for SLA monitoring

The diagnostic operations are executed asynchronously on the target devices, allowing you to schedule multiple tests and check their progress using the list endpoint.

Endpoints

POST /inventory_device_diags/

Description: Creates a new diagnostic operation for a specific device. This endpoint schedules various types of network diagnostics that will be executed on the target device, including connectivity tests, performance measurements, and network discovery operations.

Use Cases:

- Schedule automated network health checks
- Perform on-demand troubleshooting for connectivity issues
- Execute performance benchmarks for bandwidth testing
- Conduct WiFi site surveys and neighbor analysis

Full URL Example:

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

Request Body Parameters:

The request body should contain a JSON object with the diagnostic configuration:

Parameter	Type	Required	Description
id	integer	Yes	ID of the target device from your inventory

Parameter	Type	Required	Description
operation	string	Yes	Type of diagnostic operation. Options: `download`, `upload`, `ipping`, `udpecho`, `traceroute`, `dnslookup`, `wifi.neighbor`
target	string	Yes	Target URL (download/upload), IP address (ping/traceroute), hostname (DNS lookup), or parameter (WiFi diagnostics)
upload_size	string	No	File size in MB for upload operations (e.g., "100")
count	integer	No	Number of packets for ping (default: 5) or DNS lookup repetitions (default: 1)
size	integer	No	Packet size in bytes for ping operations (default: 64)
timeout	integer	No	Response timeout in seconds for ping/DNS operations (default: 1)
dns_server	string	No	Specific DNS server IP for DNS lookup operations
max_hops	integer	No	Maximum hops for traceroute operations (default: 30)
interface	string	No	Network interface name to use for the operation
number_of_connections	integer	No	Concurrent connections for download/upload tests

cURL Example - Ping Test:

```
curl -X POST "https://control.zequenze.com/api/v1/inventory_device_diags/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "id": 12345,
    "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_diags/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "id": 12345,
    "operation": "download",
    "target": "http://speedtest.example.com/100MB.bin",
    "number_of_connections": 4,
    "interface": "eth0"
  }'
```

Example Response:

```
{
  "id": 12345,
  "operation": "ipping",
  "target": "8.8.8.8",
  "count": 10,
  "size": 64,
  "timeout": 2,
  "interface": "eth0",
  "upload_size": null,
  "dns_server": null,
  "max_hops": null,
  "number_of_connections": null
}
```

Response Codes:

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

GET /inventory_device_diags/

Description: Retrieves a list of diagnostic operations, either all operations or filtered by specific criteria. This endpoint allows you to monitor the status of scheduled diagnostics and retrieve results from completed operations.

Use Cases:

- Monitor progress of scheduled diagnostic operations
- Retrieve diagnostic results for analysis and reporting
- Check the status of specific diagnostic transactions
- Update device connectivity status before retrieving diagnostics

Full URL Example:

```
https://control.zequenze.com/api/v1/inventory_device_diags/?id=67890&update_status=true
```

Parameters:

Parameter	Type	In	Required	Description
id	string	query	No	ID of a specific scheduled diagnostic transaction to retrieve
update_status	boolean	query	No	Whether to refresh device status using configured helpers before returning data

cURL Example - List All Diagnostics:

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

cURL Example - Get Specific Diagnostic:

```
curl -X GET
"https://control.zequenze.com/api/v1/inventory_device_diags/?id=67890&update_status=true" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json"
```

Example Response:

```
[
  {
    "id": 12345,
    "operation": "ipping",
    "target": "8.8.8.8",
    "count": 10,
    "size": 64,
    "timeout": 2,
    "interface": "eth0",
    "upload_size": null,
    "dns_server": null,
    "max_hops": null,
    "number_of_connections": null
  },
  {
    "id": 12346,
    "operation": "download",
    "target": "http://speedtest.example.com/100MB.bin",
    "count": null,
    "size": null,
    "timeout": null,
    "interface": "wlan0",
    "upload_size": null,
    "dns_server": null,
    "max_hops": null,
    "number_of_connections": 4
  },
  {
    "id": 12347,
    "operation": "traceroute",
    "target": "google.com",
    "count": null,
    "size": null,
    "timeout": null,
    "interface": null,
    "upload_size": null,
    "dns_server": null,
    "max_hops": 20,
    "number_of_connections": null
  }
```

```
]
```

Response Codes:

Status	Description
200	Success - Returns list of diagnostic operations
401	Unauthorized - Invalid or missing API token
404	Not Found - Specific diagnostic ID not found (when using id parameter)

Common Use Cases

Use Case 1: Automated Network Health Monitoring

Schedule regular ping and connectivity tests for critical devices to monitor network health. Use the POST endpoint to create recurring diagnostic jobs and the GET endpoint to retrieve results for dashboard displays or alerting systems.

Use Case 2: Bandwidth Performance Testing

Conduct periodic speed tests using download and upload operations to ensure devices meet SLA requirements. Configure multiple connections to test maximum throughput capabilities and identify performance degradation.

Use Case 3: Troubleshooting Connectivity Issues

When users report network problems, use traceroute and DNS lookup diagnostics to identify routing issues or DNS resolution problems. The interface parameter allows testing specific network paths.

Use Case 4: WiFi Site Survey and Optimization

Execute WiFi neighbor diagnostics to analyze wireless environments, identify interference sources, and optimize channel assignments for better performance in dense deployment scenarios.

Use Case 5: Compliance and SLA Reporting

Regularly schedule comprehensive diagnostic suites and use the results for compliance reporting, SLA verification, and network performance documentation required by regulatory or contractual obligations.

Best Practices

- **Batch Operations:** When testing multiple devices, stagger diagnostic operations to avoid network congestion and ensure accurate results
- **Parameter Optimization:** Use appropriate timeout values and packet counts based on network conditions - longer timeouts for satellite links, higher packet counts for statistical accuracy
- **Interface Selection:** Specify target interfaces when devices have multiple network connections to test specific paths (e.g., cellular vs. WiFi)
- **Error Handling:** Implement retry logic for failed diagnostics, as network conditions or device availability may cause temporary failures
- **Rate Limiting:** Respect API rate limits and device processing capabilities by spacing diagnostic requests appropriately
- **Result Polling:** Use the `update_status` parameter judiciously as it may impact performance - only use when real-time status updates are required
- **Security Considerations:** Ensure diagnostic targets (URLs, IPs) are from trusted sources to prevent devices from accessing malicious endpoints

---

Revision #4
Created 2026-02-04 05:04:13 UTC by ipena@zequenze.com
Updated 2026-02-11 02:50:40 UTC by ipena@zequenze.com