Inventory Device Serial Diags

Endpoints Summary

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

> The Inventory Device Serial Diagnostics API enables remote execution of network diagnostic operations on specific devices using their serial numbers. These endpoints allow you to schedule diagnostic tests like ping, traceroute, speed tests, and WiFi neighbor scans, then retrieve results to troubleshoot connectivity issues and monitor network performance.

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

Authentication: All endpoints require a Bearer token:

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

Overview

The inventory device serial diagnostics API provides powerful remote diagnostic capabilities for network devices in your inventory. This API enables technicians and network administrators to remotely execute various diagnostic operations on Customer Premises Equipment (CPE) and other managed devices without requiring physical access.

Key capabilities include:

- **Network Performance Testing**: Execute download/upload speed tests to measure bandwidth
- **Connectivity Diagnostics**: Perform ICMP ping tests to verify reachability
- **Route Analysis**: Run traceroute operations to identify network path issues

- **DNS Resolution Testing**: Validate DNS lookup functionality
- **WiFi Environment Scanning**: Analyze wireless neighbor networks and signal strength
- **UDP Echo Testing**: Test bidirectional UDP connectivity

How it works:

1. Schedule diagnostic operations using the POST endpoint with specific test parameters
2. The system queues the diagnostic request for the target device
3. Retrieve diagnostic results and status using the GET endpoint
4. Operations are executed asynchronously when the device connects to the management system

This API is essential for remote troubleshooting, proactive network monitoring, and validating service quality without dispatching field technicians.

Endpoints

POST /inventory_device_serial_diags/

Description: Schedules a new network diagnostic operation on a device identified by its serial number. This endpoint queues diagnostic tests that will be executed when the target device next communicates with the management platform. Use this to initiate remote troubleshooting or performance testing.

Use Cases:

- Customer reports slow internet speeds - schedule download/upload tests
- Device appears offline - execute ping tests to verify connectivity
- Intermittent connection issues - run traceroute to identify network path problems
- WiFi performance concerns - scan for interference from neighboring networks
- DNS resolution failures - test DNS lookup functionality

Full URL Example:

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

Request Body Parameters:

Parameter	Type	Required	Description
serial_number	string	Yes	Serial number of the target device to diagnose

Parameter	Type	Required	Description
operation	string	Yes	Type of diagnostic test. Options: `download`, `upload`, `ipping`, `udpecho`, `traceroute`, `dnslookup`, `wifi.neighbor`
target	string	Yes	Test target - URL for speed tests, IP address for ping/traceroute, hostname for DNS lookup, or parameter for WiFi scans
upload_size	string	No	File size in MB for upload tests (e.g., "10", "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)
max_hops	integer	No	Maximum hops for traceroute operations (default: 30)
interface	string	No	Specific network interface to use for the diagnostic operation

cURL Example - Speed Test:

```
curl -X POST "https://control.zequenze.com/api/v1/inventory_device_serial_diags/" \
  -H "Authorization: Bearer YOUR_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "serial_number": "SN123456789",
    "operation": "download",
    "target": "http://speedtest.example.com/1GB.bin"
  }'
```

cURL Example - Connectivity Test:

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

```
  -d '{
    "serial_number": "SN123456789",
    "operation": "ipping",
    "target": "8.8.8.8",
    "count": 10,
    "size": 64,
    "timeout": 2
  }'
```

Example Response:

```
{
  "serial_number": "SN123456789",
  "operation": "download",
  "target": "http://speedtest.example.com/1GB.bin",
  "upload_size": null,
  "count": null,
  "size": null,
  "timeout": null,
  "max_hops": null,
  "interface": null,
  "id": "diag_12345",
  "status": "scheduled",
  "created_at": "2024-01-15T10:30:00Z"
}
```

Response Codes:

Status	Description
201	Created - Diagnostic operation successfully scheduled
400	Bad Request - Invalid parameters or operation type
401	Unauthorized - Invalid or missing API token
404	Not Found - Device with specified serial number not found

---

GET /inventory_device_serial_diags/

Description: Retrieves diagnostic operation results and status information. Use this endpoint to check the progress and results of previously scheduled diagnostic tests. You can query for specific

operations or list all diagnostics for monitoring purposes.

Use Cases:

- Check if a scheduled diagnostic test has completed
- Retrieve speed test results to validate bandwidth performance
- Monitor diagnostic operation status across multiple devices
- Generate reports on network performance and connectivity issues

Full URL Example:

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

Parameters:

Parameter	Type	In	Required	Description
id	string	query	No	Specific diagnostic operation ID to retrieve results for
update_status	boolean	query	No	Whether to refresh device status before returning results (default: false)

cURL Example - Get Specific Diagnostic:

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

cURL Example - List All Diagnostics:

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

Example Response - Speed Test Results:

```
[
  {
    "id": "diag_12345",
```

```
    "serial_number": "SN123456789",
    "operation": "download",
    "target": "http://speedtest.example.com/1GB.bin",
    "status": "completed",
    "created_at": "2024-01-15T10:30:00Z",
    "completed_at": "2024-01-15T10:32:15Z",
    "results": {
      "download_speed_mbps": 95.4,
      "bytes_downloaded": 1073741824,
      "duration_seconds": 89.2,
      "success": true
    }
  },
  {
    "id": "diag_12346",
    "serial_number": "SN987654321",
    "operation": "ipping",
    "target": "8.8.8.8",
    "count": 5,
    "size": 64,
    "timeout": 1,
    "status": "completed",
    "created_at": "2024-01-15T11:15:00Z",
    "completed_at": "2024-01-15T11:15:08Z",
    "results": {
      "packets_sent": 5,
      "packets_received": 5,
      "packet_loss_percent": 0,
      "min_rtt_ms": 12.4,
      "max_rtt_ms": 15.8,
      "avg_rtt_ms": 14.1,
      "success": true
    }
  }
]
```


Example Response - WiFi Neighbor Scan:


```
[
  {
```

```
    "id": "diag_12347",
    "serial_number": "SN555666777",
    "operation": "wifi.neighbor",
    "target": "2.4GHz",
    "status": "completed",
    "created_at": "2024-01-15T12:00:00Z",
    "completed_at": "2024-01-15T12:00:45Z",
    "results": {
      "networks_found": 12,
      "strongest_signal_dbm": -35,
      "channel_utilization": {
        "1": 25,
        "6": 60,
        "11": 45
      },
      "recommended_channel": 1,
      "success": true
    }
  }
]
```


Response Codes:

Status	Description
200	Success - Returns diagnostic operation data
401	Unauthorized - Invalid or missing API token
404	Not Found - Specified diagnostic ID not found
500	Internal Server Error - System error processing request

Common Use Cases

Use Case 1: Customer Speed Complaint Troubleshooting

When customers report slow internet speeds, schedule download and upload tests to measure actual throughput, then compare results against service level agreements to identify if the issue is

network-related or device-specific.

Use Case 2: Device Connectivity Verification

For devices that appear offline or unreachable, execute ping tests to multiple targets (gateway, DNS servers, public IPs) to isolate whether the issue is local network connectivity, ISP routing, or device configuration.

Use Case 3: WiFi Performance Optimization

Before deploying new wireless equipment or when customers report WiFi issues, scan neighboring networks to identify channel congestion and interference sources, then recommend optimal channel configurations.

Use Case 4: Proactive Network Monitoring

Schedule regular diagnostic tests across your device fleet to establish baseline performance metrics and identify degrading connections before customers experience service issues.

Use Case 5: DNS Resolution Troubleshooting

When customers experience web browsing issues, test DNS lookup performance against multiple DNS servers to identify resolution failures or slow response times that could impact user experience.

Best Practices

- **Batch Operations**: When diagnosing multiple devices, stagger diagnostic schedules to avoid overwhelming network resources
- **Realistic Targets**: Use geographically appropriate test servers and targets that reflect real user traffic patterns
- **Error Handling**: Always check operation status before interpreting results - failed tests provide valuable troubleshooting data
- **Result Retention**: Diagnostic results may have limited retention periods, so retrieve and store important data promptly
- **Device Availability**: Remember that diagnostic operations execute asynchronously when devices connect, so factor in device communication schedules
- **Rate Limiting**: Implement appropriate delays between diagnostic requests to respect API rate limits and system resources
- **Security Considerations**: Avoid using diagnostic operations to test unauthorized targets or perform operations that could be interpreted as network scanning

Revision #4
Created 2026-02-04 05:08:02 UTC by ipena@zequenze.com
Updated 2026-02-11 03:00:40 UTC by ipena@zequenze.com