

OrpheAgent API Documentation

Version: orphe-agent-1.1.2

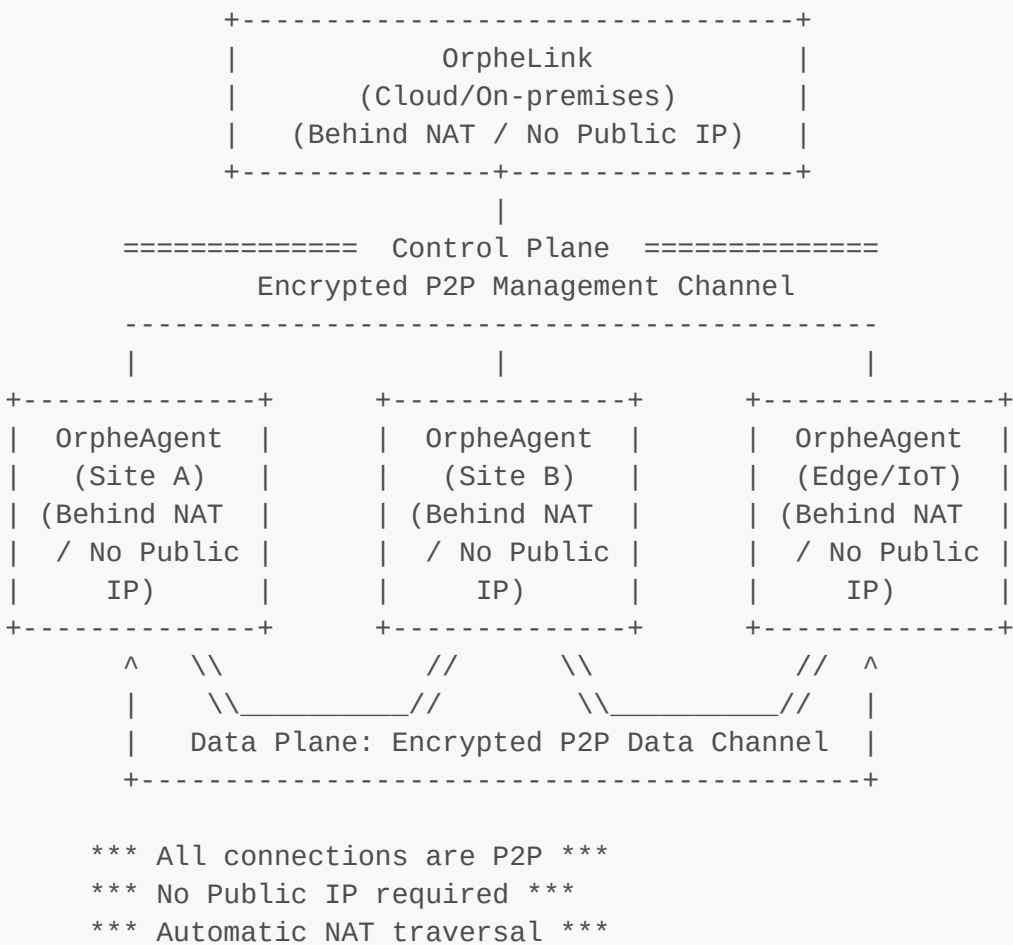
OrpheAgent is a lightweight P2P daemon purpose built for encrypted remote access. It enables centralized device management through the OrpheLink, establishing secure, private, and fully self-managed communication and data transmission without the need for VPNs or public IP addresses by automatically creating encrypted overlay networks.

OrpheAgent can be deployed across a wide range of environments, including traditional Linux systems, containerized infrastructure. This extended support allows seamless integration into routers, gateways, and other network appliances commonly found at the edge. Once installed, only a single provision key is required for the OrpheLink to take full control of management and monitoring. It also support secure and efficient site-to-site communication.

Connections between OrpheLink and OrpheAgents are handled directly, while data tunnels between OrpheAgents leverage a robust P2P architecture for site-to-site communication.

Thanks to its hardware-agnostic design, OrpheAgent can be seamlessly integrated into edge devices, embedded systems, and traditional IT infrastructures accelerating enterprise adoption of distributed networking and remote management solutions.

Architecture Overview



API Basic Information

- **Base URL:** `/api/v1`
- **Authentication Method:** Most protected API endpoints require an authorization token in the `Authorization` header. Some WebSocket APIs also accept the token through the `token` query parameter.
- **Response Format:** JSON

Swagger UI Interface

OrpheAgent provides a Swagger UI interface for users to test APIs at:

http://<ORPHEAGENT_HOST>/swagger/index.html#/

(Replace `<ORPHEAGENT_HOST>` with the IP address or domain name of the server where OrpheAgent is running.)

- **Beta version:** The Swagger service is enabled by default.
- **Production version:** You need to manually enable the Swagger service by running:

```
orphe-agent up -f -d
```

Detailed API Endpoints

Dashboards

Get Dashboard Info

- **Endpoint:** `/dashboards/get`
- **Method:** GET
- **Description:** Retrieves system dashboard information, including uptime, CPU, memory, and disk usage.
- **Authentication:** Required
- **Response Parameters:**
 - `uptime`: Uptime information containing:
 - `number`: The value of the dashboard uptime
 - `unit`: The unit of the dashboard uptime (e.g., "sec", "min", "hour")
 - `cpu_cores`: Number of CPU cores
 - `cpu_usage`: CPU usage percentage
 - `mem_total`: Total memory size (GB)
 - `mem_usage`: Used memory size (GB)
 - `cpu_temperature`: CPU temperature (°C)
 - `disk_total`: Total storage size (GB)
 - `disk_usage`: Used storage size (GB)
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "uptime": {
    "number": "5",
    "unit": "sec"
  },
  "cpu_cores": "4",
  "cpu_usage": "32",
  "mem_total": "7.74",
  "mem_usage": "23.58",
  "cpu_temperature": "41.75",
  "disk_total": "40.0",
  "disk_usage": "3",
  "status": true
}
```

System

Get System Info

- **Endpoint:** `/system/info`
- **Method:** GET
- **Description:** Get system information such as model, version, and mode.
- **Authentication:** Required
- **Response Parameters:**
 - `system_model`: Model name of the system (e.g., "OrpheAgent")
 - `version`: Current version of the system (e.g., "1.1.2")
 - `mode`: Operating mode of the system (e.g., "orphe-agent")
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "system_model": "OrpheAgent",
  "version": "1.1.2",
  "mode": "orphe-agent",
  "status": true
}
```

ControlPlane

Get Control Plane Status

- **Endpoint:** `/controlplane/status`
- **Method:** GET
- **Description:** Get the status of the control plane.
- **Authentication:** Required
- **Response Parameters:**

- **id**: The unique identifier of the agent
- **is_provision**: Boolean indicating whether the agent has been provisioned
- **node_status**: Status information of the current agent containing:
 - **status**: Boolean indicating whether the control plane status is normal
- **neighbor_status**: Array of OrpheLink connection status objects, each containing:
 - **id**: The unique identifier of the OrpheLink
 - **status**: Connection status with the OrpheLink (e.g., "direct", "relay", "offline")
 - **tx**: Transmitted bytes to the OrpheLink
 - **rx**: Received bytes from the OrpheLink
 - **latency**: Latency to the OrpheLink in milliseconds
 - **jitter**: Jitter to the OrpheLink in milliseconds
 - **protocol**: Protocol used for the connection
 - **endpoint**: Endpoint address of the OrpheLink
 - **tunnel_ip**: Tunnel IP address of the OrpheLink
 - **license_lock**: Boolean indicating whether the OrpheLink is license locked
- **status**: Boolean indicating success or failure

- **Response Example:**

```
{
  "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa",
  "is_provision": true,
  "node_status": {
    "status": true
  },
  "neighbor_status": [
    {
      "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrb",
      "status": "direct",
      "tx": 10,
      "rx": 10,
      "latency": 10,
      "jitter": 10,
      "protocol": "tcp",
      "endpoint": "10.1.1.1:8002",
      "tunnel_ip": "10.1.1.1",
      "license_lock": false
    }
  ],
  "status": true
}
```

Configuration

Save Configuration

- **Endpoint:** /config
- **Method:** POST

- **Description:** Save the configuration of the OrpheAgent.
- **Authentication:** Required
- **Request Parameters:**
 - **provision_key:** Provision key for agent registration authentication
 - **host_name:** Host name
 - **mgmt_port_http:** Management interface HTTP port
 - **mgmt_port_https:** Management interface HTTPS port
 - **controlplane_tun_port:** Control plane TUN port
 - **dataplane_tun_port:** Data plane TUN port
- **Request Body Format:**

```
{
  "provision_key": "1234567890",
  "host_name": "orphe-agent",
  "mgmt_port_http": "80",
  "mgmt_port_https": "443",
  "controlplane_tun_port": "8001",
  "dataplane_tun_port": "8002"
}
```

- **Response Parameters:**
 - **provision_key:** Provision key for agent registration authentication
 - **host_name:** Host name
 - **mgmt_port_http:** Management interface HTTP port
 - **mgmt_port_https:** Management interface HTTPS port
 - **controlplane_tun_port:** Control plane TUN port
 - **dataplane_tun_port:** Data plane TUN port
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "provision_key": "1234567890",
  "host_name": "orphe-agent",
  "mgmt_port_http": "80",
  "mgmt_port_https": "443",
  "controlplane_tun_port": "8001",
  "dataplane_tun_port": "8002",
  "status": true
}
```

Load Configuration

- **Endpoint:** `/config`
- **Method:** GET
- **Description:** Get the configuration of the OrpheAgent.

- **Authentication:** Required
- **Response Parameters:**
 - `provision_key`: Provision key for agent registration authentication
 - `host_name`: Host name
 - `mgmt_port_http`: Management interface HTTP port
 - `mgmt_port_https`: Management interface HTTPS port
 - `controlplane_tun_port`: Control plane TUN port
 - `dataplane_tun_port`: Data plane TUN port
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "provision_key": "1234567890",
  "host_name": "orphe-agent",
  "mgmt_port_http": "80",
  "mgmt_port_https": "443",
  "controlplane_tun_port": "8001",
  "dataplane_tun_port": "8002",
  "status": true
}
```

Save Basic Configuration

- **Endpoint:** `/config/basic`
- **Method:** POST
- **Description:** Save the basic configuration of the OrpheAgent (hostname and provision key only).
- **Authentication:** Required
- **Request Parameters:**
 - `provision_key`: Provision key for agent registration authentication
 - `host_name`: Host name
- **Request Body Format:**

```
{
  "provision_key": "1234567890",
  "host_name": "orphe-agent"
}
```

- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Network

Get Network Statistics

- **Endpoint:** `/network/statistics/ws`
- **Method:** GET (WebSocket)
- **Description:** Retrieve network traffic statistics over a WebSocket connection.
- **Request Parameters** (GetStatisticsInput):
 - **iface:** Network interface to monitor. Example values: "dp-tun0" for data plane, "cp-tun0" for control plane, or empty string for first priority WAN.
 - **time_length:** The time period in seconds for which to retrieve statistics.
- **Request Body Format:**

```
{
  "iface": "dp-tun0",
  "time_length": 60
}
```

- **Response Parameters** (Traffic):
 - **time:** Array of time points in Unix timestamp format
 - **tx:** Array of transmitted data values (in bytes) at each time point
 - **rx:** Array of received data values (in bytes) at each time point
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "time": ["1672531199000", "1672531200000", "1672531201000"],
  "tx": [1024.5, 1050.2, 980.7],
  "rx": [512.3, 498.6, 520.1],
  "status": true
}
```

- **Usage Notes:** This WebSocket endpoint maintains a persistent connection, continuously pushing updated network traffic statistics based on the requested network interface and time period. The client should establish a WebSocket connection to this endpoint and send a JSON request with the desired parameters to start receiving traffic data.

Jump to Service

Save Configuration

- **Endpoint:** `/jumptoservice/config`
- **Method:** POST

- **Description:** Save the configuration of jump to service.
- **Authentication:** Required
- **Request Parameters:**
 - **name:** The name of the service
 - **type:** The type of the service
 - **port:** The port number for the service
 - **jump_limit:** The jump limit for the service
- **Request Body Format:**

```
{
  "name": "service-name",
  "type": "http",
  "port": "8080",
  "jump_limit": "10"
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Load Configuration

- **Endpoint:** /jumptoservice/config
- **Method:** GET
- **Description:** Load the configuration of jump to service.
- **Authentication:** Required
- **Response Parameters:**
 - **services:** Array of service configurations, each containing:
 - **name:** The name of the service
 - **type:** The type of the service
 - **port:** The port number for the service
 - **jump_limit:** The jump limit for the service
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "services": [
    {
      "name": "service-name",
      "type": "http",
```



```

        "port": "8080",
        "jump_limit": "10"
    }
],
    "status": true
}

```

Delete Configuration

- **Endpoint:** `/jumptoservice/config/:name`
- **Method:** DELETE
- **Description:** Delete the configuration of jump to service.
- **Authentication:** Required
- **Path Parameters:**
 - `name`: The name of the service to delete
- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```

{
  "status": true
}

```

DataPlane

Node Management

Get Node

- **Endpoint:** `/dataplane/node`
- **Method:** GET
- **Description:** Get node information from the data plane.
- **Authentication:** Required
- **Response Parameters:**
 - `node`: Node information object containing:
 - `ip_addr`: The IP address of the node with subnet (e.g., "10.0.0.1/24")
 - `id`: The unique node identifier
 - `device_name`: The hostname of the node
 - `os`: The operating system of the node
 - `lan`: Array of LAN subnets managed by the node (e.g., ["192.168.0.0/24", "192.168.1.0/24"])
 - `exit_node`: Boolean indicating if this is an exit node
 - `route_to_exit`: The route to the exit node
 - `dns`: Preferred DNS server for the node
 - `site_name`: The site name of the node

- **auth**: Boolean indicating if authentication is enabled
 - **auth_timeout**: Authentication timeout in seconds
 - **expire_time**: Expiration time
 - **secure_option**: Secure option settings containing:
 - **disable_dataplane_on_boot**: Boolean to disable dataplane on boot
 - **disable_dht**: Boolean to disable DHT
 - **disable_relay_transport**: Boolean to disable relay transport
 - **disable_udp_hole_punch**: Boolean to disable UDP hole punching
 - **disable_mdns_discover**: Boolean to disable mDNS discovery
 - **neighbor**: Array of neighbor nodes (with same structure as node)
 - **status**: Boolean indicating success or failure
- **Response Example:**

```
{
  "node": {
    "ip_addr": "10.0.0.1/24",
    "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa",
    "device_name": "orphe0",
    "os": "linux",
    "lan": ["192.168.0.0/24", "192.168.1.0/24"],
    "exit_node": true,
    "route_to_exit": "10.0.0.1",
    "dns": "8.8.8.8",
    "site_name": "site1",
    "auth": true,
    "auth_timeout": 10,
    "expire_time": "2023-12-31 23:59:59 +0000 UTC"
  },
  "secure_option": {
    "disable_dataplane_on_boot": false,
    "disable_dht": false,
    "disable_relay_transport": false,
    "disable_udp_hole_punch": false,
    "disable_mdns_discover": false
  },
  "neighbor": [
    {
      "ip_addr": "10.0.0.2/24",
      "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrw",
      "device_name": "orphe1",
      "os": "linux",
      "lan": ["192.168.2.0/24"],
      "exit_node": false,
      "route_to_exit": "",
      "dns": "8.8.8.8",
      "site_name": "site2",
      "auth": true,
      "auth_timeout": 10,
      "expire_time": "2023-12-31 23:59:59 +0000 UTC"
    }
  ]
}
```

```
],  
  "status": true  
}
```

Set Node

- **Endpoint:** /dataplane/node
- **Method:** POST
- **Description:** Set Node of the data plane.
- **Authentication:** Required
- **Request Parameters:**
 - **node:** Node information object containing:
 - **ip_addr:** The IP address of the node with subnet (e.g., "10.0.0.1/24")
 - **id:** The unique node identifier
 - **device_name:** The hostname of the node
 - **os:** The operating system of the node
 - **lan:** Array of LAN subnets managed by the node
 - **exit_node:** Boolean indicating if this is an exit node
 - **route_to_exit:** The route to the exit node
 - **dns:** Preferred DNS server for the node
 - **site_name:** The site name of the node
 - **auth:** Boolean indicating if authentication is enabled
 - **auth_timeout:** Authentication timeout in seconds
 - **expire_time:** Expiration time
 - **secure_option:** Secure option settings containing:
 - **disable_dataplane_on_boot:** Boolean to disable dataplane on boot
 - **disable_dht:** Boolean to disable DHT
 - **disable_relay_transport:** Boolean to disable relay transport
 - **disable_udp_hole_punch:** Boolean to disable UDP hole punching
 - **disable_mdns_discover:** Boolean to disable mDNS discovery
- **Request Body Format:**

```
{  
  "node": {  
    "ip_addr": "10.0.0.1/24",  
    "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjsxEeTXSrwa",  
    "device_name": "orphe0",  
    "os": "linux",  
    "lan": ["192.168.0.0/24", "192.168.1.0/24"],  
    "exit_node": true,  
    "route_to_exit": "10.0.0.1",  
    "dns": "8.8.8.8",  
    "site_name": "site1",  
    "auth": true,  
    "auth_timeout": 10,  
    "expire_time": "2023-12-31 23:59:59 +0000 UTC"  
  }  
}
```

```

    },
    "secure_option": {
      "disable_dataplane_on_boot": false,
      "disable_dht": false,
      "disable_relay_transport": false,
      "disable_udp_hole_punch": false,
      "disable_mdns_discover": false
    }
  }
}

```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```

{
  "status": true
}

```

Neighbor Management

Add Neighbor

- **Endpoint:** `/dataplane/neighbor/add`
- **Method:** POST
- **Description:** Adds a new neighbor to the data plane.
- **Authentication:** Required
- **Request Parameters:**
 - **neighbor:** Array of neighbor nodes to add, each containing:
 - **ip_addr:** The IP address of the neighbor with subnet (e.g., "10.0.0.2/24")
 - **id:** The unique identifier of the neighbor node
 - **device_name:** The hostname of the neighbor node
 - **os:** The operating system of the neighbor node
 - **lan:** Array of LAN subnets managed by the neighbor node
 - **exit_node:** Boolean indicating if this neighbor is an exit node
 - **route_to_exit:** The route to the exit node (empty if not applicable)
 - **dns:** Preferred DNS server for the neighbor node
 - **site_name:** The site name of the neighbor node
 - **auth:** Boolean indicating if authentication is enabled for this neighbor
 - **auth_timeout:** Authentication timeout in seconds for this neighbor
 - **expire_time:** Expiration time
- **Request Body Format:**

```

{
  "neighbor": [
    {

```

```

    "ip_addr": "10.0.0.2/24",
    "id": "12D3KooWSeJ8HNRQBAYhqfWAKDCtJkmkJQa7pRiuhAjxEeTXSrwB",
    "device_name": "orphe1",
    "os": "linux",
    "lan": ["192.168.2.0/24"],
    "exit_node": false,
    "route_to_exit": "",
    "dns": "8.8.8.8",
    "site_name": "site2",
    "auth": true,
    "auth_timeout": 10,
    "expire_time": "2023-12-31 23:59:59 +0000 UTC"
  }
]
}

```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```

{
  "status": true
}

```

Edit Neighbor

- **Endpoint:** `/dataplane/neighbor/edit/:id`
- **Method:** POST
- **Description:** Edits an existing neighbor in the data plane.
- **Authentication:** Required
- **Path Parameters:**
 - **id:** The ID of the neighbor node to edit
- **Request Parameters:**
 - **neighbor:** Updated neighbor node information object containing:
 - **ip_addr:** The IP address of the neighbor with subnet
 - **id:** The unique identifier of the neighbor node
 - **device_name:** Updated hostname of the neighbor node
 - **os:** The operating system of the neighbor node
 - **lan:** Updated array of LAN subnets managed by the neighbor node
 - **exit_node:** Updated boolean indicating if this neighbor is an exit node
 - **route_to_exit:** Updated route to the exit node
 - **dns:** Preferred DNS server for the neighbor node
 - **site_name:** Updated site name of the neighbor node
 - **auth:** Updated boolean indicating if authentication is enabled
 - **auth_timeout:** Updated authentication timeout in seconds
 - **expire_time:** Expiration time

- **Request Body Format:**

```
{
  "neighbor": {
    "ip_addr": "10.0.0.2/24",
    "id": "12D3KooWSeJ8HNRQBAYhqfWAKDCtJkmkJQa7pRiuhAjxEeTXSrwB",
    "device_name": "orphe1",
    "os": "linux",
    "lan": ["192.168.2.0/24", "192.168.3.0/24"],
    "exit_node": true,
    "route_to_exit": "10.0.0.1",
    "dns": "8.8.8.8",
    "site_name": "site2",
    "auth": true,
    "auth_timeout": 15,
    "expire_time": "2023-12-31 23:59:59 +0000 UTC"
  }
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Delete Neighbor

- **Endpoint:** /dataplane/neighbor/delete/:id
- **Method:** POST
- **Description:** Deletes a neighbor from the data plane.
- **Authentication:** Required
- **Path Parameters:**
 - **id:** The ID of the neighbor node to delete
- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Dataplane Control

Start

- **Endpoint:** `/dataplane/start`
- **Method:** POST
- **Description:** Starts the link of the data plane.
- **Authentication:** Required
- **Request Parameters (Link):**
 - `id`: The link ID
 - `neighbor`: Array of neighbor objects, each containing:
 - `id`: The neighbor node ID
- **Request Body Format:**

```
{
  "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa",
  "neighbor": [
    {
      "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa"
    }
  ]
}
```

- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

PROF

Stop

- **Endpoint:** `/dataplane/stop`
- **Method:** POST
- **Description:** Stops the link of the data plane.
- **Authentication:** Required
- **Request Parameters (Link):**
 - `id`: The link ID
 - `neighbor`: Array of neighbor objects, each containing:
 - `id`: The neighbor node ID
- **Request Body Format:**

```
{
  "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa",
  "neighbor": [
```

```
{
  "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa"
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Restart

- **Endpoint:** /dataplane/restart
- **Method:** POST
- **Description:** Restarts the link of the data plane.
- **Authentication:** Required
- **Request Parameters (Link):**
 - **id:** The link ID
 - **neighbor:** Array of neighbor objects, each containing:
 - **id:** The neighbor node ID
- **Request Body Format:**

```
{
  "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa",
  "neighbor": [
    {
      "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa"
    }
  ]
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```


Get Status

- **Endpoint:** `/dataplane/status`
- **Method:** GET
- **Description:** Get the status of the data plane.
- **Authentication:** Required
- **Response Parameters:**
 - `id`: The unique identifier of the node
 - `node_status`: Status information of the current node containing:
 - `tunnel_ip`: Tunnel IP address of the node
 - `os`: Operating system of the node
 - `device_name`: Device name of the node
 - `daily_traffic`: Daily traffic statistics for this node
 - `tx`: Total transmitted bytes
 - `rx`: Total received bytes
 - `history_daily_traffic`: Historical daily traffic statistics for this node
 - `tx`: Historical transmitted bytes
 - `rx`: Historical received bytes
 - `status`: Boolean indicating if the node is active
 - `neighbor_status`: Array of status information for all neighbors, each containing:
 - `id`: The unique identifier of the neighbor node
 - `status`: Connection status (e.g., "direct", "relay")
 - `tx`: Transmitted bytes to this neighbor
 - `rx`: Received bytes from this neighbor
 - `latency`: Latency in milliseconds
 - `jitter`: Jitter in milliseconds
 - `quality`: Connection quality with the neighbor (e.g., "good", "bad", "offline")
 - `protocol`: Protocol used for connection (e.g., "tcp", "udp")
 - `endpoint`: The endpoint address and port
 - `tunnel_ip`: Tunnel IP address of the neighbor
 - `os`: Operating system of the neighbor
 - `device_name`: Device name of the neighbor
 - `license_lock`: Boolean indicating whether the neighbor is license locked
 - `daily_traffic`: Daily traffic statistics for this neighbor
 - `tx`: Daily transmitted bytes
 - `rx`: Daily received bytes
 - `history_daily_traffic`: Historical daily traffic statistics for this neighbor
 - `tx`: Historical transmitted bytes
 - `rx`: Historical received bytes
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjsxEeTXSrwa",
  "node_status": {
```

```

    "tunnel_ip": "10.1.1.1",
    "os": "linux",
    "device_name": "orphe0",
    "daily_traffic": {
      "tx": 1024000,
      "rx": 512000
    },
    "history_daily_traffic": {
      "tx": 5120000,
      "rx": 2560000
    },
    "status": true
  },
  "neighbor_status": [
    {
      "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwB",
      "status": "direct",
      "tx": 102400,
      "rx": 51200,
      "latency": 10.5,
      "jitter": 2.3,
      "quality": "good",
      "protocol": "tcp",
      "endpoint": "10.1.1.1:8002",
      "tunnel_ip": "10.1.1.2",
      "os": "linux",
      "device_name": "orphe1",
      "license_lock": false,
      "daily_traffic": {
        "tx": 102400,
        "rx": 51200
      },
      "history_daily_traffic": {
        "tx": 512000,
        "rx": 256000
      }
    }
  ],
  "status": true
}

```

PROF

Set Status

- **Endpoint:** `/dataplane/status/:status`
- **Method:** POST
- **Description:** Set the status of the data plane.
- **Authentication:** Required
- **Path Parameters:**
 - `status`: The status to set (e.g., "enable", "disable")
- **Response Parameters:**

- **status**: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Get Status WebSocket

- **Endpoint:** /dataplane/status/ws
- **Method:** GET (WebSocket)
- **Description:** Get real-time data plane status via WebSocket.
- **Authentication:** Required
- **Response Parameters** (StatusOutput):
 - **id**: The unique identifier of the node
 - **node_status**: Status information of the current node containing:
 - **tunnel_ip**: Tunnel IP address of the node
 - **os**: Operating system of the node
 - **device_name**: Device name of the node
 - **daily_traffic**: Daily traffic statistics for this node
 - **tx**: Total transmitted bytes
 - **rx**: Total received bytes
 - **history_daily_traffic**: Historical daily traffic statistics for this node
 - **tx**: Historical transmitted bytes
 - **rx**: Historical received bytes
 - **status**: Boolean indicating if the node is active
 - **neighbor_status**: Array of status information for all neighbors, each containing:
 - **id**: The unique identifier of the neighbor node
 - **status**: Connection status (e.g., "direct", "relay")
 - **tx**: Transmitted bytes to this neighbor
 - **rx**: Received bytes from this neighbor
 - **latency**: Latency in milliseconds
 - **jitter**: Jitter in milliseconds
 - **quality**: Connection quality with the neighbor (e.g., "good", "bad", "offline")
 - **protocol**: Protocol used for connection (e.g., "tcp", "udp")
 - **endpoint**: The endpoint address and port
 - **tunnel_ip**: Tunnel IP address of the neighbor
 - **os**: Operating system of the neighbor
 - **device_name**: Device name of the neighbor
 - **license_lock**: Boolean indicating whether the neighbor is license locked
 - **daily_traffic**: Daily traffic statistics for this neighbor
 - **tx**: Daily transmitted bytes
 - **rx**: Daily received bytes
 - **history_daily_traffic**: Historical daily traffic statistics for this neighbor

- **tx**: Historical transmitted bytes
- **rx**: Historical received bytes
- **status**: Boolean indicating success or failure
- **Response Example:**

```
{
  "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa",
  "node_status": {
    "tunnel_ip": "10.1.1.1",
    "os": "linux",
    "device_name": "orphe0",
    "daily_traffic": {
      "tx": 1024000,
      "rx": 512000
    },
    "history_daily_traffic": {
      "tx": 5120000,
      "rx": 2560000
    },
    "status": true
  },
  "neighbor_status": [
    {
      "id": "12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrb",
      "status": "direct",
      "tx": 102400,
      "rx": 51200,
      "latency": 10.5,
      "jitter": 2.3,
      "quality": "excellent",
      "protocol": "tcp",
      "endpoint": "10.1.1.1:8002",
      "tunnel_ip": "10.1.1.2",
      "os": "linux",
      "device_name": "orphe1",
      "license_lock": false,
      "daily_traffic": {
        "tx": 102400,
        "rx": 51200
      },
      "history_daily_traffic": {
        "tx": 512000,
        "rx": 256000
      }
    }
  ],
  "status": true
}
```

- **Usage Notes:** This WebSocket endpoint maintains a persistent connection, continuously pushing updated data plane status information. The client should establish a WebSocket connection to this endpoint to receive streaming updates about node status, traffic statistics, and neighbor connection quality.

Routing

Get Route Subnet to Exit

- **Endpoint:** `/dataplane/routesubnettoexit`
- **Method:** GET
- **Description:** Get subnets routed to the exit node in the data plane.
- **Authentication:** Required
- **Response Parameters** (RouteSubnetToExitOutput):
 - `route_to_exit`: The route to the exit node IP address
 - `subnet`: Array of subnet strings routed to the exit node
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "route_to_exit": "10.1.1.1",
  "subnet": ["10.1.1.0/24", "10.1.2.0/24"],
  "status": true
}
```

Set Route Subnet to Exit

- **Endpoint:** `/dataplane/routesubnettoexit`
- **Method:** POST
- **Description:** Set subnets to route to the exit node in the data plane.
- **Authentication:** Required
- **Request Parameters** (RouteSubnetToExitInput):
 - `subnet`: Array of subnet strings to route to the exit node
- **Request Body Format:**

```
{
  "subnet": ["10.1.1.0/24", "10.1.2.0/24"]
}
```

- **Response Parameters** (RouteSubnetToExitOutput):
 - `route_to_exit`: The route to the exit node IP address
 - `subnet`: Array of subnet strings routed to the exit node
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "route_to_exit": "10.1.1.1",
  "subnet": ["10.1.1.0/24", "10.1.2.0/24"],
  "status": true
}
```

Relay Addresses

Get Relay Addresses

- **Endpoint:** `/dataplane/relayaddr`
- **Method:** GET
- **Description:** Get self-owned relay addresses from the data plane.
- **Authentication:** Required
- **Response Parameters:**
 - `multi_addrs`: Array of multiaddress strings for relay servers
 - `rate_limit_mbps`: Rate limit in megabits per second
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "multi_addrs": [
    "/ip4/1.2.3.4/udp/4001/quic-
v1/p2p/12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrwa",
    "/ip4/5.6.7.8/udp/4001/quic-
v1/p2p/12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjxEeTXSrbw"
  ],
  "rate_limit_mbps": 10,
  "status": true
}
```

PROF

Set Relay Addresses

- **Endpoint:** `/dataplane/relayaddr`
- **Method:** POST
- **Description:** Set self-owned relay addresses for the data plane.
- **Authentication:** Required
- **Request Parameters:**
 - `multi_addrs`: Array of multiaddress strings for relay servers
 - `rate_limit_mbps`: Rate limit in megabits per second
- **Request Body Format:**

```
{
  "multi_addrs": [
```

```

        "/ip4/1.2.3.4/udp/4001/quic-
v1/p2p/12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjsxEeTXSrwa",
        "/ip4/5.6.7.8/udp/4001/quic-
v1/p2p/12D3KooWSeJ8HNRQBAYhqfWAKDctJkmkJQa7pRiuhAjsxEeTXSrb"
    ],
    "rate_limit_mbps": 10
}

```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```

{
  "status": true
}

```

Port Forwarding

List Port Forwarding Rules

- **Endpoint:** `/dataplane/portforwarding/list`
- **Method:** GET
- **Description:** List all port forwarding rules in the data plane.
- **Authentication:** Required
- **Response Parameters:**
 - **list:** Array of port forwarding rules, each containing:
 - **name:** The name of the port forwarding rule
 - **lan_dst_ip:** The destination LAN IP address
 - **lan_dst_mac:** The destination MAC address
 - **lan_dst_port:** The destination port on the LAN
 - **tun_src_port:** The source port on the tunnel interface
 - **protocol:** The protocol used (e.g., "tcp", "udp", "all")
 - **description:** Description of the port forwarding rule
 - **status:** Boolean indicating success or failure
- **Response Example:**

```

{
  "list": [
    {
      "name": "http",
      "lan_dst_ip": "192.168.0.1",
      "lan_dst_mac": "ee:d0:9c:8f:ea:8a",
      "lan_dst_port": "80",
      "tun_src_port": "8000",
      "protocol": "tcp",

```

```

    "description": "HTTP server"
  },
  {
    "name": "ssh",
    "lan_dst_ip": "192.168.0.2",
    "lan_dst_mac": "ee:d0:9c:8f:ea:8b",
    "lan_dst_port": "22",
    "tun_src_port": "2222",
    "protocol": "tcp",
    "description": "SSH server"
  }
],
"status": true
}

```

Add Port Forwarding Rule

- **Endpoint:** /dataplane/portforwarding/add
- **Method:** POST
- **Description:** Adds a new rule for port forwarding in the data plane.
- **Authentication:** Required
- **Request Parameters:**
 - **name:** The name of the port forwarding rule
 - **lan_dst_ip:** The destination LAN IP address
 - **lan_dst_mac:** The destination MAC address
 - **lan_dst_port:** The destination port on the LAN
 - **tun_src_port:** The source port on the tunnel interface
 - **protocol:** The protocol used (e.g., "tcp", "udp", "all")
 - **description:** Description of the port forwarding rule
- **Request Body Format:**

```

{
  "name": "http",
  "lan_dst_ip": "192.168.0.1",
  "lan_dst_mac": "ee:d0:9c:8f:ea:8a",
  "lan_dst_port": "80",
  "tun_src_port": "8000",
  "protocol": "all",
  "description": "HTTP server"
}

```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**


```
{
  "status": true
}
```

Delete Port Forwarding Rule

- **Endpoint:** `/dataplane/portforwarding/delete/:name`
- **Method:** POST
- **Description:** Deletes a port forwarding rule from the data plane by its name.
- **Authentication:** Required
- **Path Parameters:**
 - `name`: The name of the port forwarding rule to delete
- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Access Control List (ACL)

Set Mode

- **Endpoint:** `/dataplane/acl/mode/:mode`
- **Method:** POST
- **Description:** Sets the ACL mode for the data plane.
- **Authentication:** Required
- **Path Parameters:**
 - `mode`: The ACL mode (e.g., "black_list" or "white_list")
- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Get Mode

- **Endpoint:** `/dataplane/acl/mode`
- **Method:** GET

- **Description:** Retrieves the current ACL mode of the data plane.
- **Authentication:** Required
- **Response Parameters:**
 - **mode:** The current ACL mode (e.g., "black_list" or "white_list")
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "mode": "black_list",
  "status": true
}
```

List ACL Rules

- **Endpoint:** /dataplane/acl/list
- **Method:** GET
- **Description:** Lists all ACL rules in the data plane.
- **Authentication:** Required
- **Response Parameters:**
 - **acl_mode:** The current ACL mode (e.g., "black_list" or "white_list")
 - **list:** Array of ACL rules, each containing:
 - **hostname:** The hostname of the device
 - **ip:** The IP address of the device
 - **mac:** The MAC address of the device
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "acl_mode": "black_list",
  "list": [
    {
      "hostname": "device1",
      "ip": "10.0.0.1",
      "mac": "ee:d0:9c:8f:ea:8a"
    },
    {
      "hostname": "device2",
      "ip": "10.0.0.2",
      "mac": "ee:d0:9c:8f:ea:8b"
    }
  ],
  "status": true
}
```

Add ACL Rule

- **Endpoint:** `/dataplane/acl`
- **Method:** POST
- **Description:** Adds a new ACL rule to the data plane.
- **Authentication:** Required
- **Request Parameters** (ACLRuleInput):
 - `acl_mode`: The ACL mode (e.g., "black_list" or "white_list")
 - `list`: Array of ACL rules to add, each containing:
 - `hostname`: The hostname of the device
 - `ip`: The IP address of the device
 - `mac`: The MAC address of the device
- **Request Body Format:**

```
{
  "acl_mode": "black_list",
  "list": [
    {
      "hostname": "device1",
      "ip": "10.0.0.1",
      "mac": "ee:d0:9c:8f:ea:8a"
    }
  ]
}
```

- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Delete ACL Rule

- **Endpoint:** `/dataplane/acl`
- **Method:** DELETE
- **Description:** Deletes an ACL rule from the data plane.
- **Authentication:** Required
- **Request Parameters** (ACLMacInput):
 - `acl_mode`: The ACL mode (e.g., "black_list" or "white_list")
 - `list`: Array of MAC addresses to delete from the ACL
- **Request Body Format:**

```
{
  "acl_mode": "black_list",
```

```
{
  "list": ["ee:d0:9c:8f:ea:8a", "ee:d0:9c:8f:ea:8b"]
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Services

List Services

- **Endpoint:** /dataplane/services/list
- **Method:** GET
- **Description:** Lists all services in the dataplane.
- **Authentication:** Required
- **Response Parameters:**
 - **services:** Array of service objects, each containing:
 - **name:** The name of the service
 - **type:** The type of service (e.g., "http", "mysql")
 - **port:** The port number the service is running on
 - **protocol:** The protocol used by the service (e.g., "tcp", "udp")
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "services": [
    {
      "name": "web service",
      "type": "http",
      "port": "8080",
      "protocol": "tcp"
    },
    {
      "name": "database",
      "type": "mysql",
      "port": "3306",
      "protocol": "tcp"
    }
  ],
  "status": true
}
```

Edit Service

- **Endpoint:** `/dataplane/services/edit`
- **Method:** POST
- **Description:** Edits an existing service in the data plane.
- **Authentication:** Required
- **Request Parameters** (Service):
 - **name:** The name of the service
 - **type:** The type of service (e.g., "https")
 - **port:** The port number the service is running on
 - **protocol:** The protocol used by the service (e.g., "tcp", "udp")
- **Request Body Format:**

```
{
  "name": "web service",
  "type": "https",
  "port": "8443",
  "protocol": "tcp"
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

PROF

MTU (Maximum Transmission Unit)

Get MTU

- **Endpoint:** `/dataplane/mtu`
- **Method:** GET
- **Description:** Retrieves the Maximum Transmission Unit (MTU) settings.
- **Authentication:** Required
- **Response Parameters** (MTUOutput):
 - **mtu:** The current MTU value in bytes
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "mtu": 1280,
  "status": true
}
```

Set MTU

- **Endpoint:** /dataplane/mtu
- **Method:** POST
- **Description:** Updates the MTU settings.
- **Authentication:** Required
- **Request Parameters** (MTUInput):
 - **mtu:** The new MTU value in bytes
- **Request Body Format:**

```
{
  "mtu": 1280
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Tunnel Mode

PROF

Get Tunnel Mode

- **Endpoint:** /dataplane/tunnelmode
- **Method:** GET
- **Description:** Retrieves the current tunnel mode setting of the data plane (high throughput or low latency).
- **Authentication:** Required
- **Response Parameters** (TunnelModeOutput):
 - **mode:** The current tunnel mode (e.g., "high_throughput", "low_latency")
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "mode": "high_throughput",
  "status": true
}
```

Set Tunnel Mode

- **Endpoint:** `/dataplane/tunnelmode`
- **Method:** POST
- **Description:** Sets the tunnel mode for the data plane to optimize for either high throughput or low latency.
- **Authentication:** Required
- **Request Parameters** (TunnelModelInput):
 - `mode`: The tunnel mode to set (e.g., "high_throughput", "low_latency")
- **Request Body Format:**

```
{
  "mode": "high_throughput"
}
```

- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

PROF

SNAT (Source Network Address Translation)

Get SNAT

- **Endpoint:** `/dataplane/snats`
- **Method:** GET
- **Description:** Retrieves the current SNAT (Source Network Address Translation) setting of the data plane.
- **Authentication:** Required
- **Response Parameters** (SNATOutput):
 - `enable`: Boolean indicating whether SNAT is enabled or not
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "enable": true,
  "status": true
}
```

Set SNAT

- **Endpoint:** `/dataplane/snat/{status}`
- **Method:** POST
- **Description:** Sets the SNAT (Source Network Address Translation) for the data plane.
- **Authentication:** Required
- **Path Parameters:**
 - `status`: The status to set ("enable" or "disable")
- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Subnet and End Device

Get Subnet List

- **Endpoint:** `/dataplane/subnetlist`
- **Method:** GET
- **Description:** Retrieves a list of subnets in the data plane.
- **Authentication:** Required
- **Response Parameters** (SubnetListOutput):
 - `list`: Array of subnet CIDR strings (e.g., "192.168.0.0/24")
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "list": [
    "192.168.0.0/24",
    "192.168.1.0/24",
    "10.0.0.0/16"
  ],
  "status": true
}
```


Get End Device List

- **Endpoint:** /dataplane/enddeviceList
- **Method:** GET
- **Description:** Retrieves a list of end devices in the data plane.
- **Authentication:** Required
- **Response Parameters** (EndDeviceListOutput):
 - **list:** Array of end device objects (EndDevice), each containing:
 - **hostname:** The hostname of the device
 - **ip:** The IP address of the device
 - **mac:** The MAC address of the device
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "list": [
    {
      "hostname": "device1",
      "ip": "192.168.0.10",
      "mac": "ee:d0:9c:8f:ea:8a"
    },
    {
      "hostname": "device2",
      "ip": "192.168.0.11",
      "mac": "ee:d0:9c:8f:ea:8b"
    }
  ],
  "status": true
}
```

Portal Authentication

PROF

Login

- **Endpoint:** /dataplane/portal/login
- **Method:** POST
- **Description:** Handles the authentication for the portal.
- **Request Parameters** (PortalLoginInput):
 - **Username:** The username for authentication
 - **Password:** The password for authentication
- **Request Body Format:**

```
{
  "Username": "user",
  "Password": "password"
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Exit Node Configuration

Get Exit Node List

- **Endpoint:** /dataplane/exitnode/list
- **Method:** GET
- **Description:** Get the list of available exit nodes.
- **Authentication:** Required
- **Response Parameters** (ExitNodeListOutput):
 - **list:** Array of available exit nodes, each containing:
 - **device_name:** The device name of the exit node
 - **ip:** The IP address of the exit node
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "list": [
    {
      "device_name": "exit-node-1",
      "ip": "10.0.0.2"
    },
    {
      "device_name": "exit-node-2",
      "ip": "10.0.0.3"
    }
  ],
  "status": true
}
```

Get Exit Node Config

- **Endpoint:** /dataplane/exitnode/config/{id}
- **Method:** GET
- **Description:** Get the exit node configuration.
- **Authentication:** Required

- **Path Parameters:**
 - **id:** The node ID
- **Response Parameters** (ExitNodeModeOutput):
 - **enable:** Boolean indicating if the node is an exit node
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "enable": true,
  "status": true
}
```

Set Exit Node Config

- **Endpoint:** /dataplane/exitnode/config/{id}
- **Method:** POST
- **Description:** Set the exit node configuration.
- **Authentication:** Required
- **Path Parameters:**
 - **id:** The node ID
- **Request Parameters** (ExitNodeModelInput):
 - **enable:** Boolean to enable or disable exit node mode for the target node
- **Request Body Format:**

```
{
  "enable": true
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

LAN Subnet Sharing

Get LAN Subnet Sharing Config

- **Endpoint:** /dataplane/lan/subnetsharing/config/{id}
- **Method:** GET

- **Description:** Get the LAN subnet sharing configuration.
- **Authentication:** Required
- **Path Parameters:**
 - **id:** The node ID
- **Response Parameters** (LANSubnetSharingOutput):
 - **enable:** Boolean indicating if LAN subnet sharing is enabled
 - **list:** Array of LAN subnets, each containing:
 - **subnet:** The LAN subnet CIDR (e.g., "192.168.0.0/24")
 - **enable:** Boolean indicating if this subnet sharing is enabled
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "enable": true,
  "list": [
    {
      "subnet": "192.168.0.0/24",
      "enable": true
    },
    {
      "subnet": "192.168.1.0/24",
      "enable": true
    }
  ],
  "status": true
}
```

Set LAN Subnet Sharing Config

- **Endpoint:** /dataplane/lan/subnetsharing/config/{id}
- **Method:** POST
- **Description:** Set the LAN subnet sharing configuration.
- **Authentication:** Required
- **Path Parameters:**
 - **id:** The node ID
- **Request Parameters** (LANSubnetSharingInput):
 - **enable:** Boolean to enable or disable LAN subnet sharing
 - **list:** Array of LAN subnets to share, each containing:
 - **subnet:** The LAN subnet CIDR (e.g., "192.168.0.0/24")
 - **enable:** Boolean to enable or disable this specific subnet sharing
- **Request Body Format:**

```
{
  "enable": true,
  "list": [
    {
```

```

    "subnet": "192.168.0.0/24",
    "enable": true
  },
  {
    "subnet": "192.168.1.0/24",
    "enable": true
  }
]
}

```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```

{
  "status": true
}

```

Route to Exit

Get Route to Exit Config

- **Endpoint:** `/dataplane/routetoexit/config/{id}`
- **Method:** GET
- **Description:** Get the route to exit node configuration.
- **Authentication:** Required
- **Path Parameters:**
 - **id:** The node ID
- **Response Parameters** (RouteToExitOutput):
 - **enable:** Boolean indicating if routing to exit node is enabled
 - **exit_node:** Exit node configuration containing:
 - **device_name:** The device name of the exit node
 - **ip:** The IP address of the exit node
 - **subnets:** Array of subnet CIDR strings to route to the exit node
 - **status:** Boolean indicating success or failure
- **Response Example:**

```

{
  "enable": true,
  "exit_node": {
    "device_name": "exit-node-1",
    "ip": "10.0.0.2"
  },
  "subnets": ["192.168.0.0/24", "192.168.1.0/24"],
}

```

```
"status": true
}
```

Set Route to Exit Config

- **Endpoint:** `/dataplane/routetoexit/config/{id}`
- **Method:** POST
- **Description:** Set the route to exit node configuration.
- **Authentication:** Required
- **Path Parameters:**
 - `id`: The node ID
- **Request Parameters** (RouteToExitInput):
 - `enable`: Boolean to enable or disable routing to exit node
 - `exit_node`: Exit node configuration containing:
 - `device_name`: The device name of the exit node
 - `ip`: The IP address of the exit node
 - `subnets`: Array of subnet CIDR strings to route to the exit node
- **Request Body Format:**

```
{
  "enable": true,
  "exit_node": {
    "device_name": "exit-node-1",
    "ip": "10.0.0.2"
  },
  "subnets": ["192.168.0.0/24", "192.168.1.0/24"]
}
```

- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Device Configuration

Rename Device Config

- **Endpoint:** `/dataplane/config/rename/{id}`
- **Method:** POST
- **Description:** Rename the device configuration with a new device name.

- **Authentication:** Required
- **Path Parameters:**
 - **id:** The node ID
- **Request Parameters** (RenameInput):
 - **device_name:** The new device name (required)
- **Request Body Format:**

```
{
  "device_name": "new-device-name"
}
```

- **Response Parameters:**
 - **status:** Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

Hole Punch Event

Get Hole Punch Event

- **Endpoint:** /dataplane/holepunchevent/ws
- **Method:** GET (WebSocket)
- **Description:** Handles WebSocket connections for hole punching events. Provides real-time notifications about NAT traversal attempts, including address exchange, hole punch attempts, and connection status updates between peers.
- **Response Parameters** (HolePunchEvent):
 - **peer_id:** The unique identifier of the peer involved in the hole punching event
 - **mechanism:** The mechanism used for hole punching. Possible values:
 - **DPDT:** Decentralized Peer Discovery and Traversal
 - **CRAT:** Centralized Relay-Assisted Traversal
 - **event_type:** The type of event. Possible values:
 - **AddressExchange:** Address information exchange between peers
 - **StartHolePunch:** Hole punching process initiated
 - **HolePunchAttempt:** Attempt to establish direct connection
 - **EndHolePunch:** Hole punching process completed
 - **msg:** Detailed message describing the event
 - **timestamp:** RFC3339 formatted timestamp of when the event occurred
- **Response Example:**

```
{
  "peer_id": "12D3KooWAbCdEfGhIjKlMnOpQrStUvWxYz",
  "mechanism": "dcutr",
  "event_type": "StartHolePunch",
  "msg": "Initiating hole punch with peer",
  "timestamp": "2025-10-19T10:30:45Z"
}
```

- **Usage Notes:** This WebSocket endpoint provides real-time monitoring of P2P connection establishment events. The connection remains open and continuously broadcasts hole punching events as they occur. Clients should establish a WebSocket connection to receive streaming updates about NAT traversal activities between data plane nodes.

Service

Get Service Status WebSocket

- **Endpoint:** `/service/status/ws`
- **Method:** GET (WebSocket)
- **Description:** Gets real-time service status via WebSocket, providing proactive notification of OrpheAgent and OrpheLink connection status.
- **Response Parameters** (ServiceStatus):
 - `license_lock`: Boolean indicating whether the service is locked by license status
 - `status`: Boolean indicating whether the service is currently enabled and connected
- **Response Example:**

```
{
  "license_lock": false,
  "status": true
}
```

- **Usage Notes:** This WebSocket endpoint maintains a persistent connection, continuously pushing updated service status information. The client should establish a WebSocket connection to this endpoint to receive streaming updates about OrpheAgent and OrpheLink connection status. The connection status reflects whether the service is actively connected to the control plane.

Get Service Toggle Status

- **Endpoint:** `/service/toggle`
- **Method:** GET
- **Description:** Get the current service toggle state.
- **Authentication:** Required
- **Response Parameters** (ServiceToggleOutput):
 - `enable`: Boolean indicating whether the service is enabled
 - `enable_on_boot`: Boolean indicating whether the service is enabled on boot

- **status**: Boolean indicating success or failure
- **Response Example:**

```
{
  "enable": true,
  "enable_on_boot": true,
  "status": true
}
```

Set Service Toggle Status

- **Endpoint:** `/service/toggle`
- **Method:** POST
- **Description:** Enables or disables the service.
- **Authentication:** Required
- **Request Parameters** (ServiceToggleInput):
 - **enable**: Boolean to enable or disable the service
 - **enable_on_boot**: Boolean to enable or disable the service on boot
- **Request Body Format:**

```
{
  "enable": true,
  "enable_on_boot": true
}
```

- **Response Parameters:**
 - **status**: Boolean indicating success or failure
- **Response Example:**

```
{
  "status": true
}
```

PROF

Profile

Create Profile

- **Endpoint:** `/profile`
- **Method:** POST
- **Description:** Create a new profile.
- **Authentication:** Required
- **Request Parameters:**
 - **name**: Profile name

- **Request Body Format:**

```
{
  "name": "default"
}
```

- **Response Parameters:**

- **status:** Boolean indicating success or failure

- **Response Example:**

```
{
  "status": true
}
```

Get Profiles

- **Endpoint:** `/profile`

- **Method:** GET

- **Description:** Get the list of profiles.

- **Authentication:** Required

- **Response Parameters:**

- **profiles:** Array of profile objects, each containing:
 - **name:** Profile name
 - **database_name:** Database file name
 - **in_used:** Boolean indicating whether the profile is currently in use
- **status:** Boolean indicating success or failure

- **Response Example:**

```
{
  "profiles": [
    {
      "name": "default",
      "database_name": "orphe-agent.db",
      "in_used": true
    },
    {
      "name": "lab",
      "database_name": "orphe-agent-lab.db",
      "in_used": false
    }
  ],
  "status": true
}
```

Delete Profile

- **Endpoint:** `/profile`
- **Method:** DELETE
- **Description:** Delete a profile.
- **Authentication:** Required
- **Request Parameters:**
 - `name`: Profile name
- **Request Body Format:**

```
{  
  "name": "lab"  
}
```

- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{  
  "status": true  
}
```

Use Profile

- **Endpoint:** `/profile/use`
- **Method:** POST
- **Description:** Set a profile as the active profile.
- **Authentication:** Required
- **Request Parameters:**
 - `name`: Profile name
- **Request Body Format:**

```
{  
  "name": "default"  
}
```

- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Response Example:**

```
{  
  "status": true  
}
```

```
}
```

Develop

WebSocket Terminal

- **Endpoint:** `/develop/ws/terminal`
- **Method:** GET (WebSocket)
- **Description:** Open Websocket connection to the execution session of web terminal.
- **Authentication:** Required
- **Query Parameters:**
 - `token`: Authentication token for validating the WebSocket connection
- **WebSocket Communication:**
 - **Terminal Input:** Client sends text or binary data to be processed as terminal input
 - **Terminal Output:** Server sends binary data containing terminal output
 - **Terminal Resize:** Client can send JSON message with format:

```
{
  "action": "resize",
  "cols": 120,
  "rows": 30
}
```

- **Response Parameters:**
 - `status`: Boolean indicating success or failure
- **Usage Notes:** This WebSocket endpoint provides a full interactive terminal session. The connection remains open until explicitly closed by the client. Terminal mode can be either standard shell or orphe-console, depending on the current mode setting.

Common Error Codes

PROF

Most APIs will return an HTTP 503 (Service Unavailable) error when failing, along with an error message in the following format:

```
{
  "err_message": "Error description",
  "status": false
}
```

Authentication Information

Most protected API endpoints require an authorization token in the `Authorization` header.

For WebSocket APIs, the implementation may also accept the token through the `token` query parameter. The `/develop/ws/terminal` endpoint requires the `token` query parameter.