Last updated: June 8, 2023

Introduction

Base URL: `http://localhost`

This documentation aims to provide all the information you need to work with Babel.

As you scroll, you'll see code examples for working with the API in different programming languages in the dark area to the right (or as part of the content on mobile). You can switch the language used with the tabs at the top right (or from the nav menu at the top left on mobile) to suit your programming needs.

Authenticating requests

To authenticate requests, include an **`Authorization`** header with the value **`"Bearer BEARER_TOKEN"`**.

All authenticated endpoints are marked with a `requires authentication` badge in the documentation below.

FTTH Controller

API for managing FTTH connections.

Get Customer Connection

requires authentication

This endpoint allows you to get the customer connection params from Radius and status from SmartOLT.

Request Try it out ⚡

GET **`v1/ftth/customer`**

Headers

`Authorization`
Example: `Bearer BEARER_TOKEN`

`Content-Type`
Example: `application/json`

`Accept`
Example: `application/json`

Body Parameters

`username` string
The username in radius Example: `10.132.139.232`

`onu_sn` string
The ONU serial number Example: `HWTC31E0CJSD`

bash javascript php python

Example request:

```bash
curl --request GET \
    --get "http://localhost/v1/ftth/customer" \
    --header "Authorization: Bearer BEARER_TOKEN" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json" \
    --data "{
    \"username\": \"10.132.139.232\",
    \"onu_sn\": \"HWTC31E0CJSD\"
}"
```

Example response (200):


```
{
    "status": true,
    "message": "Customer connection found in Radius and ONU is available in SmartOLT.",
    "data": {
        "radius": {
            "status": "success",
            "message": "User found",
            "detail": {
                "id": 29881,
                "name": "Gustavo   Ivan   Escalon CastroESCALON CASTRO",
                "username": "10.112.239.132",
                "password": "password",
                "bandwith_plan": "61440k/61440k",
                "main_ip": "10.192.310.114",
                "node": "CASTROL FTTH",
```


Example response (206):


```
{
    "status": "success",
    "message": "Error getting ONU status, but customer connection was found in Radius.",
    "data": {
        "radius": {
            "status": "success",
            "message": "User found",
            "detail": {
                "id": 29881,
                "name": "Gustavo   Ivan   Escalon CastroESCALON CASTRO",
                "username": "10.112.239.232",
                "password": "b3afs82",
                "bandwith_plan": "61440k/61440k",
                "main_ip": "10.192.510.114",
                "node": "CASTROL FTTH",
```


Example response (400):

Example: `application/json`

Accept

Example: `application/json`

Body Parameters

name string

The customer name. Example: `Kwargs`

username string

The username in radius. Example: `1.1.1.1`

bandwith_plan string

The bandwidth plan. Example: `61440k/61440k`

node string

The node. Example: `CASTROL FTTH`

main_ip string

The main IP. Example: `10.192.310.11`

onu_sn string

The ONU serial number. Example: `HWTC31E0CJSF`


```
{
    "status": "error",
    "message": "Invalid data.",
    "detail": {
        "username": [
            "The username field is required."
        ],
        "onu_sn": [
            "The onu sn field is required."
        ]
    }
}
```


Example response (404):


```
{
    "status": "error",
    "message": "Error getting customer connection. Customer not found in Radius.",
    "data": {
        "radius": {
            "status": "error",
            "message": "User does not exist."
        }
    }
}
```



```
            "bandwith_plan": "61440k/61440k",
            "main_ip": "10.192.310.114",
            "node": "CASTROL FTTH",
```


Example response (206):


```
{
    "status": "success",
    "message": "Error getting ONU status, but customer connection was found in Radius.",
    "data": {
        "radius": {
            "status": "success",
            "message": "User found",
            "detail": {
                "id": 29881,
                "name": "Gustavo   Ivan   Escalon CastroESCALON CASTRO",
                "username": "10.112.239.232",
                "password": "b3afs82",
                "bandwith_plan": "61440k/61440k",
                "main_ip": "10.192.510.114",
                "node": "CASTROL FTTH",
```


Example response (400):


```
    \"name\": \"Kwargs\",
    \"username\": \"1.1.1.1\",
    \"bandwith_plan\": \"61440k\\/61440k\",
    \"node\": \"CASTROL FTTH\",
    \"main_ip\": \"10.192.310.11\",
    \"onu_sn\": \"HWTC31E0CJSF\"
}"
```


Example response (200):


```
{
    "status": true,
    "message": "Customer connection updated in Radius and ONU rebooted in SmartOLT.",
    "data": {
        "radius": {
            "status": "success",
            "message": "Bandwidth updated successfully",
            "detail": {
                "status": "success",
                "message": "User found",
                "detail": {
                    "id": 29881,
                    "name": "Kwargs",
                    "username": "10.112.239.232",
                    "password": "b3a7fw82",
```


Example response (206):


```
{
    "status": "success",
    "message": "Customer connection updated in Radius, but ONU is not available in SmartOLT
    "data": {
        "radius": {
```

```
            "status": "success",
            "message": "User found",
            "detail": {
                "id": 29881,
                "name": "Kwargs",
                "username": "10.112.239.232",
                "password": "b3afswe82",
                "bandwith_plan": "61440k/61440k",
                "main_ip": "10.192.510.114",
                "node": "CASTROL_FTTH"
```


Example response (422):


```
{
    "status": "error",
    "message": "Error updating customer connection.",
    "data": {
        "radius": {
            "status": "error",
            "message": "User does not exist."
        }
    }
}
```


Example response (500):


```
{
    "status": "error",
    "message": "Error getting customer connection."
}
```


Mikrotik Controller

API for managing Mikrotik

Wireless

To manage queues connections

Example response (206):


```
{
    "status": "success",
    "message": "Customer connection updated in Radius, but ONU is not available in SmartOLT",
    "data": {
        "radius": {
            "status": "success",
            "message": "User found",
            "detail": {
                "id": 29881,
                "name": "Kwargs",
                "username": "10.112.239.232",
                "password": "b3afswe82",
                "bandwith_plan": "61440k/61440k",
                "main_ip": "10.192.510.114",
                "node": "CASTROL_FTTH"
```


Example response (422):


```
{
    "status": "error",
    "message": "Error updating customer connection."
{
    "name": "Mikrotik-Cristian"
}
```


Example response (500, no route to host):


```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or body params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

Get Contracts

This endpoint allows you to get contracts information from a Mikrotik server.

ℹ This will be useful to get the clients that are in an specific router.

Request Try it out ⚡

GET `v1/contracts`

Headers

`Content-Type`
Example: `application/xml`

`Accept`
Example: `application/json`

Body Parameters

`ip` string
The server ip. Example: `192.168.1.1`

Example request:

```
curl --request GET \
    --get "http://localhost/v1/contracts" \
    --header "Content-Type: application/xml" \
    --header "Accept: application/json" \
    --data "{
    \"ip\": \"192.168.1.1\"
}"
```

Example response (200, success):

```
{
    "ip_server": "192.168.2.184",
    "clientes_activos": 1,
    "clientes_details": [
        {
            "cliente_ip": "1.1.1.1",
            "download": "102400 Kbps",
            "upload": "3072 Kbps",
            "estado": "activo"
        }
    ]
}
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or body params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

Example response (400):

```
{
    "status": "error",
    "message": "Invalid data.",
    "detail": {
        "username": [
            "The username field is required."
        ],
        "onu_sn": [
            "The onu sn field is required."
        ]
    }
}
```

Example response (404):

```
{
    "status": "error",
    "message": "Error getting customer connection. Customer not found in Radius.",
    "data": {
```

```
{
    "message": "BABEL : ¡Cola/s creada/s con éxito!"
}
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or missing params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

Update Contracts

This endpoint allows you to update the banwidth profile of a 'contract or contracts' or ip in a Mikrotik server.

ℹ With this you are able to update the "contracts" (queues with address-list).

Request Try It out ⚡

PUT `v1/contracts`

Headers

`Content-Type`
Example: `application/xml`

`Accept`
Example: `application/json`

Body Parameters

`ip` string
The server ip. Example: `192.168.1.1`

▶ **`clientes`** object
A list of objects that contains the clients params.

Example request:

```
curl --request PUT \
    "http://localhost/v1/contracts" \
    --header "Content-Type: application/xml" \
    --header "Accept: application/json" \
    --data "{
    \"ip\": \"192.168.1.1\",
    \"clientes\": {
        \"cliente_ip\": \"1.1.1.1\",
        \"download\": \"102400 Kbps\",
        \"upload\": \"3072 Kbps\",
        \"estado\": \"activo\"
    }
}"
```

Example response (200, success):

```
{
    "message": "BABEL : ¡Cola/s actualizada/s con éxito!"
}
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or missing params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

```
{
    "status": "error",
    "message": "Error getting customer connection."
}
```

Mikrotik Controller

API for managing Mikrotik

Wireless

To manage queues connections

Test RouterOS

This endpoint allows you to test the Mikrotik server connection with this API. It's a really useful endpoint and a start point to know if there is any problem.

```
{
    "message": "BABEL : ¡Cola/s eliminada/s con éxito!"
}
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or missing params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

Create Contracts

This endpoint allows you to create a 'contract or contracts' in a Mikrotik server.

ℹ With this you are able to create "contracts" (queues with address-list).

Request Try it out ⚡

POST `v1.1/contracts`

Headers

`Content-Type`
Example: `application/xml`

`Accept`
Example: `application/json`

Body Parameters

`ip` string
The server ip. Example: `192.168.1.1`

▶ **`clientes`** object
A list of objects that contains the clients params.

Example request:

```
curl --request POST \
    "http://localhost/v1.1/contracts" \
    --header "Content-Type: application/xml" \
    --header "Accept: application/json" \
    --data "{
    \"ip\": \"192.168.1.1\",
    \"clientes\": {
        \"cliente_ip\": \"1.1.1.1\",
        \"download\": \"102400 Kbps\",
        \"upload\": \"3072 Kbps\",
        \"estado\": \"activo\"
    }
}"
```

Example response (200, success):

`ip` string
The server ip. Example: `192.168.1.1`

▶ **`clientes`** object
A list of objects that contains the clients params.

```
{
    }
}"
```

Example response (200, success):

```
{
    "message": "BABEL : ¡Cola/s creada/s con éxito!"
}
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or missing params):

This endpoint allows you to update the banwidth profile of a 'contract or contracts' or ip in a Mikrotik server.

ℹ With this you are able to update the "contracts" (queues with address-list).

Request Try it out ⚡

PUT `v1.1/contracts`

Headers

`Content-Type`
Example: `application/xml`

`Accept`
Example: `application/json`

Body Parameters

`ip` string
The server ip. Example: `192.168.1.1`

▶ **`clientes`** object
A list of objects that contains the clients params.

Example request:

```
curl --request PUT \
    "http://localhost/v1.1/contracts" \
    --header "Content-Type: application/xml" \
    --header "Accept: application/json" \
    --data "{
    \"ip\": \"192.168.1.1\",
    \"clientes\": {
        \"cliente_ip\": \"1.1.1.1\",
        \"download\": \"102400 Kbps\",
        \"upload\": \"3072 Kbps\",
        \"estado\": \"activo\"
    }
}"
```

Example response (200, success):

```
{
    "message": "BABEL : ¡Cola/s actualizada/s con éxito!"
}
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or missing params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

Enable Connections

Enable Connections

This endpoint gets the ip/s in the address-list "clientes_cortados", then it puts it/them back in "clientes_activos" address-list

ℹ With this you are able to allow the "contracts" to have a valid connection to the internet

Request Try it out ⚡

PATCH `v1/connection/enableConnection`

Headers

Example request:

```
curl --request PATCH \
    "http://localhost/v1/connection/enableConnection" \
```

Content-Type

Example: `application/xml`

Accept

Example: `application/json`

Body Parameters

ip string

The server ip. Example: `192.168.1.1`

▶ **clientes** object

A list of objects that contains the clients params.

```
    --header "Content-Type: application/xml" \
    --header "Accept: application/json" \
    --data "{
    \"ip\": \"192.168.1.1\",
    \"clientes\": {
        \"cliente_ip\": \"1.1.1.1\"
    }
}"
```

Example response (200, success):

```
,
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or missing params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

Disable Connections

Disable Connections

This endpoint gets the ip/s in the address-list "clientes_activos", then it puts them in the "clientes_cortados" address-list

ℹ With this you are able to restrict the "contracts" to have a valid connection to the internet 😈

Request Try it out ⚡

PATCH **v1/connection/disableConnection**

Headers

Content-Type

Example: `application/xml`

Accept

Example: `application/json`

Body Parameters

ip string

The server ip. Example: `192.168.1.1`

▶ **clientes** object

A list of objects that contains the clients params.

Example request:

```
curl --request PATCH \
    "http://localhost/v1/connection/disableConnection" \
    --header "Content-Type: application/xml" \
    --header "Accept: application/json" \
    --data "{
    \"ip\": \"192.168.1.1\",
    \"clientes\": {
        \"cliente_ip\": \"1.1.1.1\"
    }
}"
```

Example response (200, success):

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

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or missing params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

PATCH v1/connection/revertChanges

Request Try it out ⚡

Example request:

PATCH v1/connection/revertChanges

Request Try it out ⚡

PATCH `v1/connection/revertChanges`

Headers

`Content-Type`
Example: `application/json`

`Accept`
Example: `application/json`

```
curl --request PATCH \
    "http://localhost/v1/connection/revertChanges" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json"
```

Get Data Mikrotik

This endpoint gets info from the router (clients connected, active clients, clipped, etc)

ℹ VERY USEFUL endpoint

Request Try it out ⚡

Example: `application/json`

Body Parameters

`ip` string
The server ip. Example: `192.168.1.1`

▶ **`clientes`** object
A list of objects that contains the clients params.

```
    \"clientes\": {
        \"cliente_ip\": \"1.1.1.1\"
    }
}"
```

Example response (200, success):

```
{
    "name": "Mikrotik-Cristian",
    "server_ip": "192.168.2.184",
    "queues": {
        "total_queues": 1,
        "queues_details": [
            {
                "name": "1.1.1.1",
                "target": "1.1.1.1",
                "download": "102400 Kbps",
                "upload": "3072 Kbps"
            }
        ]
    },
    "address_list": {
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (500, no payload or missing params):

```
{
    "message": "BABEL: Undefined array key 'ip'"
}
```

Radius Controller

API for managing Radius Server

Get All Users

requires authentication

This endpoint allows you to get all users from the radius database

Request Try It out ⚡

GET v1/radius/users_data

Headers

Authorization
Example: `Bearer BEARER_TOKEN`

Content-Type
Example: `application/json`

Accept
Example: `application/json`

Example request:

```
curl --request GET \
    --get "http://localhost/v1/radius/users_data" \
    --header "Authorization: Bearer BEARER_TOKEN" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json"
```

Example response (200):

```
{
    "status": "success",
    "message": "Users found",
    "detail": {
        "id": 138421,
        "name": "Kwargs",
        "username": "1.1.1.1",
        "password": "e08fsa13",
        "bandwith_plan": "11440k/11440k",
        "main_ip": "1.1.1.1",
        "node": "ROSEDAL",
        "created_at": "2023-06-07",
        "updated_at": "2023-06-08"
    }
}
```

Example response (500):

```
{
    "status": "error",
    "message": "Error retrieving users",
    "detail": "SQLSTATE[HY000] [2002] No such file or directory"
}
```

Create User

requires authentication

This endpoint allows you to create a user in the radius database.

Request Try It out ⚡

POST v1/radius/users

Headers

Authorization
Example: `Bearer BEARER_TOKEN`

Content-Type
Example: `application/json`

Accept
Example: `application/json`

Body Parameters

name string
The name of the user. Example: `Kwargs`

username string
The username of the user. Example: `1.1.1.1`

bandwith_plan string
The bandwith plan of the user. Example: `11440k/11440k`

node string
The node of the user. Example: `ROSEDAL`

main_ip string
The main ip of the user. Example: `1.1.1.1`

Example request:

```
curl --request POST \
    "http://localhost/v1/radius/users" \
    --header "Authorization: Bearer BEARER_TOKEN" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json" \
    --data "{
    \"name\": \"Kwargs\",
    \"username\": \"1.1.1.1\",
    \"bandwith_plan\": \"11440k\\/11440k\",
    \"node\": \"ROSEDAL\",
    \"main_ip\": \"1.1.1.1\"
}"
```

Example response (201):

```
{
    "status": "success",
    "message": "User created successfully."
}
```

Example response (400):

```
{
    "status": "error",
    "message": "Error creating user",
    "detail": {
        "main_ip": [
            "The main ip must be a valid IP address."
```

GET v1/test

Headers

Content-Type
Example: `application/xml`

Accept
Example: `application/json`

Body Parameters

```
curl --request GET \
    --get "http://localhost/v1/test" \
    --header "Content-Type: application/xml" \
    --header "Accept: application/json" \
    --data "{
    \"ip\": \"192.168.1.1\"
}"
```

ip string
The server ip. Example: `192.168.1.1`

Example response (200, success):

```
{
    "name": "Mikrotik-Cristian"
}
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Get User

requires authentication

This endpoint allows you to get a user from the radius database based on the username in the request

Request Try it out ⚡

GET `v1/radius/users`

Headers

Authorization
Example: `Bearer BEARER_TOKEN`

Content-Type
Example: `application/json`

Accept
Example: `application/json`

Body Parameters

username string
The username of the user. Example: `1.1.1.1`

Example request:

```
curl --request GET \
    --get "http://localhost/v1/radius/users" \
    --header "Authorization: Bearer BEARER_TOKEN" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json" \
    --data "{
    \"username\": \"1.1.1.1\"
}"
```

Example response (200):

```
{
    "status": "success",
    "message": "User found",
    "detail": {
        "id": 138421,
        "name": "Kwargs",
        "username": "1.1.1.1",
        "password": "e08fsa13",
        "bandwith_plan": "11440k/11440k",
        "main_ip": "1.1.1.1",
        "node": "ROSEDAL",
        "created_at": "2023-06-07",
        "updated_at": "2023-06-08"
    }
}
```

Example response (400):

```
{
    "status": "error",
    "message": "Error getting user",
    "detail": {
        "username": [
            "The username field is required."
        ]
    }
}
```

Example response (404):

```
{
    "status": "error",
    "message": "User does not exist."
}
```

Example response (500):

```
{
    "status": "error",
    "message": "Error retrieving users",
    "detail": "SQLSTATE[HY000] [2002] No such file or directory"
```

```
}
```

Update User

requires authentication

This endpoint allows you to update a user in the radius database.

Request Try it out ⚡

PUT v1/radius/users

Accept
Example: `application/json`

Body Parameters

ip string
The server ip. Example: `192.168.1.1`

▶ **clientes** object
A list of objects that contains the clients params.

Example request:

```
    \"ip\": \"192.168.1.1\",
    \"clientes\": {
        \"cliente_ip\": \"1.1.1.1\",
        \"download\": \"102400 Kbps\",
        \"upload\": \"3072 Kbps\",
        \"estado\": \"activo\"
    }
}"
```

Example response (200, success):

```
{
    "message": "BABEL : ¡Cola/s eliminada/s con éxito!"
}
```

Example response (500, no route to host):

```
{
    "message": "BABEL: Unable to establish socket session, No route to host"
}
```

Example response (400):

```
{
    "status": "error",
    "message": "Error updating user",
    "detail": {
        "main_ip": [
            "The main ip must be a valid IP address."
        ]
    }
}
```

Example response (500):

```
{
    "status": "error",
    "message": "An error occurred while updating the user."
}
```

Delete User

requires authentication

This endpoint allows you to delete a user in the radius database.

Request Try it out ⚡

DELETE **v1/radius/users**

Headers

Authorization
Example: `Bearer BEARER_TOKEN`

Content-Type
Example: `application/json`

Accept
Example: `application/json`

Body Parameters

username string

Example request:

```
curl --request DELETE \
    "http://localhost/v1/radius/users" \
    --header "Authorization: Bearer BEARER_TOKEN" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json" \
    --data "{
    \"username\": \"1.1.1.1\"
}"
```

Example response (200):

The username of the user. Example: `1.1.1.1`

Example response (400):

```
{
    "status": "success",
    "message": "User deleted successfully"
}
```

Example response (404):

```
{
    "status": "error",
    "message": "Error deleting user",
    "detail": {
        "username": [
            "The username field is required."
        ]
    }
}
```

Example response (500):

```
{
    "status": "error",
    "message": "User does not exist."
}
```

SmartOLT Controller

API for managing SmartOLT

Get ONU Status

requires authentication

This endpoint allows you to get the status of an ONU.

Request Try it out ⚡

GET `v1/smartolt/onu/status`

Headers

`Authorization`
Example: `Bearer BEARER_TOKEN`

`Content-Type`
Example: `application/json`

`Accept`
Example: `application/json`

Body Parameters

`onu_sn` string
The ONU serial number. Example: `HWTCA7A1236A`

Example request:

```
curl --request GET \
    --get "http://localhost/v1/smartolt/onu/status" \
    --header "Authorization: Bearer BEARER_TOKEN" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json" \
    --data "{
    \"onu_sn\": \"HWTCA7A1236A\"
}"
```

Example response (200):

```
{
    "status": "success",
    "message": "ONU Status retrieved successfully.",
    "detail": {
        "status": true,
        "onu_status": "Online"
    }
}
```

Example response (400):

Example response (404):

Example response (500):

```
{
    "status": "error",
    "message": "Error deleting user",
    "detail": "SQLSTATE[HY000] [2002] No such file or directory"
}
```

```
{
    "status": "error",
    "message": "User does not exist."
}
```


Example response (500):


```
{
    "status": "error",
    "message": "Error deleting user",
    "detail": "SQLSTATE[HY000] [2002] No such file or directory"
}
```


SmartOLT Controller

API for managing SmartOLT


```
{
    "status": "error",
    "message": "An error occurred while retrieving ONU Status."
}
```


Reboot ONU

requires authentication

This endpoint allows you to reboot an ONU by its serial number.

Request Try it out ⚡

POST `v1/smartolt/onu/reboot`

Headers

`Authorization`
Example: `Bearer BEARER_TOKEN`

`Content-Type`
Example: `application/json`

`Accept`
Example: `application/json`

Body Parameters

`onu_sn` string
The ONU serial number. Example: `HWTCA7A1236A`

Example request:

```
curl --request POST \
    "http://localhost/v1/smartolt/onu/reboot" \
    --header "Authorization: Bearer BEARER_TOKEN" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json" \
    --data "{
    \"onu_sn\": \"HWTCA7A1236A\"
}"
```

Example response (200):


```
{
    "status": "success",
    "message": "ONU rebooted successfully.",
    "detail": {
        "status": true,
        "response": "Device reboot command sent"
    }
}
```


Example response (400):


```
{
    "status": "error",
    "message": "Invalid data.",
    "detail": {
        "onu_sn": [
            "The onu sn field is required."
        ]
    }
}
```


Example response (404):


```
{
    "status": "error",
    "message": "ONU Status not found."
}
```

Example response (500):


```
{
    "status": "error",
    "message": "An error occurred while retrieving ONU Status."
}
```