API Guide

Shabodi AEP
Release 2.0

Abstract

The AEP API guide is targeted at developers of network-aware applications using network APIs to improve their application's experience and/or control of network resources. The APIs referenced in this document are Shabodi's Simplified APIs that are available in Release 2.0 of the AEP. The APIs are available through on-premise installations in either enterprise or carrier infrastructures, or available for testing in Shabodi's AEP Developer Sandbox.

For more information about Shabodi's AEP, APIs, Use Cases and Sandbox visit https://www.shabodi.com/.

Shabodi API Guide

Version 1.0 - June 2024
Contact:
https://www.shabodi.com/

Contents

1 Introduction to Shabodi 3

2 Shabodi's Network-Aware Solutions 5

3 Shabodi's AEP: making network-aware applications 6

3.1 AEP core functions 7

4 API Overview 8

4.1 Unique capabilities: 8
4.2 Benefits of AEP 8
4.3 Pioneering Network-Driven Innovation 8
4.4 Simplified APIs delivering network services 9
4.5 Features 9
4.6 Applications 9
4.7 Relevant Terms and Definitions 9
4.8 Pre-Requisites 10
4.8.1 Networks 10
4.8.2 Devices 10

5 Using AEP APIs 11

5.1 Authentication 11
5.2 Headers 11
5.3 Defaults 11
5.4 Event Subscriptions 11
5.5 Sandbox 11

6 API Reference 12

6.1 APPLICATION REGISTRATION 12
6.1.1 Pre-onboard an application 12
6.1.2 Onboard an application 12
6.2 TOKEN GENERATION 14
6.3 QUALITY OF SERVICE 14
6.3.1 Bandwidth 14
6.3.2 Latency 16
6.3.3 Jitter 17
6.4 LOCATION 18
6.5 INSIGHTS 18

Version Control

Version	Release	Description	Modified on	Modified by
v1	2.0	Initial draft	June 2024	S Gole, K Howe-Patterson

1 Introduction to Shabodi

Shabodi's **VISION** is to enable an ecosystem of network-aware solutions using advanced network services in multi-access, multi-network and multi-operator environments.

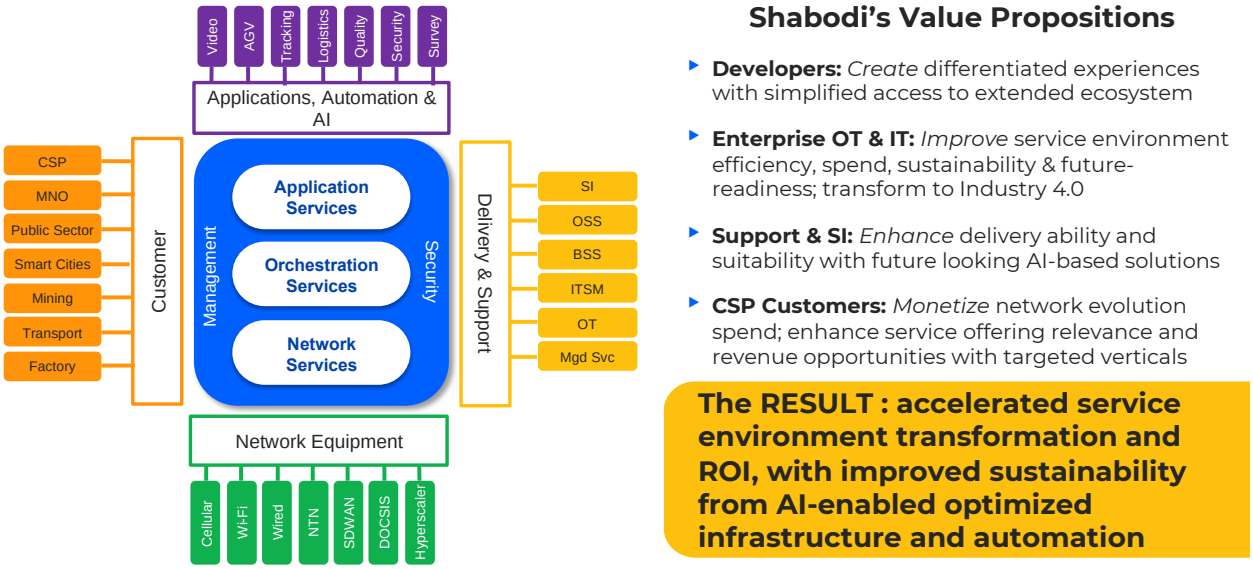


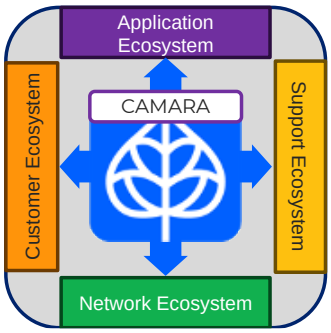
Figure 1 Shabodi's mission: unleash the power of advanced networks

Shabodi is a technology company creating products that solve some of the world's most challenging problems, primarily in the advanced networking domain. However, Shabodi's solutions often span business process, network design and implementation, system integration, technology advisory services and technical consulting. Shabodi relies upon a broad ecosystem of network equipment providers, support and integration partners, and application developers to deliver leading edge solutions to our ecosystem of customers. Shabodi is the nexus that brings these diverse ecosystems of diverse contributors together to deliver advanced network solutions.

Shabodi's **MISSION** is to unleash the power of advanced networks.

Advanced networks exist everywhere. The emergence of commercial offerings of 5G cellular networks by mobile network operators are likely the most visible. However, Wi-Fi, NTN, DOCSIS, Fibre, and SD-WAN are all forms of advanced networks. They deploy in many configurations across numerous networking domains – MNO, MVNO, CSP, MSP, Enterprise amongst the many forms. A wide array of network equipment providers play in this space as well, each with their own commercial and technical driver. The end result is a wide-opn complex network service industry with multiple players all vying to expand their business. Shabodi's solutions smooth out the bumps, making it easier for applications to thrive in this advanced networking landscape.

Shabodi is the first to deliver Enterprise Ready CAMARA API support in its Network-Aware AEP. This provides seamless portability of network-aware applications built using CAMARA API for MNO 5G networks onto enterprise infrastructures. Shabodi is an active, disruptive, innovative "first mover" in the Network-Aware AEP space.



CAMARA is a gateway to the broader ecosystem enabling:

- Wholesale services
- Differentiated experiences
- Network optimization
- Network integration
- Service disintermediation
- Service and technology bridging and migration

Figure 2 Shabodi's technology leadership and influence

While Shabodi is an independent company (that is we do not belong to nor work for MNOs/carriers, enterprises or other affiliations) we do work with a large number of standards organizations and have numerous partners that help us deliver our solutions. A significant alliance with goals aligned with Shabodi's is the CAMARA Alliance. It's primary focus is to bring applications to emerging 5G wireless network infrastructures to help that technology be successful. CAMARA's focus is MNO infrastructures.

2 SHABODI'S NETWORK-AWARE SOLUTIONS

Shabodi has created a technology stack designed to deliver advanced network-aware digital solutions. This is displayed in Figure 3 Shabodi's network-aware service technology stack.

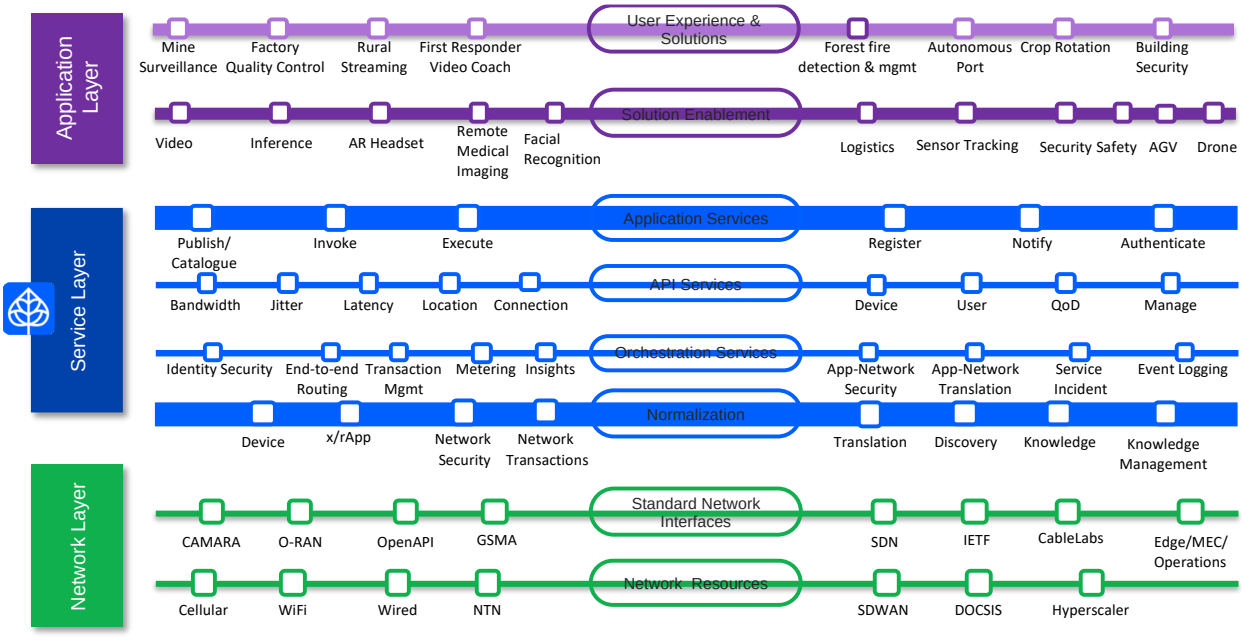


Figure 3 Shabodi's network-aware service technology stack

3 SHABODI'S AEP: MAKING NETWORK-AWARE APPLICATIONS

Shabodi's primary product is the network-aware application enablement platform (AEP). It is an API-first product. It's "northbound" APIs connect to applications in a very simplified manner: Shabodi's Simplified APIs abstract and expose the underlying network, providing the application developer with a very quick and easy method of developing applications that access the power.

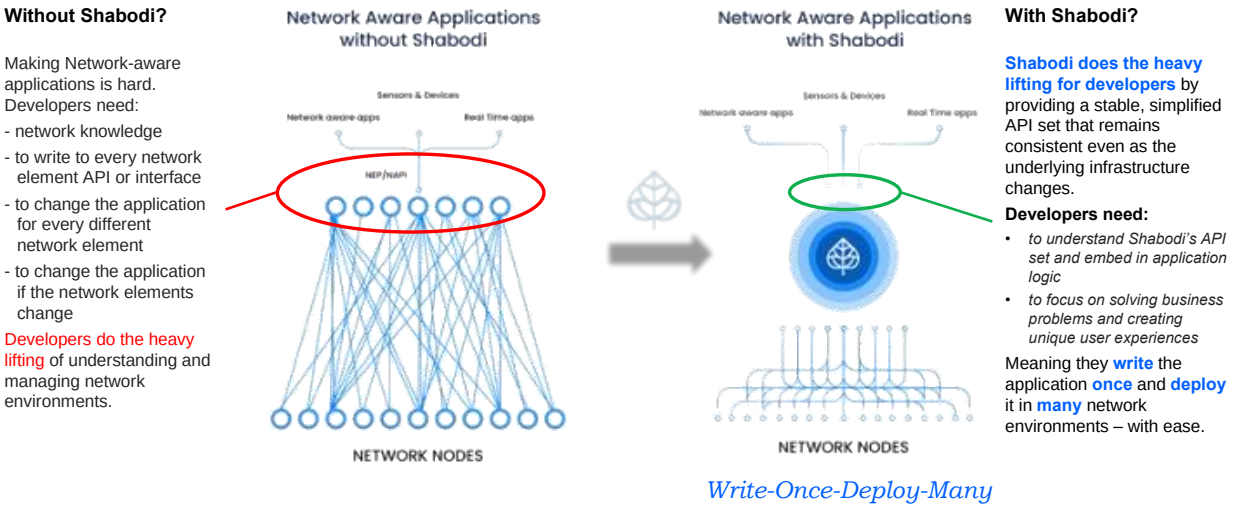


Figure 4 Simplifying Network-Awareness

Shabodi's AEP employs RESTful APIs with standardized formats designed with developers in mind. While they are simplified, that is, having a very simple and familiar format for developers to integrate into their application logic, they are very powerful in terms of their outcomes. A single API call into the AEP can replace up to 20 distinct calls to network elements and can hide from the application aspects such as network vendor, network congestion and transaction processes, network element authorization and authentication, and so on. If time is the most precious asset a developer has, Shabodi's AEP gives time back to developers so they can focus on solving business and/or operations problems delivering superior user experiences. All this without having to spend months (or years) becoming networking experts.

On the south side Shabodi discovers the network elements involved, creating transformations from application-level APIs to the language, security and data model of the network elements. The benefit here is that the network never has to know what application is controlling it, and therefore requires no special adaptations or behaviors to accommodate. The network element simply needs to do what it does best: transmit data from one connected device to another destination.

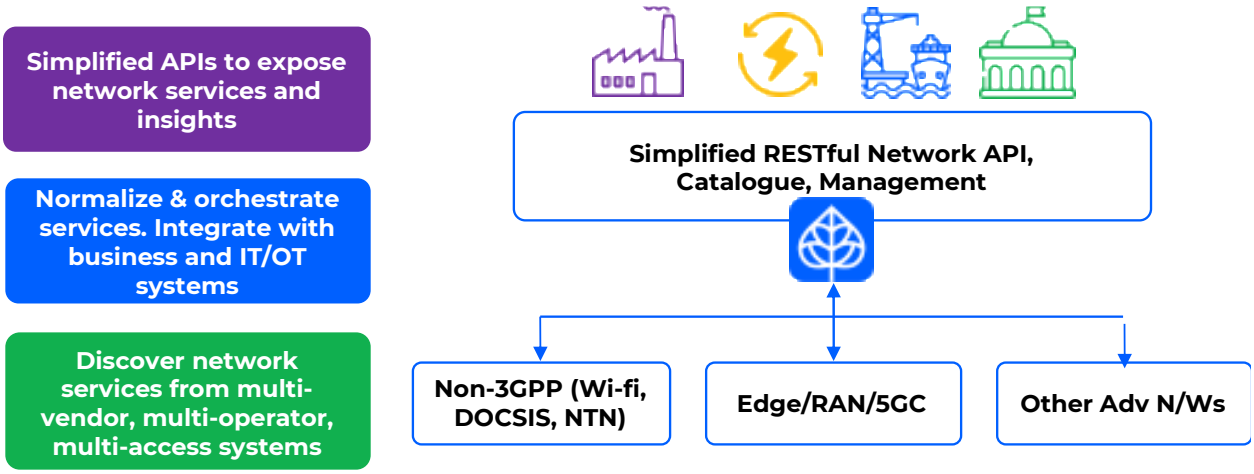
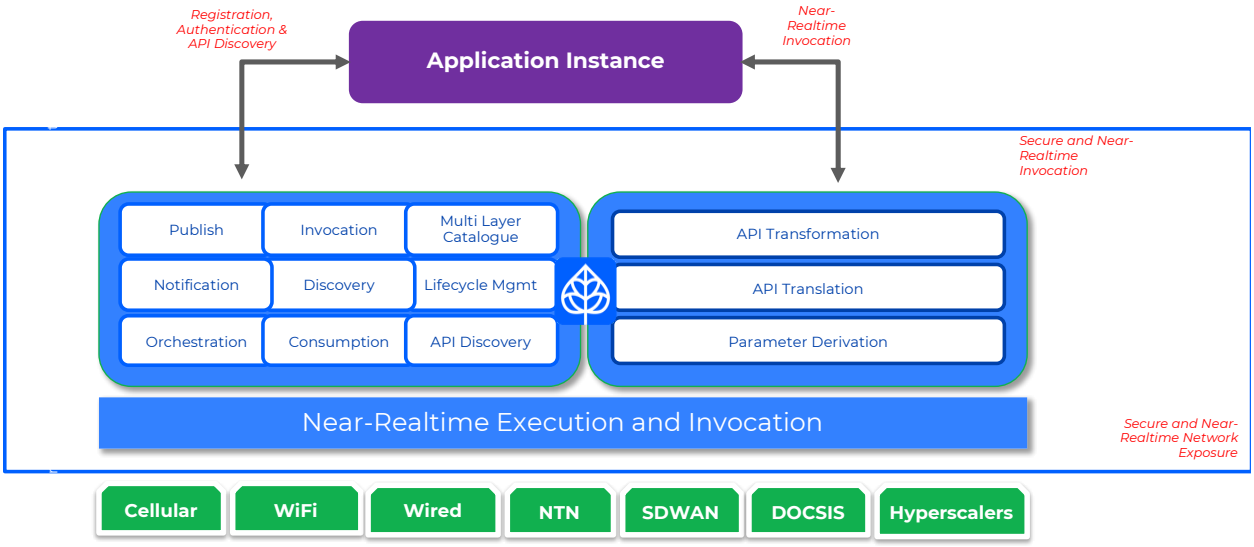


Figure 5 Multi-Network API Transactions

3.1 AEP CORE FUNCTIONS

Shabodi's AEP eliminates that complexity for both the northside applications, automations and AI through its transformation and orchestration services, performing all tasks to deliver network control plane requests in realtime. AEP also translates the transaction to adapt to network element vendors – each vendor will have its own APIs with their own language, authentication, data model and transaction processing model.

Another primary function of the AEP is to act as a security bridge between the application world and the network world.



4 API Overview

Shabodi's AEP bridges the gap between applications and the network world, making it suitable for a wide range of users across various industries. AEP allows applications to utilize network services through simplified RESTful APIs. AEP translates complex network functions into easy-to-use commands for your apps.

Shabodi's AEP goes beyond simply enabling network interaction for your applications. It acts as an intelligent bridge, providing a wealth of insights that traditional network management systems (NMS, OSS/BSS) and even the network itself may not capture.

4.1 Unique capabilities:

- **Application-Device Mapping**: Track which devices are running specific application instances, enabling better resource optimization and service delivery.
- **Application QoS Requests**: Understand the QoS (Quality of Service) requirements requested by your applications, providing valuable information for network planning and prioritization.
- **Request Fulfillment Tracking**: Gain insights into which network elements are fulfilling application requests, allowing for better performance monitoring, and troubleshooting.
- **Reason for QoS Request Failures**: Identify the root cause of application QoS request failures, whether due to network congestion, insufficient provisioning, or edge-level issues. This empowers you to take corrective actions and ensure application performance.

4.2 Benefits of AEP

- **Effortless App Development**: Focus on building innovative applications, not low-level network programming.
- **Unlock Network Potential**: Leverage advanced features like dynamic bandwidth allocation and device control.
- **New Revenue Streams**: Develop groundbreaking network-based services that create value for your users.
- **Unparalleled Network Insights**: Gain a deeper understanding of your network beyond what traditional management systems can provide.

4.3 Pioneering Network-Driven Innovation

Shabodi's AEP empowers developers to build network-aware applications across various industries [https://www.shabodi.com/solutions/aep-enterprise/]:

- **IoT Applications**: Shabodi's AEP empowers IoT devices by making them network-aware. Whether it's smart home gadgets, industrial sensors, or wearables, AEP simplifies integration with the network.
- **Telecommunications Apps**: Telecom operators benefit from AEP for call routing optimization, efficient data traffic management, and improved connectivity.
- **Edge Computing Solutions**: AEP plays a vital role in edge applications like autonomous vehicles, smart cities, and remote monitoring.
- **Healthcare and Wearables**: AEP ensures seamless communication between healthcare wearables, medical devices, and providers.
- **Retail and Supply Chain**: Real-time inventory updates, logistics optimization, and supply chain visibility are enhanced by AEP.
- **Gaming and Entertainment**: AEP improves online gaming, streaming services, and interactive entertainment experiences.

- **Smart Cities and Infrastructure**: AEP contributes to building smarter cities. Applications related to traffic management, energy efficiency, waste management, and public safety can utilize its network-aware capabilities. For instance, real-time traffic flow optimization or smart street lighting systems.

4.4 Simplified APIs delivering network services

Shabodi's Network Aware Application-Enabled Platform (AEP) enables applications to become network aware by providing access to previously inaccessible network functionalities through user-friendly APIs. These APIs are categorized into two main groups:

- **Administrative APIs:** provide the ability to register applications with APIs and to drive setup, configuration and maintenance of the AEP.
- **Service APIs:** harness the intelligence of AEP's orchestration capabilities and near-real-time engine to dynamically shape the network and obtain valuable insights into its operation.

This document serves as a comprehensive reference guide, offering visibility into both the administrative and service APIs provided by AEP. It provides detailed information regarding the features accessible to applications and describes the parameters necessary to invoke them effectively from your application.

4.5 Features

In AEP release 2.0, simplified service APIs provide applications with the ability to:

- Configure **quality of service** for a device:
  - Bandwidth:
    - Set uplink and downlink maximum bit rate
    - Set uplink and downlink guaranteed bit rate
    - Set peer-to-peer guaranteed bit rate between two devices in an SD-WAN
  - Latency
  - Jitter
- Retrieve the center point **location** of a device
- Retrieve device **insights** such as device status and event notifications:
  - In this version, the only supported event type is **'connectivity'**, which refers to the network connectivity status of a device.

4.6 Applications

A network-aware application leverages Shabodi's APIs to gain insights and control the network for connected devices it is managing. Application functions are designed to meet user requirements and leverage the underlying infrastructure's capabilities.

Provisioning: Use **Application Registration** APIs below to bind a new application to your AEP. Note that application registration is a two-step process: pre-onboarding and onboarding.

4.7 Relevant Terms and Definitions

Term	Definition
AEP	Shabodi's Network Aware Application Enablement Platform.
SNBP	Shabodi Northbound Platform responsible for registration, authentication, management, and API discovery.
Invoker	Application function invoking APIs to API providers.
Provider	Services or applications publishing capabilities to application functions.

Term	Definition
QoS	Quality of Service – Programmability of the network to dynamically adapt and allocate resources.
Insights	Provide a window into the networks managed by AEP, enabling applications and devices to adapt dynamically.
Admin APIs	Set of APIs to manage applications, registration, and services offered by SNBP.
Service APIs	Set of APIs to expose network functions to applications, including QoS, Location, and Insights.
MBR	Maximum Bit Rate defines the maximum bit rate being allocated to a device or group of devices.
GBR	Guaranteed Bit Rate defines the minimum bit rate allocation to a device or group of devices.
Downlink	Link direction from network towards device.
Uplink	Link direction from device towards network.
Subscription	Ability to subscribe to event notifications from AEP based on specific topics.
Sandbox	Shabodi-hosted AEP used for API testing, proof of concepts, and early integration of new APIs.

4.8 Pre-Requisites

4.8.1 Networks

A network refers to interconnected systems that facilitate communication and data exchange, utilizing advanced infrastructure such as routers, controllers, and radio access equipment. Shabodi's AEP accommodates various network types, including 4G, 5G, WiFi, and SD-WAN.

Configuring network information on AEP: Essential to ensure that applications and devices operate seamlessly in alignment with the services provided by the AEP. Refer to the **Networks** section of the **User Administration Guide** for details.

4.8.2 Devices

A device refers to any hardware component that can connect to the network, typically using protocols such as Wi-Fi or cellular. Devices include cameras, drones, AGVs, vehicles, smart protective equipment, and more.

Configuring devices on AEP: Devices need to be configured on the AEP to be managed by applications. Refer to the **Devices** section of the **User Administration Guide** on how to add and map devices to your applications.

5 USING AEP APIS

5.1 AUTHENTICATION

The AEP supports OAuth2.0, the industry standard protocol for authentication. OAuth2.0 uses authentication bearer tokens to make requests to the resource server.

5.2 HEADERS

The following headers are required to invoke JSON-based requests into AEP:

- Content-Type: application/json
- Authorization: Bearer <access_token>

5.3 DEFAULTS

For consistency:

- Bit rates are in kbps.
- Time and duration are in ms.
- Date-time format follows RFC 3339, section 5.6. For example, 2023-02-08T18:04:28Z.

5.4 EVENT SUBSCRIPTIONS

AEP offers the ability for applications to subscribe to insightful network events, supporting two event types:

- connectivity: Notifies the application of a device connectivity state change.
- deviceUtilization: Triggered based on active GBR sessions. Utilization is measured as (current device utilization)/(guaranteed bit rate).

5.5 SANDBOX

Shabodi offers developers a playground to test and integrate their applications with AEP's APIs. Contact developer.support@shabodi.com to request access.

6 API REFERENCE

6.1 APPLICATION REGISTRATION

Granting of permissions is required by AEP for applications to access its services and features. There are two steps in this process, pre-onboarding, and onboarding of the application.

6.1.1 Pre-onboard an application

6.1.1.1 POST

/api-invoker-management/v1/preOnboardedInvokers

6.1.1.1.1 Request:

```
{
  "name": "string"
}
```

6.1.1.1.2 Response:

- 201: Provides a Bearer token required for application onboarding.


```
{
  "name": "Drone Application",
  "ott": {
    "access_token": "eyJraWQiOiIxZGM5OTVhYi1iYWQxLTQ3ODgtODMxZC01MTA4ZTk2OGY2NTEiLCJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJTaGFib2RpIExBQVMiLCJhdWQiOiJTaGFib2RpIEFFUCIsInN1YiI6IjE3ZDAxYWExLThkY2QtNGQ3Yi05Y2FjLWZjNDMwNDY5OTQ5ZiIsInByZWZlcnJlZF91c2VybmFtZSI6IlNOQlAgREVNTyBBUFBMSUNBVElPTjYiLCJzY29wZSI6Im5iLmFwcC5jliwiaWF0IjoxNzA5OTE4Nzg5LCJleHAiOjE3MTA1MjM1ODksImp0aSI6IjI2NDFiMWYwLTY1NzEtNDQ0Ny1iYjRkLWZlM2VhMjM5ZWVkOCJ9.LwSrSxJ_HsPn1msUGDzLiEOHMDM4Qia0hfnV5-GhuwE",
    "expires_in": 604800,
    "scope": "nb.app.c",
    "token_type": "bearer"
  }
}
```


6.1.2 Onboard an application

6.1.2.1 POST

6.1.2.1.1 Request:

/api-invoker-management/v1/onboardedInvokers

6.1.2.1.2 Request Body:

```
{
  "notificationDestination": "https://a43f8337-033e-4b45-9afe-df121b62f95c.mock.pstmn.io/notifications",
  "requestTestNotification": true,
  "direction": "DL",
  "traffic": "TCP",
  "ipv4": "172.56.105.26",
  "description": "Drone controller application for QoD",
  "port": 5559
}
```

6.1.2.1.3 Response:

- 201: Provides information regarding the newly onboarded application.


```
{
  "apiInvokerId": "17d01aa1-8dcd-4d7b-9cac-fc430469949f",
  "onboardingInformation": {
    "onboardingSecret": "eok9rStqticMq8yQiueKRYvB2raBoC13"
  },
  "notificationDestination": "https://a43f8337-033e-4b45-9afe-df121b62f95c.mock.pstmn.io/notifications",
  "requestTestNotification": true,
```

```
  "ipv4": "172.56.105.26",
  "port": 5559,
  "direction": "DL",
  "traffic": "TCP"
}
```


6.1.2.2 GET

Retrieve information for all onboarded applications

6.1.2.2.1 Request:

```
/api-invoker-management/v1/onboardedInvokers
```

6.1.2.2.2 Response:

- 200: Provides an array of onboarded applications.


```
[
  {
    "apiInvokerId": "17d01aa1-8dcd-4d7b-9cac-fc430469949f",
    "onboardingInformation": {
      "onboardingSecret": "eok9rStqticMq8yQiueKRYvB2raBoC13"
    },
    "notificationDestination": "https://a43f8337-033e-4b45-9afe-df121b62f95c.mock.pstmn.io/notifications",
    "requestTestNotification": true,
    "description": "This is a demo application",
    "name": "Drone Application",
    "ipv4": "172.56.105.26",
    "port": 5559,
    "direction": "DL",
    "traffic": "TCP"
  }
]
```


6.1.2.3 PUT

Update an onboarded application

6.1.2.3.1 Request:

```
/api-invoker-management/v1/onboardedInvokers/{appId}
```

6.1.2.3.2 Parameters

- appId (required)

6.1.2.3.3 Request Body:


```
{
  "notificationDestination": "http://example.com/invoker/notification",
  "description": "This is a demo application"
}
```


6.1.2.3.4 Response:

- 200:


```
{
  "apiInvokerId": "17d01aa1-8dcd-4d7b-9cac-fc430469949f",
  "notificationDestination": "http://example.com/invoker/notification",
  "requestTestNotification": true,
  "description": "This is a demo application"
}
```


6.1.2.4 DELETE

Delete an onboarded application

6.1.2.4.1 Request:

`/api-invoker-management/v1/onboardedInvokers/{appId}`

6.1.2.4.2 Parameters

- appId (required)

6.1.2.4.3 Response:

- 200: Application removed

6.2 TOKEN GENERATION

Acquiring an OAuth2 token for invoking service APIs

6.2.1.1 POST

Acquire OAuth2 token

6.2.1.1.1 Request:

`/security/v1/token`

6.2.1.1.2 Request Body:

```
{
  "client_id": "93f2c879-c3f9-4f51-9d74-2c84cc8081dd",
  "client_secret": "Ayd9zF8tGjcb63mK54uPoJZL0T9olY5n"
}
```

6.2.1.1.3 Response:

- 200 Access token acquired


```
{
  "access_token": "eyJraWQiOiI4ZjYwYmRiMy1mYTNjLTRjZTAtYWZhOS04ZWY3MjA4ZTE0NzliLCJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJTaGFib2RpIExBQVMiLCJhdWQiOiJTaGFib2RpIEFFUCIsInN1YiI6IjkzZjJjODc5LWMzZjktNGY5MS05ZDc0LTJjODRjYzgwODFkZCIsInByZWZlcnJlZF91c2VybmFtZSI6IlNOQlAgREVNTyBBUFBMSUNBVElPTiAtIEplZmYiLCJwb3J0Ijo1NTU5LCJpcHY0IjoiMTcyLjU2LjEwNS4yNiIsImRpcmVjdGlvbiI6IkRMIiwidHJhZmZpYyI6IlRDUCIsImRldmVkZ2UiOnsiaWQiOiJ0bW9uLTItSDZoN2tyTzdxSmU3NIlZOEJOZ082cGZDNmJ0TEZSIiwic2VjcmV0IjoiM0tBOUJwaUdHSUlVWGhFTSJ9LCJ1ZXMiOlsxXSwic2NvcGUiOiJhcGk6c2V0QlcgYXBpOlBjZkF1dGhvcml6YXRpb25Ob3JtRHJ1aWROQiBhcGk6UGNmQXV0aG9yaXphdGlvbk5vcm1PcGVuNUdTTkIgYXBpOlBjZkF1dGhvcml6YXRpb25EaXNjRHJ1aWROQiBhcGk6UGNmQXV0aG9yaXphdGlvbkRpc2NPcGVuNUdTTkliLCJpYXQiOjE3MTE2NDYxMzEsImV4cCI6MTcxMTY0OTczMSwianRpIjoiMzllZjMzMjktMzdjNi00Y2M1LTgzMzMtMmFiNTE1YmEyNDBiIn0.lqPmABVWcYp153Q99j1n9TADVGszl9BpFAGyKwqyzpg",
  "expires_in": 3600,
  "scope": "string",
  "token_type": "Bearer"
}
```


6.3 QUALITY OF SERVICE

Operations to configure quality of service parameters for a device

6.3.1 Bandwidth

6.3.1.1 POST

Set the uplink or downlink maximum bitrate for a device.

6.3.1.1.1 Request:

`/qos/v1/bandwidth`

6.3.1.1.2 Example Request Body:

```
{
  "device": {
    "deviceId": 3
  },
  "maxBitRate": 4096,
  "guaranteedBitRate": 4096,
  "direction": "uplink",
  "duration": 300000
}
```

6.3.1.1.3 Response:

- 201 Provides transactionId, remaining time of the bandwidth session.

```
{
  "transactionId": "43571235",
  "remainingTime": 60000
}
```

6.3.1.2 GET:

Retrieve active bandwidth sessions.

6.3.1.2.1 Request:

```
qos/v1/bandwidth
```

6.3.1.2.2 Parameters

- **deviceId** (required)

6.3.1.2.3 Example Request Body:

```
GET /qos/v1/bandwidth?deviceId=12345
```

6.3.1.2.4 Response:

- **200** Returns an array of bandwidth sessions unless deviceId is provided as a query parameter.

```
{
  "sessions": [
    {
      "device": {
        "deviceId": 3
      },
      "peerDevice": {
        "deviceId": 4
      },
      "maxBitRate": 4096,
      "guaranteedBitRate": 4096,
      "direction": "uplink",
      "remainingTime": 60000
    }
  ]
}
```

6.3.1.3 DELETE

End an active bandwidth session for a device.

6.3.1.3.1 Request:

```
/qos/v1/bandwidth/{deviceId}
```

6.3.1.3.2 Parameters

- **deviceId** (required)

6.3.1.3.3 Example Request

```
DELETE /qos/v1/bandwidth/12345
```

6.3.1.3.4 Response:

- 200 Successful operation.

6.3.2 Latency

6.3.2.1 POST

Set latency for a device.

6.3.2.1.1 Request:

```
/qos/v1/latency
```

6.3.2.1.2 Example Request Body:

```
{
  "device": {
    "deviceId": 3
  },
  "latency": 120,
  "duration": 300000
}
```

6.3.2.1.3 Response

- 201 Provides transactionId, start, and expiry time of the latency session.

```
{
"transactionId": "43571235",
"remainingTime": 60000
}
```

6.3.2.2 GET

Retrieve active latency sessions.

6.3.2.2.1 Request:

```
/qos/v1/latency
```

6.3.2.2.2 Parameters

- **deviceId** (required)

6.3.2.2.3 Example Request Body:

```
GET /qos/v1/latency?deviceId=12345
```

6.3.2.2.4 Responses:

- 200 OK

```
{
  "sessions": [
    {
      "device": {
        "deviceId": 3
      },
      "latency": 120,
      "remainingTime": 60000
    }
  ]
}
```

6.3.2.3 *DELETE*

End an active latency session for a device.

6.3.2.3.1 Request:

```
/qos/v1/latency/{deviceId}
```

6.3.2.3.2 Parameters

- **deviceId** (required)

6.3.2.3.3 Example Request Body:

```
DELETE /qos/v1/latency/12345
```

6.3.2.3.4 Response:

- 200 Successful operation

6.3.3 Jitter

6.3.3.1 *POST*

Set jitter for a device.

6.3.3.1.1 Request:

```
/qos/v1/jitter
```

6.3.3.1.2 Example Request Body:

```
{
  "device": {
    "deviceId": 3
  },
  "jitter": 40,
  "duration": 300000
}
```

6.3.3.1.3 Responses:

- 201 Created

```
{
  "transactionId": "43571235",
  "remainingTime": 60000
}
```

6.3.3.2 *GET*

Retrieve active jitter sessions.

6.3.3.2.1 Request:

```
/qos/v1/jitter
```

6.3.3.2.2 Parameters

- **deviceId** (required)

6.3.3.2.3 Example Request Body:

```
GET /qos/v1/jitter?deviceId=12345
```

6.3.3.2.4 Response:

- 200 OK

```
{
  "sessions": [
    {
      "device": {
        "deviceId": 3
      },
```

```
    "jitter": 40,
    "remainingTime": 60000
   }
  ]
}
}
```


6.3.3.3 DELETE

End an active jitter session for a device.

6.3.3.3.1 Request:

```
/qos/v1/jitter/{deviceId}
```

6.3.3.3.2 Parameters:

- **deviceId** (required)

6.3.3.3.3 Example Request Body:

```
DELETE /qos/v1/jitter/12345
```

6.3.3.3.4 Response:

- 200 Successful operation.

6.4 LOCATION

Obtain device location with a center point.

6.4.1.1 GET

6.4.1.1.1 Request:

```
/location/v1/center
```

6.4.1.1.2 Parameters

- **deviceId** (required)

6.4.1.1.3 Example Request Body:

```
GET /location/v1/center?deviceId=12345
```

6.4.1.1.4 Response:

- 200 OK


```
{
 "location": [
  {
   "device": {
    "deviceId": 3
   },
   "latitude": "-75.75619047340287",
   "longitude": "45.3189923087387",
   "altitude": 3.56
  }
 ]
}
```


6.5 INSIGHTS

Receive insights with event-based subscriptions.

6.5.1.1 POST

Create an event-based subscription for notifications.

6.5.1.1.1 Request:

```
/insights/v1/subscriptions
```

6.5.1.1.2 Example Request Body:

```
{
  "topic": "connectivity",
  "minThreshold": 30,
  "maxThreshold": 90,
  "filters": [
    [
      "net2.*",
      "net3.device5",
      "app"
    ]
  ],
  "notification": {
    "destination": "https://application-server.com",
    "token": "---------- token ----------"
  },
  "expireTime": "2024-04-13T09:12:28Z"
}
```

6.5.1.1.3 Response:

- 201

```
{
  "transactionId": "43571235",
  "createdTime": "2024-03-13T09:12:28Z",
  "expiredTime": "2024-04-13T09:12:28Z"
}
}
```

6.5.1.2 GET

Retrieve all active subscriptions.

6.5.1.2.1 Request:

```
GET /insights/v1/subscriptions
```

6.5.1.2.2 Response:

- 200

```
[
  {
    "subscriptionDetails": {
      "subscriptionId": "25487591",
      "topic": "connectivity",
      "minThreshold": 30,
      "maxThreshold": 30,
      "filters": [
        [
          "net2.*",
          "net3.device5",
          "app"
        ]
      ],
      "notification": {
        "destination": "https://application-server.com",
        "token": "---------- token ----------"
      },
      "createdTime": "2024-03-13T09:12:28Z",
      "expiredTime": "2024-04-13T09:12:28Z"
    }
  }
]
```

6.5.1.3 GET

Retrieve details of an active subscription.

6.5.1.3.1 Request:

```
/insights/v1/subscriptions/{subscriptionId}
```

6.5.1.3.2 Parameters:

- subscriptionId (required)

6.5.1.3.3 Example Request Body:

```
GET /insights/v1/subscriptions/25487591
```

6.5.1.3.4 Response:

- 200

```
{
"subscriptionId": "25487591",
 "topic": "connectivity",
 "minThreshold": 30,
 "maxThreshold": 30,
 "filters": [
  [
   "net2.*",
   "net3.device5",
   "app"
  ]
 ],
 "notification": {
  "destination": "https://application-server.com",
  "token": "---------- token ----------"
 },
 "createdTime": "2024-03-13T09:12:28Z",
 "expiredTime": "2024-04-13T09:12:28Z"
}
```

6.5.1.4 DELETE

Delete a subscription.

6.5.1.4.1 Request:

```
DELETE /insights/v1/subscriptions/{subscriptionId}
```

6.5.1.4.2 Parameters:

- subscriptionId (required)

6.5.1.4.3 Example Request Body:

```
DELETE /insights/v1/subscriptions/25487591
```

6.5.1.4.4 Response:

- 200 Successful Operation