Expeto Platform Guide

v0.0.0

Table of contents

1. Expeto Platform Overview 3
1.1 Platform Architecture 3
1.2 Expeto Gateway 3
1.3 Expeto Edge 4
2. xControl 6
2.1 Expeto xControl 6
2.2 Getting Started 8
2.3 User Guide 11
2.4 Installation 72
3. xView 80
3.1 xView Guide 80
3.2 Metrics 86
4. API 92
4.1 Introduction 92
4.2 Getting Started 93
4.3 Exploring APIs with HAL 100
4.4 Endpoints 104
4.5 Troubleshooting API 128
5. Support 130
5.1 FAQ 130
5.2 Knowledge Base 132
6. xCore 134
6.1 Expeto xCore 134
6.2 Installation 138
6.3 Expeto CLI Guide for Operations 164
7. xRouter 166
7.1 Expeto xRouter 166
7.2 Installation 167

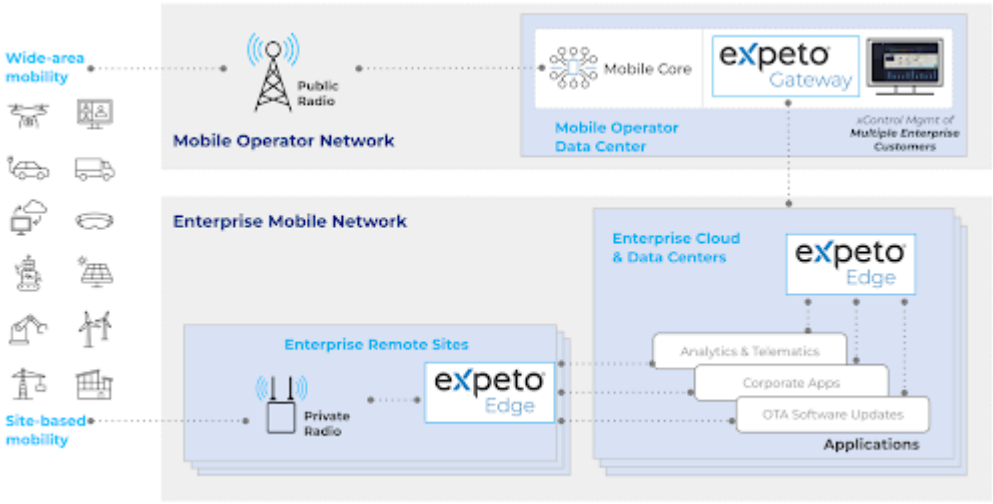
1. Expeto Platform Overview

1.1 Platform Architecture

The Expeto Platform is a software extension to a Mobile Network Operator's (MNO) primary Network Core, deployed alongside it to deliver standard 4G/5G network elements for Private Mobile Networking to Enterprise customers. The platform enables comprehensive management of connected assets, policy enforcement, and mobile network performance monitoring in globally distributed deployments. Its components provide centralized control and management capabilities to both MNOs and Enterprise customers across a single control plane.

The Expeto Platform consists of two main software components:

- Expeto Gateway - Deployed and hosted alongside the MNO Network Core
- Expeto Edge - Deployed and hosted by the Enterprise customer within their networks or hosted by the MNO as part of a managed service offering



1.2 Expeto Gateway

The Expeto Gateway serves as a multi-tenant service deployed within the MNO infrastructure, providing a centralized management layer across all customers. The Gateway comprises three integrated components:

1.2.1 xControl

xControl provides the UI/API interface for subscriber administration and network policy orchestration for MNO's and their enterprise customers. It delivers a self-service, global single pane of glass for management and configuration of Edge sites. Through xControl, users can manage mobile assets/SIMs/eSIMs, apply network policies, and administer network segments for the private network.

xControl is a cloud-native solution that does not require real-time access to the Edge platform. Edge agents collect metrics and "call home" to xControl for updates only. Any interruption in this connection does not cause network issues for devices connected to the private mobile network.

Key capabilities of xControl include:

- Access through web browser UI or standard REST API calls
- Multi-tenant architecture supporting delegated administration
- SIM/eSIM management
- Bulk actions (import, update, delete) for mobile assets
- Profile creation for service level assignment to mobile assets
- Roaming Profile creation for assigning carrier access restrictions
- Account, customer, and site management
- Role-Based Access Control (RBAC)

→ Learn More

1.2.2 xRouter

xRouter is a software application that redirects data from the public mobile network provider to Edge sites under enterprise control via a secure tunnel. xRouter handles subscriber authentication, policy management, and network routing before connecting to the Edge site.

The MNO providing the public network manages device authentication through the authentication service within the xRouter platform, which provides a proxy service to route traffic to the respective Edge sites on a per subscriber basis. All communication between the public macro network and xRouter utilizes standard 3GPP interfaces, with the xRouter software application fully integrated with the MNO network.

xRouter features include:

- Secure data path control within geographic boundaries
- Secure tunneling to carry user data packets to Edge sites
- Public RAN deployment support
- Traffic routing from MNO public core to specific network segments within customer Edge
- 3GPP standard interface utilization
- Subscriber-based data path selection
- GTP Proxy connectivity between MNO public macro network and Enterprise Edge

→ Learn More

1.2.3 xView

xView is a monitoring system based on the open source Prometheus platform, providing metrics collection and alert triggering capabilities. Its distributed architecture offers comprehensive network monitoring with a centralized aggregated view of the Expeto Platform operational status.

xView collector agents gather metrics from all platform components, federate these metrics, and forward them to the xView master. The master evaluates the metrics against alert expressions stored in the Expeto alert library, triggering alerts when expressions remain unresolved. xView integrates with third-party tools such as Grafana and Datadog for data visualization and alert management.

1.3 Expeto Edge

Expeto Edge provides a complete, standards-based, 3GPP-compliant 4G/5G packet core platform to support the necessary functionality for hybrid mobile networks. It is deployed as the **xCore** component for connection to either a local private RAN or to the public macro RAN of the MNO.

An xCore instance can be installed either physically at a remote Enterprise site or virtually in a private cloud environment. Each site supports multiple network segments ("Systems"), each with a unique CIDR/subnet, as supported by the host network. xCore is typically deployed "behind the Enterprise firewall," offering full control to the Enterprise to support their mobile networks. Alternatively, xCore can be deployed in the MNO environment and hosted for the Enterprise as a fully managed service.

The Expeto Agent is deployed on the platform and provides monitoring, watchdog functionality, and configuration synchronization. The Agent maintains an asynchronous connection to xControl to ensure configuration changes are automatically propagated to different xCore locations. Connection for the Agent requires "outbound only" traffic to the xControl platform.

xCore is installed on an edge server running the Linux operating system with Kubernetes containers. The platform provides scalability through additional compute resources to support the Expeto software.

For private RAN deployments, local authentication of devices operates without requiring continuous backhaul connection to the remote site. The local mobile asset database contains all valid subscriber details and maintains synchronization via the local Expeto Agent connected to xControl. If the Agent is disconnected, new changes are queued on xControl and forwarded when the connection is reestablished.

For public RAN deployment, xCore connects to one or more MNO-hosted xRouter sites to provide secure private networking over the public mobile network with low latency.

Expeto Edge features:

- 3GPP-compliant 4G/5G core network elements
- Deployment flexibility (corporate data center, private cloud, or remote sites with local authentication)
- Local authentication without continuous backhaul connection requirement for private deployments
- Cloud-native architecture utilizing containers and orchestration
- Enterprise deployment options (data center, cloud, or edge)
- Containerized packet core components with dynamic scalability

→ Learn More

2. xControl

2.1 Expeto xControl

xControl is a multi-tenant service hosted by Expeto or deployed in an MNO network that gives enterprises centralized management of their network segments and connected assets across a single control plane.

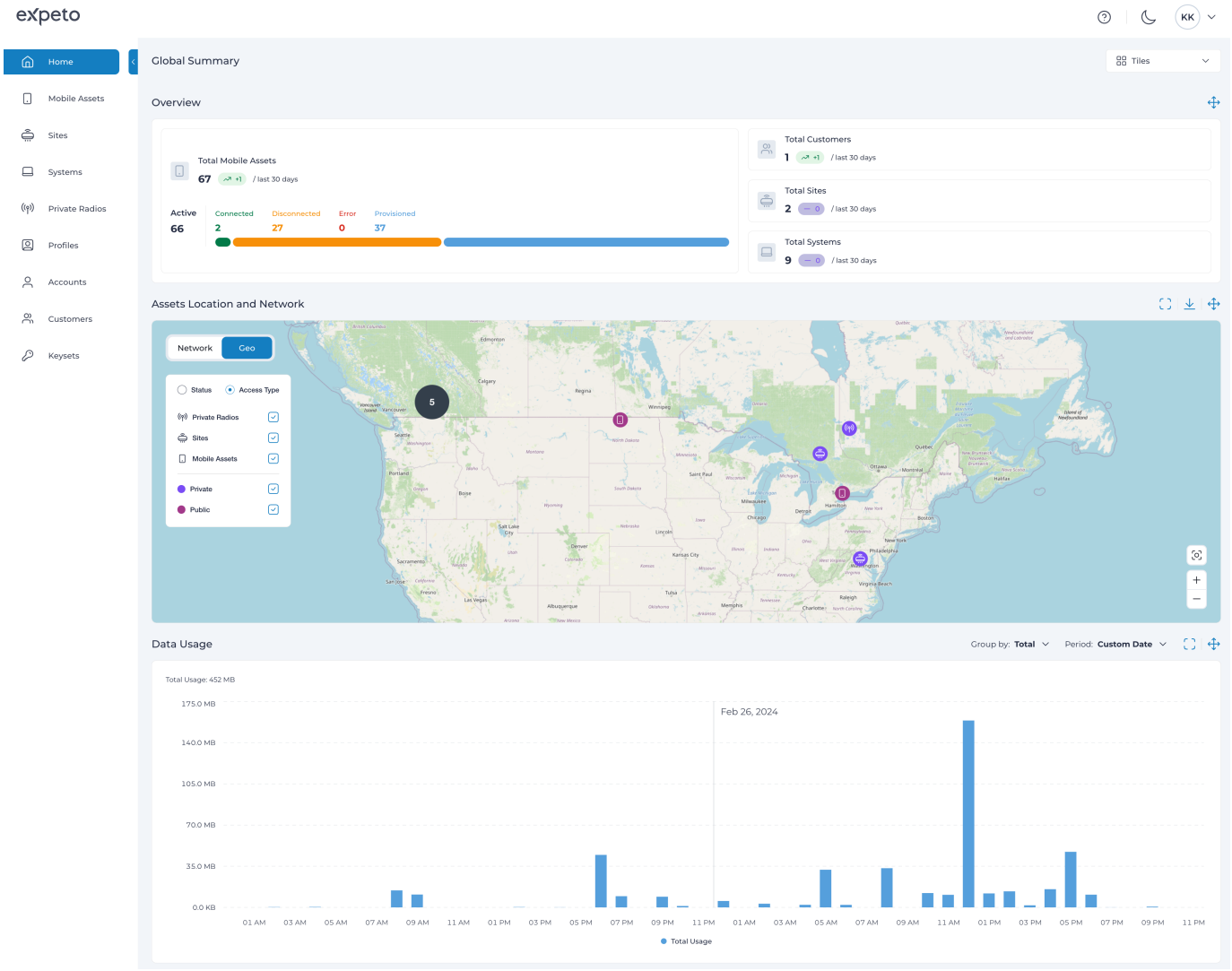
It provides visualization and control for all public and private networks, network monitoring, and secure connection to public networks.

All functionality exposed in the web interface is also available via the REST API.

Expeto xControl is a cloud-based application accessed through a web browser GUI or through REST API calls. It provides a single control plane for SIM management, networking, security, and routing of all devices and data across private and public mobile networks world-wide. The GUI interface features include an interactive dashboard, intuitive navigation, real-time updates, and deep search. As an administrator, you create and manage the systems, customers, and subscribers of your sites. You can monitor network health and subscriber status, data usage and throughput levels. You set thresholds to enforce data limits, create system profiles, and set custom network parameters.

Common tasks performed with Expeto xControl include:

- SIM and device provisioning
- Performing bulk actions (import, update, delete) to Subscribers
- Managing Subscribers
- Creating System Profiles which assign different services levels to Subscribers
- Creating Roaming Profiles and Carrier Groups to restrict 3GPP roaming access
- Segmenting Networks by creating Systems and assigning subnet ranges
- Managing Accounts, Customers and Sites
- Managing Carriers and Carrier Groups
- Adding and enabling Private Radios (eNodeB/gNodeB)
- Setting Data limits (AMBR values)
- Network monitoring through dashboard panels



2.1.1 Role Based Permissions

The Expeto xControl GUI and API is multi-tenant, supporting delegated administration of functions. Accounts can have nested customers allowing for deep hierarchies. Subordinate accounts only have access to functions and data within the restricted domain governed by the Role Based Access Control (RBAC) privilege set.

Each account used to log in to Expeto xControl has an assigned role that determines access and permissions to features. Roles are pre-defined and selected during account creation or update.

→ Learn More

2.2 Getting Started

Expeto xControl is accessed through a web browser or by using an API client.

For instructions on how to access and interact with the API, see the Expeto API documentation.

2.2.1 Log in to Expeto xControl

To log in to Expeto xControl:

1. Using a web browser, go to the URL where your xControl is hosted or access the SaaS hosted instance at admin.expeto.io
2. Log in with your Account credentials.

> **Note**
>
> The username for login is a valid email address. The use of usernames that are not email addresses has been deprecated.

2.2.2 Change Password

Account password syntax requirements are as follows:

- Must be at least 8 characters long
- Must contain a number
- Must contain an uppercase letter
- Must contain a special character

Passwords are never exposed as plain text in the Expeto xControl GUI.

To change your password:

1. Login with your existing password.
2. Click your Account name in the upper right corner of the application.
3. Select **User Account**. The Account page appears.
4. To change your login password, click **Edit**, type the new password and click **Submit**.

If you have forgotten your password, click the link on the log in prompt, provide your email address, and click **Request Password Reset**.

To change the password of any Account:

1. Navigate to Accounts.
2. Select an Account and click **Edit**. The Account parameters are displayed. Note that the Password field is empty because the existing password is never displayed.
3. Type a new Password using the syntax requirements. The password characters are hidden.
4. Click **Submit**.

2.2.3 Assign Roles to an Account

Each Account must be assigned a role. The role determines the permissions given to the Account for accessing and interacting with Expeto xControl features.

To assign a role:

1. Navigate to Accounts.
2. Click **New** or select an existing Account and click **Edit**. The Account properties appear.
3. Click **Role** and select from the available roles.

The following roles are available:

ADMIN

The highest level of access for network administration. The ADMIN role grants access to networks belonging to all Customers and child Customers.

CUSTOMER_ADMIN

The standard access for network administration. Network access is restricted to a single Customer (and any child Customers) associated with the Account.

READ_ONLY

Restricted access to the network. Provides read-only access for monitoring the network. Access is restricted to the Customer (and child Customers) associated with the Account. No create or delete permissions are granted. No changes to the network can be made.

Users assigned with either the ADMIN and CUSTOMER_ADMIN role can create a new account and assign a role, or change the role of an existing account.

You cannot assign a role with more permissions than your current role. For example, a user with a CUSTOMER_ADMIN role cannot create an account with the more powerful ADMIN role.

Each role exist to provide different levels of access to network and management features. For complete list of roles and their permissions, see Roles and Permissions.

2.2.4 Navigation

Use following navigation menu items to access xControl features:

Home

The default landing page for Expeto xControl is an interactive dashboard.

Select filters from dropdown menus to refresh graphs accordingly. Click Sites, Systems, and Mobile Asset sections to drill into more details.

Sites

A Site is the physical server where a single instance of the Expeto xCore is installed.

Systems

Systems are defined by a network IP address range expressed in CIDR notation. Create a single System for each Site, or segment your network into multiple Systems, assigning each System a unique network IP block or subnet. A System can be shared between Sites in different physical locations, allowing seamless access to the System's Mobile Assets.

Multiple Systems can be used for creating secure separation of devices using network segmentation. Extensions set at a System level are applied to all Mobile Assets belonging to the System unless explicitly overridden.

Private Radios

Private Radios (eNodeB/gNodeB) facilitate the communication from/to 4G/5G devices to private network core.

Mobile Assets

A Mobile Asset is a 4G/LTE or 5G device identified on the network by its SIM card. Each Mobile Asset has a unique IMSI value that serves as the primary identifier. Add Mobile Assets to Systems individually, or collectively by using the bulk import feature.

A Mobile Asset's *state* can be set to Active or Inactive which allows or disallows access to the network. A Mobile Asset's *status* such as 'Provisioned' or 'Connected' reflects stages in the Mobile Asset provisioning lifecycle.

Values for any Extensions (custom parameters) explicitly set at the Mobile Asset level override parameters set in the Customer, or System Profile level.

System Profiles

A System Profile is associated with a System and contains an APN name, bandwidth configuration, plus any Extensions.

A System Profile is available only for Mobile Assets belonging to the associated System. The same System Profile can be applied to multiple Mobile Assets. Extensions set in the Profile override the same parameter values set at the Customer or System level.

For example, *Gold*, *Silver*, and *Bronze* System Profiles can be configured to permit different levels of bandwidth.

A Mobile Asset must be assigned a System Profile.

Customers

A Customer can own Sites and have multiple Expeto Accounts. A Customer hierarchy can be created. For example, a parent Customer can manage any of its child Customers' networks. Each child Customer manages their own network, but cannot manage their parent Customer's network.

Extensions set at the Customer level create default settings that apply across all Sites, Systems, System Profiles, and Mobile Assets belonging to the Customer. These default settings can be explicitly overridden.

Accounts

An Account includes a username (email address) and password combination plus an assigned role. A user account must belong to an existing Customer. Expeto xControl is multi-tenant, supporting simultaneous access to multiple accounts.

The pre-defined role associated with the account determines access to Expeto xControl features and permissions.

Reports

Generation of on-demand reports is a task available to Admin and Customer Admin roles. Input parameters involve setting a time frame for the report and specifying the Customer, and optional child Customers to examine. Multiple report types are available providing different levels of data granularity such as Summary of Usage by IMSI and Summary of Usage by Site.

The generated report is in text/CSV format and can be downloaded. To generate and download files, see Generate Usage Reports.

2.3 User Guide

2.3.1 Dashboard

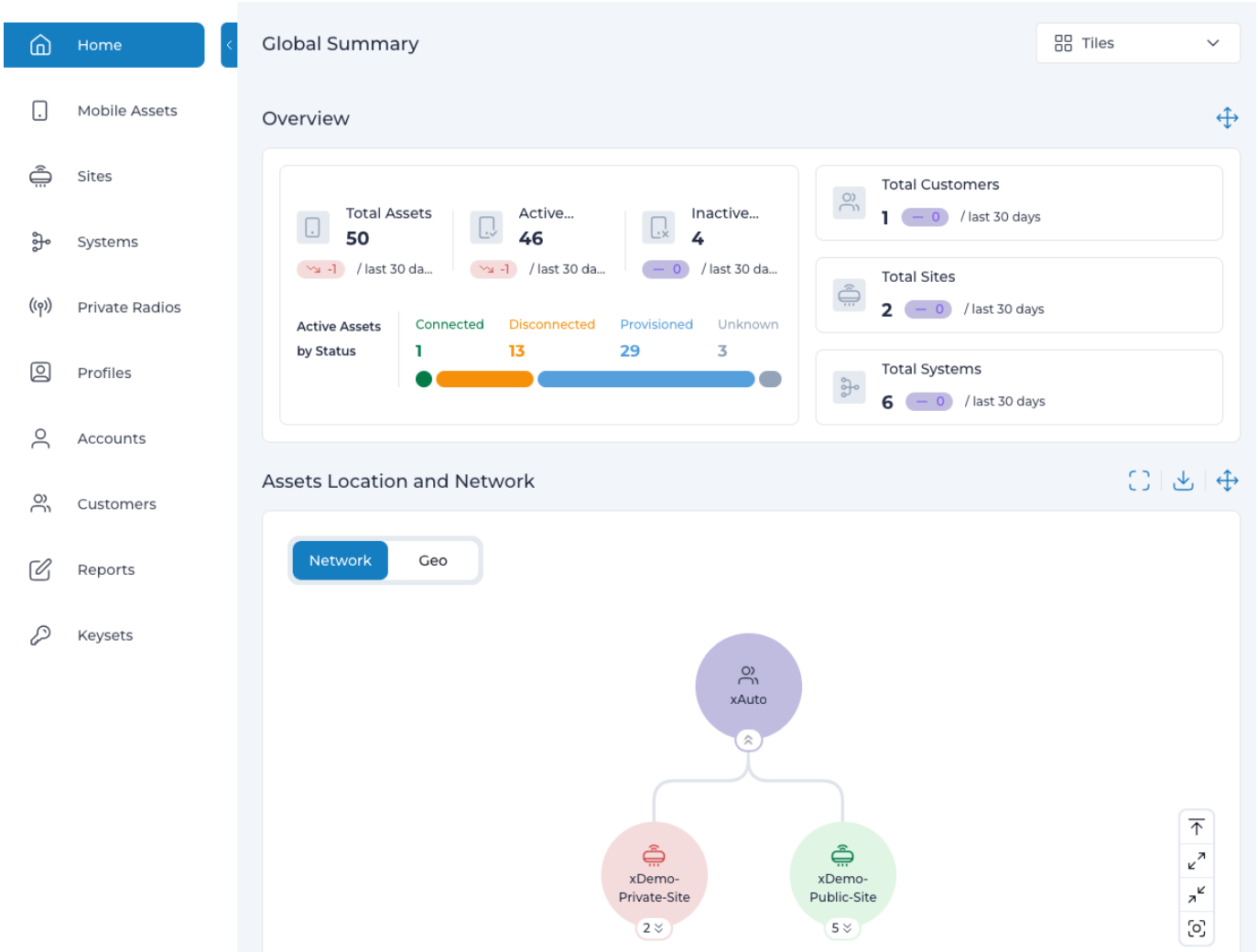
Overview

The dashboard provides administrators with real-time visual monitoring of key network components, including Subscribers, Sites, and Systems. It offers both static and interactive panels for quick insights into the network's status and trends. Context-specific panels are available on related pages, such as Customers, Sites, Systems, and Subscribers, enabling more targeted analysis.

Key features include:

- **Overview Metrics**: Displays counts and 30-day trends for Mobile Assets, Sites, and Systems, helping administrators identify issues and monitor performance changes.
- **Network Topology**: An interactive map of Customers, Sites, and Systems, showing their relationships for easier navigation and analysis.
- **Geographical Map**: Provides a location-based view of Mobile Assets, Sites, and Private Radios, with filtering options for access type and status.
- **Data Usage History**: Graphs displaying network data usage over time, configurable by date range and grouping criteria for deeper insights.

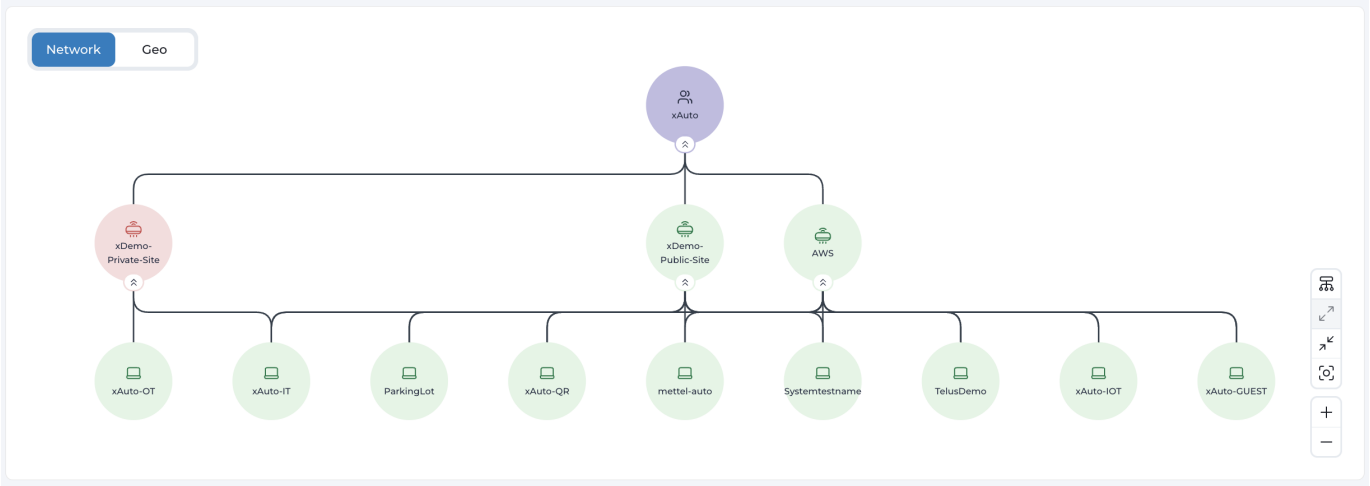
The dashboard integrates seamlessly with xControl and supports advanced monitoring through third-party tools like Datadog.



Click a panel to navigate to the corresponding component page to see more information.

Network Topology

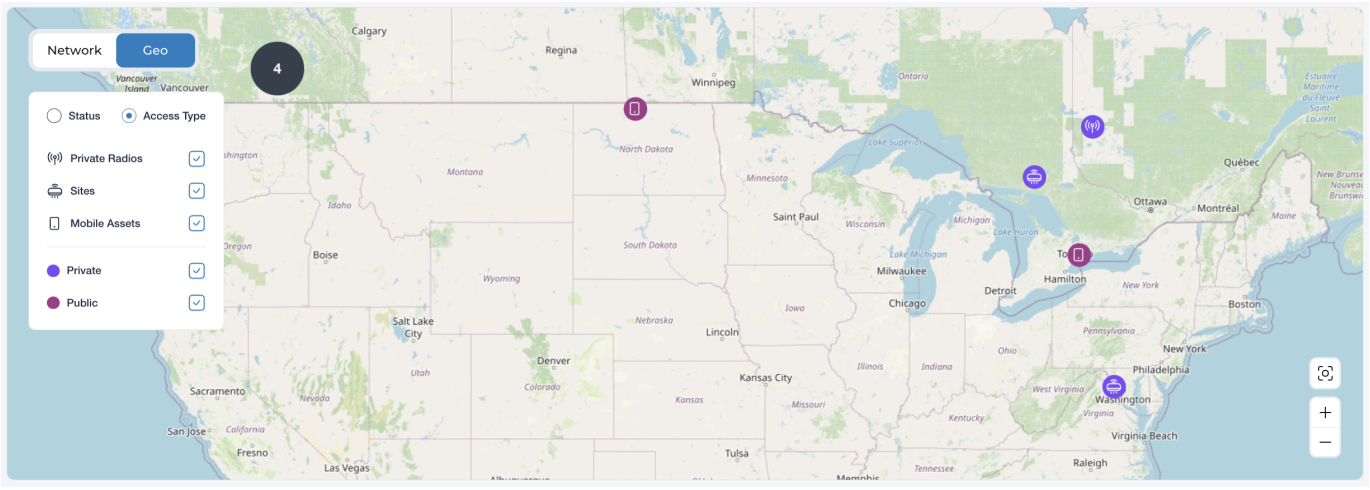
Interactive network topology view of Customers, Sites and Systems and their relations.



Hover your mouse over the nodes bars to view details and navigate to the entity.

Geographical Map

Geographical map displays locations of Mobile Assets, Sites and Private Radios.



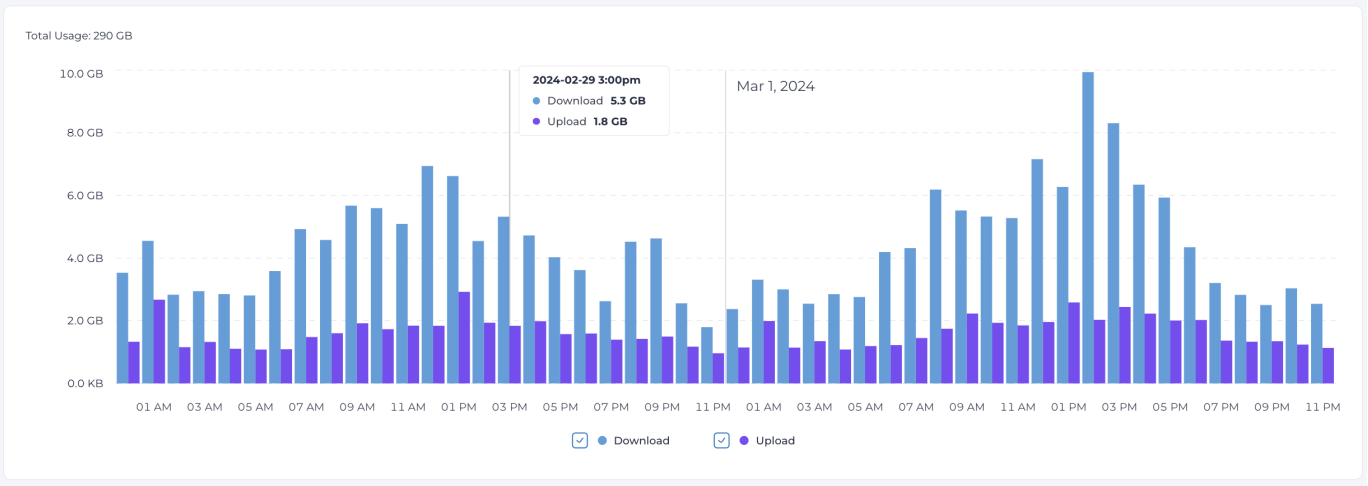
Hover your mouse over the map items to view details.

Select **Status** or **Access Type** to view Mobile Assets, Sites and Private Radios by their Access Type (Private or Public). Then select filter by specific Access Type or Status.

Click on the Fit button anytime to recenter the map to fit all the items in the view.

Data Usage

Data Usage panel panels contain graphs displaying network data usage history based on configurable criteria.



Hover your mouse over the graph\'s vertical bars to view finer statistical details.

!!! note Data usage graphs are also available in Mobile Assets, Sites, Systems, Customers pages and in their details pages.

GROUPING

Customize the graph by selecting the **Group By**: *Total*, *AccessType*, or *Direction*. This increases granularity by providing a comparative view of how much bandwidth, for example, each Subscriber or System has used.

Grouping by *Direction* displays the usage data broken into download and upload metrics.

SETTING DATE RANGE

Select one of the **Period** options from the dropdown menu, or click the existing range to access the calendar, then manually choose a start and end date.

2.3.2 Mobile Assets

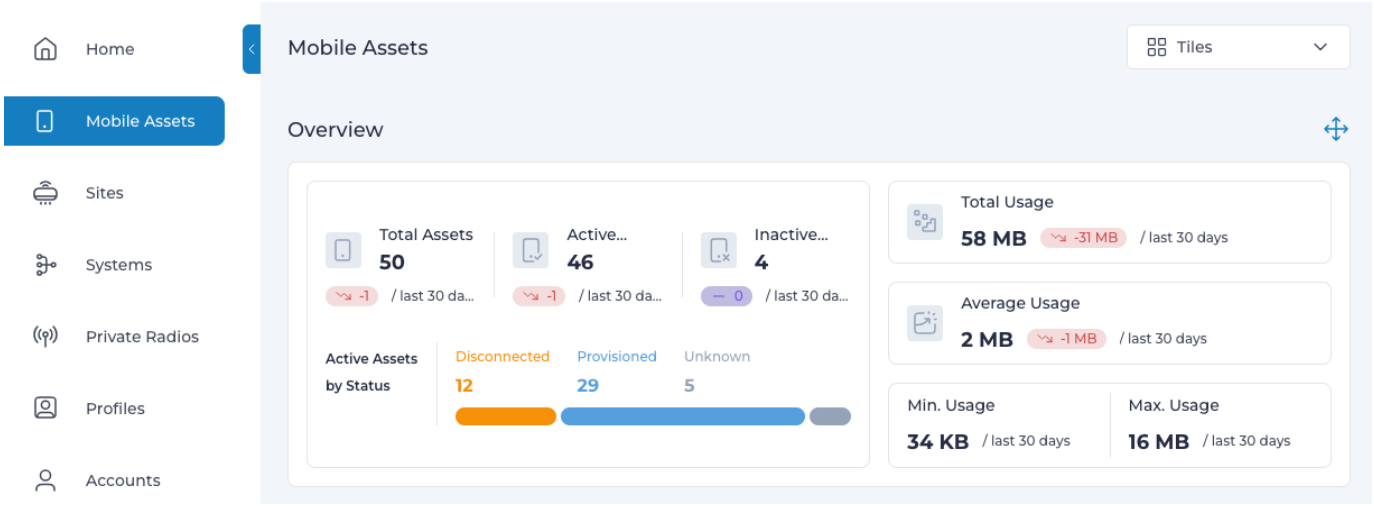
Manage Mobile Assets

Mobile Assets

Mobile Assets are 4G/LTE or 5G devices identified on the network by their SIM card and uniquely defined by their **IMSI** (International Mobile Subscriber Identity). These assets, which include smartphones, IoT devices, industrial equipment, and autonomous systems, enable data transfer, telemetry, and operational tasks within private and public mobile networks. Additional identifiers, such as the Mobile Station Integrated Services Digital Network (MSISDN) number, further distinguish Mobile Assets and facilitate their integration into network operations.

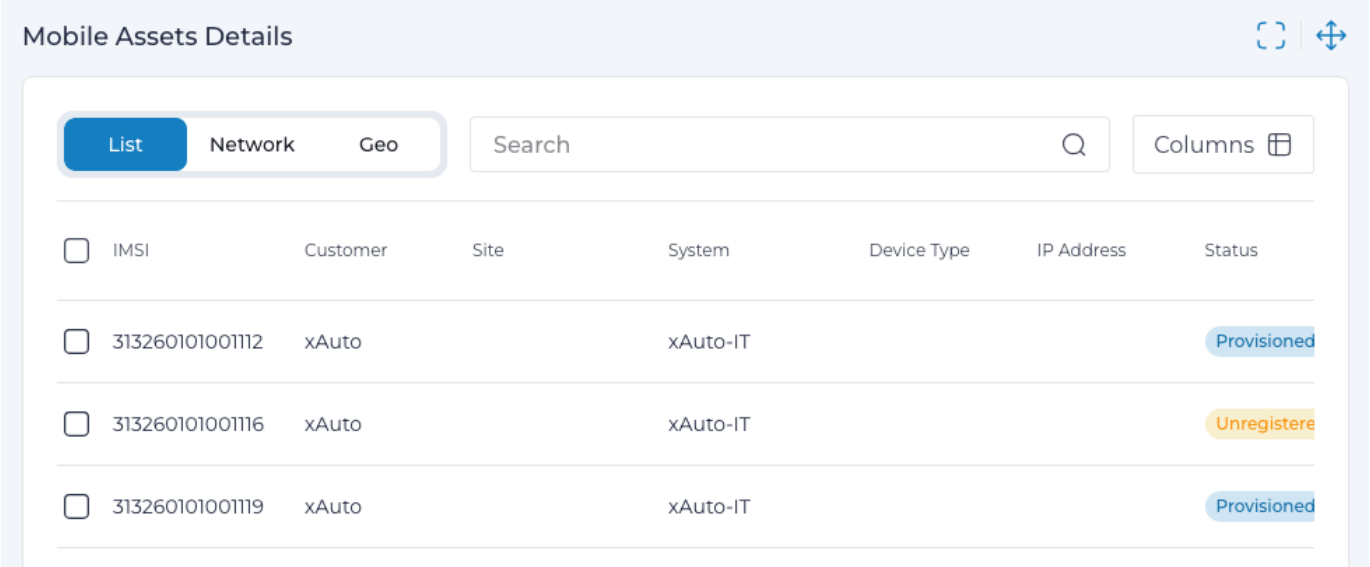
In Expeto xControl, Mobile Assets are managed to ensure proper authentication, security, and network functionality. They are configured to belong to specific **Customers**, assigned to **Systems**, and associated with **System Profiles** that define bandwidth, access permissions, and other operational parameters. Administrators can add Mobile Assets individually or in bulk using the **Bulk Import** feature, ensuring streamlined onboarding and efficient network management. Proper configuration of Mobile Assets is essential for maintaining network integrity and optimizing resource allocation.

To manage Mobile Assets, navigate to the **Mobile Assets** section in the left navigation panel. This section allows you to view and organize Mobile Assets, manage their status, terminate sessions, and configure advanced properties such as extensions.



MOBILE ASSET LIST

The **Mobile Asset List** displays key details for each Mobile Asset in a tabular format. By default, the following columns are included:



Column	Description
IMSI	A unique numeric identifier for the Mobile Asset. Click the IMSI link or expand the entry to view detailed information.
Customer	The customer associated with the Mobile Asset. A single Customer may own multiple Sites, Systems, and Mobile Assets.i
Site	The physical location where the Expeto xCore is installed and where the Mobile Asset is attached.
System	The System to which the Mobile Asset is assigned, including one or more network gateways owned by the Customer.
Device Type	The type of device used to access the System.
Status	The current operational status of the Mobile Asset, derived from Gateway and database session states.
IMEI	A unique serial number assigned to the device by the manufacturer.
IP Address	The IP address currently assigned to the Mobile Asset.

Customize Columns

You can adjust the displayed columns to suit your needs:

1. 

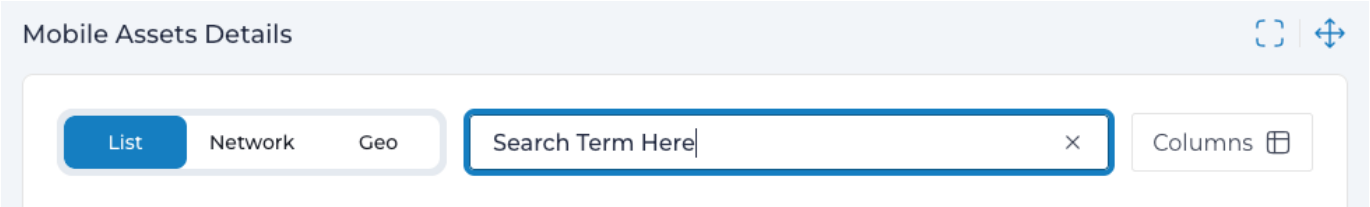
Click the **Column Chooser** icon.
2. Check or uncheck columns to add or remove them from the list view.

The flexibility of the column chooser allows you to focus on the most relevant details for managing Mobile Assets effectively.

SEARCHING

The **Deep Search** feature allows you to locate Mobile Assets based on any associated property, not just those visible in the list.

This powerful search capability supports case-insensitive queries and partial text matches, making it easy to find specific Mobile Assets.



Note

- Wildcards are not supported in the search field.
- For more help on search, see Search Tips

Using Boolean Operators

You can refine your searches further by using boolean operators to narrow results. This is particularly useful for searching by labels, properties, or statuses.

Example Searches

- **Search by Network Type**: Type `4G` to find all Mobile Assets with 4G network sessions.
- **Search by Status**: Type `err` to list all entries with an Error status.
- **Search by Username**: Enter a username associated with a Mobile Asset.
- **Search by Label**: Use labels such as `sim-type` or `sim-type=UICC` to filter specific Mobile Assets based on extensions or tags.

Info

- For more help on search, see Search Tips

ADD A MOBILE ASSET

Note

- To add a Mobile Asset, the **Add Mobile Asset** permission is required. The **CUSTOMER_ADMIN** role does not include this permission.
- Additionally, your configuration must have a - **Customer**, **Site**, **System**, and **System Profile** already created.

Steps:

1. Navigate to **Mobile Assets**.
2. Click **New** and fill in the required parameters.
3. Click **Submit** to save the new Mobile Asset.

Identification and SIM Authentication Properties

When adding a Mobile Asset, several properties must be configured to ensure proper authentication, network access, and alignment with system requirements. These properties are categorized into two main groups: **Identification and SIM Authentication** and **Network Access and Inherited Properties**.

Identification and SIM Authentication Properties

Property	Description
IMSI (Required)	The International Mobile Subscriber Identity, a unique numeric identifier for the Mobile Asset. Includes: Mobile Country Code (MCC), Mobile Network Code (MNC), and Mobile Asset Identification Number (MSIN).
MSISDN (Required)	The phone number associated with the Mobile Asset, used for communication and identification within the network.
ICCID (Optional)	The Integrated Circuit Card Identification Number, an 18–22 digit code printed on the SIM card. Includes information about the country, home network, and unique SIM identifier.
K (Required)	A 32-character hexadecimal shared secret key used with the OP value to generate the OPc value for secure authentication.
OPC (Required)	The Operator Code, a 32-character unique network-specific value. Used with the shared secret key (`K`) to ensure secure authentication.
Keyset	Select from existing Keysets for SIM authentication security. Only Keysets mapped to an HSS Keyset ID and associated with the current Site are available. A Keyset can be assigned to a Mobile Asset only once. For more details, see Add a Keyset.

Network Access and Inherited Properties

Property	Description
Status	Specifies the current state of the Mobile Asset:
	- **Active**: Allows the Mobile Asset to access the network.
	- **Inactive**: Temporarily restricts network access without deleting the Mobile Asset.
Extensions	Custom parameters applied to the Mobile Asset. Extensions at the Mobile Asset level override parameters set at the Customer, System, or System Profile levels. For more information, see Customize with Extensions.
Customer	The Enterprise owner of the Mobile Asset, linking it to the Customer's network and settings.
System	The Mobile Asset can be assigned to one System, which defines the network IP address range and related configurations.
Profile	Profiles govern bandwidth limitations, extensions, and access settings. Each Mobile Asset must have one System Profile assigned, which can apply standardized settings or tiered service levels (e.g., *Gold*, *Silver*, *Bronze*). Extensions set at the System Profile level override those set at the System or Customer level.

ADD MULTIPLE MOBILE ASSETS FROM A FILE

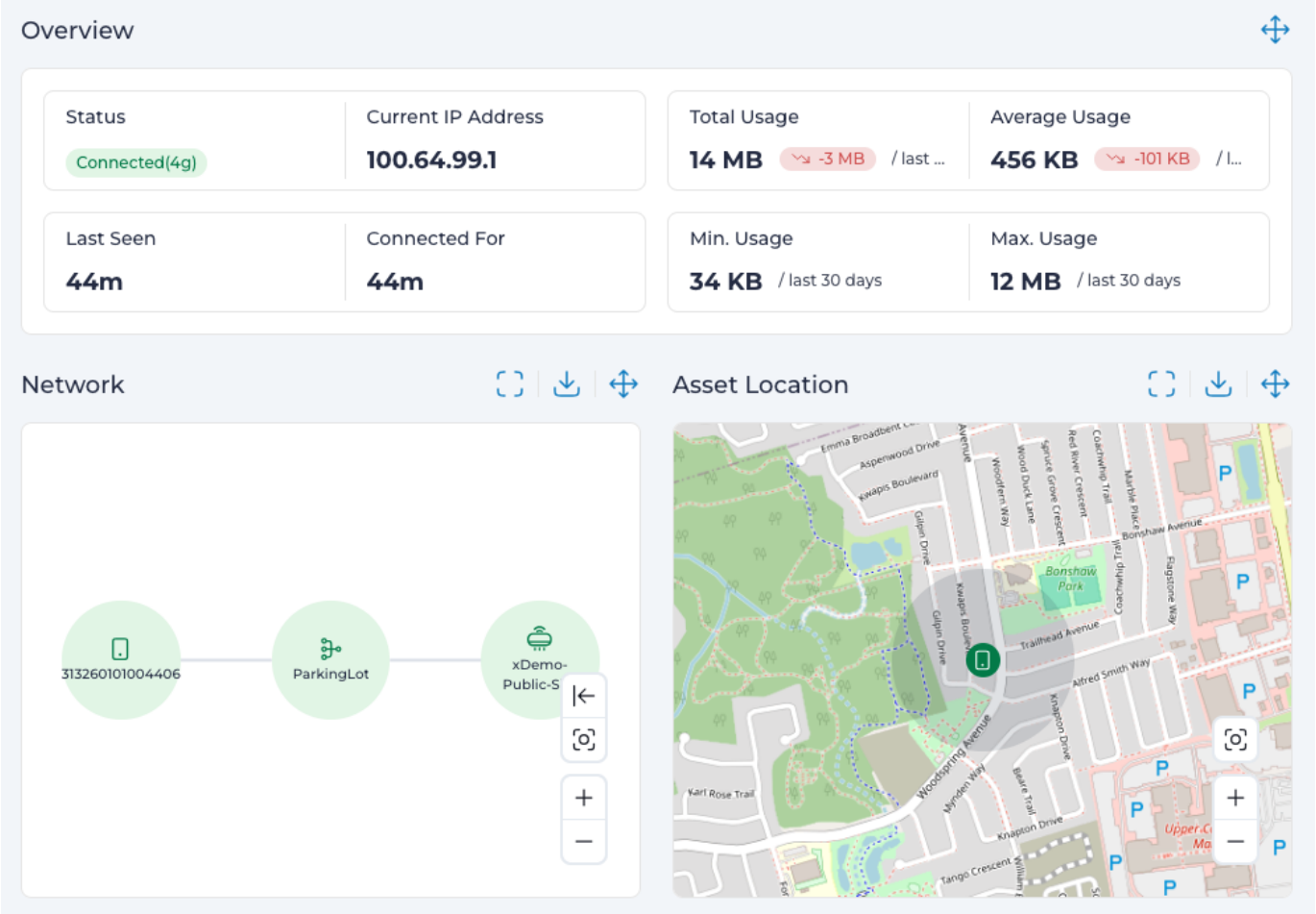
1. Navigate to **Mobile Assets**.
2. Click the **Import** button and select **Bulk Import**.
3. Follow the prompts to upload and import an `.out` file.

For detailed information on the required file format and the Bulk Import process, see Perform Bulk Actions.

VIEW / EDIT A MOBILE ASSET

Clicking on the IMSI number in the main Mobile Asset list opens the Mobile Asset Details page. This page provides detailed information about the selected Mobile Asset, including its current status, associated customer and system, IP address, and additional configuration details. Use this view to manage and analyze the Mobile Asset's settings and activity.

The Mobile Asset status is updated in real time and derived from the Mobile Asset's activity recorded in the database and network.



Section Name	Description
Status	Indicates the current connection status of the Mobile Asset.
IP Address	The current IP Address of the Mobile Asset.
Time Connected/Time Since Connected	The amount of time since the PGW session started, or the time elapsed since the previous PGW session stopped.
Last Seen	The time the last verification was made to a connected SIM, or the disconnect time of the last active session.
Total Usage	The amount of data used in during the last 30 days. Includes both upload and download.
Average Usage	The average data used in during the last 30 days. Includes both upload and download.
Minimum Usage	The minimum amount of data bandwidth in a day in the last 30 days. Includes both upload and download.
Maximum Usage	The maximum amount of data bandwidth in a day in the last 30 days. Includes both upload and download.

The following tables show normal and warning status values:

Normal Status Values

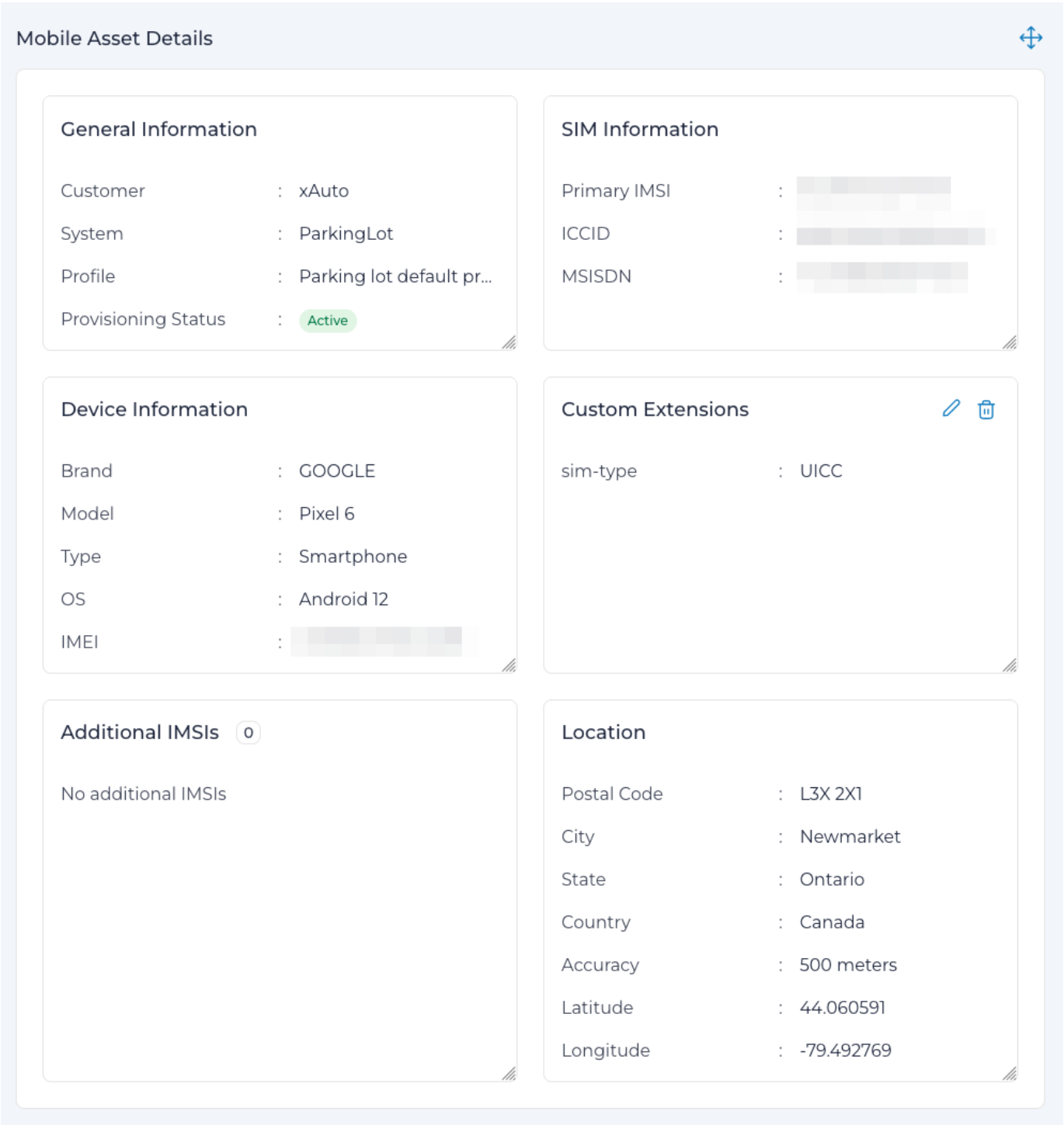
Status	Technical Notes	Reason
Provisioned	The mobile asset is registered in the database, but has never has an active session on the network.	This is the initial status for a Mobile Asset.
Registered(HSS)	The mobile asset has authenticated but does not have a current data session (over data/user plane) on the network.	The device may been powered off, moved out of range, or the SIM card removed. The Disconnected status has expired. Alternatively, the mobile asset is an SMS/IoT device where data plane session is not applicable/available, and the data is transferred over control plane.
Connected(4g)	The registered mobile asset has an active data session on the network.	The normal state of an active Subscriber Session. The IP address and network type (3G, 4G or 5G) are displayed.
Disconnected	The mobile asset's active session on the network was closed	The active session on the network ended recently. This status occurs when the device is put on airplane mode, or the SIM card is physically removed.

Warning Status Values

The following values indicate a problem with normal Mobile Asset operation. Additional troubleshooting may be required.

Status	Technical Notes	Reason
Connected(PGW)	Warning. The mobile asset is actively connected to the network; however, there is not associated database record.	The mobile asset database server may be down.
Unregistered	The mobile asset had a previously active session, but is now detach from both the mobile asset database and the network.	The device has been powered off or moved out of range.
Error(PGW)	Error state for the network connection.	Causes can include network connection problems to the Gateway, or Gateway failure.
Error(HSS)	Error state for the network connection.	Causes can include network connection problems to the database, or database failure.
Unknown	The status was unable to be determined.	Cause unknown.

MOBILE ASSET DETAILS



The **Mobile Asset Details** section provides a comprehensive overview of the Mobile Asset, organized into the following sections:

Displays essential information about the Mobile Asset's network assignment and status:

- **Customer**: The entity to which the Mobile Asset is assigned (e.g., "xAuto").
- **System**: The network system associated with the Mobile Asset (e.g., "ParkingLot").
- **System Profile**: The system profile applied to the Mobile Asset, defining its bandwidth and network settings.
- **Provisioning Status**: Indicates whether the Mobile Asset is active or inactive in the network.

Shows identifiers for the Mobile Asset's SIM card:

- **Primary IMSI**: The main IMSI associated with the SIM card.
- **ICCID**: The unique SIM card identifier.
- **MSISDN**: The phone number associated with the SIM card.

Provides details about the Mobile Asset's hardware and software:

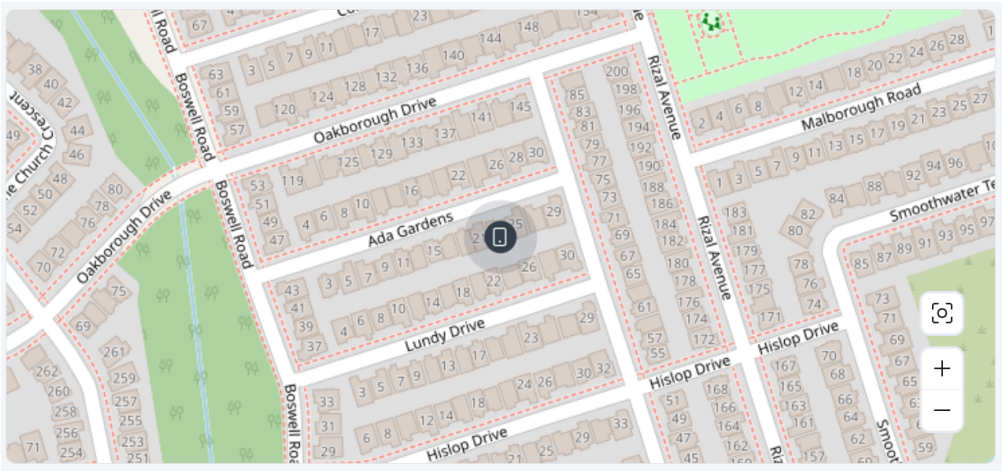
- **Brand**: The manufacturer of the device (e.g., "Google").
- **Model**: The device model (e.g., "Pixel 6").
- **Type**: The type of device (e.g., "Smartphone").
- **OS**: The operating system version (e.g., "Android 12").
- **IMEI**: The unique identifier for the device.

Lists any custom parameters or extensions applied to the Mobile Asset:

> **Example**
>
> **sim-type** set to **UICC**.

Displays alias IMSIs associated with the primary IMSI, if any. If no alias IMSIs are configured, this panel will indicate that there are no additional IMSIs.

The **Location Map** provides an approximate location of a Mobile Asset when it is connected to a public or public/private network. This feature is useful for tracking device activity and ensuring network coverage across various areas.



How Location is Determined

Location data is derived from cell tower triangulation, which estimates the Mobile Asset's position based on signals from nearby towers. Accuracy may vary depending on tower placement:

- **Accuracy Range**: Typically between 500 meters and 1500 meters.

Viewing Location Details

Hover over the location marker on the map to view additional data about the Mobile Asset's position:

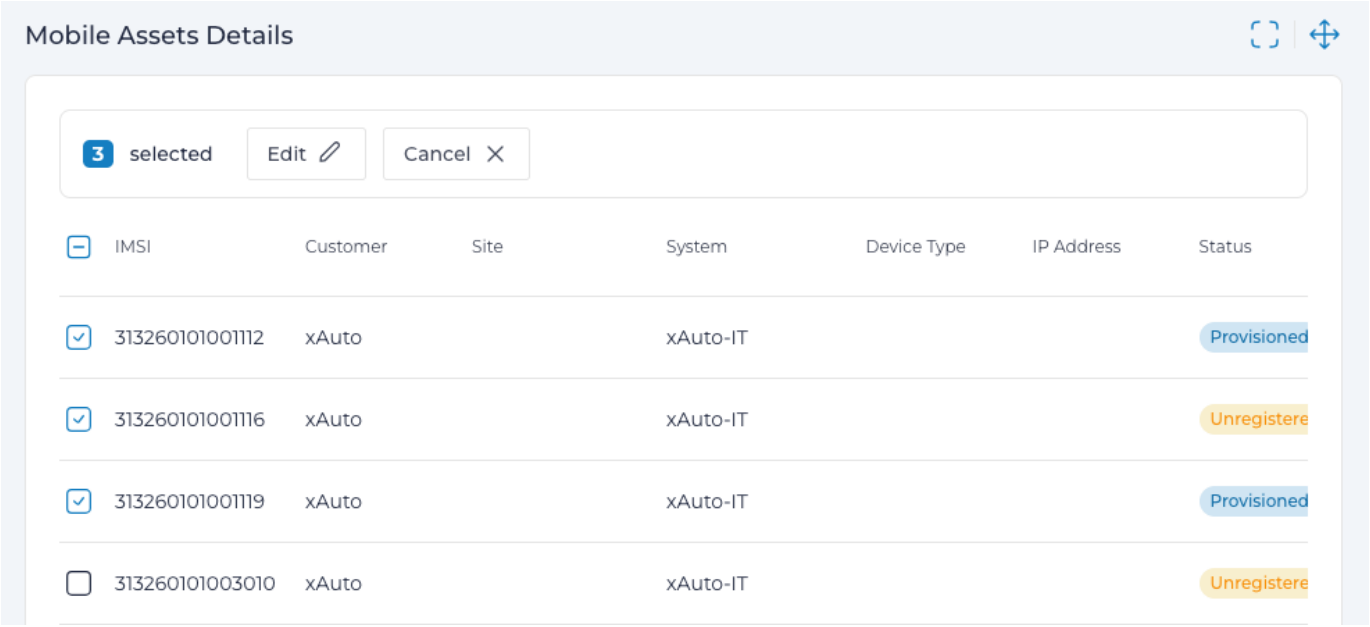
- **Coordinates**: Latitude and longitude of the Mobile Asset.
- **Accuracy Value**: Indicates the confidence in the location estimate.

> **Note**
>
> A lower accuracy value represents higher precision and a smaller margin of error.

> **Note**
>
> The location map is available only for Mobile Assets connected via public or public/private networks. It does not apply to private-only networks.

MOBILE ASSET ACTIONS

Activating or deactivating a Mobile Asset controls its ability to connect and initiate a network session:



- **Active**: The Mobile Asset can initiate and maintain active sessions.
- **Inactive**: The Mobile Asset is temporarily disabled and cannot have an active session.

This functionality is useful for temporarily restricting access without permanently deleting the Mobile Asset.

Activate or Deactivate a **Single** Mobile Asset

1. Navigate to the **Mobile Assets** section.
2. Locate the desired Mobile Asset in the list and click its **IMSI** link.
3. In the details view, click **Edit**.
4. For the **Status** field, select either **Active** or **Inactive**.
5. Click **Submit** to save your changes.

Bulk Activate/Deactivate **Multiple** Mobile Assets

1. Navigate to the **Mobile Assets** section.
2. Select the checkboxes next to the Mobile Assets you wish to modify.
3. Click the **Edit** button.
4. Leave all fields blank except for the **Status** field.
5. Choose **Active** or **Inactive** from the **Status** dropdown.
6. Click **Submit** to apply the changes.

This bulk operation streamlines the management of multiple Mobile Assets, ensuring efficient status updates across the network.

To cancel (terminate) a Mobile Asset session:

1. Navigate to the Mobile Assets.
2. Find the Mobile Asset in the list and click the IMSI link.
3. Click the more button (...) and then click **Cancel Session**.

> **Note**
>
> This action triggers a series of network events to terminate the Mobile Asset's session:
>
> - **For 4G/LTE networks**: The Home Subscriber Server (HSS) sends a **Cancel-Location-Request (CLR)** to the Mobility Management Entity (MME). This instructs the network to withdraw the user subscription associated with the IMSI, effectively deregistering the Mobile Asset.
> - **For 5G Standalone (SA) deployments**: The Unified Data Management (UDM) service notifies registered Network Functions, such as the Access and Mobility Management Function (AMF), of the session deregistration. This action signals the network to terminate the active session.
> - **Session Termination**: Once deregistration is complete, the Mobile Asset's session is fully terminated and removed from the network.
> - **Session Restoration**: After termination, the Mobile Asset may initiate a new session request, depending on the device configuration and network policies. This allows the session to be restored if necessary.

ADDITIONAL IMSIS

Expeto xControl supports **Multi-IMSI SIM cards**, which allow a single SIM card to host multiple IMSIs (International Mobile Subscriber Identities). These SIM cards provide flexibility for devices that need to operate in multiple network environments or require fallback identities for redundancy.

- The **Primary IMSI** is the main identifier for the SIM card and is used as the default identity for network operations.
- **Alias IMSIs** are secondary identities that can be added to the SIM card. These are typically used for specific use cases, such as enabling multi-network access, testing, or separating specific traffic.

Alias IMSIs can be added to the primary IMSI to extend the functionality of the Mobile Asset. To add an alias IMSI:

1. Navigate to the **IMSI** section in the Mobile Asset details.
2. Click the **Add** ⊕ button.
3. Enter the alias IMSI value in the provided field.
4. Click **Save** to confirm.

Once added:

- All IMSIs, including the primary IMSI and alias IMSIs, appear in the **IMSI** section of the Mobile Asset details.
- Only the **Primary IMSI** is displayed in the main **Mobile Assets** list for clarity.

> **Note**
>
> The primary IMSI remains the default identity for all operations, including network authentication and configuration.

- **Static IP Address**: Any static IP address assigned to the Mobile Asset applies exclusively to the primary IMSI. Alias IMSIs cannot have separate static IPs.
- **MSISDN**: The phone number (MSISDN) is also tied only to the primary IMSI and cannot be assigned to alias IMSIs.

- **Network Redundancy**: Allows devices to switch seamlessly between networks if the primary IMSI is unavailable or lacks coverage.
- **Regional Customization**: Alias IMSIs can be used for devices that operate in different countries, enabling compliance with local network requirements.
- **Testing and Validation**: Alias IMSIs can be assigned for testing new network setups without disrupting the primary IMSI's configuration.

Multi-IMSI functionality provides robust flexibility for managing Mobile Assets in diverse and complex network environments. By leveraging alias IMSIs, administrators can ensure seamless connectivity and adaptability across multiple scenarios.

SET EXTENSIONS FOR MOBILE ASSETS

Click **Edit** in the Mobile Asset Details page.

Extensions are key value pairs.

There are specific extensions used to provide functionality not yet available through the xControl GUI. For example, setting a static IP address for a Mobile Asset.

> **Example**
>
> `ip=100.64.12.123`

Other extensions are used to tag Mobile Assets with labels that help find the Mobile Asset during search queries.

> **Example**
>
> `deviceType=samsung` `user=Joe`

For more information on creating Extensions, see Extensions.

Bulk Actions

OVERVIEW

The Bulk Actions feature in Expeto xControl streamlines the management of multiple Mobile Assets by enabling the following operations:

- **Bulk Import**: Add Mobile Assets in large quantities using a `.out` file. This method ensures efficient data entry and allows configuration of parameters such as Customer, System, System Profile, Keyset, and initial status.
- **Bulk Update**: Modify or configure multiple Mobile Assets simultaneously. Updates can include changing status, assigning a System Profile, adding Extensions, or moving assets to a new System. This can be done directly or via a CSV file export/import process.
- **Bulk Delete**: Permanently remove multiple Mobile Assets from the list, freeing up resources and decluttering your network configuration.

These tools reduce manual effort, ensure data consistency, and enable administrators to efficiently manage large-scale Mobile Asset deployments.

BULK IMPORT

The bulk import feature streamlines the process of adding multiple Mobile Assets to the system by uploading an `.out` file. This file contains the required details, such as IMSI values, along with optional configuration parameters. Once imported, the Mobile Assets are listed under the chosen System in the Mobile Asset list.

Key Features:

- Supports importing Mobile Asset details in bulk using a `.out` file.
- Allows configuration of parameters such as Customer, System, System Profile, Keyset, and initial status.
- Includes validation checks to ensure file accuracy and proper mapping of columns.
- Provides flexibility to add Extensions for custom parameters.

---

1. Navigate to the **Mobile Assets** page.
2. Click the **Import** button:  , then select **Import Import**.
3. Click **Choose File** and select the `.out` file to import.
    - If the file format is incorrect or validation errors occur, you will be notified.
    - If the file format is valid, the column mapping interface will appear.
4. Assign Expeto-specific parameters to the Mobile Assets. For example:
    - **Customer**: Select the associated Customer.
    - **System**: Choose the target System.
    - **System Profile**: Assign a System Profile for the Mobile Assets.
    - **Keyset**: Link an encryption Keyset for authentication.
5. Generate a 128-bit (32-character) hexadecimal string and paste it into the **K encryption key** field. This key is used to encrypt Mobile Asset authentication values.
6. Select the **Initial Status** for the Mobile Assets:
    - **Active**: Enables network access immediately.
    - **Inactive**: Prevents network access until manually activated.
7. Add any **Extensions** to set custom parameters for all imported Mobile Assets. For example, to add an import timestamp, use:
    - Key: `importDate`
    - Value: `January20`

8. Enable the **Overwrite if Existing** checkbox to replace any existing Mobile Asset with the same IMSI.
    - If this option is disabled and duplicates are found, the import will fail.
9. Click **Submit** to complete the import process. The imported Mobile Assets will appear in the list.

> **Note**
>
> Only the `.out` file format is supported for bulk imports.

> **Note**
>
> The **K encryption key** is critical for generating the Mobile Asset authentication key (Ki).

> **Tip**
>
> Consider creating a "parking lot" System to temporarily store Mobile Assets during bulk imports.

Before importing, the `.out` file undergoes a validation process to ensure proper formatting and headers. Common validation errors include:

- Missing or invalid headers.
- Missing mappings for required fields.
- Incorrect IMSI formats.
- Mismatched column counts.

Required Fields:

- **ICCID**: The Integrated Circuit Card Identifier, engraved on the SIM card.
- **KI**: The Key Identifier, encrypted using the provided **K encryption key**.
- **IMSI**: The International Mobile Subscriber Identity, uniquely identifying each Mobile Asset.

> **Note**
>
> Auxiliary IMSI values are ignored during the import process.

Below is an example of a properly formatted `.out` file:

```
****************************************
*      HEADER DESCRIPTION          *
****************************************
Customer: Unreal Communications
Quantity: 6
Type:     PLUG-IN
Profile:  1.0
Batch:
*
Transport_key:
*
Graph_ref : 2.0

Address1: Unreal Communications
Address2: TBC Address3
Address4: TBC Address5
Artwork:  <artwork file>

****************************************
```

```
*      INPUT VARIABLES            *
*****************************************
Var_In_List:

IMSI1:    123799534412226
IMSI2:    456060048546604
SER_NB:  1924120429575334375

*****************************************
*      OUTPUT VARIABLE            *
*****************************************
var out:ICC_ID/KI/IMSI/IMSI2/OPc/PIN1/PUK1/PIN2/PUK2/ADM
1924120429575334375 '8978B79E7C104F678FA5C336509DB188 123799534412226 456060048546604 6F2E82855DEE7C893CB1F7A72FD08B57 0000 05185749 2847 60978080 76183215
1924120429575334376 '8978B79E7C104F678FA5C336509DB188 123799534412227 456060048546605 6F2E82855DEE7C893CB1F7A72FD08B57 0000 05185749 2847 60978080 76183215
1924120429575334377 '8978B79E7C104F678FA5C336509DB188 123799534412228 456060048546606 6F2E82855DEE7C893CB1F7A72FD08B57 0000 05185749 2847 60978080 76183215
1924120429575334378 '8978B79E7C104F678FA5C336509DB188 123799534412229 456060048546607 6F2E82855DEE7C893CB1F7A72FD08B57 0000 05185749 2847 60978080 76183215
1924120429575334379 '8978B79E7C104F678FA5C336509DB188 123799534412230 456060048546608 6F2E82855DEE7C893CB1F7A72FD08B57 0000 05185749 2847 60978080 76183215
1924120429575334380 '8978B79E7C104F678FA5C336509DB188 123799534412231 456060048546609 6F2E82855DEE7C893CB1F7A72FD08B57 0000 05185749 2847 60978080 76183215
```

BULK UPDATE

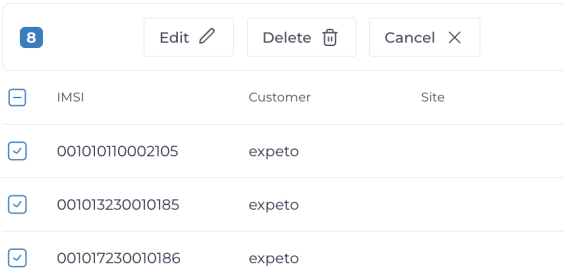
The Bulk Update feature enables administrators to modify existing values or apply new configurations to multiple Mobile Assets simultaneously. This streamlined process allows for efficient management of Mobile Assets with consistent updates.

The following tasks can be performed using the Bulk Update feature:

- **Activate or Deactivate Mobile Assets**: Change the status of selected Mobile Assets.
- **Move Mobile Assets to a New System**: Relocate all selected assets to a single target System.
- **Assign a System Profile**: Apply a System Profile to selected Mobile Assets. System Profiles are System-specific; you must select a System before assigning a System Profile.
- **Add Extensions**: Assign the same custom Extensions to all selected Mobile Assets.

> **Note**
>
> To modify the status of Mobile Assets across different Systems without changing their System assignments, use the **Bulk Update - CSV** feature.

1. Navigate to the **Mobile Assets** page.
2. Select the Mobile Assets to update:
   - Use the checkboxes to select individual Mobile Assets or select all using the "Select All" checkbox.



3. Click **Edit**. The Update page displays the number of Mobile Assets selected for modification.
4. Provide updated values for the selected Mobile Asset parameters. Existing values will be replaced by the new inputs.
5. Click **Save** to confirm and apply the updates. You will be prompted to verify the request before changes are applied.

BULK DELETE

The **Bulk Delete** feature allows you to permanently remove multiple Mobile Assets from the Mobile Asset list. This action is irreversible unless specific System-level permissions are available to restore deleted assets.

> **Warning**
>
> Once deleted, Mobile Assets cannot be recovered without System-level permission. Ensure you have the necessary access and have confirmed the selection before proceeding.

1. Navigate to the **Mobile Assets** page.
2. Select the Mobile Assets you want to delete:
    - Use the checkboxes to select individual Mobile Assets or select all using the "Select All" checkbox.
3. Click **Delete** to initiate the removal process.
4. A confirmation dialog will appear. Click **OK** to confirm and complete the deletion.

> **Note**
>
> Always double-check your selection to avoid accidental deletion of critical Mobile Assets.

EXPORT TO CSV

The **Export to CSV** feature allows you to download a file containing the current configuration of all Mobile Assets for editing or record-keeping purposes.

1. Navigate to the **Mobile Assets** page.
2. Click the **Export** button  .

> **Note**
>
> The exported CSV file contains data for all Mobile Assets in the system. It is not possible to export a file for selected Mobile Assets only.

To make changes using the exported file:

1. Open the downloaded CSV file in a text editor or spreadsheet application.
2. Modify the desired values and save the file with a `.csv` extension.
3. Navigate to the **Mobile Assets** page and click **Bulk Actions**.
4. Select **Bulk CSV Update**.
5. Click **Choose File** and upload the modified `.csv` file. Click **Next** to proceed.
6. The file is verified for formatting errors. If no issues are found, click **Submit** to apply the updates.

When preparing a CSV file for bulk updates, ensure the required columns are correctly formatted and all necessary information is included.

Required Columns

Column	Description
IMSI	The IMSI value cannot be updated. This required value identifies the Mobile Asset and must be unique within the CSV file.
ICCID	The ICCID associated with the IMSI. This cannot be updated. Leave the field empty if no change is needed.
MSISDN	The MSISDN associated with the IMSI. This cannot be updated. Leave the field empty if no change is needed.
STATUS	The status of the Mobile Asset. Valid values are **ACTIVE** or **INACTIVE**.
SYSTEM	The System ID where the Mobile Asset is assigned. Copy the System ID value from the URL of the selected System (e.g., `https://admin.example.com/ngui/sys/**123295600**`). Optionally, add the System name after the ID (e.g., `123295600 default`). The System name must match the ID.
SYSTEMPROFILE	The System Profile ID to assign to the Mobile Asset. Copy the System Profile ID value from the URL of the selected System Profile (e.g., `https://admin.example.com/%7CxControlURLPath%7C/system-profiles/**98144**`).
EXT:	Extensions are key-value pairs used for additional parameters or overrides. Each **EXT:** column represents an extension key, with corresponding values in the rows. Leave the cell empty to remove the extension. Example: For the extension `brand=acme` , use a column header `EXT:brand` and the value `acme` .

Example File Structure

An example of a CSV file structure is shown below:

```
IMSI,             ICCID,                  MSISDN,          STATUS, SYSTEM, PROFILE
001021000000000, 123098123894000000000, 101021000000000, ACTIVE, 709 Demo-6_2, 859 subscriberProfile6_2
001021000000001, 123098123894000000001, 101021000000001, INACTIVE, 708   Demo-6_1, 858

001021000000002, 123098123894000000002, 101021000000002, ACTIVE, , 858
```

To change a parameter value, edit the corresponding cell in the row for the Mobile Asset. Leave the cell blank to set the value to `null` .

Session Details

This document provides a guide for examining session and SMS event logs for Mobile Assets in Expeto xControl. Event logs allow administrators to monitor and analyze the network activity and communication history of Mobile Assets, offering insights into both subscription (control plane) and data (user plane) sessions, as well as SMS messaging details.

Session event logs include timestamped records of subscription and data session events, categorized by types such as HSS (control plane) and PGW (user plane). Administrators can review session states, network operator interactions, and manual changes like airplane mode activations. SMS event logs track the origin, status, and potential failure causes of text messages sent by or to Mobile Assets, with details about the source types and specific error codes for failed delivery attempts. These tools enable effective troubleshooting and network management.

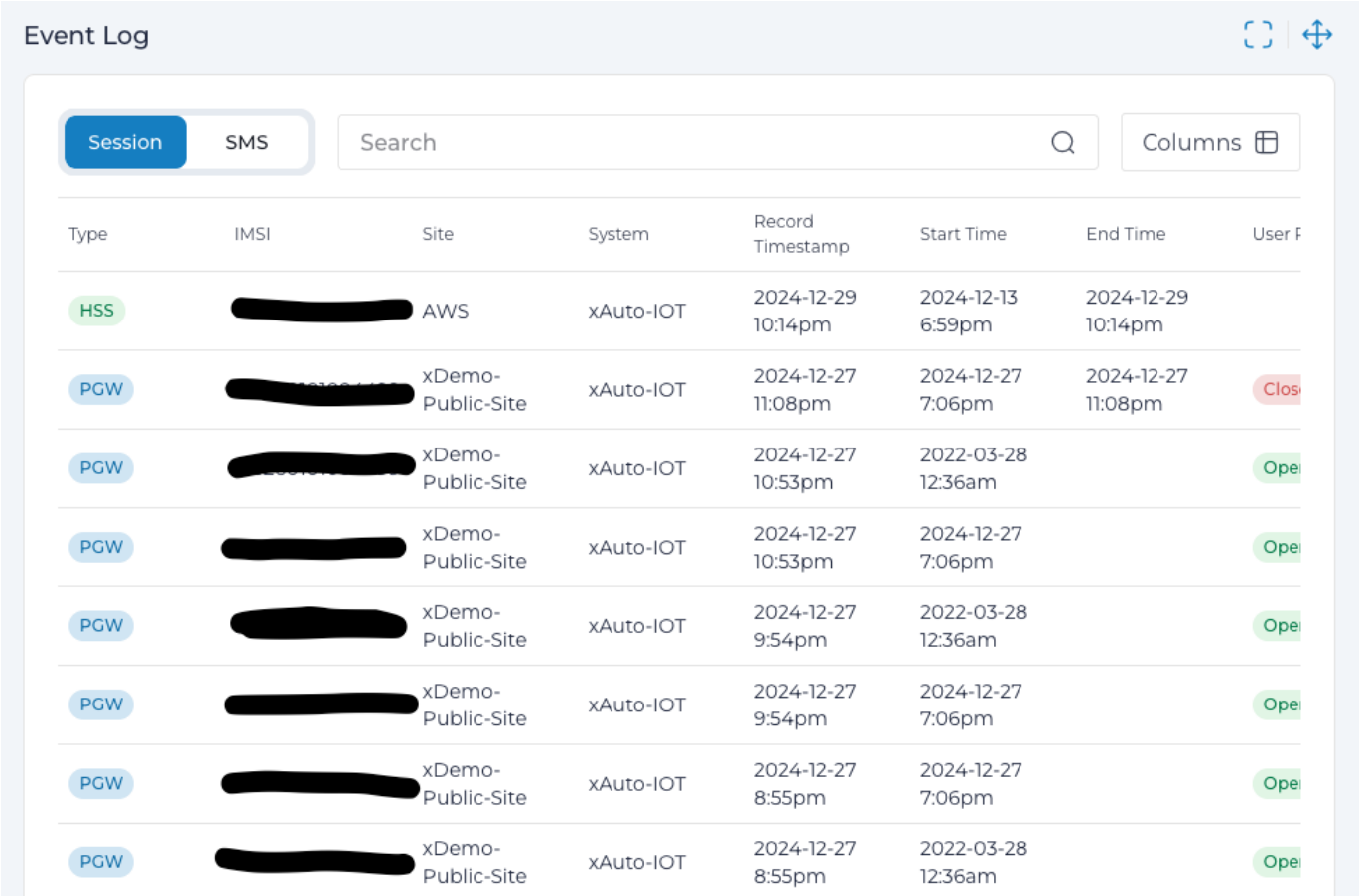
SESSION EVENT LOG

The Event Logs panel on the Mobile Asset page provides details about session activity, including the Mobile Asset's current state, last-seen timestamp, and assigned IP address. For public/private sites, the associated xRouter site (e.g., "AWS") is also displayed.

To access the **Session Event Log** for a Mobile Asset:

1. Navigate to **Mobile Assets**.
2. Click the IMSI of the Mobile Asset. The Mobile Asset details page will open.
3. In the **Event Log** section, click **Session**.

The session history panel displays events in reverse chronological order, including details such as country, network operator, IMSI and IMEI values, system and site names, and more.



To modify the displayed columns:

1. Click the **Column Chooser**  icon.

2. Check or uncheck columns to add or remove them from the view.

> **Note**
>
> Hover over a column name in the list for a brief description of its content.

- **Timestamp**: Indicates when the event occurred, session start time, and if applicable, session end time.
- **Entry Type**: Specifies whether the event is related to the HSS (control plane) or PGW (user plane).

Session events typically result from network interactions, such as keep-alive pings between the network operator and the Mobile Asset. Manual changes, like placing a device in airplane mode, are also captured.

ENTRY TYPE - HSS

HSS (Home Subscriber Server) type entries record the subscription (control plane) session events of a Mobile Asset. These events pertain to the authentication and management of Mobile Assets as they connect to the network. Mobile Assets in a public/private site connecting through a public network are routed to the xRouter site, where authentication is performed against HSS records.

The subscription session can transition through the following states:

State	Description
Attached	The Mobile Asset successfully establishes a connection to the network.
Deleted	The subscription session is removed, indicating the Mobile Asset is no longer attached.
Provisioned	The Mobile Asset's subscription information is configured and ready for use.

When a Mobile Asset (User Equipment or UE) initiates an attach request:

1. The HSS authenticates the Mobile Asset using its subscription data.
2. Upon successful authentication, the default bearer is established.
3. The attach process is completed, and the request is accepted.
4. The Mobile Asset is then able to initiate a data session with the Packet Gateway (PGW).

These events ensure that Mobile Assets are authenticated and managed securely, enabling them to access the network while maintaining control over their session states.

ENTRY TYPE - PGW

PGW (Packet Gateway) entries track data (user plane) session events for Mobile Assets, detailing the state of data sessions.

- **Open**: The data session has started and is currently active.
- **Closed**: The data session has ended and is no longer active.

- A session with a **Start Time** but no **End Time** is still open.
- A subsequent entry may show an **End Time**, indicating the session is closed, or confirm the session remains open.

These entries provide critical insights into data session activity, allowing administrators to monitor and manage connectivity effectively.

EXAMINE SMS LOG EVENTS

To view SMS event log details for a Mobile Asset:

1. Navigate to **Mobile Assets**.
2. Click the IMSI of the desired Mobile Asset to open its details page.
3. In the **Event Log** section, click **SMS** to access the SMS event logs.

SOURCE TYPES

Source types indicate the origination device or application for an SMS message. If the source is **MO** or **MAP**, the message is classified as outgoing. The table below explains the various source types:

Source Type	Description
MO	Mobile Originator.
CLI	Command Line.
MSC	Mobile Switching Center (the SMS Gateway).
SMPP	SMPP (Short Message Peer-to-Peer) Gateway.
LOCAL	Local device or application.
REMOTE	External device or application.
MAP	Mobile Application Part Protocol, such as MAP V4. Used to query HLR/HSS for customer location.
SOAP	SOAP client.
UNKNOWN	Device or application type is not known.

Failure causes indicate the result of the last attempt to deliver an SMS message. The table below lists common failure causes and their meanings:

Failure Cause	Description
None	No attempt was made to deliver the message.
NoMSCResponse	No response from the Mobile Switching Center.
NoSuchDest	Target cannot be identified.
NoUserResponse	No response from the recipient.
NoMemoryOnHandset	Recipient’s handset lacks sufficient memory.
Success	Message delivered successfully.
CommsFail	Communication failure occurred.
NetError	Network error encountered.
SMSCCongestion	SMS Center is congested.
SMPPDisconnect	Disconnection from the SMPP Gateway.
NoSuchSMPPUser	SMPP user cannot be identified.
Rejected	Message delivery was rejected.
Discarded	Message was discarded without delivery.
AbsentSub	Subscriber is not reachable.
BusyForMTSMS	Subscriber is busy for MT SMS.
IllegalSub	Subscriber credentials are invalid or illegal.
IllegalEquipment	Equipment is invalid or unauthorized.
ProtError	Protocol error encountered.
EquipNotSupp	Equipment does not support the message type.
DataError	Data-related error occurred.
Barring	Text message blocking is enabled.
NotProv	Authentication credentials not provided.
InvalidMsgRef	Invalid message reference.
IncorrectMsg	Message format or content is incorrect.
InvalidMandatoryInfo	Required information is missing or invalid.
InvalidMsgType	Message type is not valid.
NotCompatible	Device or application is not compatible.
IEError	Information Element error occurred.
TempFailure	Temporary failure, retry might succeed.
ProviderError	Error from the service provider.
SMDelFail	Short Message delivery failed.
AbsentSubSMPP	Subscriber is unreachable via SMPP.
UnidentifiedDest	Destination is not identified.
NoSuchDestOffNet	Destination is not found in off-net routing.

2.3.3 System Profiles

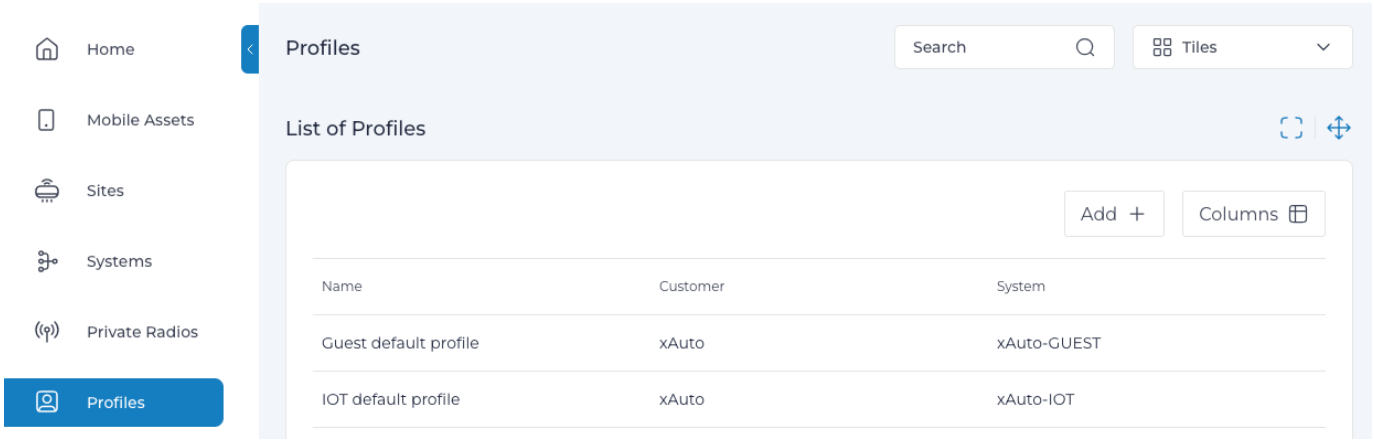
System Profiles

System Profiles are primarily used for selecting mobile asset (subscriber) service (APN) and their service quality such as bandwidth. A System Profile includes an order list of applicable APN profiles and additional extension values for setting custom parameters. Each APN profile in the System Profile, includes a reference to APN object and maximum download and upload bit rates values. The maximum download and upload values in the APN profiles overrides the values configured in the APN object for the System Profile. Otherwise, maximum download and upload vales from the APN object is used by the System Profile.

Note

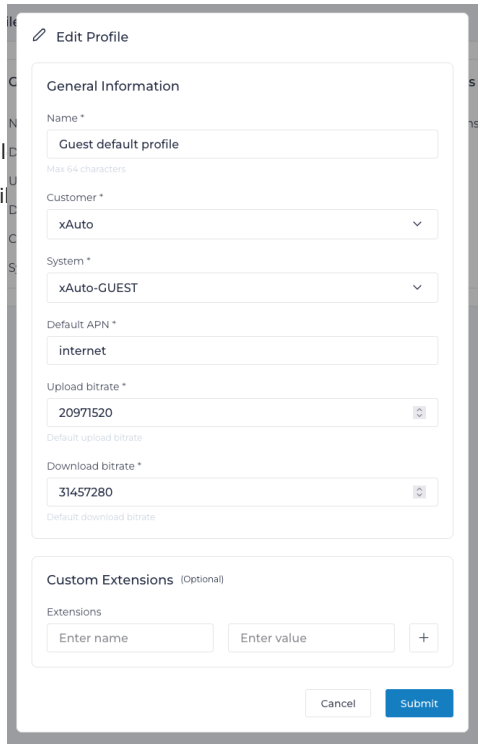
They can be applied only to Mobile Assets belonging to the Systems specified in the System Profile.

Multiple System Profiles can be created and made available for Mobile Assets of the same System. For example, create tiered service levels by creating three Profiles (bronze, silver, and gold) each with successively higher bandwidth levels. Different APN profile is applied depending on the APN used by the mobile asset to connect the network.



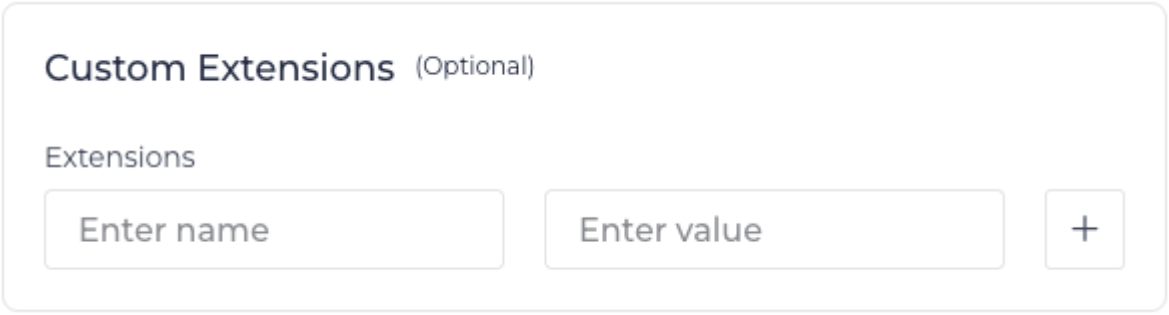
Add or Edit a System Profile

1. Navigate to System Profiles.
2. Click **Add** or click an existing System Profile **Name** to edit.
3. Type a Name of the Profile. For example: *Gold*
4. Click **Customer** and select a Customer from the list. The Profile is only available for the sel
5. Click **System** and select one or more System from the list. Once selected, the System Profi this System.
6. Click **Submit**.



Set Extensions for Profiles

In the Edit Profile dialog box, you can add custom extensions.



Note

Extensions set in the System Profile override the same Extensions set at the Customer and System level.

Info

For more information on creating Extensions, see Extensions page.

Add APN Profiles for the System Profiles

1. Select **APN Profiles** tab.

2. Click **Add APN Profile**.
3. Select **APN** from the list.
4. (Optional) Enter **Upload Bitrate** and **Download Bitrate** override values.
5. (Optional) Select **QoS** from the list.
6. (Optional) **Carrier APN Overrides**: You can define carrier-specific QoS and bitrate overrides.
    - Click **Add Carrier Override**.
    - Select a **Carrier**.
    - Specify carrier-specific **Download**, **Upload**, or **QoS** values, all are optional but at least one is required.
7. Click **Submit**.

> **Note**
>
> Carrier APN Overrides take precedence over both the APN Profile values and the APN Global values for subscribers roaming on the specified carrier.

UNDERSTANDING QOS/QCI AND CARRIER OVERRIDES

In 3GPP mobile networks, the **Quality of Service (QoS)** and specifically the **QoS Class Identifier (QCI)** determine how data traffic is prioritized and treated by the radio network. QCI is a scalar value that defines specific packet delay budgets, packet error loss rates, and priority levels.

The 3GPP specifications define two categories of QCI values: * **Standardized QCI Values:** These values (typically 1–9 and certain others like 65, 66, 69, 70) are defined globally by 3GPP. Every network operator following the spec should treat these values with the same characteristics (e.g., QCI 1 is always for conversational voice with a 100ms delay budget). * **Operator-Specific QCI Values:** Operators are permitted to define their own custom QCI values (typically in the range of 128–254). These values allow for proprietary QoS levels tailored to specific enterprise needs or unique applications.

The Importance of Carrier Overrides for Custom QCIs When roaming, using operator-specific QCIs becomes a challenge. A custom QCI value defined on your home network (HPLMN) will likely not be recognized by a roaming partner's network (VPLMN). If a device attempts to attach using a custom QCI that the VPLMN doesn't support, the session may be downgraded to a default best-effort level or, in some cases, the attach request may be rejected entirely.

By using **Carrier APN Overrides**, you can ensure that your devices always request a safe, standardized QCI value (like QCI 9) when roaming, while still utilizing your high-performance custom QCIs while on your home network.

Similarly, the **Aggregate Maximum Bit Rate (APN-AMBR)** dictates the maximum upload and download bandwidth allowed across all non-guaranteed bearers for that APN.

Why Override per Carrier? When a mobile asset leaves your home network and roams onto a partner's network (Visited PLMN / VPLMN), the network must agree on the QoS parameters. Different roaming partners often have different infrastructure capabilities, Service Level Agreements (SLAs), or pricing models.

Configuring **Carrier APN Overrides** allows administrators to dynamically adjust the requested QCI and bitrates depending on the network the device attaches to. This is crucial for: - **Cost Control:** Throttling download/upload bitrates (AMBR) when assets roam onto expensive partner networks. - **SLA Compliance:** Downgrading a high-priority QCI to a standard best-effort QCI if the roaming partner does not support or heavily charges for premium QoS classes. - **Compatibility:** Ensuring the attach request isn't rejected by a Visited PLMN that doesn't recognize a specific custom QCI value configured on your home network.

2.3.4 Sites

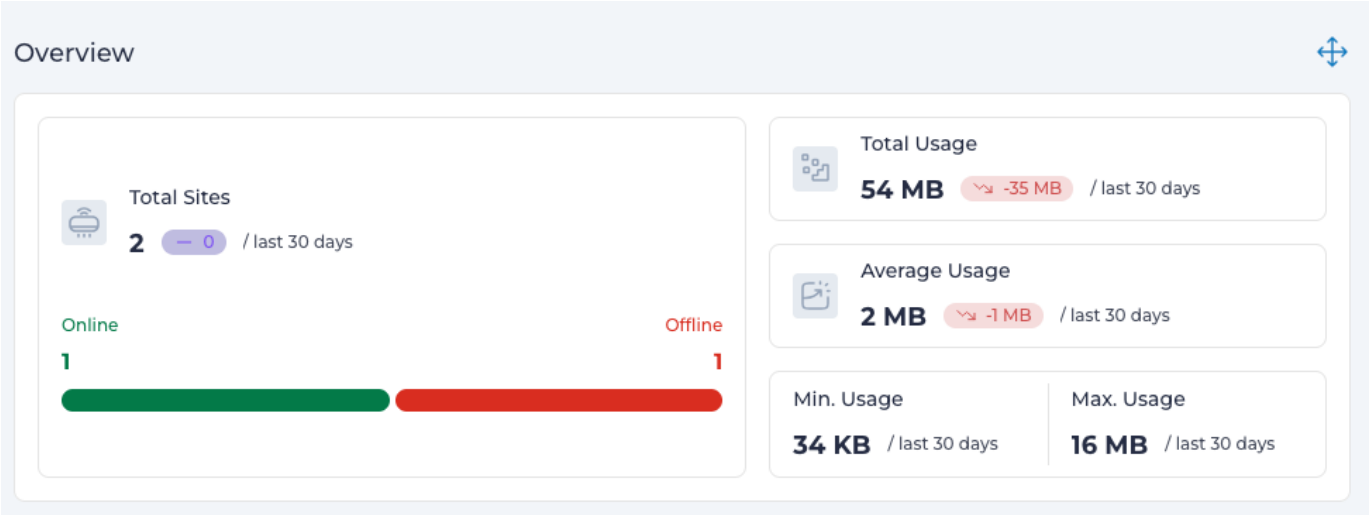
Sites

A **Site** represents a deployed instance of the Expeto xCore, which can be installed either on-premise or as a virtual instance. Sites are the foundation for managing networked systems and devices, providing centralized control, monitoring, and configuration of Expeto-enabled private mobile networks. Each Site serves as a gateway for managing Systems, Mobile Assets, and network policies.

> **Note**
>
> For SaaS-hosted xControl instances, contact Expeto Support to add or remove Sites.

Site Status

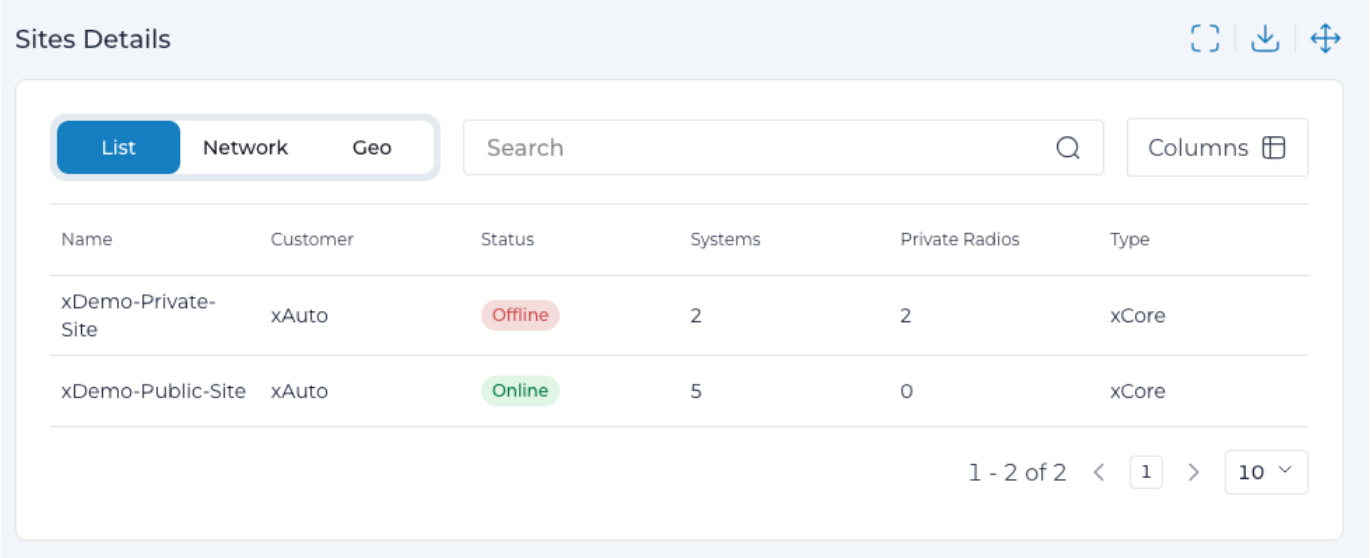
Navigate to the **Sites** section to view a list of all Sites available in your network. Each Site's status is displayed as either **Connected** or **Disconnected**:



- A Site is marked as **Offline** if it has not been detected by network agents for 5 minutes.
- Additional information available for each Site includes:
  - **Network Topology**: The network configuration of connected assets.
  - **Data Metrics**: Bandwidth usage statistics (total, average, min, max)

Site Details

Click the name of a Site in the **Sites** list to view detailed information about the Site. The following parameters are displayed:



Column Name	Description
Name	Identifies the Site in the GUI. Maximum 64 characters.
Location	Optional geographic information about the Site.
UUID	An autogenerated unique identifier for the Site in hexadecimal format.
Agent Version	The version of the Expeto Agent running on the Site (e.g., 1.1.6730).
IP Address	The IP address of a **Connected** Site.
Last Seen	The timestamp indicating when the Site was last connected.
Extensions	Custom key-value pair parameters configured for the Site.

Additional options available:

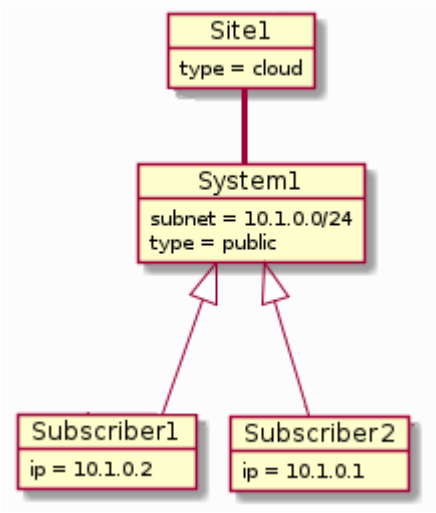
Overview Metrics, Network Topology, Geographical Location (if configured), Associated Systems, Associated Private Radios, Data Usage History.

Common Site Scenarios

Sites can be configured in various ways depending on network requirements. The following scenarios illustrate typical single-Site and dual-Site installations, highlighting their features and potential use cases.

SINGLE-SITE SCENARIO

The graphic below demonstrates a single-Site deployment with one System:

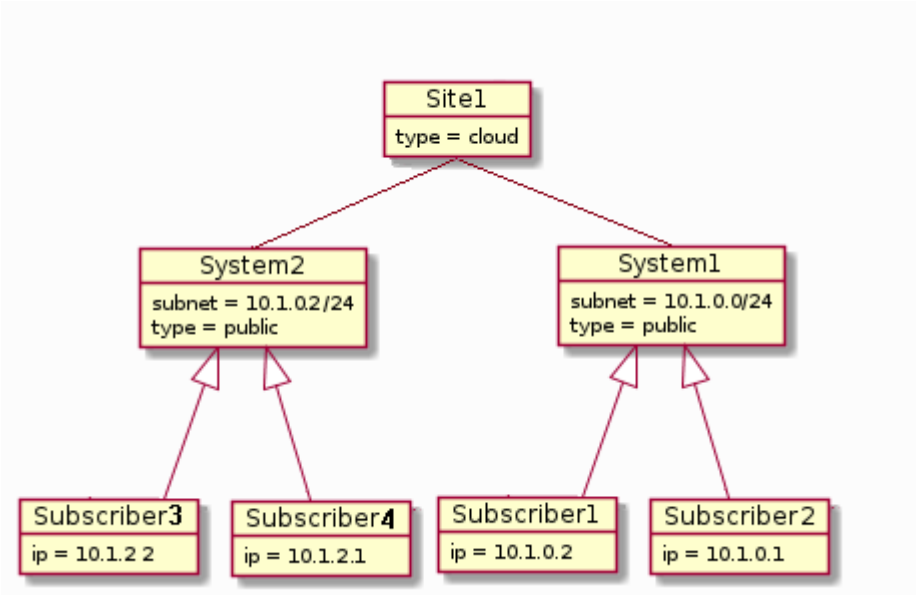


Key Features:

- Each Subscriber can be assigned a static IP address from the System's subnet.
- All Subscribers access the same network, but individual network policies can be applied.

> **Example**
>
> One Subscriber could be assigned a System Profile with higher data throughput.

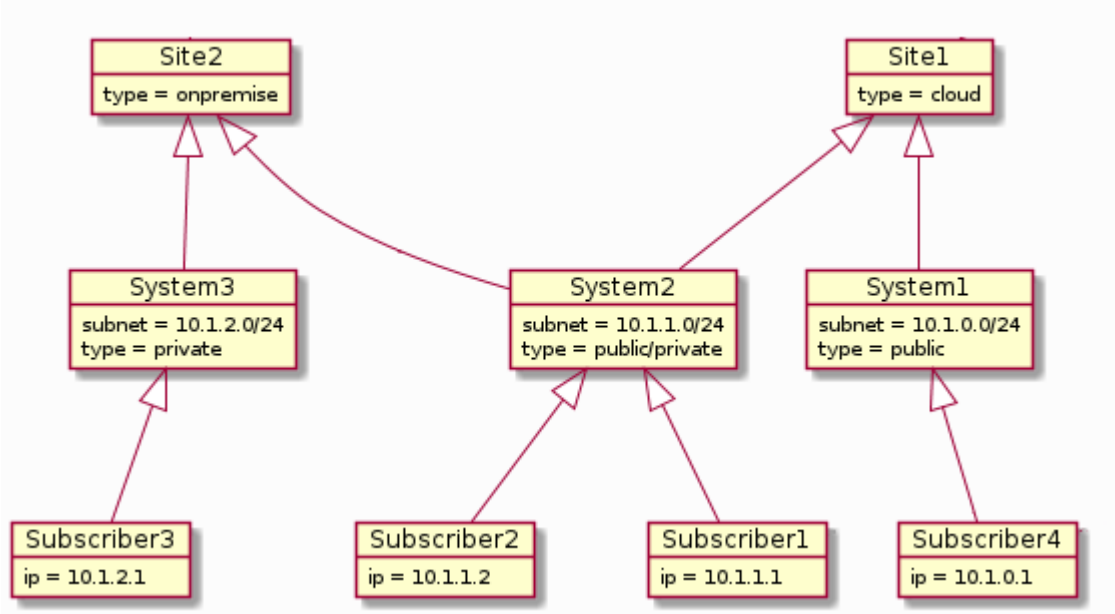
Adding an additional System within the same Site enables **network segmentation**, as shown in the diagram below:



- **Network Segmentation**: Allows restrictions on Subscribers to specific parts of the network.
- Policies can now be enforced at the System level, providing more granular control.

DUAL-SITE SCENARIO

A dual-Site setup enables segmentation of a network across two locations. The diagram below demonstrates this configuration, featuring one shared System and two Site-specific Systems:



Key Characteristics:

- One Site is **on-premise**, while the other is **virtual**.
- Each System is assigned a **unique subnet** for clear traffic separation.
- Subscribers assigned to the **shared System (System2)** can roam seamlessly between the two Sites.
- Subscribers assigned to **System1** and **System3** are restricted to **Site1** and **Site2**, respectively.

Example Use Case: Industrial Manufacturing

This dual-Site setup is ideal for industrial environments with distinct operational needs:

1. **Site1 (Cloud-based Instance)**:
   - Hosts an IT Operations network (**System1**) for office computers and printers connected via the Internet.
2. **Site2 (On-premise Instance)**:
   - Contains a dedicated SCADA network (**System3**) installed behind a corporate firewall.
   - Operates as a private, low-trust or zero-trust network with no Internet access.
   - Robotic devices within this network are assigned static IP addresses.

Employees requiring access to both the SCADA network and the IT Operations network have their devices registered with the **shared System (System2)**, enabling seamless communication across both Sites.

> **Note**
>
> The two Sites in this scenario could also be both on-premise, deployed in different geographic locations.

2.3.5 Systems

Systems

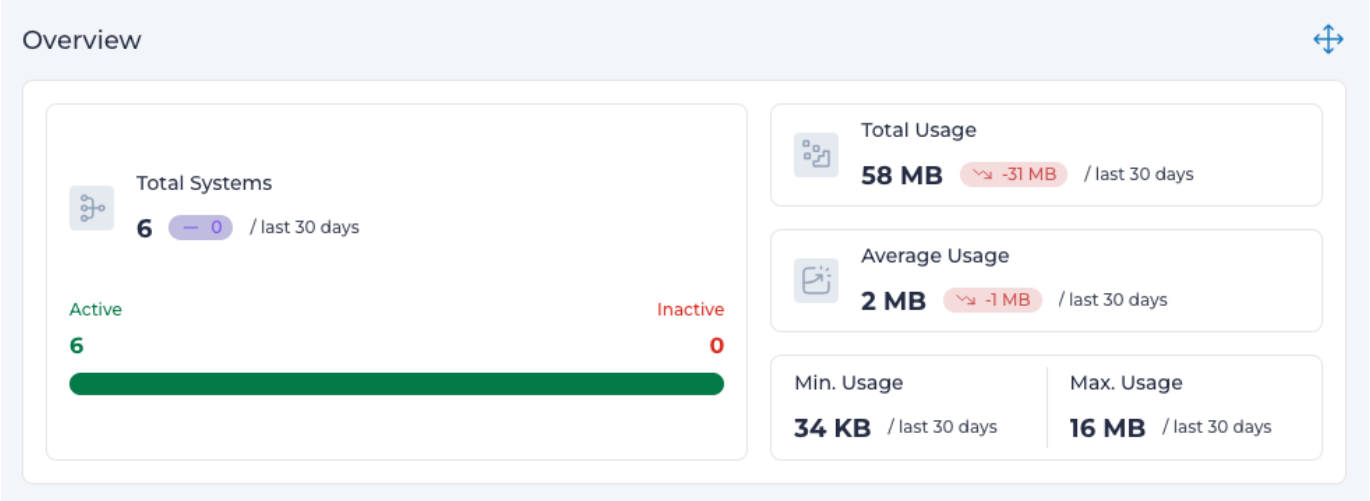
A **System** maps to a subnet in your network, creating logical divisions and ensuring secure separation for different types of Subscribers on a corporate network.

For instance, you can define one System for a subnet accessed by IoT devices and another for an employee subnet. Similarly, you might configure separate Systems for production and test environments.

Systems created in Expeto xControl are deployed as

- **Packet Gateway (PGW) instances** in 4G/LTE networks.
- **User Plane Function (UPF) instances** in 5G networks.

These Systems interface with local IT networks, enabling efficient and organized management of network resources.



Create a System

Follow these steps to create a new System:

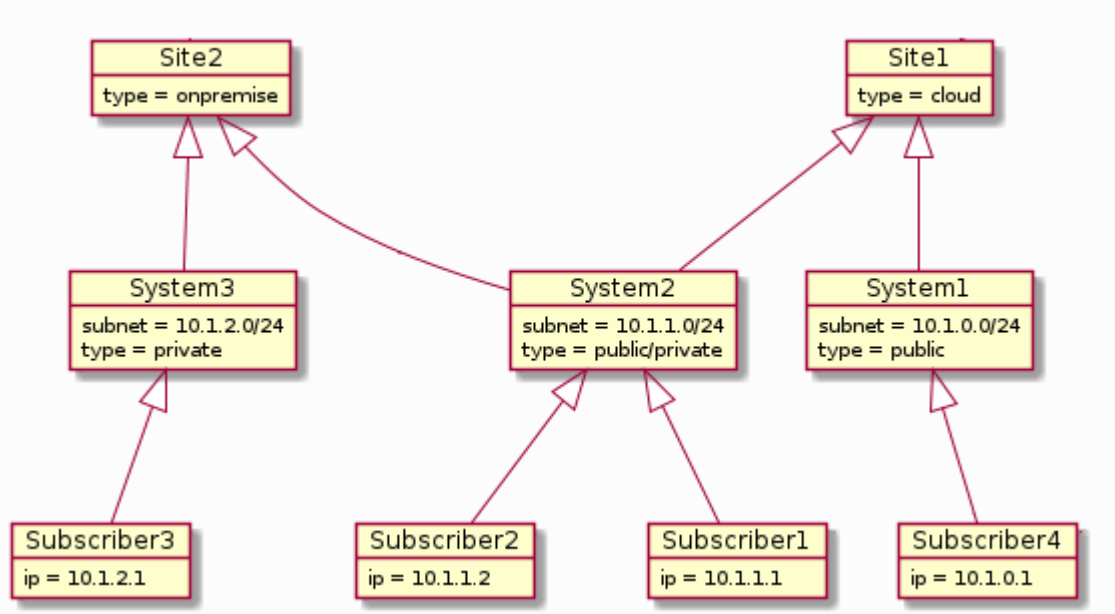
1. Navigate to the **Systems** section in Expeto xControl.
2. Enter a unique **Name** for the System.
    - Maximum 64 characters, no special characters or spaces are allowed.
3. Select **Enable** or **Disable** to activate or deactivate the System.
4. Click **Customer** and choose a Customer from the list.
    - The System will only be available for the selected Customer.
5. Click **Site(s)** and select one or more applicable Sites.
6. Enter the **Network** CIDR range for the System (e.g., `100.64.9.0/24`).
    - Each CIDR block must be unique and not conflict with other addresses in the broader network.
    - Multiple CIDR blocks can be assigned to the same System if needed.
7. Click **Submit** to save the new System.

> **Note**
>
> Your installation may include pre-configured Systems with assigned CIDR ranges. Permissions to create new Systems and assign CIDR ranges may not be granted, depending on your role.

> **Note**
>
> If your corporate network Gateway is using Network Address Translation (NAT), you can create two systems with overlapping ranges for the same network.

Create a Shared System

A **Shared System** enables seamless access for Subscribers across multiple Sites owned by the same Customer, even if the Sites are in different physical locations. This setup is ideal for ensuring consistent connectivity and unified management of network resources.



1. Navigate to the **Systems** section in Expeto xControl.
2. Choose one of the following options:
3. Click **New** to create a new System.
4. Select an existing System and click **Edit**.
5. In the **Sites** field, select one available Site.
6. Add additional Sites by selecting them from the list.
7. Multiple Sites will be separated by commas in the configuration.
8. Click **Submit** to save the Shared System.

> **Note**
>
> Shared Systems are only available for Sites under the same Customer. Ensure that all associated Sites belong to the same Customer before configuring the Shared System.

Move Subscribers Between Systems

A Subscriber belongs to one System only, but can be moved between Systems by editing the Subscriber properties.

Set Extensions

Extensions set at a System level are applied to all Subscribers belonging to the System unless explicitly overridden.

2.3.6 Configure APNs

APNs

Define an **Access Point Name (APN)** for a customer network to configure its Quality of Service (QoS), including default maximum upload and download bitrates.

In a 3GPP mobile network, an APN acts as the gateway between the cellular network and the public or private data network (e.g., the Internet, an enterprise intranet). The APN defines critical network parameters: * **Routing:** How and where the mobile asset's data traffic is routed once it leaves the radio access network. * **Quality of Service (QoS):** The priority and packet treatment of the data stream. * **Bandwidth Allocation (AMBR):** The default aggregate maximum upload and download bitrates allowed for the connection.

These base APNs (often referred to as *APN Globals*) are then incorporated into **System Profiles**, where you can further define tiered service levels, override default bandwidth limitations, or specify carrier-specific QoS overrides when a mobile asset roams onto another network.

APNS VS. ROAMING PROFILES

While APNs govern *how* a mobile asset connects to the network (its gateway, QoS, and bandwidth), **Roaming Profiles** govern *where* the asset is physically allowed to connect. Use Roaming Profiles to enforce steering of roaming policies and restrict the actual Carriers (PLMNs) the device can use.

> **Note**
>
> The **Create APN** permission is required to perform these actions.

Create or Edit an APN

1. Navigate to the **APNs** section in the admin interface.
2. Click **Add** to create a new APN or click an existing APN to edit it.
3. Type a **Name** for the APN (e.g., *internet*, *m2m.private*).
4. Click **Customer** and select the appropriate Customer from the list. The APN is only available for the selected Customer and its associated System Profiles.
5. Select the baseline **QoS** profile from the available list.
6. Specify the base **Upload Bitrate** and **Download Bitrate** in bits per second. These fields define the default Aggregate Maximum Bit Rate (AMBR) for data transfer.
7. Click **Submit**.

> **Overriding APN Defaults**
>
> The bitrates and QoS values set here are considered the baseline limits. You can override these limits on a per-profile or per-carrier basis when building out your **System Profiles**.

2.3.7 Carriers

Overview

The **Carriers** section provides tools for managing individual mobile network carriers and grouping them into **Carrier Groups**. This functionality is essential for configuring 3GPP roaming restrictions through Roaming Profiles, allowing administrators to precisely control which networks their mobile assets can attach to globally.

Key features include:

- **Carrier List**: Displays a comprehensive database of Public Mobile Network Carriers, including their Mobile Country Code (MCC), Mobile Network Code (MNC), and Public Land Mobile Network (PLMN) identifiers.
- **Carrier Groups**: Allows administrators to create logical groupings of carriers (e.g., "North American Partners", "High-Bandwidth Providers").
- **Integration with Roaming Profiles**: Carrier Groups form the foundational building blocks for Roaming Profiles, defining the allowed networks for subscribers.

Manage Carriers

The system maintains a database of mobile network carriers. While many carriers are pre-populated, you may need to view their details or add specific private or regional carriers.

CARRIER ATTRIBUTES

Each carrier is defined by several key attributes: - **MCC**: Mobile Country Code (e.g., 310 for US). - **MNC**: Mobile Network Code. - **PLMN**: The combination of MCC and MNC (e.g., 310410). - **Name/Brand**: The commercial name of the operator. - **Status**: The operational status of the network.

VIEWING CARRIERS

1. Navigate to the **Carriers** section in the admin interface.
2. The list displays all available carriers. You can use the search functionality to find specific carriers by name, country, or PLMN.
3. Click on a specific carrier to view its full details, including supported bands and operational mode.

Manage Carrier Groups

Carrier Groups are custom collections of carriers. Rather than assigning individual carriers to a Roaming Profile, you create Carrier Groups to manage roaming policies more efficiently.

CREATE A CARRIER GROUP

1. Navigate to the **Carrier Groups** section.
2. Click the **Add** button to create a new group.
3. In the dialog, provide a **Name** (e.g., "EU Roaming Partners") and an optional **Description**.
4. Navigate to the **Carriers** tab within the group configuration.
5. Select the specific carriers you want to include in this group. You can search by country or operator name to find them quickly.
6. Click **Submit** to save the Carrier Group.

EDIT A CARRIER GROUP

1. From the **Carrier Groups** list, click on the name of the group you wish to edit.
2. You can update the name, description, or modify the list of included carriers by checking or unchecking them in the **Carriers** list.
3. Click **Submit** to save your changes.

> **Warning**
>
> Modifying a Carrier Group will immediately affect all Roaming Profiles that use it, which in turn impacts the roaming capabilities of all subscribers assigned to those profiles.

Using Carrier Groups in Roaming Profiles

Once you have defined your Carrier Groups, they are ready to be used in **Roaming Profiles**.

A Roaming Profile acts as a policy that specifies which Carrier Groups are allowed. For detailed instructions on creating and applying these profiles to Systems, Sites, or individual Subscribers, please see the Configure Roaming Profiles guide.

2.3.8 Configure Roaming Profiles

========== Roaming Profiles

Roaming Profiles provide 3GPP roaming restrictions by defining which Carrier Groups a subscriber is allowed to attach to. They can be applied at the System, Site, or Subscriber level.

UNDERSTANDING 3GPP ROAMING RESTRICTIONS

In 3GPP mobile networks, roaming occurs when a mobile asset leaves the coverage area of its Home Public Land Mobile Network (HPLMN) and attempts to attach to a Visited Public Land Mobile Network (VPLMN). Without explicit restrictions, a device might connect to any available VPLMN that has a roaming agreement with the home network.

Roaming Profiles allow administrators to enforce network selection policies, effectively steering roaming traffic by strictly defining the physical networks (Carriers/PLMNs) a device is permitted to use globally. This granular control is essential for: - **Cost Management:** Blocking expensive roaming partners and forcing devices to connect only to partners with favorable data rates. - **Security & Compliance:** Restricting attachments to trusted national carriers, preventing data from traversing foreign networks, and satisfying data sovereignty requirements. - **Service Quality:** Ensuring devices attach to networks that support specific features (like 5G SA) or offer superior SLAs and coverage, rather than simply connecting to the first available signal.

Carrier Groups

A Carrier Group is a collection of Carriers used within Roaming Profiles to define sets of allowed or restricted networks. Before creating a Roaming Profile, you must define the Carrier Groups that represent the sets of networks you want to manage.

For detailed instructions on creating and managing these groups, see Manage Carriers and Carrier Groups.

Roaming Profiles

Once Carrier Groups are defined, you can create Roaming Profiles.

1. Navigate to **Roaming Profiles**.
2. Click **Add** or select an existing profile to edit.
3. Type a **Name** and **Description**.
4. Select the **Customer** this profile belongs to.
5. In the **Allowed Carrier Groups** tab, select one or more Carrier Groups that subscribers using this profile are permitted to roam on.
6. Click **Submit**.

Applying Roaming Profiles

Roaming Profiles can be applied at different levels of the hierarchy. If a Roaming Profile is set at multiple levels, the most specific one (Subscriber) takes precedence.

APPLY TO A SYSTEM

1. Navigate to **Systems**.
2. Edit a System and select the desired **Roaming Profile**.
3. Click **Submit**.

APPLY TO A SITE

1. Navigate to **Sites**.
2. Edit a Site and select the desired **Roaming Profile**.
3. Click **Submit**.

APPLY TO A SUBSCRIBER

1. Navigate to **Subscribers**.
2. Edit a Subscriber and select the desired **Roaming Profile**.
3. Click **Submit**.

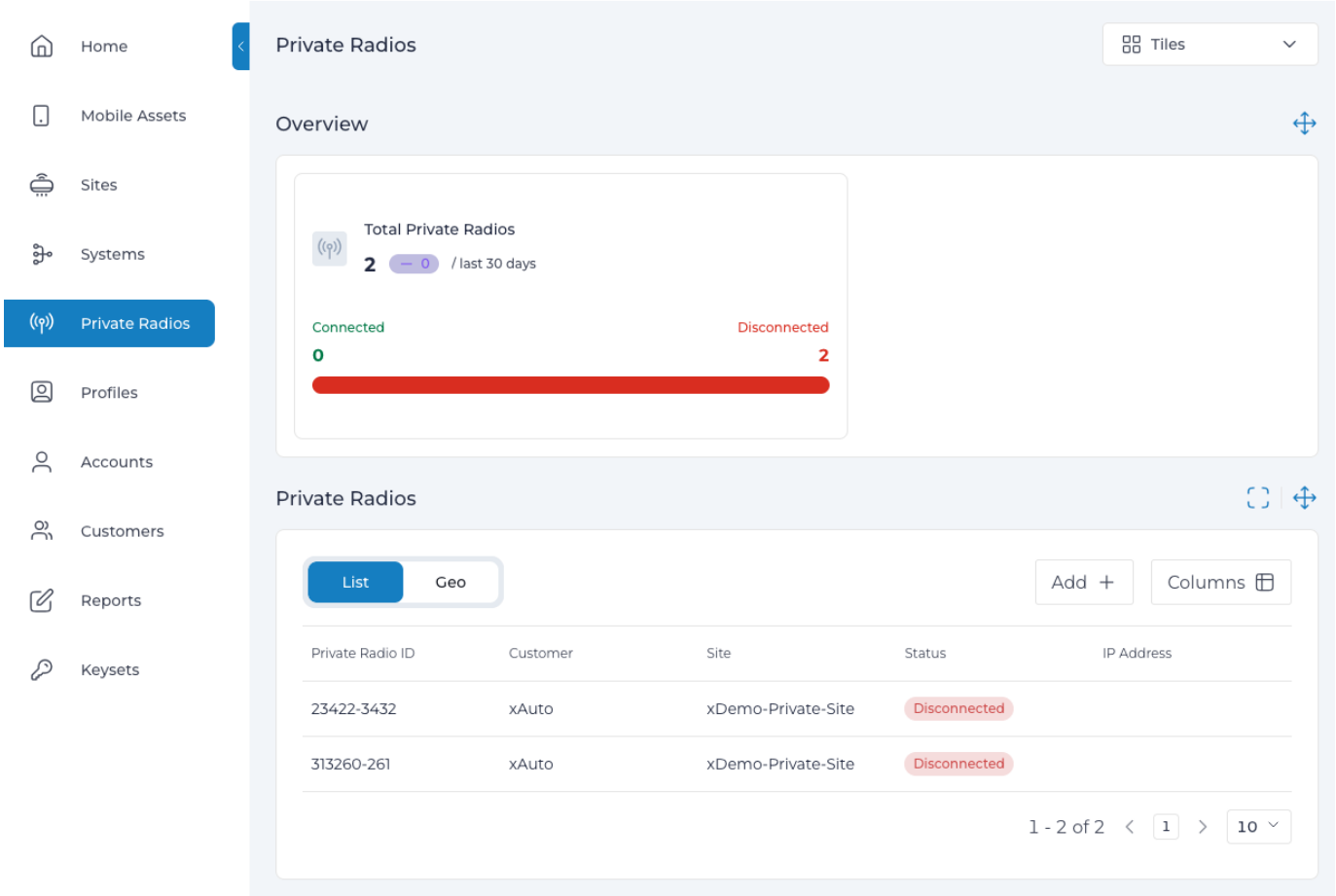
> **Note**
>
> When a Roaming Profile is assigned to a System or Site, all Subscribers within that System or Site will inherit that profile unless they have a specific Roaming Profile assigned to them directly.

2.3.9 Private Radios

Private Radios

The **Manage Private Radios** page provides tools for administering optional eNodeB/gNodeB radios within the Expeto xControl system. These radios can be added, enabled, disabled, or removed and are used to facilitate communication between 4G/5G devices and the Expeto xCore if they are part of the network configuration.

This page allows administrators to configure new Private Radios with unique identifiers, manage their operational states (enable/disable), and remove radios as needed. Real-time status updates indicate whether a Private Radio is connected, disconnected, enabled, or disabled, providing a streamlined way to manage this optional component of the network infrastructure.

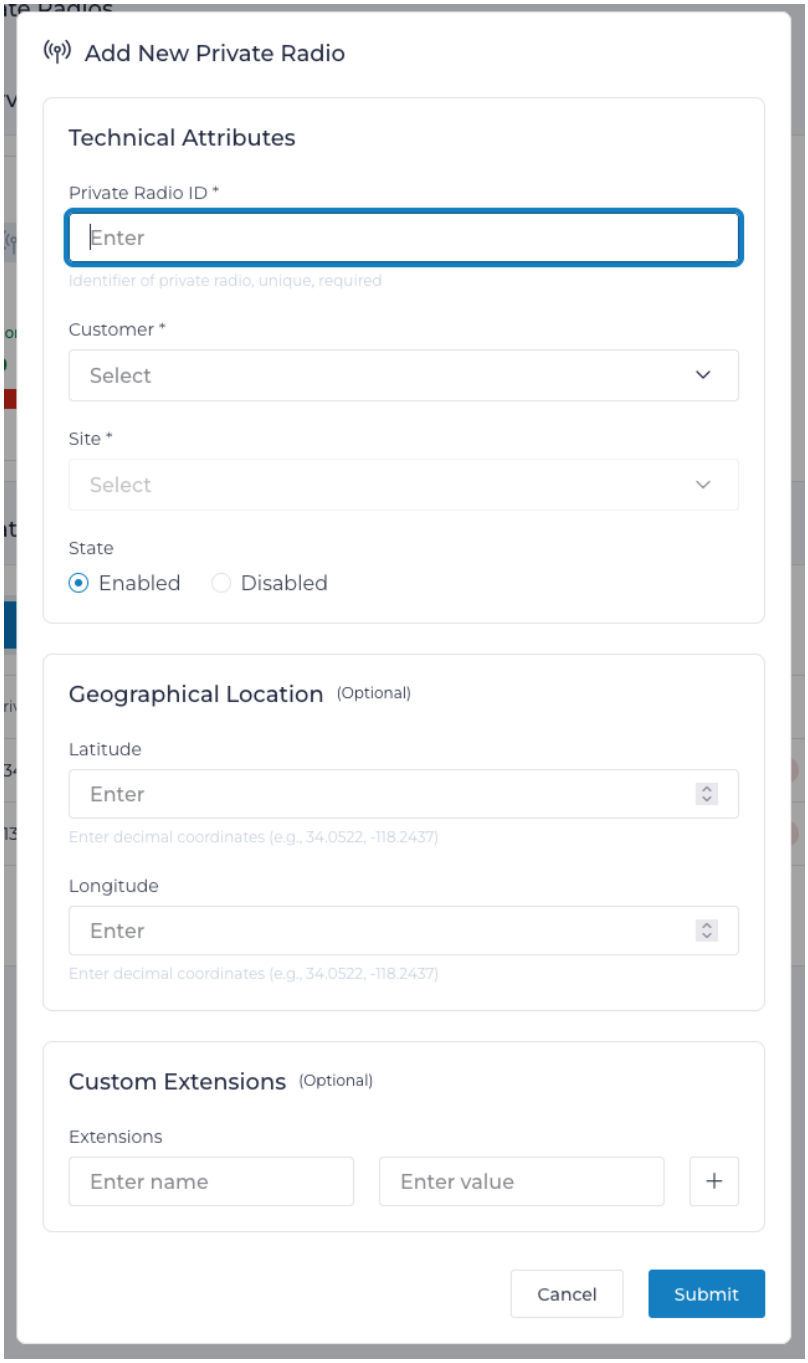


Private Radio Status

The following statuses describe the operational state of a Private Radio:

Status	Description
Enabled	The Private Radio configuration has been added to the Site and is waiting to connect with the physical unit.
Disabled	The Private Radio is configured in the System but will not accept connections.
Connected	The connection between the Expeto xCore and the Private Radio is established.
Disconnected	The connection between the Expeto xCore and the Private Radio is broken.

Add a New Radio



To add a new Private Radio, you must provide a unique identifier in the format:

`<PLMN>-<Cell ID>`

- **PLMN**: The Public Land Mobile Identifier, which consists of:
  - **MCC (Mobile Country Code)**: A 3-digit code representing the country.
  - **MNC (Mobile Network Code)**: A 2-3 digit code identifying the network carrier.
- **Cell ID**: A 1-6 digit identifier for the specific cell tower.

Example

The identifier `54206-123` breaks down as follows:

- **542**: MCC for Fiji.
- **06**: MNC for the fictional FijiFone Carrier.
- **123**: Cell tower identifier.

STEPS TO ADD A PRIVATE RADIO

1. Navigate to **Private Radios** in the Expeto xControl menu.
2. Click **Add** to open the New eNodeB page.
3. Enter the **eNodeB/gNodeB identifier** in the required format.
4. Select the **Customer** associated with the Private Radio.
5. Choose the relevant **Site** where the Private Radio will be deployed.
6. Set the initial **Status** to **Enabled** or **Disabled**.
7. Optionally, provide the **Longitude** and **Latitude** of the Private Radio's geographical location.
8. Click **Submit** to save the new Private Radio configuration.

Enable or Disable a Radio

1. Click the eNodeB ID. The details of the eNodeB appear.
2. Click **Edit**.
3. Select State **Enable** or **Disable**.
4. Click **Submit**.

Remove a Radio

---

1. Click the eNodeB ID. The details of the eNodeB appear.
2. Click **Delete**.

2.3.10 Reports

> **Note**
>
> The appropriate Generate Reports permission is required to access this functionality.

Overview

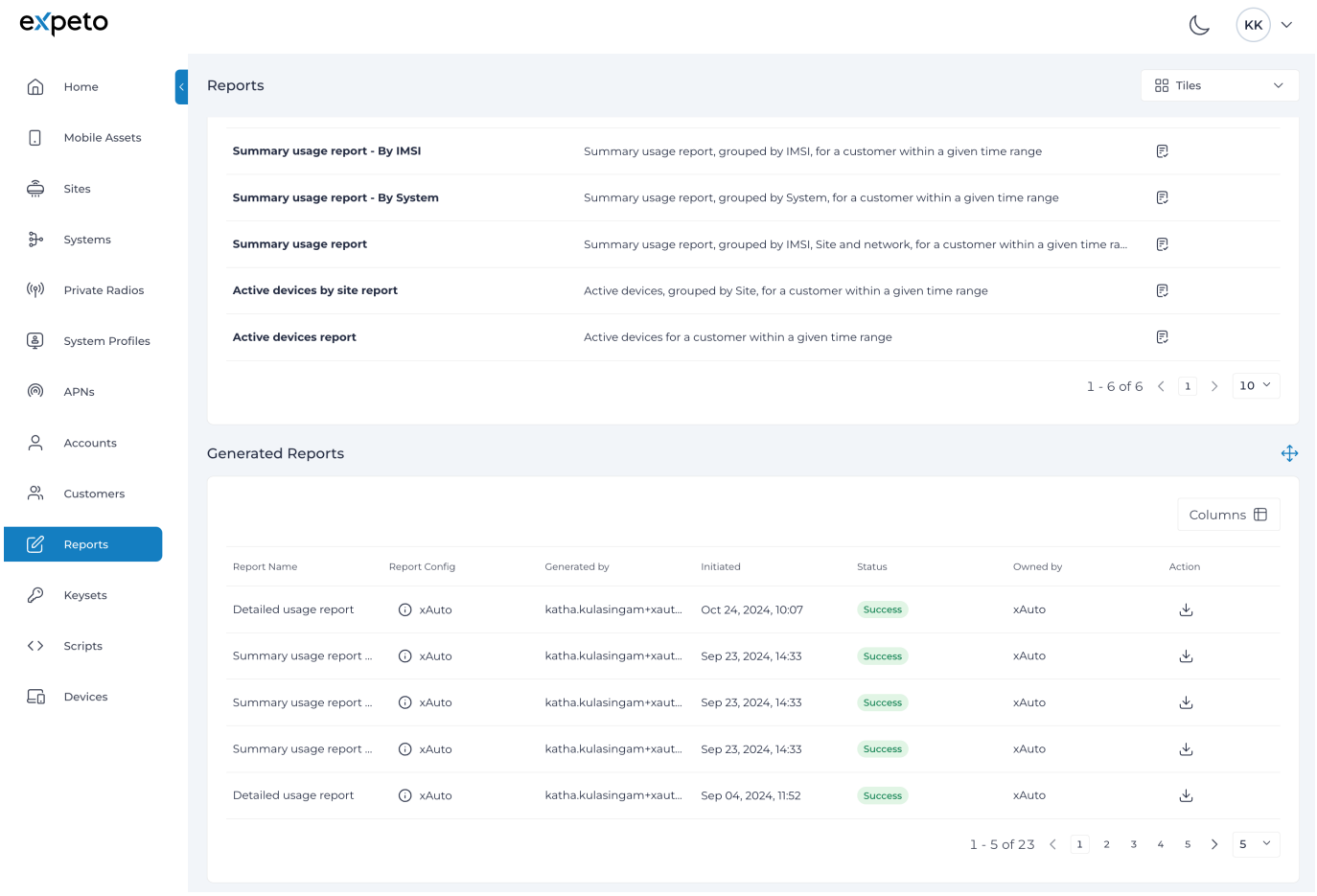
The Reports feature in xControl allows administrators to generate various types of Call Detail Reports (CDRs) on-demand for billing, auditing, and informational purposes. Reports can be customized for a specific Customer and date range, capturing critical insights into data usage, network activity, and subscriber details.

Administrators can access generated reports in the following ways:

- Download directly from **Files** within the xControl interface.
- Retrieve via a GET call to an API endpoint.
- Access through a previously configured SFTP connection.

The scope and content of each report depend on the selected report type and generation parameters. For example, reports can detail data usage per Subscriber in bytes or aggregate usage by Site.

This robust reporting system supports flexible access and granular customization, enabling organizations to monitor their network operations effectively and optimize resource allocation.



Generating a Report

To generate a report:

1. Navigate to the **Reports** section in xControl.
    - The **Available Reports** panel lists predefined report types with descriptions.
    - The **Generated Reports** panel shows previously created reports.
2. Locate the desired report type in the **Available Reports** panel and click the **Generate** icon.
3. Fill in the required parameters (e.g., date range, customer) and click **Submit**.
4. A new entry will appear in the **Generated Reports** panel with a **Pending** status.
5. Once complete, the status will update to **Success**.
    - Click the **Download** icon to retrieve the report.

Report Types

The table below lists standard report types. Additional custom templates may be available for your Enterprise.

Report Type	Description	Includes
Detailed Usage Report	Comprehensive report covering all usage activity for a Customer.	Customer Name, CustomerID, ICCID, MSISDN, IMEI, IP Address, MCC, MNC, Network Operator Name, Operator Country, Operator Mode, Start Time, End Time, Bytes, MBs, GBs, Site, SiteID, System, SystemID.
Summary Usage Report - By IMSI	Focused report showing Subscriber IMSI usage for a given Customer.	IMSI, ICCID, MSISDN, Bytes, MBs, GBs.
Summary Usage Report - By System	Report showing System-level usage for a Customer.	System Name, System ID, Bytes, MBs, GBs.
Summary Usage Report	Usage grouped by IMSI, Site, and Network Operator for a Customer.	Customer, CustomerID, Site, SiteID, IMSI, ICCID, MSISDN, MCC, MNC, Operator Name, Operator Country, Operator Mode, Bytes, MBs, GBs.
Active Devices by Site Report	Text/CSV report showing all active Subscribers grouped by Customer and Site.	Site Name, Site ID, IMSI, MSISDN, MCC, MNC, Network Operator Name, Bytes, MBs, GBs.
Active Devices Report	Text/CSV report showing the total number of connected devices for a Customer.	Count of connected devices for a Customer.

Report Generation Parameters

Use the following parameters to customize the content and scope of your report:

Parameter	Description
Date-range	Click the date field and use the calendar to specify the *start date* and *end date*. Quick-select options include **Last Week**, **This Week**, **Last Month**, and **This Month**.
Timezone	Pre-populated with your current timezone. To change it, click **Timezone** and search for a city (e.g., *Paris* applies *Europe/Paris-CEST* timezone).
Customer	Select a Customer from the list. The report is restricted to the selected Customer and optionally includes child Customers.
Customer Extensions to Extract	Specify one or more customer-level Extensions to include in the report. Separate multiple Extensions with commas. Columns are created for each extension, even if not configured (empty values appear).
Include Child Customers	Checkbox. Includes usage details for the parent Customer, its child Customers, and all subsequent child Customers.
Write Row Count in Footer	Checkbox. Adds the total number of rows at the end of the CSV file.

> **Note**
>
> Even if the customer-level Extension is not configured, the report will include a column for the extension with blank (empty) values.

View the Report

After generating a report, you can download it as a CSV file for further analysis in programs such as MS Excel. Refer to Importing to Excel for instructions on correctly handling the data.

STEPS TO DOWNLOAD A REPORT:

1. Navigate to **Reports**.
2. Locate the **Generated Reports** panel. The most recently generated report is listed at the top.
3. Click the **download** icon next to the desired report to download the file.

Once downloaded, you can open the file in a spreadsheet tool to view and analyze the report data.

Importing to Excel

> **Warning**
>
> By default, MS Excel may treat MSISDN and ICCID values as numerical data with exponents, resulting in raw data loss. Follow the steps below to prevent data display issues for these fields.

STEPS TO IMPORT THE REPORT IN MS EXCEL:

1. Open a blank workbook in MS Excel.
2. Go to the **Data** tab and click **Get Data**.
3. Select **From File** and then click **From Text/CSV**. This will open the file selection dialog.
4. Choose the CSV report file you downloaded. A preview page of the file will appear.

5. **Important!** - For Data Type Detection, select **Do not detect data types**.
    - The raw values (like MSISDN and ICCID) will now display correctly in the preview.
6. Click **Load** to import the data into your workbook.

ADDITIONAL TIP:

To ensure the data remains intact, format the relevant columns (e.g., MSISDN and ICCID) as **Text** after importing. This prevents further transformations that might alter the raw values.

2.3.11 SFTP

Overview

This document provides guidance on configuring Secure File Transfer Protocol (SFTP) connections to securely transfer Call Detail Billing Records (CDR) reports. These reports, available in GZIP format, can be accessed through REST API calls by default or delivered via SFTP for enhanced automation and security.

Key highlights include:

- **SFTP Configuration**: Detailed instructions on setting up an SFTP connection, including required parameters such as hostname, port, username, private key, and optional remote directory paths.
- **SSH Key Pair Generation**: Steps to create the public and private keys used for secure encryption and decryption in OpenSSH format. The guide also emphasizes the importance of passphrases for encrypting private keys.
- **Supported Algorithms**: Information on recommended cryptographic algorithms, such as RSA and ECDSA, with examples of their usage and customizable key sizes.

This process ensures encrypted file transfer and facilitates the secure delivery of CDR reports for billing and informational purposes.

Configure an SFTP Connection

Call Detail Billing Records (CDR) reports are provided in compressed GZIP format. These reports can be accessed via REST API calls (default) or delivered through a configured SFTP connection. Each file includes a timestamp in its name, representing the GMT time of generation in the format `yyyyMMdd_HHmmss`.

> **Example**
>
> `ExpetoDailyDetailedTraffic20210214_114423_Prod.gz` This file was generated on February 14th, 2021, at 11:44:23.

Secure File Transfer Protocol (SFTP) enables encrypted, secure server-to-server file transfers. Configuring an SFTP connection involves setting up specific parameters to establish the connection and facilitate the secure delivery of reports.

STEPS TO CONFIGURE AN SFTP CONNECTION

Provide the following information to set up an SFTP connection:

Parameter	Description
Name	A unique identifier for the SFTP connection configuration.
Hostname	The hostname or IP address of the target machine.
Port	The port number for the SFTP connection. Default: `22`.
Username	The username associated with the target SFTP server.
Private Key	The private key required to establish the connection. Paste the key contents into the provided field.
Remote Directory	(Optional) The folder path on the remote server where files will be delivered.

By completing this configuration, you enable secure and automated file delivery through SFTP.

Generate an SSH Key Pair

SFTP connections rely on an SSH key pair, which consists of a public and private key. The public key encrypts data, while the private key decrypts it. The keys are generated in OpenSSH format using the ssh-keygen command.

After generating the keys, the private key must be copied and added to the SFTP connection configuration.

STEPS TO GENERATE AN SSH KEY PAIR

1. **Run the ssh-keygen Command** Use the command line to generate the key pair, specifying the desired algorithm and key size. Example command: `ssh-keygen -t rsa -b 4096 -m PEM` For details on supported algorithms and key sizes, see the Algorithms section.
2. **Specify the Save Location** Enter the file name or accept the default location to store the generated keys.
3. **Set a Passphrase (Optional)** Enter a passphrase to encrypt the private key. Without a passphrase, the private key will be stored as plain text. Verify the passphrase by re-entering it.
4. **Locate the Generated Files** Two files will be created:
    - *key_filename* — Contains the private key.
    - *key_filename.pub* — Contains the public key.

> **Example**
>
> id_rsa id_rsa.pub

1. **Copy the Private Key** Open the private key file (e.g., id_rsa) in a text editor and copy its contents.
2. **Paste the Private Key into Expeto xControl** In Expeto xControl, click **Add SFTP Connection** and paste the private key contents into the **Private Key** field.

Once configured, the private key ensures secure authentication for your SFTP connection.

Algorithms

When generating SSH key pairs, the choice of algorithm and key size determines the level of encryption and compatibility. Below are the supported algorithms for SFTP key generation.

RSA ALGORITHM

The RSA algorithm is widely used and supports variable key sizes for enhanced security. The recommended key size is 4096 bits.

Commands:

- Generate a 4096-bit key: `ssh-keygen -t rsa -b 4096 -m PEM`
- Generate keys with alternate sizes: `ssh-keygen -t rsa -m PEM` `ssh-keygen -t rsa -b 2048 -m PEM` `ssh-keygen -t rsa -b 8192 -m PEM`

Default Size: If the `-b` flag is not specified, the default key size is 3072 bits.

ECDSA ALGORITHM

ECDSA (Elliptic Curve Digital Signature Algorithm) offers enhanced security with smaller key sizes compared to RSA.

Commands:

- Generate a key with the default size (256 bits): `ssh-keygen -t ecdsa -m PEM`
- Generate keys with specific sizes: `ssh-keygen -t ecdsa -b 384 -m PEM` `ssh-keygen -t ecdsa -b 521 -m PEM`

> **Warning**
>
> **DSA Algorithm**: DSA key types are not supported and should not be used.

2.3.12 Role Permissions

The **Roles and Permissions** section provides a comprehensive breakdown of access rights and operational capabilities in the xControl system. This ensures that administrators can manage resources efficiently while maintaining secure and role-specific access controls.

Key highlights include:

- **Object Permissions**: Defines the Create, Read, Update, and Delete (CRUD) permissions for various system objects, such as Mobile Assets, Sites, Systems, and Accounts, across roles like **ADMIN**, **CUSTOMER_ADMIN**, and **READ_ONLY**.
- **Command Permissions**: Maps specific actions and commands (e.g., bulk import, report generation, API token creation) to roles, offering clarity on who can execute these tasks.
- Visual indicators for permissions, such as icons for Create, Read, Update, and Delete, make it easy to understand access levels at a glance.

This structure provides the flexibility to manage user roles while ensuring compliance with organizational policies and operational needs.

Object Permissions

The table below lists objects in xControl and the permissions for each role to create, read, update, and delete these objects.

Object/Feature	ADMIN	CUSTOMER_ADMIN	READ_ONLY
Mobile Assets			
Site			
APN			
System			
System Profile			
Account			
Account Audit Logs			
Customer			
Keyset			

Object/Feature	ADMIN	CUSTOMER_ADMIN	READ_ONLY
			
Keyset Mapping			
Monitoring Endpoint			
Private Radio			
SFTP Access Connection			

 = Create  = Read/List  = Edit/Update  = Delete

Command Permissions

The table below maps available commands and actions in xControl to specific roles.

Commands	ADMIN	CUSTOMER_ADMIN
Mobile Assets		
Bulk Delete	✓	
Bulk Import	✓	
Bulk Update	✓	
Cancel Session	✓	
Export to CSV	✓	
Accounts		
Create API Token	✓	✓
Delete API Token	✓	✓
Reports		
Generate Reports	✓	✓

2.3.13 Search Tips and Tricks

Overview

The search functionality allows users to efficiently filter and locate specific entries across various lists, including:

- Mobile Assets
- Sites
- Systems
- System Profiles
- Accounts

Searches can target all associated properties, whether or not they are displayed in the list view. This flexibility is enhanced by support for case-insensitive searches, partial-text queries, Boolean operators, parameter-based searches, and custom extensions.

While wildcards are not supported, the wide range of search methods ensures precise and efficient filtering.

Search Examples

GENERAL SEARCH TECHNIQUES

The following table outlines common search techniques applicable across multiple domains:

Search Type	Description	Example
Full Text	Use full text to focus on specific values. To eliminate partial matches, use single quotes.	`'823'` finds 823 but not 8234.
Partial Text	Use a minimum of 3 characters to find matches containing the input string.	`andr` finds all Android devices.
Boolean	Use Boolean operators (`AND` , `OR` , `NOT`) to refine searches.	`andr AND Canada` finds all Android devices in Canada.
Parameters	Search for a specific parameter value with or without single quotes.	`ICCID=89123` finds Mobile Assets containing `89123` ; `'ICCID=8912312312312312312'` finds exact matches.
Extensions	Custom parameters can be used to tag objects and search accordingly.	`user=joe` finds Mobile Assets tagged with the user value `Joe` (and `Joey` unless quoted).
Sessions	Search by session state.	`open` finds active sessions; `closed` finds disconnected sessions.

DOMAIN-SPECIFIC SEARCHES

Field	Description	Example
ID	The numerical suffix of an API endpoint URI for a Mobile Asset.	`id=222228100`
IMSI	Search by International Mobile Subscriber Identity.	`imsi=310150123456789`
Status	Filter by Mobile Asset state. Supported states: `Connected`, `Disconnected`, `Provisioned`, `Error`.	`status=connected`
System	Search by the name of the System associated with the Mobile Asset.	`system=MainNetwork`
Customer	Search by the Customer associated with the Mobile Asset.	`customer=EnterpriseCorp`
Location	Search by city name.	`location=chicago`
Country	Search by country code or name.	`country=ca` or `country=canada`
IMEI	Search by the device's International Mobile Equipment Identity.	`imei=123456789012345`

Field	Description	Example
ID	The numerical suffix of an API endpoint URI for a Site.	`id=333338900`
Customer	Search by Customer associated with the Site.	`customer=SiteManagerCorp`

Field	Description	Example
ID	The numerical suffix of an API endpoint URI for a System.	`id=444449700`
Customer	Search by Customer associated with the System.	`customer=SystemAdminCorp`

Field	Description	Example
Email	Search by the account's email address.	`email=user@example.com`
Customer	Search by the Customer associated with the account.	`customer=AccountCorp`

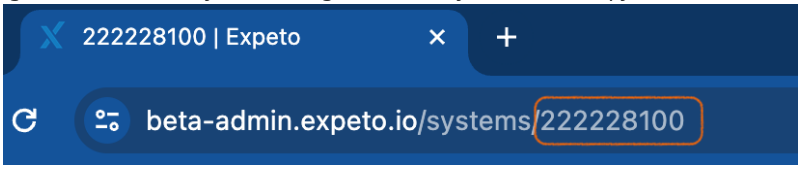
Field	Description	Example
Open	Find all currently active sessions.	`status=open`
Closed	Find all disconnected sessions.	`status=closed`

Field	Description	Example
Custom Tags	Search for objects tagged with specific custom parameters. For example, search for `user=joe` to find assets tagged with `joe`.	`user=joe`

ADDITIONAL NOTES

- Searching by **location** or **country** includes both the device location (if available) and carrier details.
- To find an object's **ID** (e.g., Mobile Asset or System), navigate to the object's details, copy the URI, and extract the numerical identifier.

Example: `id=222228100`



2.3.14 Extensions

Overview

Extensions are a powerful tool in Expeto xControl that allow users to customize configurations and extend functionalities beyond the standard GUI settings. They enable flexibility in managing Mobile Assets, System Profiles, Systems, Sites, and Customers by providing additional parameters that can enhance network operations.

Key features of Extensions include:

- Adding custom configurations, such as static IP addresses, to provide stability and reliability for connected devices.
- Tagging network elements with labels for better organization and easier searches.
- Enhancing features through scripts for tasks not available in the GUI.
- Strengthening security measures, such as IMEI locking, to restrict network access to authorized devices.

Extensions follow a hierarchical structure, enabling higher-level configurations to be overridden at more specific levels for greater control. They are instrumental in improving asset management, optimizing configurations, and ensuring secure and efficient network operations.

Searching for Mobile Assets with Extensions

Extensions enhance search functionality by allowing you to locate Mobile Assets or other entities based on custom labels or parameters. For example, if an extension labeled **Membership VIP** is applied to a Mobile Asset, searching for "vip" in the Mobile Assets list will return all assets tagged with this extension.

Extensions can be applied at different levels, such as the System Profile, System, or Mobile Asset. Conflicting extensions may exist, and searches will return all matching values.

Use Cases for Extensions

ADD A LABEL TO A MOBILE ASSET

A common use of Extensions is to add labels to Mobile Assets for easier identification and organization. Labels can also be used as search criteria in the search field. For example:

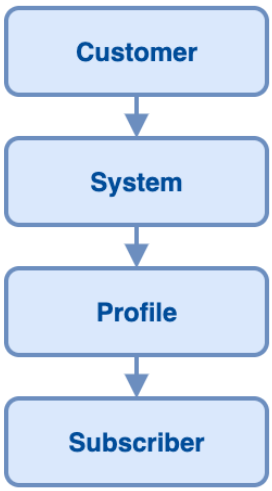
- To find all objects tagged with a specific label, search for the label name (e.g., **sim-type** or **user**).
- To find a specific value for a label, use `label=value` (e.g., **sim-type=UICC** or **user=lisa**).

USING EXTENSIONS IN SCRIPTS

Extensions can be used to enhance functionality not currently available through the GUI. By tagging a Mobile Asset with an extension, you can find it using search or integrate it into scripts for automation.

Extensions follow a hierarchy, where values set at higher-level entities can be overridden by conflicting values set at lower levels. The hierarchy is as follows:

1. Customer
2. Site
3. System
4. System Profile
5. Mobile Asset



> **Note**
>
> While lower-level values override higher-level ones for functionality, searches use a logical OR and return all matches.

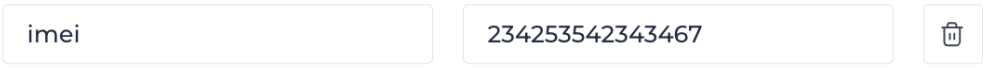
ASSIGN A STATIC IP ADDRESS

Assigning a static IP address to a Mobile Asset provides stability and reliability, especially for IoT devices requiring constant connectivity. A static IP address allows direct remote access to the device and can be whitelisted for firewall rules.

- The keyword *ip* must be in lowercase.
- The IP address must belong to the subnet of the attached Mobile Asset.
- The IP address must be designated for static assignment and excluded from the dynamic IP pool.
- The device's APN must be configured correctly; otherwise, it will default to dynamic addressing.

CONFIGURE IMEI LOCKING

The IMEI is a unique serial number assigned to a device by the manufacturer. IMEI locking enhances security by ensuring that a SIM card is used only in its designated device. If locked, the SIM card cannot access the network if moved to a different device.



- The keyword *imei* must be in lowercase.
- The value must be exactly 15 characters unless using the wildcard `*`.
- An IMSI can be locked to only one IMEI.

Note

You can display the IMEI on many devices by dialing `*#06#`.

2.3.15 Key Sets

Keysets are collections of encryption keys and algorithms used to authenticate SIM cards and establish secure communication between Mobile Assets and the network. They ensure mutual trust and encrypted interactions, safeguarding the integrity of the system.

This section provides tools to manage Keysets, including adding new Keysets, mapping them to HSS Keyset Identifiers at specific Sites, and applying them during Mobile Asset setup. Proper Keyset management is essential for secure and efficient network operations.

Enable Keyset Authentication

Keyset authentication ensures secure communication and mutual trust between the SIM card and the network. It uses encryption keys and algorithms to verify the authenticity of Mobile Assets. Keysets are collections of encryption keys and algorithms that ensure secure SIM authentication and network integrity.

- **Loading Keysets**: Performed by the Expeto Site Installer during site setup.
- **Adding and Mapping Keysets**: Performed by a Network Administrator with Keyset permissions using Expeto xControl. The Site must have Keyset Mapping enabled by the Expeto Site Installer.
- **Applying Keysets**: Network Administrators assign Keysets when adding Mobile Assets (individually or via Bulk Import).

> **Note**
>
> A Keyset can only be assigned once to a Mobile Asset and cannot be changed.

Add a Keyset

Keysets are essential for enabling encrypted authentication and ensuring secure communication between Mobile Assets and the network. Adding a Keyset creates a reference that can be used during Mobile Asset configuration.

1. Navigate to the **Keysets** section in Expeto xControl. The list of existing Keysets will be displayed.
2. Click **New** to create a new Keyset.
3. Enter the following details:
    - **Name**: Provide a unique name for the Keyset.
    - **Customer**: Select the Customer that will use this Keyset.
4. Click **Submit** to save the new Keyset.

Once added, the Keyset will appear in the Keyset list and can be used for Mobile Assets at any Site belonging to the selected Customer.

> **Note**
>
> To enable encrypted authentication, the Keyset must be mapped to the HSS Keyset Identifier. See the next section for details.

Map a Keyset

Mapping a Keyset links it to the corresponding HSS Keyset Identifier (KSI) at a specific Site, enabling secure authentication for Mobile Assets. This step is essential to complete the Keyset configuration process.

> **Note**
>
> The **Add Mobile Asset** permission is required to perform this task. The **CUSTOMER_ADMIN** role does not include the **Add Keyset Mapping** permission.

BEFORE YOU BEGIN

Ensure the following prerequisites are met:

- **Keyset Mapping**: Must be enabled for the Site by a Site Installer.
- **Keyset Loading**: The Site Installer must load the Keyset into the HSS.
- **HSS Keyset Identifier**: The Site Installer must provide the HSS Keyset Identifier.

TO MAP A KEYSET:

1. Navigate to the **Sites** section in Expeto xControl.
2. Select the Site where the Keyset will be mapped.
3. In the **Keysets** section, click **Add Keyset Mapping**.

   The New Keyset Mapping panel will appear.
4. Enter the **HSS Keyset Identifier** provided by the Site Installer.

   This value cannot be selected; it must be provided manually.
5. Optionally, enter a description for the Keyset (e.g., *Keyset for SIM batch 12*).
6. Select the Keyset name from the list.

   If no Keyset names are available, ensure you have added a Keyset.
7. Click **Submit** to save the mapping.

> **Note**
>
> - The same Keyset can be referenced by multiple Sites owned by the same Customer.
> - Keysets must be loaded individually into each Site's HSS database.
> - The Keyset name must remain consistent across Sites, but the HSS Keyset Identifier may vary depending on how the Keyset was loaded.
>
> Mapping ensures that the Keyset is properly associated with the Site and can be used for Mobile Asset authentication.

Understanding SIM Authentication Security

Ensuring secure communication and trust between the SIM card and the network is critical for mobile asset authentication. This involves mutual authentication, where both the SIM card and the network verify each other's identity.

KEY SECURITY CONCERNS

- **Secret Key Exposure**: Preventing unauthorized access to private cryptographic keys.
- **Replay Attacks**: Protecting against repeated use of intercepted authentication data.
- **HSS Spoofing**: Ensuring the authenticity of the Home Subscriber Server (HSS).

THE AUTHENTICATION PROCESS

1. **Mutual Authentication**
   - The SIM card and the HSS database validate each other using a shared secret key (`K`) and an Operator Code (`OP`), which together generate an OPc value.
   - Authentication also relies on a sequence number that increments with each challenge to prevent replay attacks.
2. **Encryption and Decryption**
   - The referenced Keyset contains private keys and algorithms for encrypting and decrypting authentication data.
   - These keys ensure that sensitive values like `K` and OPc remain secure and are never exposed publicly.
3. **Challenge-Response Mechanism**
   - The HSS sends an encrypted challenge (`AUTN`) to the SIM card.
   - The SIM decrypts the challenge using the Keyset and generates a response.
   - The SIM's response is sent back to the HSS via the MME (Mobility Management Entity).
4. **Verification**
   - The HSS decrypts the response and verifies it against the expected result.
   - If the response matches, the SIM card is authenticated, and the Mobile Asset is allowed to attach to the network.

KEY TAKEAWAYS

This process ensures that:

- Only trusted SIM cards can connect to the network.
- Sensitive authentication data is encrypted and protected at all times.
- Network integrity is maintained by verifying both the SIM card and the network.

Properly implemented SIM authentication is essential for securing mobile assets and protecting against unauthorized access or attacks.

2.4 Installation

2.4.1 Installation Overview and Preparation Guide

Expeto offers scalable and flexible solutions for deploying private and public 4G/5G networks within Kubernetes environments.

This guide provides step-by-step instructions to help you install and configure Expeto products effectively, whether you're managing a greenfield deployment or integrating it into an existing network. While the guide assumes advanced technical expertise, it is structured to accommodate the specific needs of Mobile Network Operators (MNOs), enterprise customers, and telco integrators.

Who Should Use This Guide?

- **Mobile Network Operators (MNOs):** MNOs deploy Expeto products as part of a broader ecosystem including xControl and xRouter to manage large-scale, multi-site networks for internal use or customer traffic. These users typically require advanced configurations for multi-tenancy, redundancy, and scalability, and asusme extensive expertise in telecommunications and networking such as BGP and IPX.
- **Enterprises Deploying public xCores:** These users deploy xCore as a "router" for mobile traffic to egress onto private networks without the need for full telco infrastructure. They usually have enterprise networking experience and require a streamlined installation process leveraging Expeto's managed services.
- **Integrators and Enterprise networking experts using Private Radios:** Specialists and advanced enterprise staff integrating with private radios (gNodeB/eNodeB) for customized network solutions. They possess advanced telco experience and expertise in configuring radio hardware, PLMN IDs, and complex networking interfaces.

This guide assumes that foundational decisions about deployment, such as PLMN IDs, radio hardware, and Kubernetes platform configurations, have been made. While Expeto uses a **Product Intake Form** internally to gather this information, this guide provides a comprehensive installation path even if that data hasn't been formalized.

Overview of the Installation Process

The installation process for Expedo products is structured into five main stages:

1. **Prerequisites:**
   - Ensure Kubernetes readiness, install necessary plugins, and set up required tools like `kubectl` and `helm`.
   - On OpenShift/MicroShift all commands performed with kubectl can be performed with the `oc` command instead, in this case we recommend using an alias, e.g. alias `kubectl=oc`
2. **Add Expeto Helm Repositories:**
   - Authenticate and retrieve the necessary Helm charts.
3. **Customize Configuration:**
   - Create and configure the `values.yaml` file to define behavior and connectivity, including PLMN IDs, networking interfaces, and scaling requirements.
4. **Deploy:**
   - Use Helm to install and monitor the deployment process.
5. **Validate and Optimize:**
   - Verify the deployment, ensure proper functionality, and apply performance optimizations.

Prerequisites

Before you begin the installation, ensure that your environment meets the following requirements. This section outlines system requirements, tools, and preparatory steps necessary for a smooth deployment process.

SYSTEM REQUIREMENTS

- **Version:** Kubernetes 1.23 or later.
- **Cluster Resources:**
  - At least 3 worker nodes for redundancy.
  - Minimum of 4 CPU Cores and 8 GB memory per node for a total of at least 16 cores and 32 GB per cluster.
  - Persistent storage system available from all nodes configured for `ReadWriteOnce` volumes.

  For more information on resource sizing, see System Resource / Sizing Guide
- **Plugins/Operators/Features required:**
  - DNS Resolver.
  - Cert Manager.
  - Storage plugin compatible with `ReadWriteOnce` volumes.
- **Node Kernel:**
  - Linux kernel 5.13 or later, compatible with Kubernetes.

TOOLS AND UTILITIES

- **kubectl:** Command-line tool for interacting with Kubernetes clusters.
  - Install kubectl
- **helm:** Kubernetes package manager for managing Helm charts.
  - Install Helm

Depending on your Kubernetes environment, the following tools must be configured to ensure a functional deployment. Adjust the steps based on whether you are using MicroK8s, OpenShift/MicroShift, or VMware Tanzu.

> **Important**
>
> Review the document below of your chosen Kubernetes platform to understand the common installation and configuration challenges:
>
> - Canonical MicroK8s
> - RedHat OpenShift/MicroShift
> - WMware Tanzu

DNS Resolver

A DNS resolver provides name resolution for Kubernetes services, so that services are able to communicate with each other.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

```
sudo microk8s enable dns
```

OpenShift/MicroShift have DNS capabilities enabled by default.

For further configuration options, please refer to DNS Operator in OpenShift.

VMware Tanzu comes with integrated DNS resolution capabilities via CoreDNS.

If you require public service resolution, Tanzu supports synchronizing with external DNS providers via ExternalDNS.

Cert Manager

Cert Manager is required to manage TLS certificates for secure communication.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

```
sudo microk8s enable cert-manager
```

Please refer to Cert-manager documentation for configuring cert-manager with OpenShift.

Please refer to Tanzu Cert Manager Installation for installation steps with VMware Tanzu.

Persistent Storage

A storage class must be configured to support persistent volumes.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

MicroK8s does not ship with a storage class plugin to span multiple worker nodes. For Demo purposes the following can be used on single node installs only. This will not work for multi-node installs because the storage will not be replicated across nodes causing Pods to loose data if they migrate between Nodes. Other options are available for larger MicroK8s installs, see Cannonical documentation for examples and guidelines.

```
sudo microk8s enable hostpath-storage
```

OpenShift supports a wide variety of persistent storage options.

Read more at Understanding Persistent Storage in OpenShift.

See following document for MicroShift MicroShift Storage Configuration.

Tanzu also supports a variety of persistent storage options.

Read more at Using Persistent Storage in vSphere with Tanzu.

CLUSTER PREPARATION

1. Get a copy of the kube-config file for your cluster, name it config and place it in the .kube directory residing in your home directory, be sure to set permissions to 0600 or equivalent on non Linux platforms.
2. **Verify Kubernetes Access:**
   - Ensure you can connect to your Kubernetes cluster using kubectl:

     ```
     kubectl get nodes
     ```

     Confirm all nodes are in a Ready state.
3. **Helm Installation:**
   - Verify that Helm is installed and functioning:

     ```
     helm version
     ```
4. **Ulimits:**

   The Expeto solution ships with a DaemonSets/MachineConfigs that will automatically configure things like kernel modules, buffer sizes, sysctl parameters and ulimits on most platforms but in some cases slight additional modifications might be recommended.

   For MicroK8s: Edit the container environment configuration and increase the ulimit to 1048576

   ```
   echo 'ulimit -n 1048576 || true' >> '/var/snap/microk8s/current/args/containerd-env'
   ```
5. **Configure allowed NodePort range:**

   The default installation depends on the availability of NodePorts and having a wider range of ports enabled than is default with most k8s deployments.

Unless your install only uses Multus based interfaces make sure to allow node ports in the range of 30000-38413 before installing the chart.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMWare Tanzu**

```
echo '--service-node-port-range=1024-38413' >> /var/snap/microk8s/current/args/kube-apiserver
```

Then, restart microk8s (`sudo microk8s.stop && sudo microk8s.start`)

```
oc patch network.config.openshift.io cluster --type=merge -p \
'{
   "spec":
   { "serviceNodePortRange": "1025-38413" }
}'
```

Note that it might take several minutes for the new config to be applied.

For a multi-master cluster see the following doc for how to identify the leader: Leader identification

For a single master cluster or once the cluster leader has been identified make the following change:

```
echo '- "--service-node-port-range=-1024-38413"' >> /var/vcap/jobs/kube-apiserver/config
```

Then, restart the kube-apiserver

6. **Advanced Networking:**

   - **Using LoadBalancers:**

     Please refer to Services, Load Balancing, and Networking in Kubernetes documentation for detailed information regarding setting up load balancers and other advanced networking topics.

   - **Using Multus for Advanced Networking:**

     For platforms that support Multus it can either be installed separately or installed by setting `multus.enabled` to true during install.

     It's recommended to also install Whereabouts for IPAM at the same time.

     For installations that have a Multus plugin, it is recommended to use the built-in version.

     Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

     ```
     sudo microk8s enable community
     sudo microk8s enable multus
     ```

     Please refer to Multus CNI documentation for configuring Multus and Whereabouts with OpenShift.

     Please refer to Cert-manager documentation for configuring Multus with VMWare Tanzu.

     For further information regarding installation and configuration of Multus on other platforms, please refer to Multus Quick-Start Guide or Multus Documentation.

7. **Role-Based Access Control (RBAC):**

   Note: If you plan to use Role-Based Access Control (RBAC) in your Kubernetes cluster, **it must be enabled before deploying Expeto products.**

   If RBAC is enabled after deployment, the cluster must be redeployed before it will function correctly.

   When the cluster is deployed, the helm chart will automatically configure the required roles, bindings, and permissions for RBAC.

   Refer to Using RBAC Authorization in Kubernetes documentation for further information.

Adding the Expeto Helm Repositories

All Expeto deployments rely on Helm charts hosted in Expeto's private repositories. This step outlines how to configure access to these repositories, ensuring that the necessary charts and container images can be downloaded during installation.

Before proceeding, ensure you have:

1. **Helm Repository Credentials:** Provided by Expeto Support for accessing container images and Helm charts.
2. **Deployment-Specific Information:**
    - PLMN IDs, RAN details, and other configuration specifics (if applicable).
    - Network interface and CIDR details for northbound (N2/N3/N9) and southbound (N6) interfaces.

1. Use the provided credentials to add the Expeto Helm repositories to your local Helm configuration.

```
helm repo add expeto-ngc https://repo.expeto.io/repository/ngc --username '<your_username>' --password '<your_password>'
```

   Replace `<your_username>` and `<your_password>` with the credentials provided by Expeto Support.
2. Update your local Helm repository cache to ensure you have the latest chart versions:

```
helm repo update
```

After adding the repositories, confirm that they are accessible and contain the required charts:

1. List the added repositories:

```
helm repo list
```

2. Search for available charts to confirm connectivity:

xCore **xRouter** **xControl**

```
helm search repo xcore
```

```
helm search repo xrouter
```

```
helm search repo xcontrol
helm search repo expeto-docs
```

These commands should return a list of available charts for each product in the Expeto repository.

If you encounter issues adding the repositories or accessing charts:

- Ensure your username and password are correct and match the credentials provided by Expeto Support.
- Confirm that your machine has access to the internet and can reach `https://repo.expeto.io`.
- Ensure that Helm is installed and functioning properly and is more recent than 3.0.9 than by running:

```
helm version
```

- Contact Expeto Support for assistance if the problem persists.

Customize Configuration

The `values.yaml` file is a critical part of the Expeto xCore deployment process.

It defines parameters for the entire deployment, including networking, resource scaling, radio configuration and advanced features. This guide will help you customize the file to align with your deployment requirements.

> **Important**
>
> - All changes to deployments must be made in the `values.yaml` file. This file serves as the **source of truth** for your Helm deployments. **Any manual changes made directly to resources (e.g., via kubectl) will be overwritten when scaling occurs or when the `values.yaml` file is applied again.**
> - For more details on applicable parameters in the `values.yaml`, see values.yaml Reference

To ensure consistency and prevent loss of changes, always update the values.yaml file and reapply it with Helm commands.

Make sure you store a copy of the values.yaml file in a secure place, this file acts as your disaster recovery and will allow you to rebuild the entire solution with minimal effort very rapidly.

To access the `values.yaml` file:

1. Pull the Helm chart from the `expeto-ngc` repository:

   xCore **xRouter** **xControl**

   ```
   helm show values expeto-ngc/xcore
   ```

   ```
   helm show values expeto-ngc/xrouter
   ```

   ```
   helm show values expeto-ngc/xcontrol
   ```

   This is to get the default values, multiple examples are also available to demonstrate different setup options and possibilities for a variety of Kubernetes distributions

   The `values.yaml` file will be in the extracted directory.

2. Pull examples from the `expeto-ngc` repository:

   xCore **xRouter**

   ```
   export tmp_dir="$(mktemp -d)" && helm pull ngc/xcore  --untar --untardir ${tmp_dir} && mv "${tmp_dir}/xcore/examples" ./ && rm -Rf "${tmp_dir}"    === "xCore"
   ```

   ```
   export tmp_dir="$(mktemp -d)" && helm pull ngc/xrouter  --untar --untardir ${tmp_dir} && mv "${tmp_dir}/xrouter/examples" ./ && rm -Rf "${tmp_dir}"
   ```

   This will download the chart, extract it to a temporary directory and then move the examples directory to the current directory. Alternatively the whole chart can simply be downloaded with "helm pull" and extract.

- **Edit the File:** Open `values.yaml` and update the sections relevant to your deployment. Reference the comments in the file for additional guidance.
- **Validate Changes:** Use Helm to validate the configuration before applying it:

xCore **xRouter** **xControl**

```
helm template expeto-ngc/xcore -f values.yaml > validated_output.yaml
```

```
helm template expeto-ngc/xrouter -f values.yaml > validated_output.yaml
```

```
helm template expeto-ngc/xcontrol -f values.yaml > validated_output.yaml
```

Review the `validated_output.yaml` for any errors or issues before proceeding.

- **Start with Defaults:** Use the default values.yaml file as a baseline and modify only the required fields.
- **Document Changes:** Add comments to track customizations for easier troubleshooting or future updates.
- **Test Incrementally:** Validate and test changes incrementally to catch errors early.

Deploy and Verify

With your environment prepared, Helm repositories configured, and the `values.yaml` file customized, you are ready to deploy into your Kubernetes cluster. Following the links below for details of the deployment process, verification steps, and troubleshooting tips of the Expeto platform components:

- **xCore**
- **xRouter**
- **xControl**

2.4.2 Deployment & Verification

3. xView

3.1 xView Guide

3.1.1 Introduction

xView is a network monitoring system (NMS) based on the open source Prometheus platform, providing metrics collection and alert triggering capabilities. Its distributed architecture offers comprehensive network monitoring with a centralized aggregated view of the Expeto Platform operational status.

The NMS collector agents gather metrics from all platform components, federate these metrics, and forward them to the NMS master. The master evaluates the metrics against alert expressions stored in the Expeto alert library, triggering alerts when expressions remain unresolved. xView integrates with third-party tools such as Grafana and Datadog for data visualization and NetCool for alert management.

KEY FEATURES:

- Distributed and configurable architecture
- Dynamic service discovery of endpoints to monitor
- Centralized aggregated view
- Integration with third-party systems

3.1.2 How xView Works

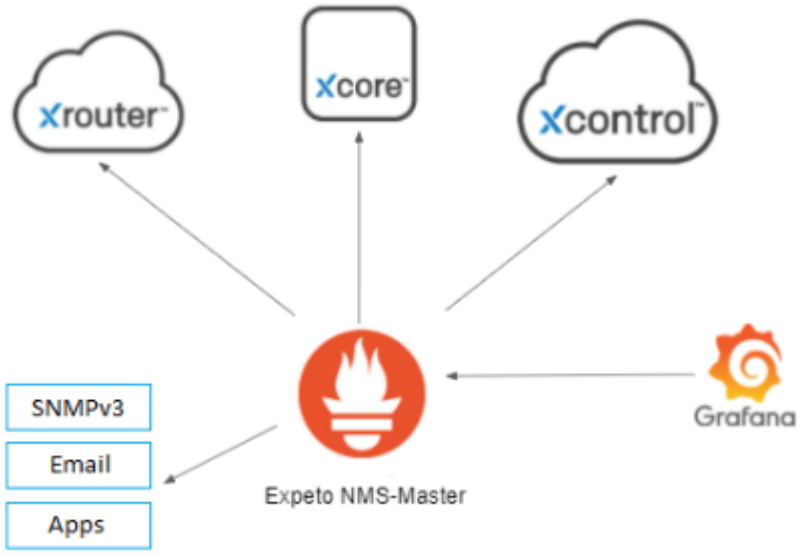
Monitored endpoints are created on deployments by Expeto at strategic component locations.

At regular polling times, NMS-collectors scrape metrics from the monitored endpoints. The metrics are federated across NMS-collectors and sent to the NMS-Master where they are used to evaluate alert expressions stored in the Expeto Alert Library.

If an alert expression evaluates to `true`, an alert is created with a status of `pending`. The pending state prevents the creation of multiple alert instances for the same incident (alert flapping). If the alert remains true for the time interval defined in the alert expression, the alert status changes to `fired`. The alert clears when the polled metrics cause the alert expression to evaluate to `false`.

xView integrates with third-party alert monitoring and management tools. In the xView Architecture example below:

- Metrics from the Expeto platform components are collected by Prometheus using native and third-party exporters.
- Grafana fetches only the metrics needed to populate its custom dashboard configuration.
- Alerts can be sent to a third-party tool to kick off a ticket flow.



Metric Exporters

Metric exporters collect and send metrics from the Expeto platform components to the Prometheus master. Each exporter focuses on specific categories of metrics. They are configured as

Exporter Name	Prometheus Job Name	Notes
Node Exporter	job="node"	Provides VM node metrics and server health.
Docker Exporter	job="docker"	Collects metrics about Docker containers.
Prometheus	job="prometheus"	Collects metrics about Prometheus exporters.
xAgent Exporter	job="xAgent"	Provides metrics about the xCore host site.
Secure Gateway	job="SecGw"	
Alert Manager	job="alertManager"	
AUSF Exporter	job="ausf"	
Exporter	job="blackbox-q-ls"	
Exporter	job="broker"	
Exporter	job="grafana"	
HSS Exporter	job="hss"	
Kubernetes Exporter	job="k8s"	
Kublet Exporter	job="kubelet"	
Exporter	job="local_config_file"	
MariaDB Exporter	job="mariadb"	
MariaDB Galera Exporter	job="mariadb-galera"	
PostgresSQL Exporter	job="postgres"	
xCore Exporter	job="site"	
UDM Exporter	job="udm"	
UDR Exporter	job="udr"	
xControl Exporter	job="xControl"	
Rec Exporter	job="x_rec"	

3.1.3 Enable or Suspend Monitoring

Suspending monitoring is recommended to avoid a flood of unnecessary alerts when taking a Site down for maintenance.

To suspend monitoring at the Site level:

1. Open xControl, navigate to Sites, and click the name of the Site you want to exclude from monitoring.
2. Click **Edit**.
3. Create or edit the following Extension: label.site_state.
4. Assign one of the following values:
    - maintenance
    - decommissioned
5. Click **Save**.

The extension appears listed with other key parameters for the Site.

To enable monitoring for the Site, either delete the extension or set it to enabled.

3.1.4 Anatomy of an Alert

All Expeto alerts are metrics-based and defined in the alerts.rules file in YAML format located in `/etc/prometheus` on the Prometheus server. The actual triggering (firing) of the alert occurs only after the metrics-based expression in the alert definition evaluates to true and remains true over a period of time.

The metrics-based expression is written in PromQL (Prometheus Query Language) in the **expr** alert parameter.

The time period is defined in the **for** alert parameter.

For example, the following **ExpetoContainerDown** alert checks whether any container starting with the name “expeto” is equal to 0 (indicating the container is down). If any Expeto container remains down for 2 minutes, the alert is fired.

```
name: ExpetoContainerDown
expr: docker_container_running_state{name=~"expeto.*"} == 0
for: 2m
```

Some alerts contain additional tiers (Tier2, Tier3), firing an additional alert after more time has elapsed and indicating that conditions causing the initial alert still exist.

For example, the xCoreAgentOutOfContact alert initially fires after being evaluated as true for 5 minutes. The tier2 version of the alert fires after being true for 1 hour. The tier3 version of the alert fires after being true for 4 hours.

```
name: xCoreAgentOutOfContact
expr: up{job="site",site_state!~"decommissioned|maintenance|poc",tier=~"1|"} == 0
for: 5m
```

```
name: xCoreAgentOutOfContactTier2
expr: up{job="site",site_state!~"decommissioned|maintenance|poc",tier="2"} == 0
for: 1h
```

```
name: xCoreAgentOutOfContactTier3
expr: up{job="site",site_state!~"decommissioned|maintenance|poc",tier="3"} == 0
for: 4h
```

Labels attached to the alert provide troubleshooting information, such as customer, site, monitoring endpoint instance, and monitoring job name. Use labels for search queries in Prometheus or NetCool to find all alerts belonging to a specific site or customer.

Alert Parameters

Each alert contains the following parameters:

- **Name**: The name of the alert.
- **Expr**: The Prometheus alert expression that includes the metric variables for evaluation. If the expression evaluates as true, the alert is activated. If the alert has a **for** time greater than 0, the alert is pending until the **for** time interval elapses, at which point the alert is fired. Note that some alerts contain additional conditions that prevent the alert expression from evaluating as true.

  For example, the Node Down alert contains the following alert expression that filters out Sites set to decommissioned or maintenance:

  ```
  up{job="node",site_state!~"decommissioned|maintenance"} == 0
  ```

- **For**: The amount of time the alert must remain evaluated as true before the alert is actually fired. This prevents alert flapping. The **for** value can be expressed in seconds, minutes, hours, or days. The following PrometheusCollectorDown alert rests in a pending state for 3 minutes before firing:

  ```
  name: PrometheusCollectorDown
  expr: up{job="prometheus",site_state!~"decommissioned|maintenance"} == 0
  for: 3m
  ```

  An alert with a **for** value of 0 (or no **for** value), is fired immediately. There is no pending state.
- **Labels**: Metadata for the alert. Currently used to set the severity level of the alert. Other labels are used in the annotations section. See Labels section for more information.
- **Annotations**: A section that includes the **debug**, **description**, and **summary** information.
- **Debug**: Lists all labels attached to the alert. Useful for filtering queries and troubleshooting.

  ```
  debug: {{ $labels }}
  ```

- **Description**: A verbose alert message using label values to describe the details of the alert. Includes expanded endpoint URIs that provide links to resources in xControl.

  ```
  Disk space usage is at 93 percent on device /mnt on VmName001 at
  SiteName001 (https://myCompany.com/v1/sites/27351) for
  Customer001 (https://myCompany.com/v1/customers/1851).
  ```

- **Summary**: An alert message using label values to provide enough information about the alert to start troubleshooting.

  ```
  High disk space usage on VmName001 at SiteName001 for Customer001.
  ```

Labels

Labels are key-value pairs that are used in the alert annotations and can also be used as filters in search queries on third-party alert management tools or ticket systems. For example, you could find all alerts for a given site or customer.

Many labels, such as `siteId` or `customerId`, are automatically generated by xControl. Others are set by Prometheus, such as `xName`, which provides the name of the Prometheus job that scraped the metrics. Additional custom labels can be manually set as extensions at the site or at the monitoring endpoint.

USE OF LABELS IN ALERT ANNOTATIONS

The annotations section uses labels as variables to generate the alert messages.

```
annotations:

summary: "The network interface down {{ if $labels.vmName }}on {{ $labels.vmName }} {{ end }}{{ if $labels.site }}at {{ $labels.site }} for {{ $labels.customer }}{{ end }}"

description: "The network interface {{ $labels.device }} is down {{ if $labels.vmName }}on {{ $labels.vmName }} {{ end }}{{ if $labels.site }}at {{ $labels.site }}{{{ $labels.siteId }}) for {{ $labels.customer }} ({{{ $labels.customerId}}}{{ end }}"

debug: "{{ $labels }}"
```

The resulting alert messages look like this:

summary: The network interface down on VmName001 at SiteName001 for Customer001

description: The docker container expeto-hss is down on VmName001 at SiteName001 (...) for Customer001 (...)

Site State Labels

Another important use of labels is to tag a site as being under maintenance or out of commission. When a site is tagged, no alerts are generated for that site. This is achieved with the following extensions at the site level:

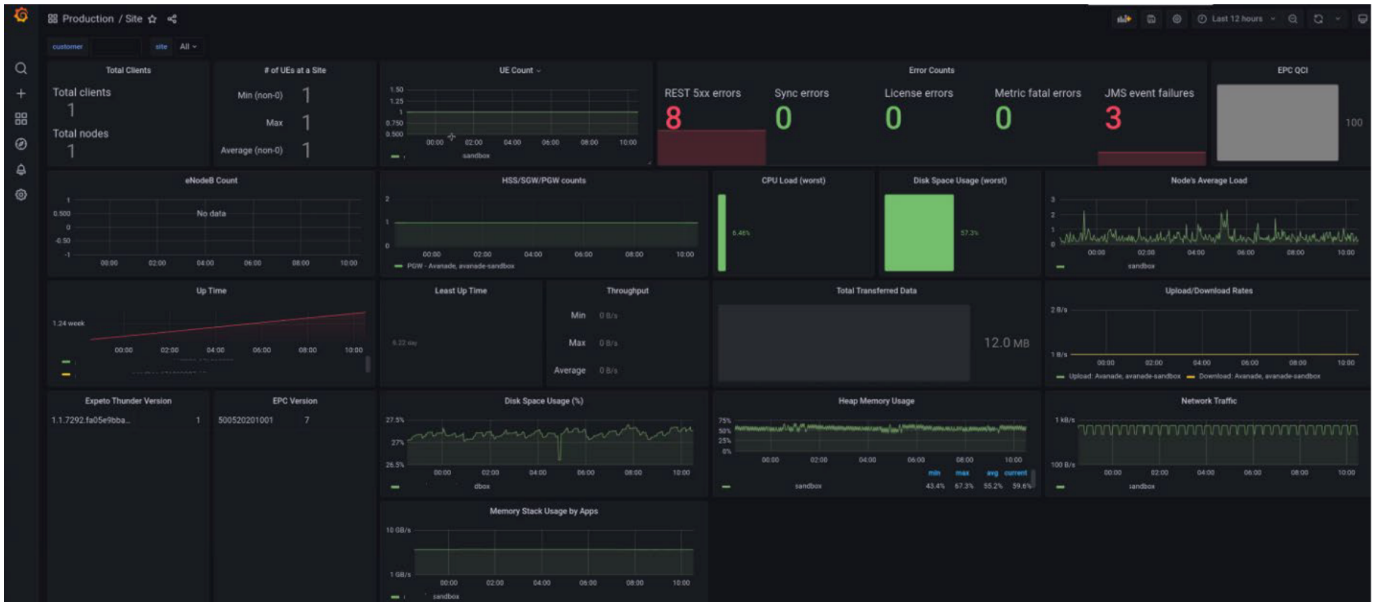
- **label.site_state = maintenance**
- **label.site_state = decommissioned**

For more details, refer to the **Enable or Suspend Monitoring** section.

3.1.5 Metrics Visualization

The collected metrics are available for display and evaluation in data visualization applications such as Splunk, Datadog, or Grafana regardless of whether conditions to trigger an alert are met.

The following screenshots show integration with Grafana.



Only the metrics required to populate the configured dashboard are fetched.

Active Alerts	Alert Changes 1d	Min Uptime	Min xUptime	LogErrors 1d	MaxLoad	MaxCpu	MaxMemUsage	DiskUsage
0	12	15.7 week	0	525	0	2.17%	29.81%	26.24%
0	27	3.81 week	2.20 week	23	0.03	11.61%	59.62%	47.73%
0	2	1.70 week	6.22 day	5	0.31	12.28%	24.53%	57.27%
0	0	25.1 week	15.3 week	7	0.36	12.88%	12.39%	35.12%
0	0	5.52 week	5.52 week	7	0.16	11.49%	30.71%	28.39%
0	27	8.40 week	2.81 week	7	0.13	3.27%	54.84%	22.50%
1	1	1.26 week	1.26 week	531	0.08	3.01%	26.34%	52.57%
0	7	10.8 week	2.23 hour	45	0.14	13.82%	59.04%	68.93%
0	3	1.28 year	1.75 week	9	0.14	6.01%	56.56%	60.71%
2	0	3.12 week	1.07 day	3	0.02	10.62%	31.21%	63.49%
0	3	3.05 day	3.05 day	21	0.48	6.51%	9.23%	54.16%
0	123	1.28 year	2.27 week	27	0.18	17.84%	83.23%	75.41%

3.2 Metrics

3.2.1 Expeto xCore Metrics

xCore Metrics Used in Default Alerts

The following metrics appear in PromQL expressions within the default alerts of Expeto NMS. These metrics are commonly used to monitor key aspects of system health and performance.

Category	Data Type	Prometheus Metric	Description
Client Metrics	Total Clients	`xcore_clients`	Number of active clients connected to the xCore.
Throughput Metrics	Upload Bytes	`xcore_upload_bytes_total`	Total bytes uploaded through the xCore.
	Download Bytes	`xcore_download_bytes_total`	Total bytes downloaded through the xCore.
Authentication Metrics	3G MAP Auth Requests	`xcore_q_hlr_map_c_cs_send_auth_RqRcvd_total`	Total 3G MAP authentication requests received.
	4G S6A Auth Requests	`xcore_q_hss_s6a_auth_retrieval_RqRcvd_total`	Total 4G S6A authentication requests received.
Session Metrics	S1 Data Session Requests	`xcore_q_mme_s1_s1ap_initialCtx_setup_Rq_total`	Number of S1 initial context setup requests.
	S5/S8 Session Requests	`xcore_q_pgw_s5_create_session_ReqRcvd_total`	Total S5/S8 session requests received.
	S5/S8 Failed Responses	`xcore_q_pgw_s5_create_session_RespSentFail_total`	Number of failed responses for S5/S8 session creation requests.
Error Metrics	License Errors	`xcore_license_error`	Tracks license-related errors in the system.
Uptime Metrics	System Uptime	`xcore_uptime_minutes`	Total uptime of the system in minutes.
Radio Metrics	eNodeB Radio Count	`xcore_privateradio_count` , `xcore_q_mme_procedures_s1_instances`	Number of private radios connected to the xCore.
Component Metrics	HSS/SGW/PGW/GGSN Count	`xcore_sgw_count` , `xcore_pgw_count` , `xcore_hss_active` , `xcore_ggsn_count`	Counts of various network components in use.
Sync Metrics	Sync Errors	`xcore_sync_mismatch` , `xcore_sync_missing` , `xcore_sync_extra`	Tracks synchronization mismatches, missing syncs, or extra syncs.

Category	Data Type	Prometheus Metric	Description
REST API Metrics	REST Errors	`xcore_rest_5xx_responses_total`	Tracks total 5xx responses from the REST API.
Version Metrics	Software Version	`xcore_version`	Tracks the current software version.
Log Metrics	Error Logs	`log_metrics_error_total` , `log_metrics_fatal_total`	Tracks error and fatal log counts.

List of All Available xCore Metrics

The following `xcore-` prefixed metrics can be used in PromQL expressions to create custom alerts for your xCore installation.

CLIENT AND THROUGHPUT METRICS

- `xcore_clients` : Number of clients connected to the xCore.
- `xcore_download_bytes_total` : Total bytes downloaded through the xCore.
- `xcore_upload_bytes_total` : Total bytes uploaded through the xCore.

AUTHENTICATION METRICS

- `xcore_q_hlr_map_c_cs_send_auth_FailMsgSent_total` : Number of failed MAP authentication messages sent.
- `xcore_q_hlr_map_c_cs_send_auth_RqRcvd_total` : Number of MAP authentication requests received.
- `xcore_q_hlr_map_c_cs_send_auth_RspSent_total` : Number of MAP authentication responses sent.
- `xcore_q_hss_s6a_auth_retrieval_FailMsgSent_total` : Number of failed S6A authentication messages sent.
- `xcore_q_hss_s6a_auth_retrieval_RqRcvd_total` : Number of S6A authentication requests received.
- `xcore_q_hss_s6a_auth_retrieval_RspSent_total` : Number of S6A authentication responses sent.

SESSION METRICS

- `xcore_q_mme_s1_s1ap_initialCtx_setup_Rq_total` : Total number of S1 initial context setup requests.
- `xcore_q_pgw_s5_create_session_ReqRcvd_total` : Number of S5/S8 session requests received.
- `xcore_q_pgw_s5_create_session_RespSentFail_total` : Number of failed responses for S5/S8 session requests.

NODE AND COMPONENT METRICS

- `xcore_privateradio_count` : Number of private radios connected to the xCore.
- `xcore_q_mme_procedures_s1_instances` : Instances of S1 procedures in MME.
- `xcore_sgw_count` : Count of SGW components in use.
- `xcore_pgw_count` : Count of PGW components in use.
- `xcore_hss_active` : Number of active HSS instances.
- `xcore_ggsn_count` : Count of GGSN components in use.

ERROR METRICS

- `xcore_license_error` : Tracks license-related errors in the system.
- `xcore_sync_mismatch` : Number of synchronization mismatches detected.
- `xcore_sync_missing` : Number of missing synchronization events detected.
- `xcore_sync_extra` : Number of extra synchronization events detected.

REST API METRICS

- `xcore_rest_5xx_responses_total` : Total number of 5xx errors from the REST API.
- `xcore_rest_2xx_responses_total` : Total number of 2xx successful responses from the REST API.
- `xcore_rest_404_responses_total` : Total number of 404 errors from the REST API.

UPTIME AND VERSION METRICS

- `xcore_uptime_minutes` : Total system uptime in minutes.
- `xcore_version` : Current version of the xCore software.

QUEUE AND PERFORMANCE METRICS

- `xcore_queue_status` : Status of the xCore queue system.
- `xcore_quortus_version` : Version information for Quortus.

LOGGING METRICS

- `log_metrics_error_total` : Total number of error-level log entries.
- `log_metrics_fatal_total` : Total number of fatal-level log entries.

MISCELLANEOUS METRICS

- `xcore_epc_connections` : Number of EPC connections established.
- `xcore_unhandledevents_total` : Total number of unhandled events.

For further details, refer to the Prometheus Node Exporter documentation for system-level metrics and exporter configurations.

Node Metrics

Prometheus node exporters provide insights into system-level metrics. Below are some commonly used metrics for monitoring the health of the system.

Category	Metric	Description
Disk Metrics	`node_filesystem_avail_bytes`	Available disk space in bytes.
CPU Metrics	`node_cpu_seconds_total`	Total CPU usage in seconds.
Network Metrics	`node_network_transmit_bytes_total`	Total transmitted network bytes.
	`node_network_receive_bytes_total`	Total received network bytes.
Memory Metrics	`node_memory_MemTotal_bytes`	Total system memory in bytes.
	`node_memory_MemFree_bytes`	Free system memory in bytes.
	`node_memory_Cached_bytes`	Cached memory usage in bytes.

Many more metrics are available. For a comprehensive list, refer to the Prometheus Node Exporter documentation.

Metric Exporters

Metric exporters collect and send metrics from the xCore to the Prometheus master. Each exporter focuses on specific categories of metrics.

Exporter Name	Prometheus Job Name	Notes
Node Exporter	`job="node"`	Provides VM node metrics and server health.
**Docker Exporter **	`job="docker"`	Collects metrics about Docker containers.
Prometheus	`job="prometheus"`	Collects metrics about Prometheus exporters.
xAgent Exporter	`job="xAgent"`	Provides metrics about the xCore host site.

4. API

4.1 Introduction

Welcome to the Expeto API Reference! This API gives you the power to seamlessly manage and interact with your private mobile networks, helping you unlock the full potential of the Expeto platform.

With the Expeto API, you can:

- Manage network resources like subscribers, devices, and sites.
- Retrieve real-time insights into network usage and performance.
- Handle customer accounts, authentication settings, and profiles.
- Automate workflows such as subscriber provisioning and rate limiting.

Designed with hypermedia-driven navigation (HATEOAS), the API enables dynamic exploration of resources and their relationships, making integration straightforward and efficient. Whether you're building custom applications, streamlining operations, or integrating with third-party systems, the Expeto API provides everything you need to succeed.

4.2 Getting Started

The xControl Expeto API is a HATEOAS (Hypermedia as the Engine of Application State) driven REST API that enables precise management and customization of your xCore network. Through API endpoints and hypermedia links provided in JSON responses, you can dynamically explore resource relationships and perform a variety of operations using resource IDs.

4.2.1 Authentication Methods

The Expeto API supports two authentication methods:

1. **Cookie-based Authentication**

   Leverages your xControl Admin login credentials and requires an active browser session using HAL Explorer.

   Cookie-based authentication allows you to access the Expeto API using your xControl Admin login credentials through HAL Explorer. This method requires an active administrator session in xControl and cookies enabled in your browser.

   > **Info**
   >
   > For more information on how to use HAL Explorer, see Exploring APIs with HAL.

2. **API Key (Token) Authentication**: Uses an API token for secure access, typically from your terminal or your applications.

   API access token authentication enables secure interaction with the Expeto API, either through a browser-based HATEOAS application or the command line. The access token must be included in the `X-API-KEY` header for each API request.

   Key Details:

   - **Roles**: Both `Admin` and `Customer_Admin` roles can use API tokens to access the API. However:
     - Only `Admin` accounts with the **Add API Token** feature enabled can generate and manage tokens.
     - `Customer_Admin` accounts can use tokens but cannot create or manage them.
   - **Token Scope**: The scope of your API token determines which endpoints and methods you can access.
   - **Multiple Tokens**: A single account can have multiple valid API tokens simultaneously.
   - **View Endpoints**: Make a `GET` request to the API Base URL to see available endpoints for your role:

     `api.example.com/v1/`

   > **Important**
   >
   > - All API requests must be made over HTTPS to ensure secure communication.
   > - The **Add API Token** feature must be explicitly enabled by Expeto Support for an admin account to generate tokens. If enabled, the API Tokens section will be visible on the account page in xControl.

4.2.2 Generating an API Token

The API token can be generated from xControl and through an API endpoint. Only users with 'Admin' privilege can generate the API token; it is not accessible to `Customer_Admin` accounts.

From xControl

As an `Admin` account user with the **Add API Token** feature enabled, you can:

- Generate API tokens for your account or other accounts.
- Delete API tokens for your account.
- Generate API tokens for accounts within the same Customer or Child Customers.

To generate a token for your Admin account from xControl:

1. Open xControl and navigate to your account page.
2. Locate the **API Tokens** section (only visible if the feature is enabled for your account).
3. Click **Add**, then provide a name for the token and click **Save**.
4. A panel will display the API token string and a warning that this is the only time the full token will be shown.
5. Copy the token and save it securely (e.g., in a password manager or text editor).
6. Close the panel.

The new token will appear in the **API Tokens** section, but its value will be obfuscated, showing only the last four characters for identification.

Using API

To generate a token for your Admin account using an API endpoint:

1. Open your API client.
2. Make a `POST` request to the following endpoint, providing your existing API token in the request header:

   `/v1/accounts/command/createToken`

EXAMPLE CURL REQUEST:

```
curl --request POST \
  --url https://api.example.com/v1/accounts/command/createToken \
  --header 'Content-Type: application/json' \
  --header 'X-API-KEY: <apiKey>' \
  --data '{
  "name": "exampleToken",
  "account": "123456"
  }'
```

REQUEST PARAMETERS:

Parameter	Description
`name`	A user-defined name for the token record. Visible in xControl.
`account`	The Admin account ID for which the token is being created.

Once the request is successfully processed, the API will return the new token. Be sure to copy and securely store the token, as it will not be displayed again.

4.2.3 Using the API Token

API tokens are available to both `Admin` and `Customer_Admin` roles and are required for secure access to API endpoints. Tokens are included in the request header using the `X-API-KEY` key.

Include the API token in the `X-API-KEY` header when making API requests via CLI. For example:

```
curl --request GET \
  --url https://api.example.com/v1/subs \
  --header 'X-API-KEY: <APIkeyValue>'
```

4.2.4 Token Expiration

If the API token is nearing expiration (within the next 30 days), the API server includes the following additional header in the response:

```
X-API-KEY-EXPIRING=true
```

> **Important**
>
> Expired tokens cannot access any API endpoint, including the refresh token endpoint. Ensure your tokens are refreshed or replaced before expiration to avoid service disruptions.

4.2.5 Refreshing the Token

> **Note**
>
> The `refreshToken` endpoint currently supports refreshing **site tokens only**. If your user account token is expiring, please contact Expeto Support.

To refresh an API token, send a `POST` request to the following endpoint:

```
/v1/api-tokens/command/refreshToken
```

REQUEST REQUIREMENTS:

- The request header must include:
  - **Key**: `X-API-KEY`
  - **Value**: The token that needs to be refreshed.
- Tokens cannot be refreshed if they are less than one hour old.

EXAMPLE CURL REQUEST:

```
curl --request POST \
  --url https://api.example.com/v1/api-tokens/command/refreshToken \
  --header 'X-API-KEY: <existingToken>' \
  --header 'Content-Type: application/json'
```

RESPONSE BODY:

The JSON response contains the following fields:

Field	Description
`token`	The new token, if the request was successful.
`message`	A description of the action taken or the reason a new token was not created.
`status`	A status code indicating the result. Refer to the table below for possible status values and meanings.

STATUS CODES:

STATUS	MEANING
`CREATED`	A new refresh token was successfully created.
`YOU_HAVE_THE_LATEST`	The token sent in the header is an unactivated refresh token. No new token is created. Use the token in a valid request.
`ALREADY_ACTIVATED`	The token sent in the header is an activated refresh token created within the past hour. No new token is created.
`ALREADY_REFRESHED`	The token sent in the header has already been used to generate a new refresh token within the past hour. No new token is created.

> **Important**
>
> Be sure to securely store the new token after it is created, as it replaces the previous token. Expired tokens cannot be refreshed or reused.

4.2.6 Deleting the token

To delete the API Token:

```
curl --request DELETE \
  --url http://localhost/v1/account-feature-api-tokens/{id} \
  --header 'X-API-KEY: <apiKey>'
```

4.2.7 Making an API Request

API calls are sent over HTTPS to Expeto's API server endpoints to access and interact with resources. Each request triggers a response from the API server, containing the requested data or the result of an operation. The main components of an API request include:

COMPONENTS OF AN API REQUEST

Component	Description
Request Method	The HTTP method that specifies the action being performed (e.g., `GET`, `POST`, `PUT`, `PATCH`, `DELETE`).
Base URL	The API server's host URL. For example: `https://api.example.com`.
Resource Path	The endpoint path that specifies the resource being accessed. Includes the API version number. Example: `/v1/accounts`.
Query Parameters	Optional parameters that refine the request, such as filtering by subscriber IMSI or systemID.

EXAMPLE REQUEST WITH CURL

By default, a cURL request uses the `GET` method unless another method is explicitly specified. Below is an example of a `GET` request with an API token:

```
curl --request GET \
  --url https://api.example.com/v1/accounts \
  --header 'X-API-KEY: <APIkey>'
```

Note

- Use the `X-API-KEY` header to include your API token for authentication.
- Specify other HTTP methods (`POST`, `PUT`, etc.) based on the action you wish to perform on the resource.

4.2.8 API Request Methods

Expeto's REST API supports the following HTTP methods to interact with resources:

Method	Description
GET	Retrieves a resource or all resources in a collection.
POST	Creates a new resource. It can also be used to update an existing resource by specifying key-value pairs.
PUT	Updates all attribute values of an existing resource. Requires a complete set of attributes, with unchanged values set to `null`.
PATCH	Updates specific attributes of a resource. Unspecified attributes retain their existing values. Ideal for editing single parameters.
DELETE	Removes an existing resource.

The `POST`, `PUT`, and `PATCH` methods require a `Content-Type` header with a value of `application/json` and a JSON-formatted body containing the key-value pairs to be created or updated.

A `POST` or `PATCH` request can update an existing resource without including all attributes. However, a `PUT` request requires all attributes to be specified, and any omitted attributes will be set to `null`.

4.2.9 Base URL and Resource Path

The **Base URL** for the Expeto API is:

```
https://api.example.com/
```

An API version number always follows the Base URL. For example:

```
https://api.example.com/v1
```

To complete the URL, append the endpoint name corresponding to the desired resource. For example, to access the `accounts` resource:

```
https://api.example.com/v1/accounts
```

> **Note**
>
> The endpoint path name may differ from the resource name. For example, the `subscribers` resource uses the endpoint:
>
> /v1/subs

4.2.10 Query Parameters

When retrieving resources, query parameters allow you to filter, sort, and customize the response. For example, a `GET` request for all subscribers may return thousands of results. Query parameters such as `page`, `size`, `sort`, and `projection` help refine how the response is displayed.

COMMON QUERY PARAMETERS

**Parameter **	Description	Default Value
`page`	The page number to start with. The first page is `0`.	`0`
`size`	The number of resources displayed on each page.	`20`
`sort`	The order in which resources are listed. Use `asc` (ascending) or `desc` (descending). Multiple fields can be sorted.	`asc`
`projection`	Specifies the level of detail in the response:	`detail`
	- `brief`: A minimal response.	
	- `detail`: A standard response with default details.	
	- `deep`: A verbose response with additional attributes.	

EXAMPLE REQUESTS

URI TEMPLATE

The following URI template includes placeholders for query parameters:

```
https://api.expeto.io/v1/subs{?page,size,sort,projection}
```

SAMPLE REQUEST

A `GET` request to retrieve subscribers on page 23, with 10 results per page, sorted by `imsi` in descending order:

```
https://api.example.com/v1/subs?page=23&size=10&sort=imsi,desc
```

If no `projection` parameter is specified, the response defaults to `detail`.

EXAMPLE JSON RESPONSES

PAGINATION METADATA

The `page` object in the response body provides details about pagination:


```
{
  "page": {
    "size": 10,
    "totalElements": 64752,
    "totalPages": 6476,
    "number": 23
  }
}
```


SUBSCRIBER EXAMPLE

A single subscriber resource in the `detail` projection might appear as:


```
{
  "imsi": "123480002339875",
  "msisdn": "+123480002339875",
  "k": "###MASKED###",
  "opc": null,
  "iccid": "89618802340002598755",
  "status": "Inactive",
  "type": "Production",
  "extensions": {
     "batch": "2022-04-12"
  }
}
```


SUBSCRIBER WITH `deep` PROJECTION

A `deep` projection includes additional details such as system profile settings:


```
{
  "systemProfile": {
    "_links": {
      "self": {
        "href": "https://api.expeto.io/v1/system-profiles/123456789{?projection}",
        "templated": true
      }
    },
    "name": "internet-209715200-104857600"
  }
}
```


4.2.11 API Error Responses

The Expeto API server returns responses in JSON format, accompanied by HTTP status codes to indicate the outcome of each request:

HTTP STATUS CODES

Status Code	Description
`200 OK`	The request was successfully completed.
`4xx` Error	Client-side error. Review your request for issues.
`5xx` Error	Server-side error. Contact support if the issue persists.

SUCCESSFUL RESPONSE

A successful `200 OK` response contains the resource's key-value attributes, as defined by the Expeto API schema. For example:


```
{
  "version": -89287109,
  "deleted": -16312438,
  "tac": "35607009",
  "brandName": "SAMSUNG",
  "modelName": "SM-G5323G/DS",
  "deviceType": "Smartphone",
  "os": "Android 6.0 Marshmallow",
  "source": "manual",
  "recordTimestamp": "2012-03-28T16:19:27.128Z",
  "id": "Grob's Phone",
```

```
  "_links": {}
}
```

ERROR RESPONSES

If an error occurs, the API server returns a JSON response with additional information about the issue. For example, if you lack permissions to access a resource, a `403 AccessDenied` error is returned:

```
{
  "status": 403,
  "error": "AccessDenied",
  "message": "The Security Framework denied access for this request. You are probably not allowed to access this resource.",
  "path": "/v1/notifications",
  "timestamp": 1626387747246,
  "incident": "40530304"
}
```

4.2.12 Additional Resources

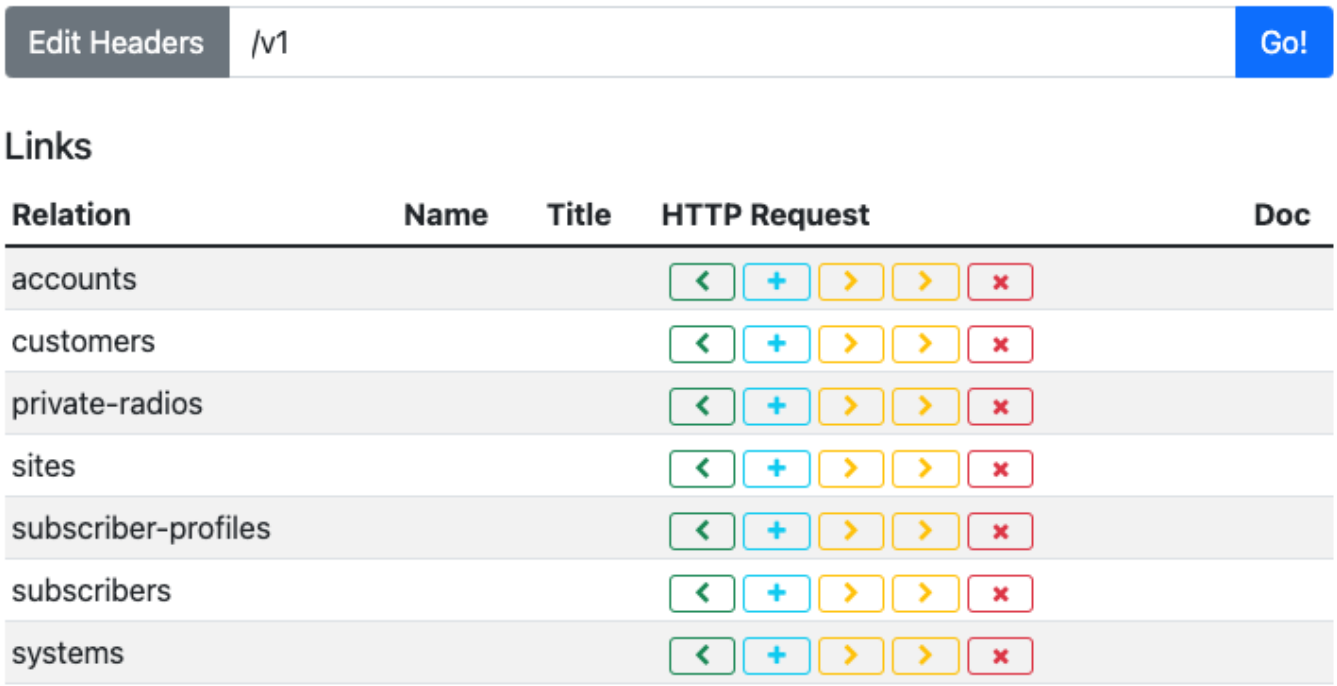
For more information on error codes and troubleshooting, see Troubleshooting API.

4.3 Exploring APIs with HAL

The REST API is **HATEOAS-enabled** (Hypermedia as the Engine of Application State), meaning API clients can dynamically interact with the API using hypermedia links. These links enhance API exploration by providing extended URIs to related resources. For example, a `GET` request on a Subscriber resource returns its attributes along with hyperlinks to associated resources such as Customer, Group, or keysets, even though these are not defined as Subscriber resource attributes.

Take full advantage of hypermedia links by using a client like **HAL Explorer**. Follow these steps to explore resources:

1. Log into **xControl** with your admin credentials.
2. Open a new browser tab and navigate to the API root URL in HAL Explorer. For example: `https://api.example.com/v1/explorer`
3. The **Links** section displays all available resources. Click the green arrow (**GET**) to open the HTTP Request Input template for a resource.
4. Click **Go!** to execute the request.



Alternatively, you can use the **Edit Headers** field to type the path to target a specific resource root endpoint. For example, the following path targets the `subs` endpoint and click:

```
/v1/subs
```

> **Note**
>
> - Endpoint paths may differ from resource names. For example, the `subscribers` resource uses the `/v1/subs` endpoint.
> - Refer to HAL Explorer documentation site for more detail on how to use.

Error Handling

If you do not have permissions for a resource, the API returns an **AccessDenied** error:

```
{
  "status": 403,
  "error": "AccessDenied",
  "message": "The Security Framework denied access for this request. You are probably not allowed to access this resource.",
  "path": "/v1/customers",
  "timestamp": 1626387747246,
```

```
  "incident": "40530304"
}
```


4.3.1 Pagination

Responses in HAL include a **page** element to manage data display and navigation. For example, a `GET` request to `/subs` returns all subscribers, organized across multiple pages. The `page` element specifies:

- `size` : Maximum number of resources per page.
- `totalElements` : Total number of resources.
- `totalPages` : Total number of pages.
- `number` : Current page (0-based index).

Example Response:


```
{
 "page": {
  "size": 20,
  "totalElements": 74752,
  "totalPages": 3738,
  "number": 0
 }
}
```


Navigate pages in HAL Explorer using the **first**, **prev**, **next**, and **last** links in the response.

4.3.2 Resource Attributes

Each resource includes attributes that form its schema. For example, the `subscribers` resource returns:


```
{
 "imsi": "123456313212312",
 "msisdn": "+199938382204417",
 "k": "###MASKED###",
 "opc": null,
 "iccid": "12348901260230613137",
 "status": "Active",
 "type": null,
 "extensions": {
  "sim-type": "UICC",
  "user": "SirGrob"
 }
}
```


4.3.3 Hypermedia Links

Resources include a `_links` section with hypermedia links for related operations. For example, the `Subscriber` resource for a `customer_admin` role includes:

- `self`
- `subscriber`
- `profile`

To retrieve the System Profile, send a `GET` request using the `systemProfile` link:

```
{baseUrl}/v1/subs/<IMSI>/systemProfile
```

Example Response:


```
{
 "name": "default",
 "defaultApn": "internet",
 "uploadBitRate": 20971520,
 "downloadBitRate": 31457280,
 "extensions": {}
}
```

4.3.4 Example Request

The following example shows how to use a deep projection request and hyperlinks to get extensive information about a specific customer.

To enhance response details for the customer resource using projection parameters:

1. Make an GET request to return a list of customers:

```
{baseUrl}/v1/customers
```

2. Select one of the customers to view the standard response for that customer, consisting of the name and any extensions. The JSON properties and the Links sections appear.

customers [2]

JSON Properties

```
{
  "name": "xRoboticsCorp",
  "extensions": {}
}
```

Links

Relation	Name	Title	HTTP Request	Doc
self				
customer				
children				
sites				
parent				
systems				
accounts				
features				

3. Click the green arrow (GET) on the **customer** hyperlink. The template appears with an Expanded URI containing the customerID:

   {baseUrl}/v1/customers/87089251

4. For the **projection** parameter, type **deep**. The parameter is appended to the URI:

   {baseUrl}/v1/customers/123854?projection=deep

5. Click **Go**.

Comprehensive responses include statistics and hierarchical relationships:

```
{
  "siteCount": 1,
  "systemCount": 5,
  "subscriberCount": 26
}
```

If no results are found, the response will indicate `"totalElements": 0` :

```
{
  "page": {
    "size": 10,
    "totalElements": 0,
    "totalPages": 0,
    "number": 0
  }
}
```

There is no error because the search was performed successfully, even if no results were found.

4.4 Endpoints

4.4.1 Searching Efficiently

This page concentrates on searching for Subscribers, which is the resource with the most search options. However, the same logic can be applied to searching for other resources such as Sites or Systems.

The API method for searches is always GET.

Find By Deep Search

//api.example.com/v1/subs/search/findByDeepSearch

Deep search allows you to search for Subscribers using any extensions that have been set for the user. Extensions are key value pairs expressed as key=value. It also allows you to search using partial match for specific values such as ICCID.

The following API call returns all Subscribers with the extension user=sam:

Example: *user=sam*

```
curl --request GET \
  --url 'http://localhost/v1/subs/search/findByDeepSearch?searchString=user%3Dsam
  --header 'Accept: application/hal+json'
```

Note that the \"=\" character in the extension is expressed as its Unicode equivalent: **%3D**.

The same deep search could be performed by omitting the key and the Unicode and just using the value, however this is less precise.

Example: *sam*

```
curl --request GET \
  --url 'http://localhost/v1/subs/search/findByDeepSearch?searchString=sam
  --header 'Accept: application/hal+json'
```

You may return an instance where the same value you\'re searching with is set for a different extension key. For example, the deep search shown here for *sam* would return Subscribers with the user of Sam but also with a device type of Samsung.

Find by Site, System, or Group IDs

Subscribers belong to certain organizational resources in your network. The Site is the largest entity, often containing multiple Systems. A single System can span Sites. A Subscriber is assigned a single System Profile.

The following findBys extend the /subs/search/ endpoint path to search for Subscribers within organizational resources:

- /v1/subs/search/findBySite
- /v1/subs/search/findBySystem
- /v1/subs/search/findByGroup
- /v1/subs/search/findBySiteAndSystem

NOTE

To search by the organizational resource, you must user the associated ID value not the name of the Site, System, or Group. You can find the ID value by making an additional API call on the organizational entity.

--

You can discover the ID by using findByName on the resource, then applying the ID to the path.

Find By Name

The findByName path allows you to discover the resource ID value as well as other details.

- /v1/customers/search/findByName
- /v1/sites/search/findByName
- /v1/sys/search/findByName

To discover the ID value, make a findByName call on the resource. For example, make the following call and specify the System Name:

*//api.example.com/v1/sys/search/findByName?name=*systemName**

The following API call searches for the System ID for the site \"xAuto\".

```
curl --request GET \
  --url 'http://localhost/v1/sys/search/findByName?name=xAuto' \
  --header 'Accept: application/hal+json'
```

This returns the system ID of 12319402.

APPLY THE ID VALUE TO THE PATH

Now you know the system ID, you can use it in the search field

//api.example.com/v1/subs/search/findBySystem?system= *SystemID*

For example, the following call returns all Subscribers assigned to the system **xAuto** that has a system ID of 12319402.

```
curl --request GET \
  --url 'http://localhost/v1/subs/search/findBySystem?system=12319402
  --header 'Accept: application/hal+json'
```

Find By SIM card Values

You can search for a Subscriber based on the following SIM card values:

- /v1/subs/search/findByIccid
- /v1/subs/search/findByMsisdn

ICCID -- a unique 18-22 digit code often printed on the back of the SIM card that includes a country code, home network, and identifier.

> //api.example.com/v1/subs/search/findByIccid?iccid= *ICCID number*

MSISDN -- a unique code maximum 15 characters that identifies a SIM card in its mobile network. Often the phone number to which you call or send an SMS message.

> //api.example.com/v1/subs/search/findByMsisdn?msisdn= *MSISIDN number*

The entire value must be entered. Use findByDeepSearch to enter partial values.

The following example searches for a Subscriber using a known MISIDN value.

```
curl --request GET \
  --url 'http://localhost/v1/subs/search/findByMsisdn?msisdn=123438386906424'
  --header 'Accept: application/hal+json'
```

4.4.2 Accounts

Path

```
/v1/accounts
```

An administrator account is used to manage the network. It consists of a username and email address. The account is associated with a Customer and is assigned an account role that determines feature access and permissions.

Resource Attributes

The attributes for the accounts resource are detailed below:

Attribute	Type	Required?	Description
id	Integer	YES	The account ID. Use this value to specify the account in requests.
username	String	YES	The username, identical to the email address.
externalId	String	NO	Allows you to set an alternate ID value for the account, used for searches.
emailAddress	String	YES	The email address provided for the account, used to generate the username.

EXAMPLE

```
{
  "id": 123271682,
  "username": "dasGrob@expeto.io",
  "externalId": null,
  "emailAddress": "dasGrob@expeto.io"
}
```

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/accounts/<id>/role
```

Where `<id>` is the account ID associated with the user.

```
{baseUrl}/v1/accounts/<id>
```

```
{baseUrl}/v1/accounts/search/findByDeepSearch?searchString=<searchCriteria>
```

Where `<searchCriteria>` is the text string used to find the account.

```
{baseUrl}/v1/accounts/command/currentUser
```

UPDATE ACCOUNT PASSWORD

To update the account password, make a POST request:

```
curl --request POST \
  --url http://localhost/v1/accounts/command/update
```

4.4.3 APN Globals

Path

```
/v1/apn-globals
```

An APN Global is where you define APN names for your subscriber fleet, and their associated default maximum upload and download values to be used in conjunction with APN Profiles and System Profiles.

Multiple APN Globals can be created if your SIM requires more than one APN.

Resource Attributes

The attributes for the sub-profiles resource are detailed below:

Attribute	Type	Required?	Description
name	String	YES	The name of the apn global.
upload	Integer	YES	The maximum upload bitrate in bits per second.
download	Integer	YES	The maximum download bitrate in bits per second.
apnQos	String	NO	The default QoS/QCI associated with this APN.

EXAMPLE

```
{
  "name": "internet",
  "upload": 210000000,
  "download": 310000000,
  "apnQos": "{baseUrl}/v1/apn-qos/8"
}
```

Endpoints

REQUEST EXAMPLES

{baseUrl}/v1/apn-globals

{baseUrl}/v1/apn-globals/

The `<apnGlobalID>` is discovered by selecting or searching for an apn-global.

```
{baseUrl}/v1/apn-globals/search/findByDeepSearch?searchString=<searchCriteria>
```

4.4.4 APN Profiles

Path

```
/v1/apn-profiles
```

An APN Profile is used to associate an APN Global to a System Profile in an ordered list. An APN Profile includes an APN from the available APN Globals, and specifies an optional maximum download and upload bitrate, where the maximum download and upload bitrate defined in the APN Profile can override the values defined in the APN Global.

If a System Profile needs to support multiple APNs, you must define multiple APN Profiles in an ordered list in the System Profile. The `ordering` attribute shows in which (0 indexed) order the APN Profile is in the list within the System Profile.

Resource Attributes

The attributes for the sub-profiles resource are detailed below:

Attribute	Type	Required?	Description
ordering	Integer	YES	The index order of the APN in the APN list.
upload	Integer	NO	The maximum upload bitrate in bits per second.
download	Integer	NO	The maximum download bitrate in bits per second.
apnQos	String	NO	The QoS profile to apply, overriding the APN Global default.
apnOverrides	Set	NO	A set of carrier-specific QoS and bitrate overrides.

EXAMPLE

```
{
  "ordering": 0,
  "upload": 20971520,
  "download": 31457280,
  "apnQos": "{baseUrl}/v1/apn-qos/1",
  "apnOverrides": [
    {
      "carrier": "{baseUrl}/v1/carriers/123",
      "apnQos": "{baseUrl}/v1/apn-qos/456",
      "download": 10485760,
      "upload": 5242880
    }
  ]
}
```

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/apn-profiles
```

```
{baseUrl}/v1/apn-profiles/<profileID>
```

```
{baseUrl}/v1/apn-profiles/<profileID>/apnGlobal
```

```
{baseUrl}/v1/apn-profiles/<profileID>/systemProfile
```

The `<profileID>` is discovered by selecting or searching for all apn profiles.

4.4.5 Carriers

Path

```
/v1/carriers
```

A Carrier represents a Mobile Network Operator (MNO) identified by its Mobile Country Code (MCC) and Mobile Network Code (MNC).

Resource Attributes

The attributes for the carriers resource are detailed below:

Attribute	Type	Required?	Description
mcc	String	YES	Mobile Country Code (3 digits).
mnc	String	YES	Mobile Network Code (2-4 digits).
type	String	YES	Carrier type. Options: `INTERNATIONAL`, `NATIONAL`, `TEST`.
status	String	YES	Carrier status. Options: `ALLOCATED`, `IMPLEMENT_DESIGN`, `NOT_OPERATIONAL`, `ONGOI` `TEST_NETWORK`, `UNKNOWN`.
mode	String	YES	Carrier mode. Options: `PUBLIC`, `PRIVATE`, `UNKNOWN`.
countryName	String	NO	Name of the country.
countryCode	String	NO	Two-letter country code (ISO 3166-1 alpha-2).
brand	String	NO	Carrier brand name.
operator	String	NO	Carrier operator name.
bands	String	NO	Supported frequency bands.
notes	String	NO	Additional notes about the carrier.

EXAMPLE


```
{
  "mcc": "302",
  "mnc": "220",
  "type": "NATIONAL",
  "status": "OPERATIONAL",
  "mode": "PUBLIC",
  "countryName": "Canada",
  "countryCode": "CA",
  "brand": "Telus",
  "operator": "Telus Communications",
  "bands": "1, 2, 4, 5, 7, 12, 13, 17, 29, 30, 66",
  "notes": "Main carrier for region"
}
```


Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/carriers
```

```
{baseUrl}/v1/carriers/<carrierID>
```

```
{baseUrl}/v1/carriers/search/findByDeepSearch?searchString=<searchCriteria>
```

```
{baseUrl}/v1/carriers/search/findByOperator?searchString=<operatorName>
```

The is discovered by selecting or searching for a carrier.

4.4.6 Carrier Groups

Path

```
/v1/carrier-groups
```

A Carrier Group is a collection of Carriers. Carrier Groups are used within Roaming Profiles to define sets of allowed or restricted networks.

Resource Attributes

The attributes for the carrier-groups resource are detailed below:

Attribute	Type	Required?	Description
name	String	YES	The name of the carrier group.
description	String	NO	A description of the carrier group.

EXAMPLE

```
{
  "name": "North American Carriers",
  "description": "All major carriers in US and Canada"
}
```

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/carrier-groups
```

```
{baseUrl}/v1/carrier-groups/<groupID>
```

```
{baseUrl}/v1/carrier-groups/<groupID>/carriers
```

```
{baseUrl}/v1/carrier-groups/search/findByDeepSearchWithCounts?searchString=<searchCriteria>
```

The is discovered by selecting or searching for a carrier group.

4.4.7 Customers

Path

```
/v1/customers
```

The customer resource is associated with one or more sites. A customer hierarchy can exist where a parent customer oversees one or more child customers. Parent customers can manage the sites of all associated child customers.

Resource Attributes

The attributes for the customers resource are detailed below:

Attribute	Type	Required?	Description
name	String	YES	The name of the customer.
extensions	Object	NO	A container for any additional properties or metadata.

EXAMPLE

```
{
  "name": "acmeCorp",
  "extensions": {}
}
```

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/customers
```

This command returns all customers that your account can access, including parent and child customers.

```
{baseUrl}/v1/customers/<customerID>/accounts
```

The `<customerID>` is discovered by selecting a specific customers resource.

```
{baseUrl}/v1/customers/<customerID>/sites
```

4.4.8 Notifications

Path

```
/v1/notifications
```

The notifications resource contains information about data thresholds.

> **Note**
>
> By default, this feature is not available. It must be enabled by Expeto.

Resource Attributes

The attributes for the notifications resource are detailed below:

Attribute	Type	Required?	Description
issued	String	YES	The timestamp when the notification was issued (ISO 8601 format).
type	String	YES	The type of notification, such as `ThresholdReached`.
hash	String	YES	A unique hash identifying the notification.
payload	Object	YES	Contains details about the notification.
payload.imsi	String	YES	The IMSI (International Mobile Subscriber Identity) of the subscriber.
payload.dataLimit	Integer	YES	Defines the data limit usage for a subscriber in bytes.
payload.dataLimitPeriod	String	YES	Defines the period for calculating data limit usage. Can be **absolute** or **monthly**.
payload.dataLimitCustomDate	Integer	NO	Day of the month (1–31) for data-limit reset if `dataLimitPeriod` is monthly.
payload.totalUsage	Integer	YES	The total data usage of the subscriber in bytes.
payload.thresholdValue	Integer	YES	Threshold defined as a percentage of data usage reached, e.g., 90 for 90%.
consumed	Boolean	YES	Indicates whether the notification has been consumed.

EXAMPLE


```
{
  "issued": "2020-01-30T14:03:02.231Z",
  "type": "ThresholdReached",
  "hash": "AB1230A01ABC51DDB00A19126143A12319ECAB02C2FDC446CD7C16E42A0E8FB6",
  "payload": {
    "imsi": "123470101231123",
    "dataLimitPeriod": "monthly",
    "dataLimitCustomDate": 1,
    "dataLimit": 1073741824,
    "totalUsage": 966367641,
    "thresholdValue": 90
  },
  "consumed": false
}
```

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/notifications
```

You can filter your response using any of the following search options:

- findByHash
- findByConsumed
- findBySystem
- findBySystemAndConsumed
- findBySubscriber
- findBySubscriberAndConsumed
- findByDeepSearch

```
{baseUrl}/v1/notifications/search/findBySystem?system=<systemID>
```

```
{baseUrl}/v1/notifications/search/findBySubscriber?subscriber=<IMSI>
```

4.4.9 Private Radios

Path

```
/v1/private-radios
```

The private radio resource returns the eNodeB information for the site. A site can have multiple eNodeBs.

Resource Attributes

The attributes for the private radios resource are detailed below:

Attribute	Type	Required?	Description
enodebId	String	YES	The unique identifier of the eNodeB.
status	String	YES	The current status of the eNodeB. Can be **Enabled** or **Disabled**.
extensions	Object	NO	A container for any additional properties or metadata.

EXAMPLE

```
{
  "enodebId": "10101-101",
  "status": "Enabled",
  "extensions": {}
}
```

NOTE

- The **enodebId** is not used as the resource ID in API requests.
- The **resourceID** is found using the GET command.

For example, a GET response might show the following:

- enodebId: `00000-99121`
    - resourceID: `185123749`

```
{
   "enodebId": "00000-99121",
   "status": "Enabled",
   "extensions": {},
   "_links": {
      "self": {
         "href": "https://api.expeto.io/v1/private-radios/185123749"
      },
      "private-radio": {
         "href": "https://api.expeto.io/v1/private-radios/185123749{?projection}",
         "templated": true
      },
      "feature": {
         "href": "https://api.expeto.io/v1/private-radios/185123749/feature"
      },
      "latestCdr": {
         "href": "https://api.expeto.io/v1/private-radios/185123749/latestCdr"
      }
   }
}
```

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/private-radios/search/findBySite?site=<siteID>
```

Where `<siteID>` is a number representing the site. You can discover the site ID using the `/site` endpoint.

```
{baseUrl}/v1/private-radios/<resourceID>
```

1. Select a private-radios instance.
2. Click GET for the **private-radio** link. The `<resourceID>` is discovered and populated in the request URL.
3. For the projection parameter, type **deep**.

The following request:

```
{baseUrl}/v1/private-radios/<resourceID>?projection=deep
```

Might elicit the following response:

```
{
  "lastSeen": "2021-02-19T20:58:36.041547Z",
  "connected": true,
  "name": "ipaccess31",
  "status": "Enabled",
  "ipAddress": "192.111.222.333",
  "enodebId": "123264-45"
}
```

4.4.10 Profile

Path

```
/v1/profile
```

The profile resource refers to your account profile (not a subscriber profile). It immediately shows the resources you can access from your current Administrator role. It has no attributes and lists the HATEOAS hypermedia links.

Resource Attributes

None.

Supported Methods

The hypermedia links available to your account appear when you go to the resource URL.

4.4.11 Roaming Profiles

Path

```
/v1/roaming-profiles
```

A Roaming Profile defines 3GPP roaming restrictions for Subscribers, Sites, or Systems. It specifies which Carrier Groups a subscriber is allowed to attach to.

Resource Attributes

The attributes for the roaming-profiles resource are detailed below:

Attribute	Type	Required?	Description
name	String	YES	The name of the roaming profile.
description	String	NO	A description of the roaming profile.
customer	String	YES	The customer associated with this roaming profile (link).

EXAMPLE

```
{
  "name": "Standard Roaming",
  "description": "Standard roaming restrictions for employees",
  "customer": "{baseUrl}/v1/customers/123"
}
```

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/roaming-profiles
```

```
{baseUrl}/v1/roaming-profiles/<profileID>
```

```
{baseUrl}/v1/roaming-profiles/<profileID>/allowedCarrierGroups
```

```
{baseUrl}/v1/roaming-profiles/<profileID>/subscribers
```

```
{baseUrl}/v1/roaming-profiles/<profileID>/sites
```

```
{baseUrl}/v1/roaming-profiles/<profileID>/systems
```

```
{baseUrl}/v1/roaming-profiles/search/findByDeepSearchWithCounts?searchString=<searchCriteria>
```

```
{baseUrl}/v1/roaming-profiles/search/findByNameAndCustomerWithParents?name=<name>&customer=<customerRef>
```

The is discovered by selecting or searching for a roaming profile.

4.4.12 Sites

Path

```
/v1/sites
```

Resource Attributes

The attributes for the sites resource are detailed below:

Attribute	Type	Required?	Description
name	String	YES	The name of the site.
uuid	String	YES	A unique identifier for the site.
location	Object	NO	The location of the site. Can be null if not provided.
extensions	Object	NO	A container for any additional properties or metadata.
roleNameFull	String	YES	The full role name associated with the site (e.g., ROLE_SITE).
roleName	String	YES	The short role name associated with the site (e.g., SITE).
roamingProfile	String	NO	The roaming profile assigned to the site (link).

EXAMPLE


```
{
  "name": "AcmeCorp",
  "uuid": "1234a123-123b-12a1-aa12-a1a2a345a1d2",
  "location": null,
  "extensions": {},
  "roleNameFull": "ROLE_SITE",
  "roleName": "SITE",
  "roamingProfile": "{baseUrl}/v1/roaming-profiles/123"
}
```


Endpoints

REQUEST EXAMPLES

Use `findByName` or `findByDeepSearch`:

```
{baseUrl}/v1/sites/search/findByDeepSearch?searchString=<searchCriteria>
```

Where `<searchCriteria>` is the text string used to find the site.

```
{baseUrl}/v1/sites/<siteID>/systems
```

Where `<siteID>` is the unique identifier for the site.

4.4.13 Subscribers

Path

```
/v1/subs
```

The Subscriber refers to the User Equipment (UE) accessing the network, such as a cell phone with a SIM card.

Resource Attributes

The attributes for the subscribers resource are detailed below:

Attribute	Type	Required?	Description
imsi	String	YES	The unique IMSI value to identify the subscriber.
msisdn	String	YES	The phone number (MSISDN) associated with the subscriber.
k	String	NO	The secret key for the subscriber. Masked by default.
opc	String	NO	The Operator Code (OPC), used in authentication. Can be null.
iccid	String	NO	The Integrated Circuit Card Identifier (ICCID) of the SIM card. Can be null.
status	String	YES	The current status of the subscriber (e.g., Active or Inactive).
type	String	YES	The type of subscriber (e.g., Production).
systemProfile	String	YES	The system profile assigned to the subscriber (link).
roamingProfile	String	NO	The roaming profile assigned to the subscriber (link).
extensions	String	NO	A container for additional metadata or properties.

EXAMPLE

```
{
  "imsi": "206018123123123",
  "msisdn": "+126018012345789",
  "k": "###MASKED###",
  "opc": null,
  "iccid": null,
  "status": "Active",
  "type": "Production",
  "systemProfile": "{baseUrl}/v1/system-profiles/123",
  "roamingProfile": "{baseUrl}/v1/roaming-profiles/123",
  "extensions": ""
}
```

Endpoints

SEARCH REQUESTS

```
{baseUrl}/v1/subs/search/findByDeepSearch?search=<searchCriteria>
```

```
{baseUrl}/v1/subs/search/findBySystem?system=<systemName>
```

```
{baseUrl}/v1/subs/search/findByMsisdn?msisdn=<msisdnValue>
```

Search filters include:

- findByDeepSearch
- findByDeepSearchBrief
- findByFeature
- findByIccid
- findByMsisdn
- findBySite
- findBySystem
- findBySiteAndSystem

CREATE A NEW SUBSCRIBER

Make a POST request to create a new subscriber:

```
{baseUrl}/v1/subs/
```

Example JSON body:

```
{
  "imsi": "1232601012344251",
  "msisdn": "+126018012345789",
  "k": "yoursecretkey",
  "opc": null,
  "iccid": null,
  "status": "Active",
  "type": "Production",
  "extensions": ""
}
```

UPDATE EXISTING ATTRIBUTES

Make a PATCH request to update the attributes for a subscriber. Example:

```
{baseUrl}/v1/subs/<IMSI>
```

JSON body:

```
{
  "status": "Inactive"
}
```

DELETE A SUBSCRIBER

Make a DELETE request to remove a subscriber:

```
{baseUrl}/v1/subs/<IMSI>
```

BULK IMPORT SUBSCRIBERS

Make a POST request to bulk import subscribers:

```
{baseUrl}/v1/subs/command/bulkImport
```

Parameter	Type	Required?	Description
overwriteIfExisting	Boolean	YES	Overwrite existing subscribers if true. Default is false.
file	String	YES	The .out file containing SIM card data.
amf	String	NO	Authentication Management Field (4 hex digits).
op	String	NO	Operator Code (128-bit unique code).
system	String	YES	Target system name.
id	String	NO	Value assigned to the import batch.
systemProfile	String	NO	Name of the system profile.
keyset	String	NO	Name of the encryption keyset.
status	String	YES	Initial status (Active or Inactive).
extensions	Object	NO	Custom extension parameters (e.g., key-value pairs).

BULK UPDATE SUBSCRIBERS

Make a POST request to apply the same update to multiple subscribers:

```
{baseUrl}/v1/subs/command/bulkUpdate
```

JSON body:

```
{
  "status": "Active",
  "subscribers": [
    "123260331041372",
    "123456234556627"
  ]
}
```

BULK DELETE SUBSCRIBERS

Make a POST request to bulk delete subscribers:

```
{baseUrl}/v1/subs/command/bulkDelete
```

JSON body:

```
{
  "subscribers": [
    "123260331041372",
    "123456234556627"
  ]
}
```

GET ALL SUBSCRIBERS FOR A SYSTEM

```
{baseUrl}/v1/subs/search/findBySystem?system=<systemID>
```

GET SUBSCRIBER SESSION EVENT LOGS

```
{baseUrl}/v1/subs/search/findByImsiWithSessionReport?searchString=<subscriberID>
```

Example:

```
curl --request GET \
    --url 'http://api.example.com/v1/subs/search/findByImsiWithSessionReport?searchString=123302123091235' \
    --header 'accept: application/hal+json'
```

GET THE SMS HISTORY REPORT FOR A SUBSCRIBER

```
{baseUrl}/v1/subs/search/smsCdrs?subscriber=<subscriberID>
```

Example:

```
curl --request GET \
    --url 'http://api.example.com/v1/subs/search/smsCdrs?subscriber=123302123091235' \
    --header 'accept: application/hal+json'
```

GET A LIST OF ACTIVE/INACTIVE SUBSCRIBERS

Use `findByDeepSearch` with keywords `active` or `inactive`:

```
{baseUrl}/v1/subs/search/findByDeepSearch?searchString=active&projection=brief
{baseUrl}/v1/subs/search/findByDeepSearch?searchString=inactive&projection=brief
```

GET A LIST OF CONNECTED/DISCONNECTED SUBSCRIBERS

Use `findByDeepSearch` with keywords `connected` or `disconnected`:

```
{baseUrl}/v1/subs/search/findByDeepSearch?searchString=connected&projection=brief
{baseUrl}/v1/subs/search/findByDeepSearch?searchString=disconnected&projection=brief
```

4.4.14 System Profiles

Path

```
/v1/system-profiles
```

A System Profile is used to apply a fixed set of parameters to one or more System(s) and/or Subscribers. It comprises one or more APN Profiles.

Multiple System Profiles can be created and made available for Subscribers of the same System. For example, establish tiered service levels by creating three System Profiles (bronze, silver, and gold) each with successively higher bandwidth levels.

Resource Attributes

The attributes for the sub-profiles resource are detailed below:

Attribute	Type	Required?	Description
name	String	YES	The name of the system profile.
extensions	Object	NO	A container for any additional custom parameters or metadata.

EXAMPLE

```
{
  "name": "ITprofile",
  "extensions": {}
}
```

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/system-profiles
```

```
{baseUrl}/v1/system-profiles/<profileID>
```

```
{baseUrl}/v1/system-profiles/<profileID>/apnProfiles
```

```
{baseUrl}/v1/system-profiles/<profileID>/customer
```

```
{baseUrl}/v1/system-profiles/<profileID>/subscribers
```

```
{baseUrl}/v1/system-profiles/<profileID>/systems
```

The `<profileID>` is discovered by selecting or searching for a system profile.

```
{baseUrl}/v1/system-profiles/search/findByDeepSearch?searchString=<searchCriteria>
```

4.4.15 System

Path

```
/v1/sys
```

The System resource maps to a network subnet. Subscribers are assigned to one System. A System can be shared by multiple Sites but is owned by only one Customer.

Resource Attributes

The attributes for the System resource are detailed below:

Attribute	Type	Required?	Description
name	String	YES	The name of the System.
status	String	YES	The current status of the System (e.g., Active or Inactive).
roamingProfile	String	NO	The roaming profile assigned to the System (link).
extensions	Object	NO	A container for additional metadata or custom parameters.

EXAMPLE

```
{
  "name": "RobotNetwork1",
  "status": "Active",
  "roamingProfile": "{baseUrl}/v1/roaming-profiles/123",
  "extensions": {}
}
```

NOTE

Systems are identified in API calls by a `<systemID>`. The `<systemID>` can be discovered when a specific System instance is selected in the HAL Explorer.

Endpoints

REQUEST EXAMPLES

```
{baseUrl}/v1/sys/
```

```
{baseUrl}/v1/sys/search/findByName?name=RobotNetwork1
```

```
{
  "name": "RobotNetwork1",
  "status": "Active",
  "extensions": {
    "subscriberServiceEnabled": "true",
    "network": "Private Cloud"
  }
}
```

```
{baseUrl}/v1/sys/<systemID>/sites
```

Where `<systemID>` is a unique identifier for the System. A System can be shared by multiple Sites.

Create System

Send a POST to `{baseUrl}/v1/sys` with the following parameters to create a new System: name, status, and customer.

```
{
   "name":"RobotNetwork1",
   "status":"Active",
   "customer":"{baseUrl}/v1/customers/<customerID>",
}
```

4.4.16 System Features

Path

```
/v1/sys-features
```

The `sys-features` resource is used to explore the system's gateway and subscribers.

Resource Attributes

The attributes for the `sys-features` resource are detailed below:

Attribute	Type	Required?	Description
extensions	Object	NO	A container for additional metadata or custom parameters.
type	String	YES	The type of system feature, such as `gateway_feature`.

EXAMPLE

```
{
  "extensions": {},
  "type": "gateway_feature"
}
```

4.5 Troubleshooting API

API call errors return a response including the error status and additional details.

4.5.1 Error Code Samples

Error Code 403 AccessDenied

In this example the API token did not allow access to existing endpoints.


```
{
 "status": 403,
 "error": "AccessDenied",
 "message": "The Security Framework denied access for this request. You are probably not allowed to access this resource.",
 "path": "/v1/account-roles",
 "timestamp": 1625896060372,
 "incident": "3245A2AC"
}
```


Error Code 404 Not Found

In this example, the endpoint path specified in the request does not exist.


```
{
 "status": 404,
 "error": "Not found",
 "message": "Not found",
 "path": "/v1/accounts/subs",
 "details": [],
 "timestamp": 1625785873772,
 "incident": "10F7E18D"
}
```


Error Code 406 Invalid Input


```
{
  "status": 406,
  "error": "Invalid input",
  "message": "The parameter 'subscriber' with value '3576730602192802' was not found",
  "path": "/v1/session-usage/search/findBySubscriber",
  "timestamp": 1625683231340,
  "incident": "1ECC1944"
}
```


Error Code 500 UNKNOWN


```
{
 "status": 500,
 "error": "Mapping",
 "message": "Sorry, an unexpected error occurred. Please report the details given here to Support for debug and analysis.",
 "path": "/v1/session-usage/search/findBySubscriber",
 "timestamp": 1625680929976,
 "incident": "65F1DA6A"
}
```

4.5.2 Error Codes

Code	Status	Description
200	OK	The request was successfully completed.
201	Created	A new resource was successfully created.
400	Bad Request	The request was invalid.
401	Unauthorized	The request did not include an authentication token or the
403	Forbidden	The client did not have permission to access the requested resource.
404	Not Found	The requested resource was not found.
405	Method Not Allowed	The HTTP method in the request was not supported by the resource. For example, the DELETE method cannot be used with the Agent API.
406	Invalid Input	The value provided for the parameter is formatted incorrectly or cannot be found.
409	Conflict	The request could not be completed due to a conflict. For example, POST ContentStore Folder API cannot complete if the given file or folder name already exists in the parent location.
500	Internal Server Error	The request was not completed due to an internal error on the server side.
503	Service Unavailable	The server was unavailable.

5. Support

5.1 FAQ

I have a Docker based install and I want to migrate to Kubernetes

We recommend setting up a Kubernetes cluster with separate hardware/nodes and install using the helm chart, in-place upgrades are not supported.

Can Expeto set up Kubernetes for us?

Expeto does not offer Kubernetes support or services but there are many other providers out there that offer day 0,1,2 for various Kubernetes platforms. A short list of supported platforms includes:

- RedHat OpenShift / MicroShift
- VMWare Tanzu
- Cannonical MicroK8s
- AWS EKS
- Microsoft AKS
- Digital Ocean Kubernetes Platform
- Google GKE

Note that not all cloud platforms support advanced routing and scalability features required for large deployments.

How many subscribers can each UPF support?

The number of supported subscribers is configurable and can be changed by using the --set nfcm.dataNetwork.numSessions= flag, default number of sessions is 3000. Each session requires additional memory on the UPF, the maximum number of supported sessions per UPF replica/pod is currently 50.000

How do I attach additional networks to the UPF or AMF?

Each network has to be configured in the networks section under the global values. There are several different kinds of network types available

How do I add static routes?

Static routes are handled using the route-override-cni which is bundled with both xCore and xRouter, the routes can be configured in the network section under the global values:

```
n6:
  create: false
  cniVersion: 0.3.1
  plugins:
    - type: macvlan # (macvlan/ipvlan/macvlan/host-device/sr-iov/bridge)
      master: eth0 # (eth0/enp3s0/nic-1/wlp2s0)
    - type: route-override
      flushroutes: false
      #delroutes:
      #  - dst: 192.168.255.0/24
      #addroutes:
      #  - dst: 192.168.255.0/24
      #    gw: 198.18.0.1
```

I updated the DNS server on a single-host MicroK8s install but still seeing

errors with name resolution

You need to restart the MicroK8s cluster for it to use the new DNS servers `sudo microk8s.stop && sudo microk8s.start`

I get certificate errors when using kubectl logs or when starting

containers on MicroK8s

If the IP address of your host has changed you might need to recreate the CA cert to include the new IP address, try running: `sudo microk8s refresh-certs --cert ca.crt`

I can see GTP packets arrive on a node, but why don't they arrive at the SMF or UPF pod?

Check for services configured as NodePort. If you've installed the helm chart with the intention of using a NodePort service but decided not to, that lingering service can cause issues. Services configured as NodePorts create firewall rules on the kubernetes worker nodes which can prevent packets arriving to pods **unless** they come through the NodePort service itself.

You can double check this by verifying iptables on the worker node:

```
iptables -L -t nat
```

5.2 Knowledge Base

How do I add extra network interfaces to my UPF, PGW, SMF or AMF?

Network segments (subnets/VLANs/VPCs/etc) are configured under the global section of the values.yaml file used to drive the xCore/xRouter installation and configuration, e.g:

```
global:
  networks:
    n6:
      create: true
      cniVersion: 0.3.1
      type: macvlan # (macvlan/ipvlan/macvlan/host-device/sr-iov/bridge)
      master: enp4s0 # (eth0/enp3s0/nic-1/wlp2s0)
      ipam:
        type: host-local # (whereabouts/multus-dhcp)
        subnet: 198.19.19.0/24
        rangeStart: 198.19.19.254
        rangeEnd: 198.19.19.254
```

For details on available types and options see the official Multus documentation and the official MetalLB documentation.

Each Multus interface will show up as an extra netX interface in the Kubernetes Pod, Static Routes can be modified and added using the route-override CNI and dynamic routes can be modified using the FRR routing daemon section.

Multus network with Static Route:

```
global:
  networks:
    n6:
      create: true
      cniVersion: 0.3.1
      plugins:
      - type: macvlan
        master: eth0
        ipam:
          type: whereabouts
          subnet: 198.19.19.0/24
          rangeStart: 198.19.19.254
          rangeEnd: 198.19.19.254
      - type: route-override
        flushroutes: false
        delroutes:
          - dst: 192.168.255.0/24
        addroutes:
          - dst: 192.168.255.0/24
            gw: 198.19.19.1
```

Multus network with Dynamic Routes:

```
global:
  networks:
    n6:
      create: true
      cniVersion: 0.3.1
      plugins:
      - type: macvlan
        master: eth0
        ipam:
          type: whereabouts
          subnet: 198.19.19.0/24
          rangeStart: 198.19.19.254
          rangeEnd: 198.19.19.254
  frr:
    - name: n6bgp
      bgp:
        routers:
          - asn: 64512
            neighbors:
            - address: 198.19.19.1
              asn: 64512
              disableMP: false
              port: 179
              toAdvertise:
                allowed:
                  mode: all
                  #mode: filtered
                  #prefixes:
                  #- 172.30.0.0/16
                  #- 10.128.0.0/14
              toReceive:
                allowed:
```

```
            mode: all
            #mode: filtered
            #- prefix:  192.168.255.0/24
            #  ge: 24
            #  le: 32
          prefixes:
          - 172.30.0.0/16
          - 10.128.0.0/14
```

MetalLB interfaces are created dynamically by the MetalLB daemonset and traffic is forwarded to the eth0 interface of the Pod, additional routes can be added/removed using the FRR section. Note that FRR configuration can affect the advertisement of MetalLB interfaces when allowed advertisements are specified.

```
global:
  networks:
    n2mlb:
      # Note that any addresses for networks wiht advertisementType "bgp" will
      # need to be included in the allowed prefixes of bgp routers defined in
      # the frr section.
      create: false
      apiVersion: metallb.io/v1beta1
      advertisementType: bgp
      addresses:
        - 192.168.10.0/24
        - 192.168.9.1-192.168.9.5
        - fc00:f853:0ccd:e799::/124
```

There is no limit on the number of networks that can be configured and the naming is left to the discression of the operator. Some networking types might not be available on cloud hosted K8s implementation, notable GKE and AKS don't allow Multus or MetalLB networking and the default NodePort interfaces need to be used.

6. xCore

6.1 Expeto xCore

Expeto xCore is a hybrid combined 3/4/5G core that can be installed and configured for both private and public use-cases.

If IPX connectivity is required please see xRouter documentation.

xCore is distributed and installed and configured using Helm charts and Kubernetes.

As a fully containerized 3GPP compliant core xCore is optimised for for ease of deployment into enterprise-controlled cloud or edge locations, supporting connection to both public and private networks.

Expeto xCore exists in two configurations: private and public.

The private RAN implementation of Expeto xCore is a complete 3GPP compliant Packet Core providing private networking over a 4G/5G Mobile Network. It includes fundamental 3GPP E UTRAN components such as the HSS/AUSF/UDR/UDM AMF/MME, SGW/SMF, and UPF/PGW. These components are fully compliant with 3GPP specification standards. The private Expeto xCore is always installed within the private network. The public RAN implementation of Expeto xCore provides added external to an existing private 5G/LTE installation.

Installed outside the private network in a data center or corporate headquarters, public Expeto xCore and the Expeto xRouter service enable private SIM devices connected over a public network to access the enterprise 5g/LTE network. Tasks such as subscriber authentication are performed before access to the private network is granted.

4G/LTE/5G Expeto xCore services:

- HSS/AUSF/UDR/UDM – Store subscriber information for authentication and authorisation. Stores all the information about provisioned IMSIs
- SGW/SMF – Performs routing responsibilities as a GTP-Proxy and/or delegates them to UPF instances
- PGW/UPF – Routes inbound and outbound packets through the enterprise network for defined CIDR ranges
- MME/AMF – Control plane for radio access. Connects eNodeBs/gNodeBs and authenticates devices. Manages/stores UE contexts, creates temporary IDs, sends pages, controls authentication functions, and selects the SGW/SMF and PGW/UPFs

Expeto Agent

Expeto xCore includes the Expeto Agent which connects to Expeto xControl. The agent monitors changes and synchronizes configuration. Network changes made through xControl are handled by the Agent and automatically propagated to distributed locations using an “outbound only” traffic connection to xControl. The Agent communicates with xControl by connecting using TLS on port 443. If the Agent is disconnected, changes are queued in xControl and sent when the Agent reconnects.

Private Network Scenarios

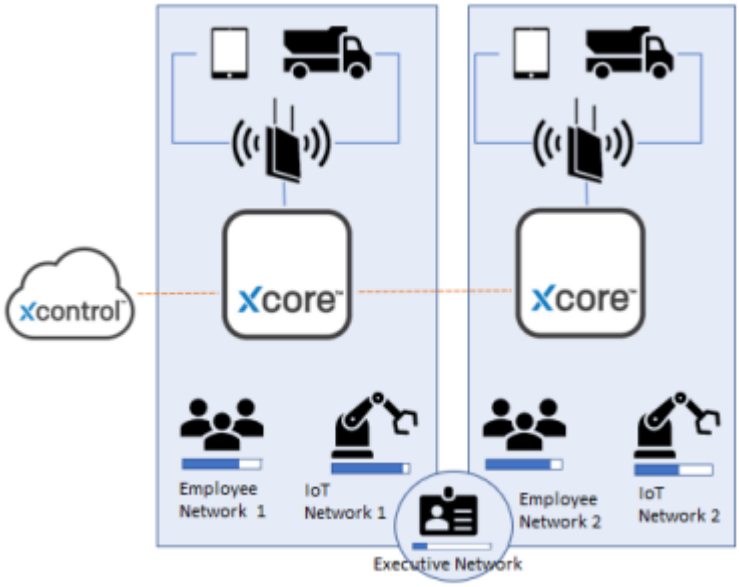
Adopted by industries that rely on secure mission-critical low latency wireless connectivity, private 4G/5G networks provide improved security, low-latency data transfer, network reliability, and cost effectiveness. In a private mobile network scenario, enterprise data is secured within the Enterprise network. The Enterprise itself becomes the mobile network provider, segmenting networks into subnets, managing subscribers, and monitoring network performance. Data in a 4/5G private network is routed through private Radio Access Network (RAN) physically installed at key locations providing site coverage and speed superior to Wi-Fi networks. The site itself can be a factory, hospital, mining location, university campus, airport, shipyard, or anywhere that can benefit from having its own secure 4G/5G network.

Private Network Characteristics:

- SIM card or eSIM is added to devices which connect to the private network through the eNodeB/gGodeB RAN.
- The Enterprise network can be segmented using Systems, each system can egress to a separate subnet with it's own CIDR range.
- Each geographically remote site requires an additional instance of Expeto xCore.
- Does not require use of the Expeto xRouter service.

DEFAULT ENCRYPTION SETTINGS:

- 3GPP Cipher and integrity settings 33.501 (5.11.1)
- Ciphering Value "0001" 128-NEA1 128-bit SNOW 3G based algorithm
- Integrity Value "0001" 128-NIA1 128-bit SNOW 3G based algorithm



How To Build This Network

There are several ways to set up a Network spanning physical sites depending on your requirements and any existing network, the following describes an example use-case.

NETWORK CONFIGURATION:

- Each instance of Expeto xCore corresponds to a Site in Expeto xControl.
- Two Systems are defined for each site: Employee Network and IoT Network.
- The Executive Network System is a single Shared System assigned to both Sites. This provides executives with roaming access when visiting the other site location.
- Each System is assigned a unique CIDR range for mobile devices, each system is associated with a separate network interface on the xCore cluster (vlan/macvlan/ipvlan/vpn/vpc)
- Each private SIM card in a device is a Subscriber, known to the network by the IMSI number.
- Each Subscriber is assigned to one of the Systems.

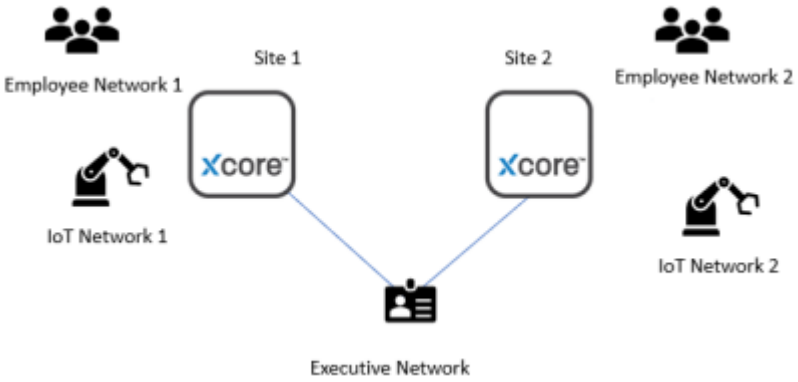
MANAGING SUBSCRIBERS BY SYSTEMS

Systems represent subnets, logical divisions of your network to which you can apply policies and assign groups and subscribers. The bar in the diagrams illustrate the number of Subscribers assigned to the System and the capacity of the System. The capacity of the System is defined by the associated CIDR range of available IP addresses. Although Sites are at different physical locations, Systems can be logically assigned to one or more network Sites. This allows Subscribers assigned to a shared System roaming access to multiple Sites. In the previous use case, if all Systems are assigned to both Sites, roaming of all Subscribers is allowed. Any employee (or autonomous vehicle!) of Site 1

could drive to Site 2 and still access the network. To restrict access to a single location, create Systems that are exclusive to a single Site . To allow access to multiple locations (enable roaming), create Systems that are assigned to multiple sites.

In the following example, the following Systems are created:

- Employee Network 1 and IoT Network 1 are Systems assigned to Site 1.
- Employee Network 2 and IoT Network 2 are Systems assigned to Site 2.
- Executive Network is a Shared System assigned to Site 1 and Site 2.



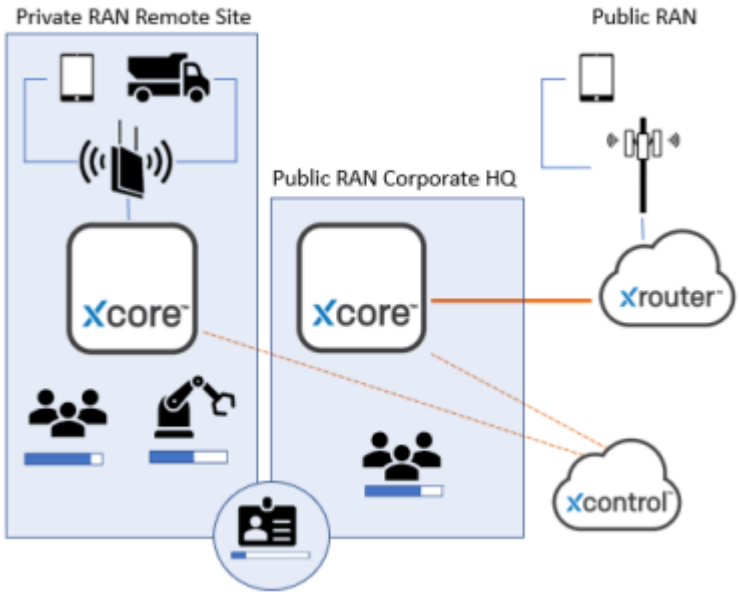
Each Subscriber is assigned to a single System only. Subscribers can either have dynamic or static IP addresse, dynamic addreses are allocated from a system wide CIDR block. If a Subscriber does not have a static IP address and is moved between Systems the Subscriber loses the old IP address and is given a new one that falls within the CIDR block of the new System. If the Subscriber has been assigned a static IP address, the IP address does not change. An Admin can assign a static IP address to a Subscriber in Expeto xControl by using an Extension. The diagram also shows the mixture of a public and private network. The public/private scenario is examined in the next section.

PUBLIC AND PRIVATE (HYBRID) NETWORK SCENARIO

The public and private network scenario adds the ability for user equipment (phones, tablets, IoT devices) to connect to the private network over a public mobile network. For example, employees can leave a remote Site, even leave the country, but can still use their phones with private SIM cards to access the private network. The public and private connections are managed as a single network through Expeto xControl. The scenario comprises a mixture of public and private RAN connectivity provided by two instances of Expeto xCore (one public, one private), and Expeto xRouter. Expand the network by deploying additional private Expeto xCores at each remote location. Improve global coverage by deploying additional public Expeto xCores at globally dispersed data centers.

NETWORK SCENARIO CHARACTERISTICS:

- Private SIM cards in all devices provide global access to the private network
- Roaming between public and private mobile networks with the same SIM
- Private deployment of Expeto xCore runs in a Kubernetes cluster on the private site
- Public deployment of Expeto xCore runs in a Kubernetes cluster in a datacenter or corporate headquarters
- Expeto xRouter runs in a datacenter with connectivity to IPX network and routes traffic between public mobile network operators and xCores
- Expeto xControl manages public/private as a single network
- Each geographically remote private network Site requires an additional instance of Expeto xCore
- Secure data transfer throughout the network with the ability to establish data sovereignty
- The Enterprise network can be segmented into subnets by assigning a CIDR range to a specific network associated with a System



Scenario Details Control Plane (Public RAN) * Private SIM attaches to the public RAN Mobile Network Operator (MNO) network. * The MNO directs SIMs using the Private IMSI range to the Expeto xRouter * In the Expeto xRouter, an Expeto HSS/UDM authenticates the SIM (Public RAN) * Expeto xRouter PGW/UPF receives data from the public RAN MNO network and sends it to the public Expeto xCore PGW/UPF network slice based on the subscriber's system membership through an private link or VPN. * Private Expeto xCore SGW/SMF routes the data to the selected PGW/UPF network slice based on the subscriber's system membership.

CONTROL PLANE (PRIVATE RAN)

- SIM attaches to private RAN.
- Expeto xCore with local MME/AMF and HSS/AUSF/UDR/UDM installed manages authentication and authorization.

DATA PLANE (PRIVATE RAN)

- Data is routed to the PGW/UPF network slice based on the subscriber's system membership in the Expeto xCore located in the remote site.
- Expeto xCore has PGW/UPF instances for each subnet in the Enterprise site.

4G/5G THROUGHPUT

Throughput depends on the following factors: * Bandwidth -- The wider the bandwidth, the higher the throughput. 4G/LTE Channel bandwidth can be 1.4, 3, 5, 10, 15, or 20 MHz. 5G Channel bandwidth adds 40, 80 and 160Mhz and channel multiplexing. * Conditions -- The eNodeB/gNodeB selects Modulation and Coding Scheme (MCS) based on the current channel quality for radio conditions. The higher MCS, the more bits can be transmitted by a single resource element (RE). * Multi-Antenna Use -- MIMO 2x2 or MIMO 4x4 can be used to increase performance by providing additional data streams. The bits per RE multiplied by the number of streams provide the throughput. Actual throughput will be slightly lower due to the additional pilot signals. * Network Load -- Resources are divided amongst active subscribers. * Bandwidth of 20MHz with the best modulation (256QAM) without MIMO (SIMO) provides a maximum throughput of 97.896 Mbps. For more bandwidh larger channels can be used or more channels added with multiplexing.

5G MMWAVE THROUGHPUT

In addition to considerations for LTE/4G throughput 5G adds more spectrum, this is especially true in the millimeter wave frequencies where throughput can reach multy-gigabit speeds. 5G connections also allow more flexibility when it comes to combined attach and using multiple parts of the spectrum together.

6.2 Installation

6.2.1 Installation Overview and Preparation Guide

Expeto offers scalable and flexible solutions for deploying private and public 4G/5G networks within Kubernetes environments.

This guide provides step-by-step instructions to help you install and configure Expeto products effectively, whether you're managing a greenfield deployment or integrating it into an existing network. While the guide assumes advanced technical expertise, it is structured to accommodate the specific needs of Mobile Network Operators (MNOs), enterprise customers, and telco integrators.

Who Should Use This Guide?

- **Mobile Network Operators (MNOs):** MNOs deploy Expeto products as part of a broader ecosystem including xControl and xRouter to manage large-scale, multi-site networks for internal use or customer traffic. These users typically require advanced configurations for multi-tenancy, redundancy, and scalability, and asusme extensive expertise in telecommunications and networking such as BGP and IPX.
- **Enterprises Deploying public xCores:** These users deploy xCore as a "router" for mobile traffic to egress onto private networks without the need for full telco infrastructure. They usually have enterprise networking experience and require a streamlined installation process leveraging Expeto's managed services.
- **Integrators and Enterprise networking experts using Private Radios:** Specialists and advanced enterprise staff integrating with private radios (gNodeB/eNodeB) for customized network solutions. They possess advanced telco experience and expertise in configuring radio hardware, PLMN IDs, and complex networking interfaces.

This guide assumes that foundational decisions about deployment, such as PLMN IDs, radio hardware, and Kubernetes platform configurations, have been made. While Expeto uses a **Product Intake Form** internally to gather this information, this guide provides a comprehensive installation path even if that data hasn't been formalized.

Overview of the Installation Process

The installation process for Expedo products is structured into five main stages:

1. **Prerequisites:**
   - Ensure Kubernetes readiness, install necessary plugins, and set up required tools like `kubectl` and `helm`.
   - On OpenShift/MicroShift all commands performed with kubectl can be performed with the `oc` command instead, in this case we recommend using an alias, e.g. alias `kubectl=oc`
2. **Add Expeto Helm Repositories:**
   - Authenticate and retrieve the necessary Helm charts.
3. **Customize Configuration:**
   - Create and configure the `values.yaml` file to define behavior and connectivity, including PLMN IDs, networking interfaces, and scaling requirements.
4. **Deploy:**
   - Use Helm to install and monitor the deployment process.
5. **Validate and Optimize:**
   - Verify the deployment, ensure proper functionality, and apply performance optimizations.

Prerequisites

Before you begin the installation, ensure that your environment meets the following requirements. This section outlines system requirements, tools, and preparatory steps necessary for a smooth deployment process.

SYSTEM REQUIREMENTS

- **Version:** Kubernetes 1.23 or later.
- **Cluster Resources:**
  - At least 3 worker nodes for redundancy.
  - Minimum of 4 CPU Cores and 8 GB memory per node for a total of at least 16 cores and 32 GB per cluster.
  - Persistent storage system available from all nodes configured for `ReadWriteOnce` volumes.

  For more information on resource sizing, see System Resource / Sizing Guide
- **Plugins/Operators/Features required:**
  - DNS Resolver.
  - Cert Manager.
  - Storage plugin compatible with `ReadWriteOnce` volumes.
- **Node Kernel:**
  - Linux kernel 5.13 or later, compatible with Kubernetes.

TOOLS AND UTILITIES

- **kubectl:** Command-line tool for interacting with Kubernetes clusters.
  - Install kubectl
- **helm:** Kubernetes package manager for managing Helm charts.
  - Install Helm

Depending on your Kubernetes environment, the following tools must be configured to ensure a functional deployment. Adjust the steps based on whether you are using MicroK8s, OpenShift/MicroShift, or VMware Tanzu.

> **Important**
>
> Review the document below of your chosen Kubernetes platform to understand the common installation and configuration challenges:
>
> - Canonical MicroK8s
> - RedHat OpenShift/MicroShift
> - WMware Tanzu

DNS Resolver

A DNS resolver provides name resolution for Kubernetes services, so that services are able to communicate with each other.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

```
sudo microk8s enable dns
```

OpenShift/MicroShift have DNS capabilities enabled by default.

For further configuration options, please refer to DNS Operator in OpenShift.

VMware Tanzu comes with integrated DNS resolution capabilities via CoreDNS.

If you require public service resolution, Tanzu supports synchronizing with external DNS providers via ExternalDNS.

Cert Manager

Cert Manager is required to manage TLS certificates for secure communication.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

```
sudo microk8s enable cert-manager
```

Please refer to Cert-manager documentation for configuring cert-manager with OpenShift.

Please refer to Tanzu Cert Manager Installation for installation steps with VMware Tanzu.

Persistent Storage

A storage class must be configured to support persistent volumes.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

MicroK8s does not ship with a storage class plugin to span multiple worker nodes. For Demo purposes the following can be used on single node installs only. This will not work for multi-node installs because the storage will not be replicated across nodes causing Pods to loose data if they migrate between Nodes. Other options are available for larger MicroK8s installs, see Cannonical documentation for examples and guidelines.

```
sudo microk8s enable hostpath-storage
```

OpenShift supports a wide variety of persistent storage options.

Read more at Understanding Persistent Storage in OpenShift.

See following document for MicroShift MicroShift Storage Configuration.

Tanzu also supports a variety of persistent storage options.

Read more at Using Persistent Storage in vSphere with Tanzu.

CLUSTER PREPARATION

1. Get a copy of the kube-config file for your cluster, name it config and place it in the .kube directory residing in your home directory, be sure to set permissions to 0600 or equivalent on non Linux platforms.
2. **Verify Kubernetes Access:**
   - Ensure you can connect to your Kubernetes cluster using kubectl:

     ```
     kubectl get nodes
     ```

     Confirm all nodes are in a Ready state.
3. **Helm Installation:**
   - Verify that Helm is installed and functioning:

     ```
     helm version
     ```
4. **Ulimits:**

   The Expeto solution ships with a DaemonSets/MachineConfigs that will automatically configure things like kernel modules, buffer sizes, sysctl parameters and ulimits on most platforms but in some cases slight additional modifications might be recommended.

   For MicroK8s: Edit the container environment configuration and increase the ulimit to 1048576

   ```
   echo 'ulimit -n 1048576 || true' >> '/var/snap/microk8s/current/args/containerd-env'
   ```
5. **Configure allowed NodePort range:**

   The default installation depends on the availability of NodePorts and having a wider range of ports enabled than is default with most k8s deployments.

Unless your install only uses Multus based interfaces make sure to allow node ports in the range of 30000-38413 before installing the chart.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMWare Tanzu**

```
echo '--service-node-port-range=1024-38413' >> /var/snap/microk8s/current/args/kube-apiserver
```

Then, restart microk8s (`sudo microk8s.stop && sudo microk8s.start`)

```
oc patch network.config.openshift.io cluster --type=merge -p \
'{
   "spec":
   { "serviceNodePortRange": "1025-38413" }
}'
```

Note that it might take several minutes for the new config to be applied.

For a multi-master cluster see the following doc for how to identify the leader: Leader identification

For a single master cluster or once the cluster leader has been identified make the following change:

```
echo '- "--service-node-port-range=-1024-38413"' >> /var/vcap/jobs/kube-apiserver/config
```

Then, restart the kube-apiserver

6. **Advanced Networking:**

   - **Using LoadBalancers:**

     Please refer to Services, Load Balancing, and Networking in Kubernetes documentation for detailed information regarding setting up load balancers and other advanced networking topics.

   - **Using Multus for Advanced Networking:**

     For platforms that support Multus it can either be installed separately or installed by setting `multus.enabled` to true during install.

     It's recommended to also install Whereabouts for IPAM at the same time.

     For installations that have a Multus plugin, it is recommended to use the built-in version.

     Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

     ```
     sudo microk8s enable community
     sudo microk8s enable multus
     ```

     Please refer to Multus CNI documentation for configuring Multus and Whereabouts with OpenShift.

     Please refer to Cert-manager documentation for configuring Multus with VMWare Tanzu.

     For further information regarding installation and configuration of Multus on other platforms, please refer to Multus Quick-Start Guide or Multus Documentation.

7. **Role-Based Access Control (RBAC):**

   Note: If you plan to use Role-Based Access Control (RBAC) in your Kubernetes cluster, **it must be enabled before deploying Expeto products.**

   If RBAC is enabled after deployment, the cluster must be redeployed before it will function correctly.

   When the cluster is deployed, the helm chart will automatically configure the required roles, bindings, and permissions for RBAC.

   Refer to Using RBAC Authorization in Kubernetes documentation for further information.

Adding the Expeto Helm Repositories

All Expeto deployments rely on Helm charts hosted in Expeto's private repositories. This step outlines how to configure access to these repositories, ensuring that the necessary charts and container images can be downloaded during installation.

Before proceeding, ensure you have:

1. **Helm Repository Credentials:** Provided by Expeto Support for accessing container images and Helm charts.
2. **Deployment-Specific Information:**
    - PLMN IDs, RAN details, and other configuration specifics (if applicable).
    - Network interface and CIDR details for northbound (N2/N3/N9) and southbound (N6) interfaces.

1. Use the provided credentials to add the Expeto Helm repositories to your local Helm configuration.

```
helm repo add expeto-ngc https://repo.expeto.io/repository/ngc --username '<your_username>' --password '<your_password>'
```

   Replace `<your_username>` and `<your_password>` with the credentials provided by Expeto Support.
2. Update your local Helm repository cache to ensure you have the latest chart versions:

```
helm repo update
```

After adding the repositories, confirm that they are accessible and contain the required charts:

1. List the added repositories:

```
helm repo list
```

2. Search for available charts to confirm connectivity:

xCore **xRouter** **xControl**

```
helm search repo xcore
```

```
helm search repo xrouter
```

```
helm search repo xcontrol
helm search repo expeto-docs
```

These commands should return a list of available charts for each product in the Expeto repository.

If you encounter issues adding the repositories or accessing charts:

- Ensure your username and password are correct and match the credentials provided by Expeto Support.
- Confirm that your machine has access to the internet and can reach `https://repo.expeto.io`.
- Ensure that Helm is installed and functioning properly and is more recent than 3.0.9 than by running:

```
helm version
```

- Contact Expeto Support for assistance if the problem persists.

Customize Configuration

The `values.yaml` file is a critical part of the Expeto xCore deployment process.

It defines parameters for the entire deployment, including networking, resource scaling, radio configuration and advanced features. This guide will help you customize the file to align with your deployment requirements.

> **Important**
>
> - All changes to deployments must be made in the `values.yaml` file. This file serves as the **source of truth** for your Helm deployments. **Any manual changes made directly to resources (e.g., via kubectl) will be overwritten when scaling occurs or when the `values.yaml` file is applied again.**
> - For more details on applicable parameters in the `values.yaml`, see values.yaml Reference

To ensure consistency and prevent loss of changes, always update the values.yaml file and reapply it with Helm commands.

Make sure you store a copy of the values.yaml file in a secure place, this file acts as your disaster recovery and will allow you to rebuild the entire solution with minimal effort very rapidly.

To access the `values.yaml` file:

1. Pull the Helm chart from the `expeto-ngc` repository:

   xCore **xRouter** **xControl**

   ```
   helm show values expeto-ngc/xcore
   ```

   ```
   helm show values expeto-ngc/xrouter
   ```

   ```
   helm show values expeto-ngc/xcontrol
   ```

   This is to get the default values, multiple examples are also available to demonstrate different setup options and possibilities for a variety of Kubernetes distributions

   The `values.yaml` file will be in the extracted directory.

2. Pull examples from the `expeto-ngc` repository:

   xCore **xRouter**

   ```
   export tmp_dir="$(mktemp -d)" && helm pull ngc/xcore  --untar --untardir ${tmp_dir} && mv "${tmp_dir}/xcore/examples" ./ && rm -Rf "${tmp_dir}"    === "xCore"
   ```

   ```
   export tmp_dir="$(mktemp -d)" && helm pull ngc/xrouter  --untar --untardir ${tmp_dir} && mv "${tmp_dir}/xrouter/examples" ./ && rm -Rf "${tmp_dir}"
   ```

   This will download the chart, extract it to a temporary directory and then move the examples directory to the current directory. Alternatively the whole chart can simply be downloaded with "helm pull" and extract.

- **Edit the File:** Open `values.yaml` and update the sections relevant to your deployment. Reference the comments in the file for additional guidance.
- **Validate Changes:** Use Helm to validate the configuration before applying it:

xCore **xRouter** **xControl**

```
helm template expeto-ngc/xcore -f values.yaml > validated_output.yaml
```

```
helm template expeto-ngc/xrouter -f values.yaml > validated_output.yaml
```

```
helm template expeto-ngc/xcontrol -f values.yaml > validated_output.yaml
```

Review the `validated_output.yaml` for any errors or issues before proceeding.

- **Start with Defaults:** Use the default values.yaml file as a baseline and modify only the required fields.
- **Document Changes:** Add comments to track customizations for easier troubleshooting or future updates.
- **Test Incrementally:** Validate and test changes incrementally to catch errors early.

Deploy and Verify

With your environment prepared, Helm repositories configured, and the `values.yaml` file customized, you are ready to deploy into your Kubernetes cluster. Following the links below for details of the deployment process, verification steps, and troubleshooting tips of the Expeto platform components:

- **xCore**
- **xRouter**
- **xControl**

6.2.2 Deployment & Verification

6.2.3 Testing and Validation Procedures

- UE initiated Detach
- Network-initiated Detach
- UE Idle Mode
- UE Service Request
- UE Paging
- Multiple UE Attach
- Unknown UE Attach
- Inactive Subscriber attach
- Expeto Edge (xCore) Session Management High Availability K8s
- Deactivate Subscriber
- Reactivate Subscriber
- Limit Bandwidth (AMBR)
- Expeto Edge (xCore) restart recovery
- Expeto Gateway (xRouter) restart recovery
- Expeto xControl restart recovery
- Expeto Gateway (xRouter) Subscriber Management high availability K8s
- Expeto Edge (xCore) Geographical Redundancy
- Expeto Edge (xCore) Datapath High Availability K8s
- Expeto Gateway (xRouter) Geographical Redundancy
- Prevent UE to UE communication
- Prevent UE communication with internal xCore components
- Expeto Edge (xCore) maximum subscribers (UEs)

UE Initiated Attach

Description

A UE (subscriber) initiates an attach request to the network where it is authenticated.
After the authentication is passed, the network obtains the UE's subscription data from the subscriber database.
Based on the default APN and PDN/PDU context in the subscription data, the network creates default session; and then the UE is attached to the network.

Prerequisite

- Log into xControl with an appropriate user role
- UE (Mobile Asset/Subscriber) is provisioned to an appropriate Expeto ‘System’ in xControl with the status 'Provisioned'
- UE is powered off or in airplane mode
- UE has the APN configured
- UE has a web browser application or terminal application with ICMP
- Start signal tracing (pcap) using preferred tool to capture 3GPP signalling traffic

Action	Expected Result
1. Power on the UE or come out of the Airplane mode 2. Wait for it to connect to the cellular network 3. Initiate traffic by navigating to an internet page or ping a known available resource	1. Verify the UE has a cellular data connection 2. Verify internet traffic or ping is working on the UE 3. In xControl, verify the UE (Mobile Asset) is in Connected status and has an IP address 4. In xControl, verify the UE's Session Event log has appropriate entries 5. Verify signalling traces match the expected 3GPP flow.

UE initiated Detach

Description

UE initiates a detach procedure and release the session.

Prerequisite

- Log into xControl as Admin role
- Start signal tracing (pcap) using a preferred tool to capture 3GPP signalling traffic

Action	Expected Result
Execute Test: UE Initiated Attach	
1. Power off the UE or put it in the Airplane mode 2. Wait for the UE to disconnect from the network	1. In xControl, verify the UE (Mobile Asset) is in Disconnected status within reasonable time (nominally within 30 seconds) 2. In xControl, verify the UE's Session Event log has appropriate entries 3. Verify signalling traces match the expected 3GPP flow.

Network-initiated Detach

Description The network (core) initiates a detach procedure and releases the session.

Prerequisite

- Log into xControl as Admin role
- Start signal tracing (pcap) using prefered tool to capture 3GPP signalling traffic

Action	Expected Result
Execute Test: UE Initiated Attach	
1. In xControl, invoke the "Cancel Session" action for the subscriber (Mobile Asset) 2. Wait for the UE to disconnect from the network	1. In xControl, verify the UE is in Disconnected status 2. In xControl, verify the Session Event log has appropriate entries 3. Verify signalling traces match the expected 3GPP flow. NOTE: the UE may try to reconnect after the disconnect

UE Idle Mode

Description Validate UE can go into inactivity (ECM-IDLE) mode

Prerequisite

- UE has a web browser application or terminal application with ICMP
- Start signal tracing (pcap) using a preferred tool to capture 3GPP signalling traffic

Action	Expected Result
Execute Test: UE Initiated Attach	
1. Ensure UE is not generating data traffic. 2. Wait the required time as defined by the radio configuration. Note: The wait time is dependent on the radio's idle-mode timer. If this is unknown, use an Android or similar app.	1. For private radios, use CLI to verify the MME session is marked as Idle 2. Alternatively, use the app on UE to verify it is Idle (inactive) 3. Verify signalling traces match the expected 3GPP flow.

UE Service Request

Description

When a UE is registered at a network, but its connection was released due to inactivity (ECM-IDLE), new traffic is generated from the UE, changing the network connection state to connected.

Prerequisite

- UE has a web browser application or terminal application with ICMP
- Start signal tracing (pcap) using a prefered tool to capture 3GPP signalling traffic
- Log into xControl as Admin role

Action	Expected Result
Execute Test: UE Initiated Attach	
Execute Test: UE Idle Mode	
1. Initiate traffic by navigating to an internet page or ping a known available resource	1. Verify the UE has a cellular data connection 2. Verify internet traffic or ping is working on the UE 3. In xControl, verify the UE (Mobile Asset) is in Connected status 4. Verify signalling traces match the expected 3GPP flow.

UE Paging

Description

The network side initiates a service request procedure to establish a connection with UE in idle mode.

Prerequisite

- Tool to generate ping (or similar other traffic) from the network towards the UE's IP address, which could be on actual PGW/UPF or another environment close to it
- Start signal tracing (pcap) using a preferred tool to capture 3GPP signalling traffic
- Log into xControl as Admin role

Action	Expected Result
Execute Test: UE Initiated Attach	
Execute Test: UE Idle Mode	
Use the preferred tool (ping) to initiate traffic from a network toward the IP address of the UE.	1. Verify that the ping returns the expected result - If UE does not respond to the ping, try another protocol/tool to initiate network data 2. Verify signalling traces match the expected 3GPP flow.

Multiple UE Attach

Description

Test connecting multiple UEs to the network.

Prerequisite

- The same preconditions as "UE Initiated Attach" test cases, except multiple UEs are provisioned in xControl.

Action	Expected Result
Execute Test: UE Initiated Attach	
Repeat the Action from Step 1 for each of the provisioned Subscribers (UEs)	Repeat the Expected Result from Step 1 for each of the provisioned Subscribers (UEs)

Unknown UE Attach

Description

When a UE not provisioned in xControl initiates an attach request to the network, the network rejects the attach attempt.

Prerequisite

- Log into xControl as with an appropriate user role
- UE (Mobile Asset/Subscriber) is NOT provisioned in xControl
- UE is powered off or in airplane mode
- Start signal tracing (pcap) using prefered tool to capture 3GPP signalling traffic"

Action	Expected Result
1. Power on the UE or come out of the Airplane mode 2. Wait for the UE to attempt to connect to the cellular network	1. Verify the UE fails to connect the the network 2. Verify signalling traces match the expected 3GPP flow.

Inactive Subscriber attach

Description An inactive subscriber in the xControl, initiates an attach request to the network and the network rejects the attach attempt

Prerequisite

- Log into xControl with an appropriate user role xControl but marked as inactive/disabled
- UE is powered off or in airplane mode
- Start signal tracing (pcap) using a preferred tool to capture 3GPP signalling traffic

Action	Expected Result
Execute Test: Deactivate Subscriber	
1. Power on the UE or come out of the Airplane mode 2. Wait for the UE to attempt to connect to the cellular network	1. Verify the UE fails to connect the the network 2. Verify signalling traces match the expected 3GPP flow.

Expeto Edge (xCore) Session Management High Availability K8s

Description

Expeto Edge, managed by Kubernetes, continues to operate even when the node(s) hosting the session management services fails.

Prerequisite

- An xCore is deployed on Kubernetes (K8s) and operational
- Identify the node (VM) where the pods responsible for session management (SMF/PGW-C) are running
- Ensure there are sufficient nodes available to host the session management
- Ensure a subscriber (device) is connected and passing data

Action	Expected Result
1. Shutdown the node hosting the session management pod(s) (SMF/PGW-C) 2. Put the device in airplane mode 3. Then turn off the airplane mode 4. Perform speedtest	1. Verify that all pods on the node start successfully on the other nodes. 2. Verify the subscriber can authenticate and connect to the network. 3. Run a speedtest and verify the device can upload and download data.

Deactivate Subscriber

Description An active subscriber is deactivated in the xControl, UE initiates an attach request to the network and the network rejects the attach attempt.

Prerequisite

- Log into xControl with an appropriate user role
- Start signal tracing (pcap) using a preferred tool to capture 3GPP signalling traffic

Action	Expected Result
Execute Test: UE Initiated Attach	
1. Change the status of the subscriber (mobile asset) to 'inactive' in xControl 2. Wait for the subscriber's status to change to Disconnected.	1. Verify the UE disconnected from the network and any reconnection attempts fail 2. Verify signalling traces match the expected 3GPP flow.

Reactivate Subscriber

Description

An active subscriber is deactivated in the xControl, UE initiates an attach request to the network, and the network rejects the attach attempt.

Prerequisite

- UE is powered on and not in airplane mode
- UE has APN correctly configured
- UE has a web browser application or terminal application with ICMP
- Start signal tracing (pcap) using a prefered tool to capture 3GPP signalling traffic
- Log into xControl as Admin role

Action	Expected Result
Execute Test: Deactivate Subscriber	
1. Change the status of the subscriber (mobile asset) to 'active' in xControl 2. Wait for the device to reconnect and re-establish the connection 3. Wait for the subscriber's status to change to Connected.	1. Verify the UE has a cellular data connection 2. Verify internet traffic or ping is working on the UE 3. In xControl, verify the UE (Mobile Asset) is in Connected status 4. Verify signalling traces match the expected 3GPP flow.

Limit Bandwidth (AMBR)

Description

When the subscriber UL/DL AMBR is reduced in the xControl with a new profile, the UL/DL rate of the UE is reduced.

Prerequisite

- Log into xControl with an appropriate user role
- UE has a web browser application
- A tool that can measure upload and download speeds.
- Start signal tracing (pcap) using a preferred tool to capture 3GPP signalling traffic

Action	Expected Result
Execute Test: UE Initiated Attach	
1. Run a speedtest to get initial bandwidth numbers 2. Create a new or use existing profile in xControl with significantly reduced UL/DL AMBR 3. In xControl, edit the subscriber, change the profile to the new profile and save. 4. Wait for the UE to detach and attach back the network 5. Run the speed test on the UE browser and note down the upload and download speeds.	1. In xControl, verify the UE's Session Event log has appropriate entries as the UE is detached and attaches back to the network. 2. Verify that the upload and download speeds on the UE are lower after the new profile is applied to the subscriber. 3. Verify signalling traces match the expected 3GPP flow.

Expeto Edge (xCore) restart recovery

Description

Non-redundant Expeto Edge (xCore) recovers after all resource(s) hosting the xCore are power cycled (turned off and back on)

Prerequisite

- An xCore is deployed and operational
- Ensure a subscriber (device) is connected and passing data

Action	Expected Result
1. Shutdown all node(s)/VM(s) hosting the xCore site 2. Start all the node(s)/VM(s) that were shutdown	1. Verify all the services of the xCore is running 2. Verify the subscriber reconnects to the network 3. Run a speedtest and verify the device can upload and download data.

Expeto Gateway (xRouter) restart recovery

Description

Non-redundant Expeto Gateway (xRouter) recovers after all resource(s) hosting the xRouter are power cycled (turned off and back on)

Prerequisite

- An xRouter is deployed and operational
- The xRouter is integrated with an xCore
- Ensure a subscriber (device) is connected the xCore through the xRouter and passing data

Action	Expected Result
1. Shutdown all node(s)/VM(s) hosting the xRouter site 2. Start all the node(s)/VM(s) that were shutdown	1. Verify all the services of the xRouter is running 2. Verify the subscriber reconnects to the network through the xRouter 3. Run a speedtest and verify the device can upload and download data.

Expeto xControl restart recovery

Description

Non-redundant Exepto xControl recovers after all resource(s) hosting xControl are power cycled.

Prerequisite

- An Control is deployed and operational
- An xCore site is deployed and managed by the xControl
- You have an account in xControl

Action	Expected Result
1. Log into the xControl and confirm the status of the xCore site 2. Shutdown all node(s)/VM(s) hosting the xControl 3. Start all the node(s)/VM(s) that were shut down	1. Verify that xControl starts without any errors 2. You able to log back into the xControl 3. Verify the status of the xCore status

Expeto Gateway (xRouter) Subscriber Management high availability K8s

Description

Expeto Gateway, managed by Kubernetes, continues to operate when the node(s) hosting the subscriber management services fail.

Prerequisite

- An xRouter is deployed on Kuberntes (K8s) and operational
- Identify the node (VM) where the pods responsible for subscriber mangement (for 5G - AUSF, UDR and UDM and for 4G - HSS) are running
- Ensure there are sufficient nodes available to host the subscriber mangement after node shutdown
- Ensure a subscriber (device) is connected and passing data

Action	Expected Result
1. Shutdown the subscriber node(s) 2. Put the device in airplane mode 3. Then turn off the airplane mode 4. Steps 2 and 3 might need to be repeated until authentication or update location operations occur due to how some visited networks may cache authentication data fetched prior to node shutdown	1. Verify all pods on node start successfully on the other nodes 2. Verify the subscriber can authenticate and connect to the network.

Expeto Edge (xCore) Geographical Redundancy

Description

When an xCore deployed in a geo-redundant configuration fails, subscribers from the xCore reconnect to the redundant xCore and continue to get service.

Prerequisite

- Geo-redundant xCores deployed in active-active configuration
- Subscribers (devices) connected to both xCores and passing traffic

Action	Expected Result
1. Run a speedtest on a device connected to one of the xCore sites 2. Record the speed and the latency 3. Shutdown the xCore site	1. The device connected to the xCore that was shutdown should disconnect from the xCore 2. The device should then reconnect to the redundant xCore site within a reasonable time 3. Run a speedtest on the device 4. Verify the speed and the latency are acceptable

Expeto Edge (xCore) Datapath High Availability K8s

Description

Expeto Edge, managed by Kubernetes, continues to operate even when the node(s) hosting the datapath services fails.

Prerequisite

- An xCore is deployed on Kubernetes (K8s), and operational
- Identify the node (VMs) where the pods responsible for datapath (UPF/PGW-U) is running
- Ensure there are sufficient nodes available to host the datapath
- Ensure a subscriber (device) is connected and passing data

Action	Expected Result
1. Shutdown the datapath node (UPF/PGW-U) 2. Put the device in airplane mode 3. Then turn off the airplane mode 4. Perform speedtest	1. Verify all pods on node start successfully on the other nodes 2. Verify the subscriber can authenticate and connect to the network. 3. Run speedtest and verify the device can upload and download data.

Expeto Gateway (xRouter) Geographical Redundancy

Description

When an xRouter deployed in a geo-redundant configuration fails, subscribers connected through the xRouter will reconnect through the redundant xRouter and continue to get service.

Prerequisite

- Geographically redundant xRouters are deployed and operational
- Both the xRouters integrated with an xCore
- Subscribers (devices) connected through both xRouters to the xCore and passing traffic

Action	Expected Result
1. Run speedtest on a device connected through one of the xRouter sites 2. Record the speed and the latency 3. Shutdown the xRouter site	1. The device connected through the xRouter that was shutdown should disconnect from the xRouter 2. The device should then reconnect to the xCore site through the other geo-redundant xRouter 3. Run speedtest on the device 4. Verify the speed and the latency are acceptable

Prevent UE to UE communication

Description

The network rejects communication between UEs (subscribers) provided on the Expeto platform.

Prerequisite

- An xCore is deployed and operational
- The xCore is configured to prevent UE to UE communication (default configuration)
- The xCore has a System
- Two UEs are provisioned into the System

Action	Expected Result
Execute Test: Multiple UE Attach	
1. Log into xControl 2. Find the target UE 3. Note down the target UE's IP address 4. From the source UE, ping the target UE's IP address	1. Verify ping fails to reach the target UE

Prevent UE communication with internal xCore components

Description

The network rejects any communication from the UE to any software components of the Expeto platform.

Prerequisite

- An xCore is deployed and operational
- The xCore is configured to prevent UE to UE communication (default configuration)
- The xCore has a system where the UE are provisioned

Action	Expected Result
Execute Test: UE Initiated Attach	
1. Log into xControl 2. Find the UE (Mobile Asset) on the Mobile Assets page 3. Navigate to the Site details page of UE and node down the IP addresses of the Site and monitoring endpoints. 4. From the source UE, ping the IP addresses	1. Verify ping fails to reach the target IP addresses

Expeto Edge (xCore) maximum subscribers (UEs)

Description

Non-redundant Expeto Edge (xCore) shall support the maximum number of active subscribers as designed.

Prerequisite

- An xCore is deployed and operational
- The xCore has a System

Action	Expected Result
Execute Test: UE Initiated Attach	
1. Repeat the previous step until the number of active subscribers (UEs) connected reaches the maximum the xCore can support.	1. Verify all the UEs are connected 2. Verify the UEs are able to pass traffic

6.2.4 Installation Considerations

Installation Considerations - MicroK8s

Deploying **Expeto xCore** on **MicroK8s** can be an efficient way to manage private and public 4G/5G networks within a lightweight Kubernetes environment. However, MicroK8s' design choices, such as a simplified architecture, snap-based lifecycle management, and its focus on small-scale or edge deployments, may present challenges during installation and configuration.

This document outlines the most common challenges you might encounter when using MicroK8s as the Kubernetes platform for xCore. By understanding these potential issues upfront, you can proactively address them to ensure a smoother deployment experience. Each section categorizes and explains these challenges to help you better prepare and adapt the deployment process to MicroK8s' environment.

NETWORKING CHALLENGES

- NodePort traffic for xCore requires configuring a custom port range (e.g., `30000-38413`). MicroK8s updates or reboots may reset these configurations, necessitating reapplication.
- MicroK8s does not include a native load balancer. Deploying MetalLB or similar solutions is required for external traffic. Misconfigured BGP routes or IP pools can result in connectivity issues.
- MicroK8s does not enforce network policies by default. This could lead to security vulnerabilities or unintentional exposure of services. Users should define explicit network policies for xCore.
- Multus requires manual configuration of `NetworkAttachmentDefinitions`. Misconfigurations may prevent xCore pods from utilizing additional network interfaces effectively.

PERSISTENT STORAGE ISSUES

- The `hostpath-storage` add-on (`microk8s enable hostpath-storage`) is not production-grade and unsuitable for shared access in multi-node environments. Users should configure external storage solutions like NFS, Ceph, or dynamic storage classes.
- Snap-based updates may reset storage configurations, potentially leading to data loss if external storage is not configured.

RESOURCE LIMITATIONS

- MicroK8s' lightweight design may allocate insufficient CPU, memory, or storage resources. Users must ensure nodes meet xCore's minimum requirements (4 CPUs, 8 GB RAM per node).
- Adding nodes in MicroK8s is a manual process, requiring careful configuration to maintain consistent resource availability and avoid scheduling issues.

CLUSTER LIFECYCLE MANAGEMENT

- Automatic updates to MicroK8s may disrupt xCore deployments by altering configurations or introducing Kubernetes version mismatches. Users should disable automatic updates or carefully monitor updates.
- Custom configurations like `service-node-port-range` or SCTP kernel module enablement may not persist across reboots. Users should create startup scripts to reapply these settings.

SCTP KERNEL MODULE

- SCTP support is required for private radio integrations (e.g., gNodeB/eNodeB) but must be manually enabled (`modprobe sctp`). Users should also configure persistence across reboots.
- Certain kernel versions may lack SCTP support, requiring updates or custom builds.

RBAC AND HELM CONFIGURATION

- MicroK8s uses a simplified RBAC setup. Missing service accounts or role bindings for Helm deployments can lead to failed installations of xCore components.
- Network issues, proxy restrictions, or outdated Helm versions may block access to Expeto's private Helm repositories. Users should validate connectivity and Helm configurations before deployment.

INGRESS AND TLS CERTIFICATES

- While MicroK8s includes a built-in cert-manager (`microk8s enable cert-manager`), it may conflict with Helm-based configurations. Users may need to manually install cert-manager via Helm for xCore.
- Integrating custom TLS certificates often requires additional steps, such as importing certificates into the cluster or modifying ingress configurations.

VALIDATION AND DEBUGGING

- Users may skip verifying that all MicroK8s components (e.g., ingress, storage) are in a `Ready` state. Running `microk8s inspect` can help identify common issues.
- Skipping validation steps like `helm template` or `kubectl apply --dry-run` can result in unnoticed syntax or logic errors in `values.yaml` .
- Debugging MicroK8s components (e.g., ingress) can be challenging due to limited logging by default. Users may need to enable verbose logging or access component-specific logs for troubleshooting.
- Users should test service connectivity between xCore components using tools like `kubectl exec` and `ping` to identify potential networking or DNS issues early.

ADVANCED OPTIMIZATIONS

- High-throughput deployments require optimized kernel parameters (`sysctl`). Users can apply these changes using `nodeConfig` in `values.yaml` .
- Users must review resource requests and limits in `values.yaml` to ensure they align with the cluster's capacity. Misaligned settings can lead to resource contention or pod evictions.
- If autoscaling is configured, users should verify it is functioning (`kubectl get hpa`) and adjust thresholds based on expected load.

Installation Considerations - OpenShift

INTRODUCTION

Deploying Expeto xCore on OpenShift can be a powerful way to manage private and public 4G/5G networks in a Kubernetes environment. However, OpenShift's unique architecture, stricter security policies, and specific networking and resource management features may present potential stumbling blocks during installation and configuration.

This document highlights potential issues that may not be fully addressed in the initial installation guide, providing guidance to proactively address them and ensure a smoother deployment experience.

NETWORKING CHALLENGES

- OpenShift's default NodePort range (`30000–32767`) differs from the recommended range (`30000–38413`) in the installation guide. Adjusting the `serviceNodePortRange` requires administrative access and can conflict with cluster policies. Consider using OpenShift Routes as an alternative to NodePorts.

- Multus or VLAN integration can add complexity in OpenShift due to its networking stack (e.g., OVN-Kubernetes or OpenShift SDN). Ensure the OpenShift Network Operator is configured correctly if additional network interfaces or attachments are required.

- OpenShift's native Route mechanism may conflict with external ingress controllers like NGINX or Traefik. It is recommended to use OpenShift Routes for xCore instead of setting up a custom ingress controller.

DEPLOYMENT AND CONFIGURATION CHALLENGES

- OpenShift does not natively support Helm, though it is compatible. You may encounter RBAC issues when creating resources across namespaces. Ensure all required permissions are granted to the Helm service account in OpenShift.

- OpenShift's dynamic storage provisioning may not meet the `ReadWriteOnce` volume requirements outlined in the installation guide. If necessary, define a custom storage class to support the deployment.

SECURITY AND COMPLIANCE CHALLENGES

- OpenShift enforces stricter pod security via Security Context Constraints (SCCs). Some xCore components (e.g., `upf` , `amf`) may require custom SCCs to operate correctly. Ensure pods are granted the necessary permissions.

- Enabling SCTP for private radios in OpenShift might require custom SCCs and tolerations. Test the kernel module (`modprobe sctp`) on OpenShift nodes to ensure compatibility.

OPENSHIFT-SPECIFIC INTEGRATIONS

- OpenShift provides integrated tools like EFK and Prometheus, which might require configuration to align with xCore logs and metrics. Verify compatibility and customize as needed to ensure proper monitoring.

- OpenShift uses CRI-O as its container runtime, which can differ from Docker or containerd. Review custom container images or runtime configurations to ensure compatibility with CRI-O.

OPERATIONAL CONSIDERATIONS

- OpenShift's managed upgrade process can disrupt compatibility with Helm charts or custom components. Validate configurations and compatibility after each upgrade to ensure a smooth transition.

- OpenShift includes DNS functionality by default, but resolving DNS issues (`oc get dns`) may not be straightforward if misconfigurations arise. Verify DNS functionality before proceeding with installation.

Installation Considerations - VMware Tanzu

INTRODUCTION

Deploying Expeto xCore on VMware Tanzu provides a flexible and robust platform for managing private and public 4G/5G networks. However, Tanzu's architecture, specific tooling, and unique resource management strategies may introduce challenges not fully addressed in the initial installation guide.

This document identifies potential issues specific to VMware Tanzu and offers guidance for overcoming them to ensure a smooth deployment process.

TANZU-SPECIFIC CHALLENGES

- Tanzu leverages custom tools and configurations (e.g., Tanzu CLI) that may require modifications to generic Kubernetes instructions provided in the guide.
- Tanzu's default network policies might conflict with xCore's NodePort, LoadBalancer, or Multus networking setups, necessitating additional adjustments.
- Tanzu's vSphere CSI driver integration may need extra setup to align with xCore's storage requirements.
- Configuring Tanzu's Load Balancer often requires NSX-T or IP pool configurations, which are not detailed in this guide.
- Tanzu's lifecycle management tools may overwrite or disrupt manual configurations for xCore during cluster upgrades.

INTEGRATION AND RESOURCE MANAGEMENT

- Resource scheduling in Tanzu may require specific optimizations, such as CPU pinning or pod affinity, for latency-sensitive components.
- Configuring advanced networking setups, such as BGP routing for MetalLB, can be challenging and are not covered in the guide.
- TLS Management: Tanzu's unique certificate handling process may require additional steps to deploy and manage TLS certificates using Cert Manager.

MONITORING AND TROUBLESHOOTING

- Integrating xCore with Tanzu's monitoring tools, such as vRealize Operations, may involve significant additional effort beyond the scope of this guide.
- Errors related to NSX-T integration, Tanzu-specific Helm chart mismatches, or advanced networking configurations may require consulting VMware documentation or support.

SPECIALIZED FEATURES

- Enabling and validating SCTP for private radio integrations may necessitate non-standard kernel configurations in Tanzu environments.
- Multi-Zone and Hybrid Setups: Deployments in multi-zone or hybrid setups with Tanzu may involve complexities not addressed in this guide.

6.2.5 values.yaml Reference

- **imagePullSecrets:** Secrets for accessing private container registries.
- **imageCredentials:** Credentials (username, password, and registry information) for downloading container images.
- **serviceAccount:** Configures Kubernetes service accounts for xCore components.
- **role:** Defines Role-Based Access Control (RBAC) permissions for the deployment.

These sections configure the main xCore functions:

- **agent:** Handles communication with xControl and monitors the deployment.
- **amf (Access and Mobility Management Function):** Manages signaling for connected devices and session handoff.
- **nfcm (Network Function Configuration Manager):** Stores configuration and state for other components.
- **faultmanager:** Gathers fault and log information for monitoring.
- **udsf (Unstructured Data Storage Function):** Handles data storage for various network functions.
- **upf (User Plane Function):** Routes data streams between devices and external networks.
- **nrf (Network Repository Function):** Maintains a registry of other network functions.
- **idallocator:** Allocates unique IDs for sessions, components, and users.
- **ausf (Authentication Server Function):** Authenticates sessions and communicates with other components.
- **udm (Unified Data Management):** Stores user data and handles session and policy management.
- **udr (Unified Data Repository):** A database backend for user and session data.
- **smf (Session Management Function):** Manages sessions and coordinates data flow with AMF and UPF.
- **pcef (Policy and Charging Enforcement Function):** Handles policies and billing for network services (optional).

These sections configure networking and optional features:

- **createNetworks:** Automates the creation of network attachment definitions.
- **metallb:** Provides load balancing using Layer 2 or BGP protocols.
- **multus-cni:** Enables advanced networking with support for multiple network interfaces.
- **whereabouts:** Configures multi-node IP address management.
- **cni-route-override:** Adds custom route overrides for Multus networks.

These sections help configure scaling and optimize resource usage:

- **karpenter:** Configures Kubernetes auto-scaling for dynamic node provisioning.
- **global replica settings:** Enables autoscaling and sets minimum/maximum replicas for components.

These sections enable monitoring and testing features:

- **kubemonitoring:** Enables monitoring for the Kubernetes cluster and xCore components.
- **openspeedtest:** Configures a web-based speed test server.
- **iperf3:** Enables a terminal-based speed test server for bandwidth measurements.

These sections manage database configurations and storage backends:

- **galeracluster:** Configures MariaDB Galera Cluster for relational data storage.
- **redis:** Provides in-memory caching for performance optimization.
- **couchbase:** Configures Couchbase for distributed database storage.

These sections ensure secure operation of the xCore deployment:

- **tls:** Manages TLS certificates and encryption for secure communication.
- **securityConstraints:** Configures security context constraints, such as privileges and file system settings.
- **nodeConfig:** Specifies kernel and network configurations for cluster nodes.

6.3 Expeto CLI Guide for Operations

The Expeto CLI is a powerful command-line interface designed for IT operations personnel. It provides real-time access to manage and troubleshoot applications and data within xCore, xControl, and xRouter environments. This tool is essential for diagnosing network issues, querying subscriber states, and executing operational commands directly against the core network components.

6.3.1 Accessing the CLI

The CLI is accessed by executing an interactive shell within the designated CLI pod running in your Kubernetes cluster.

To start the CLI, use the following `kubectl` command, replacing `<release name>` with your deployment's release name (e.g., `local`):

```
kubectl exec -it <release name>-agent-0 -c cli -- cli
```

Once connected, you will be greeted by the `expeto_shell>` prompt.

6.3.2 Basic Navigation and Help

The CLI features a hierarchical, folder-like structure categorized by network types and functions (e.g., `lte` for 4G Core, `nr` for 5G Core, `debug` for operational commands).

- **List Directory (`ls`)**: Displays sub-directories, connected instances, and available commands (nouns and verbs) in the current context.
- **Change Directory (`cd`)**: Navigates through the command hierarchy. For example, `cd lte` moves into the 4G Core command set, and `cd ../nr` moves back and into the 5G Core command set.
- **Auto-complete (Tab)**: Press `Tab` at any time to auto-complete commands, noun names, or see available options. This is highly recommended to explore available parameters.
- **Help (`help` or `?`)**: Type `help` to see descriptions of commands in the current folder. You can also append `help` to a specific command to view its usage and options (e.g., `cmd sync-sub help`).

6.3.3 Command Structure

Commands in the Expeto CLI generally follow a **Verb + Noun + [Options]** pattern.

Verbs (Operations)

Verbs define the action to be taken on a resource. The available verbs depend on the noun: * **`read`** : Lists available records (e.g., a list of subscribers) or retrieves the detailed state of a specific record. * **`add`** : Adds a new record. * **`update`** : Modifies an existing record. * **`delete`** : Deletes a record. * **`show`** : Displays the current operational state of a component. * **`set`** : Changes the operational state of a component. * **`cmd`** : Executes a specific operational command or action.

Nouns (Resources)

Nouns represent the network components or data entities you are interacting with. To see available nouns for your current context, use the `ls` command. * **4G (LTE) Examples**: `hss` , `mme` , `sgw` , `pgw` , `enodeb` . * **5G (NR) Examples**: `amf` , `smf` , `upf` , `udr` , `udsf` .

6.3.4 Common Operations Examples

1. Exploring Available Commands

Upon logging in, you can see the top-level directories:

```
expeto_shell>ls
Sub-directories:
-debug
-lte     4G Core
-nr      5G Core
```

2. Listing Components and Supported Actions

Navigate into a specific core (e.g., 5G) and list available resources:

```
expeto_shell>cd nr
expeto_shell>nr>ls

Instances:
- name: birch-udr1
  description: HttpEndpoint(url=http://local-udr.default.svc.cluster.local:8080)

┌──────────────┬──────────────┐
│Noun          │Verb(s)       │
├──────────────┼──────────────┤
│amf           │read, delete  │
│associations  │read          │
│smf           │delete, read  │
│udsf          │read, delete  │
│...           │...           │
└──────────────┴──────────────┘
```

3. Reading and Paginating Records

To view records for a specific component, like the Unstructured Data Storage Function (UDSF):

```
expeto_shell>nr>read udsf
```

If there are many records, you can paginate through the results using the `-o` (offset) option:

```
expeto_shell>nr>read udsf -o 10
```

4. Viewing Detailed Record Information

To see the full configuration or state of a specific record, append the record ID to the `read` command:

```
expeto_shell>nr>read udsf UPF:Controller:enterprise1:slice1:SmfInitHbInfoKey
```

5. Executing Operational Actions with `cmd`

Operational actions, such as forcing a subscriber synchronization, are found in the `debug` directory. Always use `help` to understand the required parameters before executing a `cmd`.

```
expeto_shell>nr>cd ../debug/ops/
expeto_shell>debug>ops>cmd sync-sub help
```

The output will detail the options available, such as targeting a specific `imsi`, running it `siteWide`, or performing a specific `type` of sync (e.g., `UPDATE`, `RECREATE`).

7. xRouter

7.1 Expeto xRouter

Expeto xRouter is a public 3/4/5G core which can route traffic to public/private xCores and connect to IPX networks for public roaming use.

xRouter is distributed and installed and configured using Helm charts and Kubernetes.

As a containerized 3GPP compliant core with IPX and BGP support xRouter is optimised for for ease of deployment into enterprise-controlled cloud or bare metal environments, supporting connectivity using both public and private networks

The Expeto xRouter acts as the “home network”, receiving traffic redirected from a mobile network operator, then securely sends the traffic to Expeto xCore sites for public, or hybrid public/private implementations. The Expeto xRouter assumes the tasks of subscriber authentication, policy management, and network routing before connecting to the Public Expeto xCore.

Expeto xRouter features: - Enables seamless roaming of Subscribers between segmented Systems. - Secures sensitive data by controlling data path within geographic boundaries. - BGP route exchange for both north and south bound links - Dynamic Linear scalability of dataplane components for increased throughput and flexible cost efficient deployment - Built-in high availability using Kuberenetes flatform - Geographic failover and traffic optimisation for lower latency and increased availability

Policy & Traffic Engines

The Policy Engine maintains a database of all subscriber related information needed for routing policy decisions. The Policy Engine keeps the database in sync with all other xRouters and xCores. The Traffic Engine routes device (IMSI) connections to different destinations depending on metadata rules, these rules can be based on any part of the connection information including IMEI or IMSI value itself. It can also reject a device that fails to meet allow list criteria.

The Traffic Engine acts as a PGW/UPF towards the mobile network operator’s SGW/SMF. From the SGW/SMF perspective the Expeto xRouter is the terminating PGW/UPF. It then acts as a SGW/SMF sending traffic to a Public Expeto xCore. The Public Expeto xCore can then forwards data to the Private Expeto xCore or distributes it to local Systems (PGW/UPFs).

Subscriber Repository (HSS/AUSF/UDR/UDM)

The Subscriber Service Repository is a multi-tenant subscriber repository for all signalling traffic from the public/private radio networks. It contains the SIM database that represents subscriber registrations.

Expeto Agent

Expeto xRouter, like Expeto xCore, also includes Expeto Agent which synchronizes configuration between the Subscriber Repository and Expeto xControl to quickly create and expand 4G or 5G global networks. The Kubernetes based platform allows you to install multiple instances in different combinations across your network, enabling global coverage, high availability, performance scaling, test environments, and network evolution. Expeto network components can be installed behind a corporate firewall, on a DMZ, in a private cloud and at geographically remote sites according to your needs. Enterprises and Operators can manage the entire network from a browser through Expeto xControl or through the Expeto API.

7.2 Installation

7.2.1 Installation Overview and Preparation Guide

Expeto offers scalable and flexible solutions for deploying private and public 4G/5G networks within Kubernetes environments.

This guide provides step-by-step instructions to help you install and configure Expeto products effectively, whether you're managing a greenfield deployment or integrating it into an existing network. While the guide assumes advanced technical expertise, it is structured to accommodate the specific needs of Mobile Network Operators (MNOs), enterprise customers, and telco integrators.

Who Should Use This Guide?

- **Mobile Network Operators (MNOs):** MNOs deploy Expeto products as part of a broader ecosystem including xControl and xRouter to manage large-scale, multi-site networks for internal use or customer traffic. These users typically require advanced configurations for multi-tenancy, redundancy, and scalability, and asusme extensive expertise in telecommunications and networking such as BGP and IPX.
- **Enterprises Deploying public xCores:** These users deploy xCore as a "router" for mobile traffic to egress onto private networks without the need for full telco infrastructure. They usually have enterprise networking experience and require a streamlined installation process leveraging Expeto's managed services.
- **Integrators and Enterprise networking experts using Private Radios:** Specialists and advanced enterprise staff integrating with private radios (gNodeB/eNodeB) for customized network solutions. They possess advanced telco experience and expertise in configuring radio hardware, PLMN IDs, and complex networking interfaces.

This guide assumes that foundational decisions about deployment, such as PLMN IDs, radio hardware, and Kubernetes platform configurations, have been made. While Expeto uses a **Product Intake Form** internally to gather this information, this guide provides a comprehensive installation path even if that data hasn't been formalized.

Overview of the Installation Process

The installation process for Expedo products is structured into five main stages:

1. **Prerequisites:**
   - Ensure Kubernetes readiness, install necessary plugins, and set up required tools like `kubectl` and `helm`.
   - On OpenShift/MicroShift all commands performed with kubectl can be performed with the `oc` command instead, in this case we recommend using an alias, e.g. alias `kubectl=oc`
2. **Add Expeto Helm Repositories:**
   - Authenticate and retrieve the necessary Helm charts.
3. **Customize Configuration:**
   - Create and configure the `values.yaml` file to define behavior and connectivity, including PLMN IDs, networking interfaces, and scaling requirements.
4. **Deploy:**
   - Use Helm to install and monitor the deployment process.
5. **Validate and Optimize:**
   - Verify the deployment, ensure proper functionality, and apply performance optimizations.

Prerequisites

Before you begin the installation, ensure that your environment meets the following requirements. This section outlines system requirements, tools, and preparatory steps necessary for a smooth deployment process.

SYSTEM REQUIREMENTS

- **Version:** Kubernetes 1.23 or later.
- **Cluster Resources:**
  - At least 3 worker nodes for redundancy.
  - Minimum of 4 CPU Cores and 8 GB memory per node for a total of at least 16 cores and 32 GB per cluster.
  - Persistent storage system available from all nodes configured for `ReadWriteOnce` volumes.

  For more information on resource sizing, see System Resource / Sizing Guide
- **Plugins/Operators/Features required:**
  - DNS Resolver.
  - Cert Manager.
  - Storage plugin compatible with `ReadWriteOnce` volumes.
- **Node Kernel:**
  - Linux kernel 5.13 or later, compatible with Kubernetes.

TOOLS AND UTILITIES

- **kubectl:** Command-line tool for interacting with Kubernetes clusters.
  - Install kubectl
- **helm:** Kubernetes package manager for managing Helm charts.
  - Install Helm

Depending on your Kubernetes environment, the following tools must be configured to ensure a functional deployment. Adjust the steps based on whether you are using MicroK8s, OpenShift/MicroShift, or VMware Tanzu.

> **Important**
>
> Review the document below of your chosen Kubernetes platform to understand the common installation and configuration challenges:
>
> - Canonical MicroK8s
> - RedHat OpenShift/MicroShift
> - WMware Tanzu

DNS Resolver

A DNS resolver provides name resolution for Kubernetes services, so that services are able to communicate with each other.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

```
sudo microk8s enable dns
```

OpenShift/MicroShift have DNS capabilities enabled by default.

For further configuration options, please refer to DNS Operator in OpenShift.

VMware Tanzu comes with integrated DNS resolution capabilities via CoreDNS.

If you require public service resolution, Tanzu supports synchronizing with external DNS providers via ExternalDNS.

Cert Manager

Cert Manager is required to manage TLS certificates for secure communication.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

```
sudo microk8s enable cert-manager
```

Please refer to Cert-manager documentation for configuring cert-manager with OpenShift.

Please refer to Tanzu Cert Manager Installation for installation steps with VMware Tanzu.

Persistent Storage

A storage class must be configured to support persistent volumes.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

MicroK8s does not ship with a storage class plugin to span multiple worker nodes. For Demo purposes the following can be used on single node installs only. This will not work for multi-node installs because the storage will not be replicated across nodes causing Pods to loose data if they migrate between Nodes. Other options are available for larger MicroK8s installs, see Cannonical documentation for examples and guidelines.

```
sudo microk8s enable hostpath-storage
```

OpenShift supports a wide variety of persistent storage options.

Read more at Understanding Persistent Storage in OpenShift.

See following document for MicroShift MicroShift Storage Configuration.

Tanzu also supports a variety of persistent storage options.

Read more at Using Persistent Storage in vSphere with Tanzu.

CLUSTER PREPARATION

1. Get a copy of the kube-config file for your cluster, name it config and place it in the .kube directory residing in your home directory, be sure to set permissions to 0600 or equivalent on non Linux platforms.
2. **Verify Kubernetes Access:**
   - Ensure you can connect to your Kubernetes cluster using kubectl:

     ```
     kubectl get nodes
     ```

     Confirm all nodes are in a Ready state.
3. **Helm Installation:**
   - Verify that Helm is installed and functioning:

     ```
     helm version
     ```
4. **Ulimits:**

   The Expeto solution ships with a DaemonSets/MachineConfigs that will automatically configure things like kernel modules, buffer sizes, sysctl parameters and ulimits on most platforms but in some cases slight additional modifications might be recommended.

   For MicroK8s: Edit the container environment configuration and increase the ulimit to 1048576

   ```
   echo 'ulimit -n 1048576 || true' >> '/var/snap/microk8s/current/args/containerd-env'
   ```
5. **Configure allowed NodePort range:**

   The default installation depends on the availability of NodePorts and having a wider range of ports enabled than is default with most k8s deployments.

Unless your install only uses Multus based interfaces make sure to allow node ports in the range of 30000-38413 before installing the chart.

Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMWare Tanzu**

```
echo '--service-node-port-range=1024-38413' >> /var/snap/microk8s/current/args/kube-apiserver
```

Then, restart microk8s (`sudo microk8s.stop && sudo microk8s.start`)

```
oc patch network.config.openshift.io cluster --type=merge -p \
'{
   "spec":
   { "serviceNodePortRange": "1025-38413" }
}'
```

Note that it might take several minutes for the new config to be applied.

For a multi-master cluster see the following doc for how to identify the leader: Leader identification

For a single master cluster or once the cluster leader has been identified make the following change:

```
echo '- "--service-node-port-range=-1024-38413"' >> /var/vcap/jobs/kube-apiserver/config
```

Then, restart the kube-apiserver

6. **Advanced Networking:**

   - **Using LoadBalancers:**

     Please refer to Services, Load Balancing, and Networking in Kubernetes documentation for detailed information regarding setting up load balancers and other advanced networking topics.

   - **Using Multus for Advanced Networking:**

     For platforms that support Multus it can either be installed separately or installed by setting `multus.enabled` to true during install.

     It's recommended to also install Whereabouts for IPAM at the same time.

     For installations that have a Multus plugin, it is recommended to use the built-in version.

     Canonical MicroK8s **RedHat OpenShift/MicroShift** **VMware Tanzu**

     ```
     sudo microk8s enable community
     sudo microk8s enable multus
     ```

     Please refer to Multus CNI documentation for configuring Multus and Whereabouts with OpenShift.

     Please refer to Cert-manager documentation for configuring Multus with VMWare Tanzu.

     For further information regarding installation and configuration of Multus on other platforms, please refer to Multus Quick-Start Guide or Multus Documentation.

7. **Role-Based Access Control (RBAC):**

   Note: If you plan to use Role-Based Access Control (RBAC) in your Kubernetes cluster, **it must be enabled before deploying Expeto products.**

   If RBAC is enabled after deployment, the cluster must be redeployed before it will function correctly.

   When the cluster is deployed, the helm chart will automatically configure the required roles, bindings, and permissions for RBAC.

   Refer to Using RBAC Authorization in Kubernetes documentation for further information.

Adding the Expeto Helm Repositories

All Expeto deployments rely on Helm charts hosted in Expeto's private repositories. This step outlines how to configure access to these repositories, ensuring that the necessary charts and container images can be downloaded during installation.

Before proceeding, ensure you have:

1. **Helm Repository Credentials:** Provided by Expeto Support for accessing container images and Helm charts.
2. **Deployment-Specific Information:**
    - PLMN IDs, RAN details, and other configuration specifics (if applicable).
    - Network interface and CIDR details for northbound (N2/N3/N9) and southbound (N6) interfaces.

1. Use the provided credentials to add the Expeto Helm repositories to your local Helm configuration.

```
helm repo add expeto-ngc https://repo.expeto.io/repository/ngc --username '<your_username>' --password '<your_password>'
```

   Replace `<your_username>` and `<your_password>` with the credentials provided by Expeto Support.
2. Update your local Helm repository cache to ensure you have the latest chart versions:

```
helm repo update
```

After adding the repositories, confirm that they are accessible and contain the required charts:

1. List the added repositories:

```
helm repo list
```

2. Search for available charts to confirm connectivity:

xCore **xRouter** **xControl**

```
helm search repo xcore
```

```
helm search repo xrouter
```

```
helm search repo xcontrol
helm search repo expeto-docs
```

These commands should return a list of available charts for each product in the Expeto repository.

If you encounter issues adding the repositories or accessing charts:

- Ensure your username and password are correct and match the credentials provided by Expeto Support.
- Confirm that your machine has access to the internet and can reach `https://repo.expeto.io`.
- Ensure that Helm is installed and functioning properly and is more recent than 3.0.9 than by running:

```
helm version
```

- Contact Expeto Support for assistance if the problem persists.

Customize Configuration

The `values.yaml` file is a critical part of the Expeto xCore deployment process.

It defines parameters for the entire deployment, including networking, resource scaling, radio configuration and advanced features. This guide will help you customize the file to align with your deployment requirements.

> **Important**
>
> - All changes to deployments must be made in the `values.yaml` file. This file serves as the **source of truth** for your Helm deployments. **Any manual changes made directly to resources (e.g., via kubectl) will be overwritten when scaling occurs or when the `values.yaml` file is applied again.**
> - For more details on applicable parameters in the `values.yaml`, see values.yaml Reference

To ensure consistency and prevent loss of changes, always update the values.yaml file and reapply it with Helm commands.

Make sure you store a copy of the values.yaml file in a secure place, this file acts as your disaster recovery and will allow you to rebuild the entire solution with minimal effort very rapidly.

To access the `values.yaml` file:

1. Pull the Helm chart from the `expeto-ngc` repository:

   xCore **xRouter** **xControl**

   ```
   helm show values expeto-ngc/xcore
   ```

   ```
   helm show values expeto-ngc/xrouter
   ```

   ```
   helm show values expeto-ngc/xcontrol
   ```

   This is to get the default values, multiple examples are also available to demonstrate different setup options and possibilities for a variety of Kubernetes distributions

   The `values.yaml` file will be in the extracted directory.

2. Pull examples from the `expeto-ngc` repository:

   xCore **xRouter**

   ```
   export tmp_dir="$(mktemp -d)" && helm pull ngc/xcore  --untar --untardir ${tmp_dir} && mv "${tmp_dir}/xcore/examples" ./ && rm -Rf "${tmp_dir}"    === "xCore"
   ```

   ```
   export tmp_dir="$(mktemp -d)" && helm pull ngc/xrouter  --untar --untardir ${tmp_dir} && mv "${tmp_dir}/xrouter/examples" ./ && rm -Rf "${tmp_dir}"
   ```

   This will download the chart, extract it to a temporary directory and then move the examples directory to the current directory. Alternatively the whole chart can simply be downloaded with "helm pull" and extract.

- **Edit the File:** Open `values.yaml` and update the sections relevant to your deployment. Reference the comments in the file for additional guidance.
- **Validate Changes:** Use Helm to validate the configuration before applying it:

xCore **xRouter** **xControl**

```
helm template expeto-ngc/xcore -f values.yaml > validated_output.yaml
```

```
helm template expeto-ngc/xrouter -f values.yaml > validated_output.yaml
```

```
helm template expeto-ngc/xcontrol -f values.yaml > validated_output.yaml
```

Review the `validated_output.yaml` for any errors or issues before proceeding.

- **Start with Defaults:** Use the default values.yaml file as a baseline and modify only the required fields.
- **Document Changes:** Add comments to track customizations for easier troubleshooting or future updates.
- **Test Incrementally:** Validate and test changes incrementally to catch errors early.

Deploy and Verify

With your environment prepared, Helm repositories configured, and the `values.yaml` file customized, you are ready to deploy into your Kubernetes cluster. Following the links below for details of the deployment process, verification steps, and troubleshooting tips of the Expeto platform components:

- **xCore**
- **xRouter**
- **xControl**

7.2.2 Deployment & Verification

Expeto Platform Sizing Technical Guide

Introduction

This document outlines the compute and storage requirements for Expeto products running in a Kubernetes deployment environment.

Platform Components Overview

The Expeto platform consists of several key components that work together to deliver network services:

xCore / xRouter

- Built to scale horizontally based on data throughput capacity needs
- Configurable redundancy from none to full geographic redundancy
- xRouter deployment:
  - Typically hosted centrally by the network operator
  - Geo-redundant deployments available for High Availability, Data Sovereignty, or low latency localization
- xCore deployment:
  - Typically hosted by the end customer
  - Flexible deployment model with one to many xCores to match specific use cases

xControl / xView

- Deployment model:
  - Typically deployed centrally by the network operator
  - Geo-redundant deployments available for High Availability
- Scaling:
  - Built to scale vertically depending on node specifications
  - Sized according to required active subscribers
- Redundancy options:
  - Single site with failover capability
  - Full geographic redundancy

All platform components are deployed in Kubernetes Clusters for consistent management and orchestration.

Kubernetes Infrastructure Key Definitions

Node: A physical or virtual machine with defined capacity to run Kubernetes workloads

Cluster: A group of nodes working together to run an Expeto Component in a single geographic location

Redundancy: Each cluster contains 2+ nodes to provide service continuity, ideally 3+ nodes depending on Kubernetes environment.

Geographic Redundancy: Utilizes 2+ cluster locations, each with 2+ nodes, for multi-site resilience, ideally 3+ nodes depending on Kubernetes environment.

Assumptions:

- The underlying infrastructure is composed of Kubernetes nodes running either on bare metal or on virtual machines with a modern Linux kernel (recommended versions 5.13+), or using AKS, GKS, EKS, or Digital Ocean hosted Kubernetes offerings.
- The same nodes can be used for both the control plane and the worker nodes though for production systems it is strongly recommended to use separate nodes. At minimum, 3 nodes are always needed to guarantee redundancy and availability for both control-plane and worker nodes.
- Expeto is not recommending the size, specification, or number of the underlying physical servers, only the resources allocated to the Kubernetes nodes required to run the Expeto components. A compute partner may recommend the physical server and network switch designs.
- This design is for a single cluster deployment at a co-location or cloud provider running on x86_64 (amd64) based processors and does not take into consideration geographic or other types of 'failover'. It is not recommended to have clusters spanning multiple geographically separated co-locations or geographic regions.
- High Availability (HA) is achieved using multiple clusters in an active-active setup.
- The choice of hypervisors/operating systems for nodes, and their maintenance and support, is left to the discretion of the operations team as per best practices.

Node-selectors, tolerations, and affinity

Workloads can be associated and restricted to specific nodes but special care must be taken to prevent single points of failure or compromise performance or load balancing.

The simplest way to associate a workload with a specific node is to use node-selectors in combination with labels. This allows pods from a specific workload to be assigned to dedicated nodes which can be either VMs or bare metal. This is generally only recommended when physical constraints are in place because of network interface cards or other infrastructure components.

Affinities are more powerful and flexible than node-selectors and allow "soft" preferences. These soft preferences allow failover to other nodes and the ability to choose if pods should move back when a node does come back. Anti-affinities are also supported and can be used to achieve redundancy without restricting load-balancing.

Taints and tolerations are the most powerful but also the most complex. They allow advanced scheduling logic to be written to the exact specifications and requirements of the hardware and customer.

These topics should be familiar to most experienced Kubernetes administrators and as such will not be covered in more detail here but for more information see: - Assign Pods to Nodes - Taint and Toleration

Node Configuration and Performance Considerations

For best performance and resource utilization, configuration changes should be made to the underlying hardware and host operating systems. This includes but is not limited to changing kernel settings and parameters, network interface settings and queue lengths, NIC drivers, and settings.

These settings and modules are configured automatically by MachineConfigs or DaemonSets but can be configured manually if desired.

Supported Kubernetes Container Platforms

- VMware Tanzu
- RedHat OpenShift/MicroShift
- MicroK8s
- EKS
- AKS*
- GKE*
- Digital Ocean Kubernetes*

* Some hosted Kubernetes platforms have limitations on performance and networking configurations that should be considered.

xCore / xRouter Scaling Key Assumptions:

- Default size descriptions are for production deployments using built-in Kubernetes networking.
- VPP/DPDK/SR-IOV networking functionality can be enabled for substantially better networking performance.
- Capacity limits are estimated based on typical deployments. Actual limits depend on traffic model of end customers
- A k8s node can be deployed bare metal or on a VM. For redundancy, multiple VMs should always be distributed on as many physical servers as specified.
- HA Sizing recommendations assume an active-active configuration.
- Unless otherwise specified, CPU/Core count refers to physical cores and not hyperthreaded vCPUs.
- All disks should be enterprise SSD storage in a redundant setup, made available as PersistentVolumes via PersistentVolumeClaims from any node in the cluster with ReadWriteOnce access.

Sizing Methodology

1. **Determine redundancy requirements (cluster size)**
   - 1 node – no redundancy (total capacity = 1 node)
   - 2 nodes – failover redundancy (total capacity = 1 node, MicroK8s only)
   - 3+ nodes – horizontal scalable redundancy (total capacity = n nodes - 1)
   - Geographic requires 2+ clusters (total capacity = n clusters - 1)
2. **Select scale per node**
   - 4 CPU / 32 GB – up to 2 Gbps or 50,000 subscribers
   - 4 CPU / 64 GB with VPP/DPDK/SR-IOV – up to 10 Gbps or 250,000 subscribers
   - 8 CPU / 64 GB – up to 4 Gbps or 100,000 subscribers
   - 8 CPU / 96 GB with VPP/DPDK/SR-IOV – up to 20 Gbps or 500,000 subscribers
3. **Build clusters based on capacity requirements**

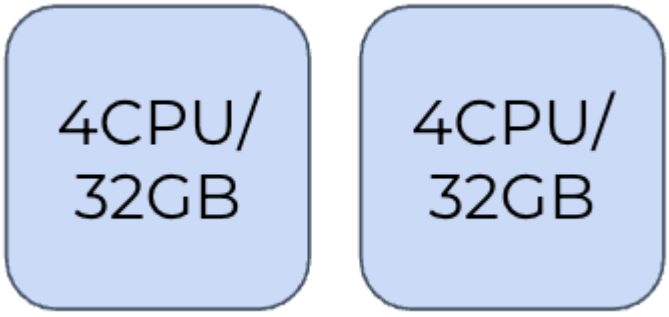
Storage Requirements

Disk per cluster: 100GB per 2 Gbps capacity

Example Configurations **2 Gbps (No Redundancy):**

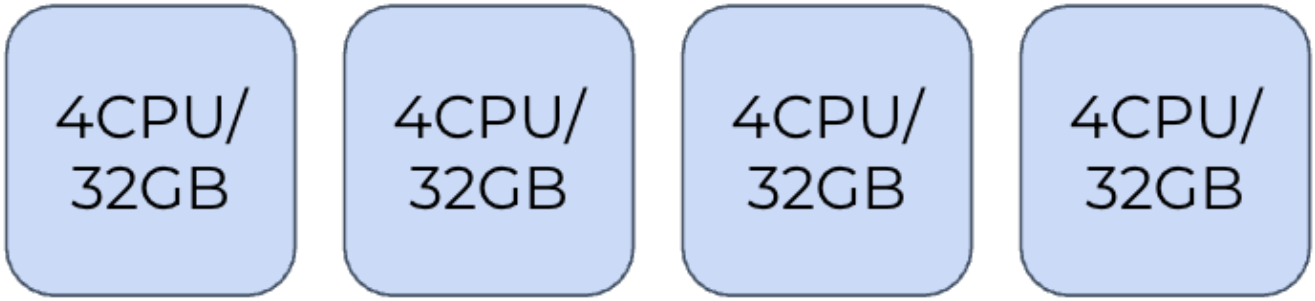
- Single node: 4 CPU / 32 GB
- 100 GB Disk Storage

2 Gbps (With Redundancy):



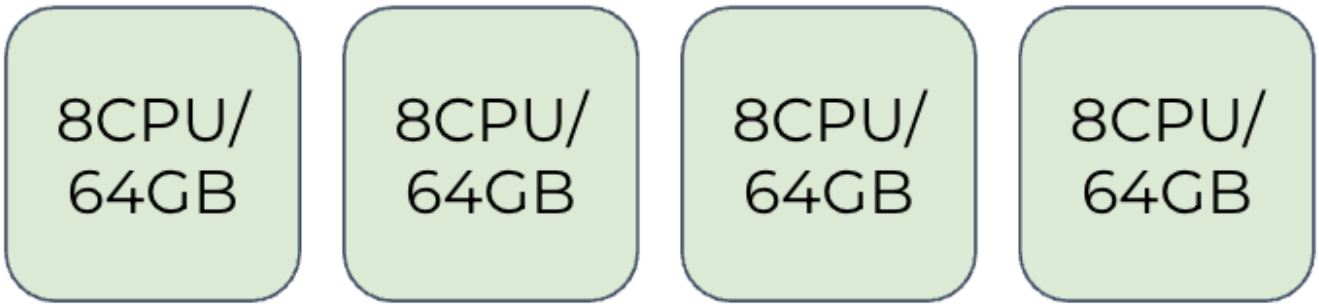
- Two nodes: 4 CPU / 32 GB each
- 200 GB Disk Storage

6 Gbps (Scalable Redundant):



- Four nodes: 4 CPU / 32 GB each
- 300 GB Disk Storage

12 Gbps (Scalable Redundant):



- Four nodes: 8 CPU / 64 GB each
- 600 GB Disk Storage
- Higher capacity per node for increased throughput

xControl / xView Scaling Key Assumptions:

- Capacity limits are estimated based on typical deployments. Actual limits depend on traffic model of end customers
- A k8s node can be deployed bare metal or on a VM. For redundancy, multiple VMs should always be distributed on as many physical servers as specified.
- HA Sizing recommendations assume the use of an active-active configuration.
- Unless otherwise specified, CPU/Core count refers to physical cores and not hyperthreaded vCPUs.
- All disks should be enterprise SSD storage in a redundant setup, made available as PersistentVolumes via PersistentVolumeClaims from any node in the cluster with ReadWriteOnce access.

Sizing Methodology

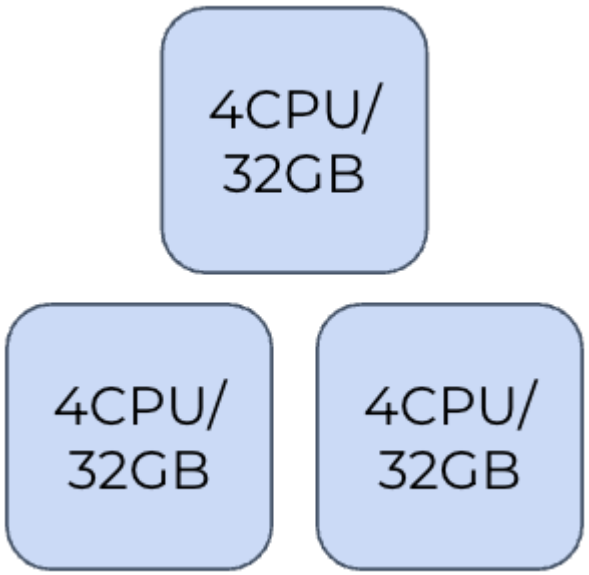
1. **Determine redundancy requirements (cluster size)**
   - Single Location (2+ nodes)
   - Geographic failover (2 clusters)
2. **Scale per node**
   - 4 CPU / 32 GB – up to 100,000 subscribers
   - 8 CPU / 64 GB – up to 500,000 subscribers
   - 16 CPU / 128 GB – up to 1,000,000 subscribers
3. **Build clusters based on capacity requirements**

Storage Requirements

Disk requirements per cluster:

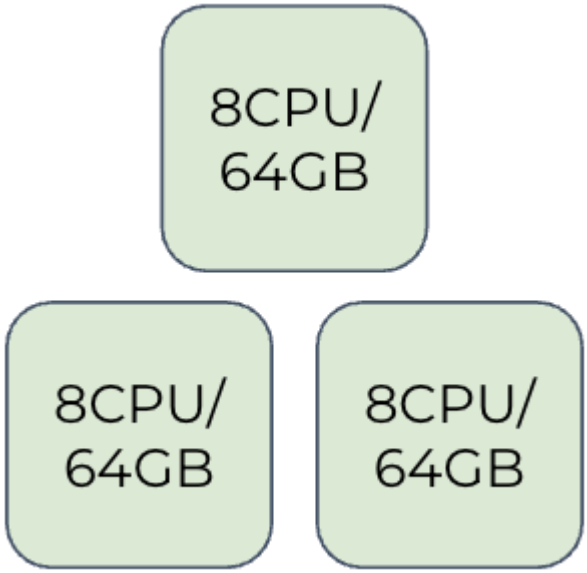
- 100K subscribers = 1 TB
- 500K subscribers = 4 TB
- 1M subscribers = 8 TB

Example Configurations **100,000 Subscribers:**



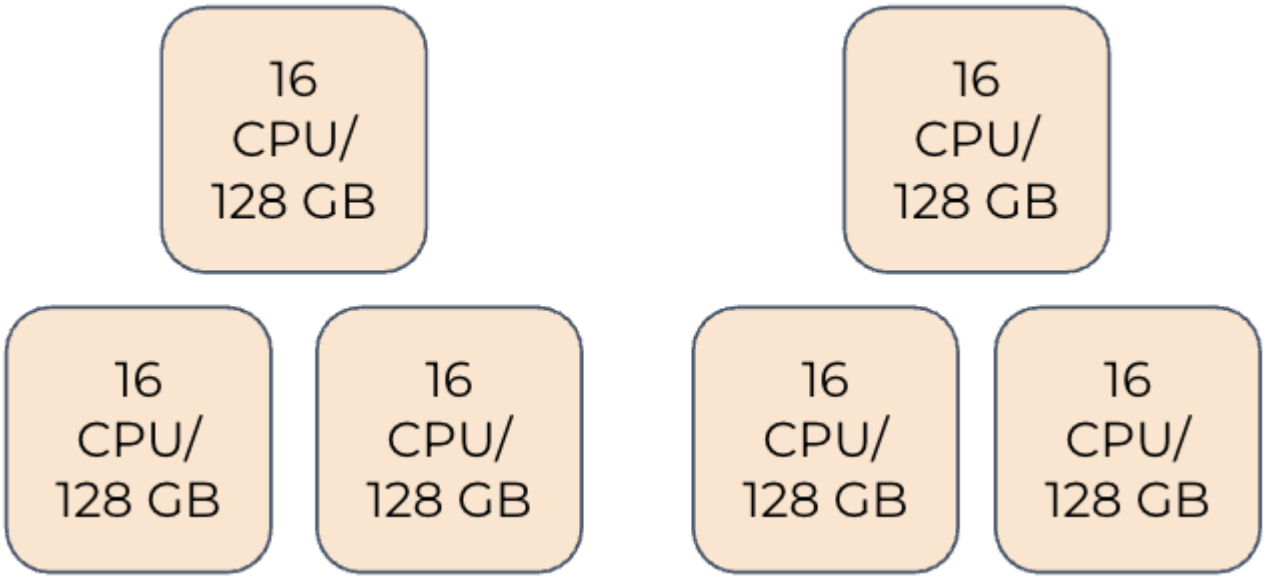
- Three nodes: 4 CPU / 32 GB each
- 1 TB Disk Storage

500,000 Subscribers:



- Three nodes: 8 CPU / 64 GB each
- 4 TB Disk Storage

1,000,000 Subscribers with Geographic Failover:



- Location A: Three nodes of 16 CPU / 128 GB each
- Location B: Three nodes of 16 CPU / 128 GB each
- Provides complete geographic redundancy
- 8 TB Disk Storage

Conclusion

This technical guide provides the foundation for properly sizing an Expeto platform deployment. When planning your implementation, consider:

1. Expected throughput requirements (in Gbps)
2. Number of subscribers to support
3. Redundancy needs (local and/or geographic)
4. Growth projections to ensure adequate scaling capacity

For specific sizing recommendations tailored to your environment, consult with Expeto technical specialists.

7.2.3 values.yaml Reference

- **imagePullSecrets:** Secrets for accessing private container registries.
- **imageCredentials:** Credentials (username, password, and registry information) for downloading container images.
- **serviceAccount:** Configures Kubernetes service accounts for xCore components.
- **role:** Defines Role-Based Access Control (RBAC) permissions for the deployment.

These sections configure the main xCore functions:

- **agent:** Handles communication with xControl and monitors the deployment.
- **amf (Access and Mobility Management Function):** Manages signaling for connected devices and session handoff.
- **nfcm (Network Function Configuration Manager):** Stores configuration and state for other components.
- **faultmanager:** Gathers fault and log information for monitoring.
- **udsf (Unstructured Data Storage Function):** Handles data storage for various network functions.
- **upf (User Plane Function):** Routes data streams between devices and external networks.
- **nrf (Network Repository Function):** Maintains a registry of other network functions.
- **idallocator:** Allocates unique IDs for sessions, components, and users.
- **ausf (Authentication Server Function):** Authenticates sessions and communicates with other components.
- **udm (Unified Data Management):** Stores user data and handles session and policy management.
- **udr (Unified Data Repository):** A database backend for user and session data.
- **smf (Session Management Function):** Manages sessions and coordinates data flow with AMF and UPF.
- **pcef (Policy and Charging Enforcement Function):** Handles policies and billing for network services (optional).

These sections configure networking and optional features:

- **createNetworks:** Automates the creation of network attachment definitions.
- **metallb:** Provides load balancing using Layer 2 or BGP protocols.
- **multus-cni:** Enables advanced networking with support for multiple network interfaces.
- **whereabouts:** Configures multi-node IP address management.
- **cni-route-override:** Adds custom route overrides for Multus networks.

These sections help configure scaling and optimize resource usage:

- **karpenter:** Configures Kubernetes auto-scaling for dynamic node provisioning.
- **global replica settings:** Enables autoscaling and sets minimum/maximum replicas for components.

These sections enable monitoring and testing features:

- **kubemonitoring:** Enables monitoring for the Kubernetes cluster and xCore components.
- **openspeedtest:** Configures a web-based speed test server.
- **iperf3:** Enables a terminal-based speed test server for bandwidth measurements.

These sections manage database configurations and storage backends:

- **galeracluster:** Configures MariaDB Galera Cluster for relational data storage.
- **redis:** Provides in-memory caching for performance optimization.
- **couchbase:** Configures Couchbase for distributed database storage.

These sections ensure secure operation of the xCore deployment:

- **tls:** Manages TLS certificates and encryption for secure communication.
- **securityConstraints:** Configures security context constraints, such as privileges and file system settings.
- **nodeConfig:** Specifies kernel and network configurations for cluster nodes.

7.2.4 NF Configuration

HSS

CONFIGURATION QUESTIONS

- What are your home PLMNs? (please provide list with their realm)
- Do you want a diameter connection? if yes
- Do you want SCTP connection or TCP connection for diameter?
- Do you have specific IP to use (primary and secondary for multi-homing)? (default listen IP is 0.0.0.0)
- Do you have specific port for diameter connections? (default is 3868)
- What is the diameter host name of HSS?
- What is the diameter realm of HSS?
- What are diameter host routing rules, if any? (list of peer realms and their host list)
- Are you expecting us to initiate diameter exchange? if yes please provide a list of IP, port, host, realm of external systems that we need to connect to.
- Do you want a map connection? if yes
- Please provide list of IP:port for our server that are whitelisted by STP links
- Please provide sccp protocol version to be used in m3ua (defaults to ANSI if not provided)
- Please provide below details for this HSS
    - point code
    - HLR global title
    - SCCP header patterns with below details (currently we support only one pattern)
    - address indicator (defaults to 11 = GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_ONLY, ssn, dpc present)
    - global title type (defaults to GT0010 = GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_ONLY)
    - translation type (defaults to 10 = no translation required)
    - numbering plan (defaults to 1 = ISDN)
    - nature of address (defaults to 4 = INTERNATIONAL)
- Please provide list of STP network that will be connecting to. For each STP link please provide
    - primary and backup points codes of STP link
    - network indicator (defaults to 2 = NATIONAL if not provided)
    - network appearance (default is not to use this field)
    - traffic mode (defaults to 2 = load share, other value supported is 1 = override)
    - list of SCCP header patterns with below details (usually there will be only one pattern)
    - address indicator (defaults to 11 = GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_ONLY, ssn, dpc present)
    - global title type (defaults to GT0010 = GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_ONLY)
    - translation type (defaults to 10 = no translation required)
    - numbering plan (defaults to 1 = ISDN)
    - nature of address (defaults to 4 = INTERNATIONAL)
    - list of IP:PORT of the STP link
    - routing contexts, only if the STP requires it. Usually most STPs do not use routing contexts.

- Please provide list of routing rules based on global title with below details
    - address indicator (defaults to 11 = GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_ONLY, ssn, dpc present)
    - global title type (defaults to GT0010 = GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_ONLY)
    - translation type (defaults to 10 = no translation required)
    - numbering plan (defaults to 1 = ISDN)
    - nature of address (defaults to 4 = INTERNATIONAL)
    - list of global title prefixes to match
    - routeTo list to decide which PointCode and which SCCP format to use
- Do you want a SMSc service? if yes
- Please provide below MAP details for this SMSc
    - point code
    - SMSc global title
    - SCCP header patterns with below details (currently we support only one pattern)
    - address indicator (defaults to 11 = GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_ONLY, ssn, dpc present)
    - global title type (defaults to GT0010 = GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_ONLY)
    - translation type (defaults to 10 = no translation required)
    - numbering plan (defaults to 1 = ISDN)
    - nature of address (defaults to 4 = INTERNATIONAL)
- Do you want a SMPP service? If yes
- Please provide systemId and password
- Please provide routing rules based on MSISDN
- Do you want geo-redundant setup? if yes
- please provide three VMs where one VM will be arbitrator/witness-server for split brain scenario

CONFIGURATION ANSWERS TO VALUES MAPPING

- Home PLMNs are configured in agent values, e,g.

```
agent:
  hss:
    nodeConfig:
      hssConfig:
        homePlmns:
          - plmn:
              mcc: '313'
              mncs:
                - '260'
              realm: epc.mnc260.mcc313.3gppnetwork.org
            gts:
              - 11111111114 # smsc global title
              - 11111111111 # hlr global title
        # below values are for diameter that has been agreed with DRA links
        hostUri: hss03.node
        originHost: hss03.node.epc.mnc260.mcc313.3gppnetwork.org # diameter host name
        originRealm: epc.mnc260.mcc313.3gppnetwork.org # diameter host realm
        # below values are for map and will match with STP link details
        hlrGt: 11111111111 # hlr global title
        pointCode: 111111 # hlr point code
        hlrTt: 0
        hlrEncoding: BCD_ODD
        hlrGti: GLOBAL_TITLE_INCLUDES_TRANSLATION_TYPE_NUMBERING_PLAN_ENCODING_SCHEME_AND_NATURE_OF_ADDRESS
```

- Diameter config

```
diameter:
  enabled: true
  tcp: true # Do you want SCTP connection or TCP connection for diameter?
  primaryIp: 1.2.3.4 # default is 0.0.0.0
  secondaryIp: 2.3.4.5 # default is not to use secondary IP
  port: 1111 # default is 3868
  host: hss # What is the diameter host name?
  realm: epc.mnc260.mcc313.3gppnetwork.org # What is the diameter realm?
  routingRules:
    - peerRealm: *
```

```
    peerHosts:
      - dra1.comfone.com
  peers: # Are you expecting us to initiate diameter exchange? if yes, e.g. below
   - ip: 1.2.3.4 # IP of the DRA
     port: 1234 # port of the DRA
     diameterRealm: bics.3gpp.org
     diameterHost: node01.bics.3gpp.org
     protocol: <SCTP|TCP>
     hssIp: 0.0.0.0 # IP to listen to when initiating connection with the peer
     hssPort: 2345 # port to listen to when initiating connection with the peer
```

- MAP config yaml

```
map:
  enabled: true # Do you want a map connection?
  sccpVersion: ITU # sccp protocol version
  hlr:
    pointCode: 3000
    globalTitle: '11111111112'
    sccp:
    - addressIndicator: '11'
      globalTitleType: 'GT0010'
  smsc:
    pointCode: 2010
    globalTitle: '11111111111'
    sccp:
    - addressIndicator: '11'
      globalTitleType: 'GT0010'
  servers: # Do you want us to be server or client?
    - ip: 0.0.0.0
      port: 2905
      name: SCTP1
  stps:
    - routingContexts: []
      pointCodes: [ '4444', '4445' ] # point code of the STP and it's backup point code
      networkIndicator: '0'
      networkAppearance: 0
      trafficMode: 2
      sccp:
      - addressIndicator: '19'
        globalTitleType: 'GT0100'
        translationType: '0'
        encodingScheme: '1'
        numberingPlan: '0'
        natureOfAddress: '0'
        name: stp1_sccp1
      ips:
        - ip: 192.168.2.20
          port: 64504
          name: unique_name # optional, unique name for this link, helpful in cli link status
          linkTo: SCTP1 # when we are server
    - routingContexts: [ '71', '72' ]
      pointCodes: [ '22222', '22223' ] # point code of the STP and it's backup point code
      networkIndicator: '0'
      sccp:
      - addressIndicator: '19'
        globalTitleType: 'GT0100'
        translationType: '0'
        encodingScheme: '1'
        numberingPlan: '0'
        natureOfAddress: '0'
```

```
      name: stp2_sccp1
    ips:
      - ip: 192.168.2.20
        port: 64505
        hlrIp: 192.168.3.4 # need this when we are client
        hlrPort: 2906 # need this when we are client
  routingRules:
    - addressIndicator: '19'
      globalTitleType: 'GT0100'
      translationType: '0'
      encodingScheme: '1'
      numberingPlan: '0'
      natureOfAddress: '0'
      globalTitlePrefixes:
        - 3933*
      routeTo:
        - pointCode: 4444
          sccp: stp1_sccp1
        - pointCode: 22222
          sccp: stp2_sccp1
```

- SMSc config

```
smsc:
  enabled: false
  smpp:
  clients:
  - systemId: expeto
    password: 12345678
  routingRules: []
  - destinationAddress: 111111111
    destinationRoute: SMPP
    sourceRoute: ANY
    systemId: expeto
```