

# MNI Guide

---

- [1. Merkator Legal Statements](#)
  - [1.1. Export restrictions](#)
  - [1.2. Disclaimer](#)
  - [1.3. Limitation of liability](#)
  - [1.4. Trademarks](#)
  - [1.5. Patent marking notice](#)
- [2. Target Audience](#)
- [3. Introduction](#)
  - [3.1. What does Merkator do?](#)
  - [3.2. What is the problem?](#)
  - [3.3. How is Merkator MarlinDT Network Intelligence \(MNI\) different?](#)
  - [3.4. Why should your business care?](#)
- [4. API-First](#)
- [5. Predictions](#)
  - [5.1. Simple](#)
  - [5.2. Complex](#)
- [6. Alerts](#)
- [7. Resources](#)
- [8. Time-Series](#)
  - [8.1. Concept](#)
  - [8.2. Behind the Scenes](#)
  - [8.3. Shadow Time-Series Tables](#)
    - [8.3.1. Bulk Create \(Import\)](#)
  - [8.4. Bulk Read \(Export\)](#)
    - [8.4.1. Create](#)
    - [8.4.2. Read](#)
    - [8.4.3. Update](#)
    - [8.4.4. Delete](#)
- [9. Resource Related Dependencies](#)
- [10. Database Schema](#)
- [11. Architectures](#)
  - [11.0.1. Application](#)
  - [11.0.2. DevOps CI/CD Pipeline](#)

- [11.0.3. Functional](#)
- [11.0.4. Integration](#)
- [11.0.5. Logical \(Cloud\)](#)
- [11.0.6. Logical \(On-Premise\)](#)
- [11.0.7. Cluster Replication](#)
- [11.0.8. Security](#)
  - [11.0.8.1. Through the API Gateway](#)
- [11.0.9. Internally](#)
  - [11.0.9.1. API Server Encrypted Storage](#)
  - [11.0.9.2. Role-Based Access Control \(RBAC\)](#)
- [11.0.10. Deployment](#)
- [12. DNS Resolution](#)
- [13. MNI Host Configuration](#)
  - [13.1. MNI Wildcard SSL Certificates](#)
  - [13.2. MNI Environment File](#)
    - [13.2.1. Common](#)
    - [13.2.2. OAuth2.0 / OpenID Connect](#)
    - [13.2.3. Alert Service](#)
    - [13.2.4. API Gateway](#)
    - [13.2.5. API Server](#)
    - [13.2.6. Fetch Service](#)
    - [13.2.7. User Interface \(UI\) Server](#)
    - [13.2.8. Identity and Access Management \(IAM\) Server](#)
    - [13.2.9. Predict Service](#)
    - [13.2.10. DNS Server](#)
    - [13.2.11. Non-Unique per Component](#)
    - [13.2.12. SSL Certificates](#)
    - [13.2.13. Example](#)
  - [14. Hosting Specifications](#)
    - [14.1. Standalone](#)
      - [14.1.1. Combined Single Host](#)
      - [14.1.2. Individual Hosts](#)
    - [14.2. Cluster](#)
      - [14.2.1. Per API Server Host](#)
      - [14.2.2. Other](#)

- [14.3. Operating Systems](#)
- [14.4. Host Firewalls](#)
- [15. License Generator](#)
- [16. Building the release](#)
  - [16.1. JavaScript Code Scanning](#)
  - [16.2. Individual Build Scripts](#)
  - [16.3. Release artifacts](#)
- [17. Deploying the release](#)
  - [17.1. Example Component Installation](#)
  - [17.2. Example Component Update](#)
  - [17.3. Example Component Upgrade](#)
  - [17.4. Example Component Removal](#)
- [18. Integration Lab](#)
  - [18.1. Host](#)
    - [18.1.1. DNS Records](#)
  - [18.2. User Interface](#)
  - [18.3. API Server](#)
  - [18.4. API Gateway](#)
  - [18.5. Alert Service](#)
  - [18.6. Fetch Service](#)
  - [18.7. Predict Service](#)
- [19. Third-Party](#)
  - [19.1. Map Rendering](#)
  - [19.2. Foreign Exchange Rates](#)
  - [19.3. Common Vulnerabilities and Exposures \(CVE\) Repository](#)
  - [19.4. Sample Data](#)
  - [19.5. World Boundaries](#)
  - [19.6. Third-Party Elements & Libraries](#)
    - [19.6.1. API Gateway](#)
    - [19.6.2. API Server](#)
    - [19.6.3. DNS Server](#)
    - [19.6.4. UI Server](#)
    - [19.6.5. Predict Service](#)
    - [19.6.6. Alert Service](#)
    - [19.6.7. Fetch Service](#)

- [19.6.8. License Generator](#)
- [19.7. Development Tooling - EchoAPI](#)
- [20. MNI OpenAPIs](#)
- [21. MNI User Interface \(UI\)](#)
  - [21.1. Login](#)
  - [21.2. Readiness](#)
  - [21.3. Country Selector](#)
  - [21.4. Dashboard](#)
  - [21.5. Cables](#)
    - [21.5.1. Common](#)
    - [21.5.2. Coax](#)
    - [21.5.3. Copper](#)
    - [21.5.4. Ethernet](#)
    - [21.5.5. Fiber](#)
      - [21.5.5.1. Single-Fiber](#)
      - [21.5.5.2. Multi-Fiber](#)
  - [21.6. Ducts](#)
  - [21.7. Poles](#)
  - [21.8. Network Equipment](#)
  - [21.9. Racks](#)
  - [21.10. Sites](#)
  - [21.11. Services](#)
  - [21.12. Trenches](#)
    - [21.12.1. Detail](#)
    - [21.12.2. Lifetime](#)
    - [21.12.3. Per Country](#)
  - [21.13. Quote-to-Cash \(Q2C\)](#)
  - [21.14. Providers](#)
    - [21.14.1. Email](#)
    - [21.14.2. Map](#)
    - [21.14.3. Kafka](#)
    - [21.14.4. Workflow Engines](#)
  - [21.15. Alerts](#)
  - [21.16. Settings](#)
  - [21.17. OpenAPIs](#)

- [21.18. Help/About](#)
- [21.19. Quit \(Logout\)](#)
- [22. MNI API Server Internals](#)
  - [22.1. Database](#)
  - [22.2. Geometry](#)
  - [22.3. Data Longevity](#)
  - [22.4. Database Backup](#)
  - [22.5. Tracking Changes](#)
  - [22.6. MNI Fetch](#)
  - [22.7. Alert Service](#)
- [23. MNI Alert Internals](#)
- [24. MNI Fetch Internals](#)
- [25. MNI Predict Internals](#)
  - [25.1. Duplication](#)
  - [25.2. Simple Prediction](#)
  - [25.3. Complex Prediction](#)
    - [25.3.1. Advantages for Time Series Prediction](#)
- [26. MNI User Interface Internals](#)
  - [26.1. Distribution Directories](#)
  - [26.2. Metadata](#)
- [27. Additional Q&A](#)
  - [27.1. Business / Functional / Sales Context](#)
  - [27.2. Feature Identification & Versioning](#)
  - [27.3. Data Relationships](#)
  - [27.4. Prediction Functionality](#)
  - [27.5. Architecture / Technology Choices](#)
- [28. Merkator Technical Services](#)
  - [28.1. By email](#)
  - [28.2. On the web](#)
  - [28.3. Technical documentation](#)
- [29. Corporate Headquarters](#)

---

# 1. Merkator Legal Statements

No part of this content may be reproduced in any form or by any means or used to make any derivative work (such as translation, transformation, or adaptation) without written permission from Merkator nv/sa and/or its affiliates ("Merkator"). Merkator reserves the right to revise or change this content from time to time without obligation on the part of Merkator to provide notification of such revision or change.

## **1.1. Export restrictions**

---

These products and associated technical data (in print or electronic form) may be subject to export control laws of the United States of America. It is your responsibility to determine the applicable regulations and to comply with them. The following notice is applicable for all products or technology subject to export control. These items are controlled by the U.S. government and authorized for export only to the country of ultimate destination for use by the ultimate consignee or end-user(s) herein identified. They may not be resold, transferred, or otherwise disposed of, to any other country or to any person other than the authorized ultimate consignee or end-user(s), either in their original form or after being incorporated into other items, without first obtaining approval from the U.S. government or as otherwise authorized by U.S. law and regulations.

## **1.2. Disclaimer**

---

THIS CONTENT AND ASSOCIATED PRODUCTS OR SERVICES ("MATERIALS"), ARE PROVIDED "AS IS" AND WITHOUT WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED. TO THE FULLEST EXTENT PERMISSIBLE PURSUANT TO APPLICABLE LAW, Merkator DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON- INFRINGEMENT, FREEDOM FROM COMPUTER VIRUS, AND WARRANTIES ARISING FROM COURSE OF DEALING OR COURSE OF PERFORMANCE. Merkator does not represent or warrant that the functions described or contained in the Materials will be uninterrupted or error-free, that defects will be corrected, or are free of viruses or other harmful components. Merkator does not make any warranties or representations regarding the use of the Materials in terms of their completeness, correctness, accuracy, adequacy, usefulness, timeliness, reliability, or otherwise. As a condition of your use of the Materials, you warrant to Merkator that you will not make use thereof for any purpose that is unlawful or prohibited by their associated terms of use.

## **1.3. Limitation of liability**

---

IN NO EVENT SHALL Merkator, Merkator AFFILIATES, OR THEIR OFFICERS, DIRECTORS, EMPLOYEES, AGENTS, SUPPLIES, LICENSORS, AND THIRD-PARTY PARTNERS, BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER, EVEN IF Merkator HAS BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, WHETHER IN AN ACTION UNDER CONTRACT, TORT, OR ANY OTHER THEORY ARISING FROM YOUR ACCESS TO, OR USE OF, THE MATERIALS. Because some jurisdictions do not allow limitations on how long an implied warranty lasts, or the exclusion or limitation of liability for consequential or incidental damages, some of the above limitations may not apply to you.

## **1.4. Trademarks**

---

All trademarks identified by ™ or ® are trademarks or registered trademarks in the US and may be registered in other countries. All product names, trademarks, and registered trademarks are property of their respective owners.

## **1.5. Patent marking notice**

---

For applicable patents, see [www.cs-pat.com](http://www.cs-pat.com). That website is intended to give notice under 35 U.S.C. § 287(a) of articles that are patented or for use under the identified patents. That website identifies the patents associated with each of the patented articles.

---

## **2. Target Audience**

---

It is expected Network Engineering IT/Developer resources will undertake any integrations and such resources are proficient in the following areas:

- RESTful HTTP
- Data Engineering

**Note:**

- *Professional Services* are available from Merkator to provide knowledge transfer and/or undertake any such integration.
- 

## **3. Introduction**

---

### **3.1. What does Merkator do?**

---

Merkator Group is an absolute market leader and trusted player in the field of geospatial solutions with global deployments of our physical and logical network inventory products

### **3.2. What is the problem?**

---

Every business is constantly faced with finding current accurate information but also attempting to plan, coping with the ever changing “What if ?” questions from their own employees in addition to shareholders, customers and external market influences

### **3.3. How is Merkator MarlinDT Network Intelligence (MNI) different?**

---

MNI takes historical and actual data and applies Machine Learning predictions providing insight from the past and foresight for the future by envisioning possibilities, from network intelligence delivering business intelligence

### **3.4. Why should your business care?**

---

MNI serves to help you make decisions or, at the very least, streamline a company's decision-making processes by collecting, visualising, and communicating data associated with the telecommunications infrastructure and the services it offers

**Every location and every network resource has a story, with MNI we bring it to life !**

**MarlinDT Network Intelligence** provides analytical and prediction insights for telecommunications infrastructure and services.

from hindsight to insight, delivering network intelligence foresight...

Hindsight: the ability to learn from the past by looking back at what has already happened.

Insight: the ability to interpret and respond to the present by developing self-awareness and honest perception.

Foresight: the ability to predict and prepare for the future by envisioning possibilities.

When combined, hindsight, insight, and foresight can be a foundation for better decision-making and outcomes.

## 4. API-First

MNI does not replace the fine-grain capabilities of either MarlinDT PNI or LNI, but serves to help you make decisions or, at the very least, streamline a company's decision-making processes by collecting, visualising, and communicating data associated with the telecommunications infrastructure and services it offers API-first Business Support System (BSS)

MNI is API-first meaning it's built from the ground up to support integration with existing eco-systems – any product that controls the data and does not act as barrier to the data has deployment longevity

- All Open API definitions published externally for machine-to-machine (M2M)
- All Open API definitions & tooling included within UI for interactive queries
- All MNI components use the same APIs – there are no hidden routes (microservices)
- 230 APIs are defined, documented and adhere to same security mechanisms

Refer to section MNI [MNI OpenAPIs](#) for further details.

## 5. Predictions

Due to the MNI Time-Series architecture, predictions are treated as updates to historical resource data with calculated/determined values being applied as modifications with a future date/time associated.

### 5.1. Simple

Inspecting the historical changes of a single resource, example a rack, and calculating the likelihood of the rack slot usage change and what slot usage would change to, i.e. free to used

### 5.2. Complex

Involve combining historical changes of multiple resources and potentially updated external data, example post address, and calculating the need for a resource to change to meet the external data conditions, or the need to extend an existing trench to cover new premises, or provide bandwidth to a new cell tower, data center etc.

## 6. Alerts

---

MNI aims to provide notification alerts via numerous mechanisms:

- NE Common Vulnerability (CVE) Passive Scan (requested previously by Accenture)
- Trench Utilization 75% / 90% / 100%
- Duct Utilization 75% / 90% / 100%
- Cable Utilization 75% / 90% / 100%
- Pole Utilization 75% / 90% / 100%
- Rack Utilization 75% / 90% / 100%
- NE Port Utilization 75% / 90% / 100%
- Transmission Utilization 75% / 90% / 100%
- Service Bandwidth Utilization 75% / 90% / 100%
- NE Error Rates 75% / 90% / 100%
- NE Classifiers Utilization 75% / 90% / 100%
- Work Planning Efficiency 110% / 130% / 150%
- Data Quality - geometry alignment / missing connected To / trench / cable / duct / pole / NE / service / site / offNet Postal Addresses

Providers can be defined via the MNI UI and API Server for:

- Low-Code/No-Code workflow
  - Camunda BPMN (pending)
  - ELSA (pending)
- Webhook
  - RESTful async callback (pending)
- Stream
  - Kafka Bus (pending)
- Email
  - SMTP (implemented)
  - Microsoft Exchange (pending)
- MNI UI (pending)

---

## 7. Resources

---

MNI resources include:

- Cable: coax, copper, ethernet, single/multi fiber, placement against a pole, duct, trench or to a piece of network equipment

- Duct: placement (within trench), gas, power, cable, water, configuration (x-way), type (duct or sub-duct)
- Pole: Height/Type
- Network Equipment & Ports: Vendor agnostic & Coax, ethernet, loopback, xDSL, Fiber, virtual
- Rack: size, slots
- Service: broadband, circuit, ethernet, FTTx, optical, SoGEA, unclassified, voice
- Site: ADU, COLO, commercial, exchange, MDU, SDU, street, unclassified
- Trench: depth/width, build job

For full details of what is possible to hold against a specific resource, refer to the published MNI OpenAPIs.

**Note:**

- Demographics on all resources when appropriate (residential, rural, commercial, urban, highways, unclassified)
- Each resource, if supplied with coordinates, has its country of existence checked and if necessary updated after creation.
- All resources, once when the resource is created, either on a volume basis i.e. distance, a unit basis per piece of equipment, have a cost calculated and applied.
- Cost rates can be managed separately through the published MNI API Server APIs or the included defaults changed prior to installation.

## 8. Time-Series

### 8.1. Concept

Every MNI resource has possibility of multiple diverging states including deleted but remains as the same originally defined resource Historical & predicted relationships remain between resources, for example a duct in a trench, such scenarios include:

- Trench is extended
- Trench has additional ducting installed
- Ducting has cables added, removed
- Cable is moved to different port, fiber splice is connected, disconnected etc.
- Network Equipment has additional cables connected, removed etc.
- Rack has equipment installed, removed etc.
- Rack slot becomes unusable, reserved etc.
- Service is moved between endpoints etc.
- New premises and sites appear etc.

As data is either collected by MNI (fetch) or received by MNI the historical data is updated, the resource container remains as is, not duplicated, nor replaced All relationships between resources are preserved during changes

- No need to reform relationships, for example, changes to rack such as equipment being installed are to the same physical rack, not a different incarnation of the rack
- Adding a service through two endpoints, automatically updates the referenced equipment ports – no need for separate update to mark equipment ports as used etc.

Entire record from external source can be supplied again without external processing needs – MNI works it out

- No need to supply the incremental/delta difference
- No need to delete MNI data before supplying data again

As time passes (now), older predicted data that is earlier than the last historical data is removed and new predictions performed. Boundaries for earliest preserved historical data and latest predicted data are configurable ensuring quantifiable storage requirements (days, months, quarters or years)

## 8.2. Behind the Scenes

---

The data required for MNI must be expanded overtime during normal operations as MNI requires a historical dataset in order to predict the future changes, this requires certain adjustments to normal Create-Read-Update-Delete (CRUD) concepts in addition to all non-administrative datasets including metadata fields:

1. `point` a timestamp (specific point in time)
2. `source` to indicate if either supplied as actual (`historical`), supplied as proposed (`planned`), or generated (`predicted`) internally via machine learning predictive algorithms
3. `delete` to indicate if marked for deletion

All records, regardless of type, are subject to the administrative data pruning mechanisms when the historical retention period ( HTTP GET `/mni/v1/admin/data/historical/duration` ) of the record has been exceeded

## 8.3. Shadow Time-Series Tables

---

Each data (parent) table has a corresponding shadow (child) table prefixed by `_`, for example Alerts are held in the database table alert and its shadow table is `_alert`.

The parent table only consists of two fields:

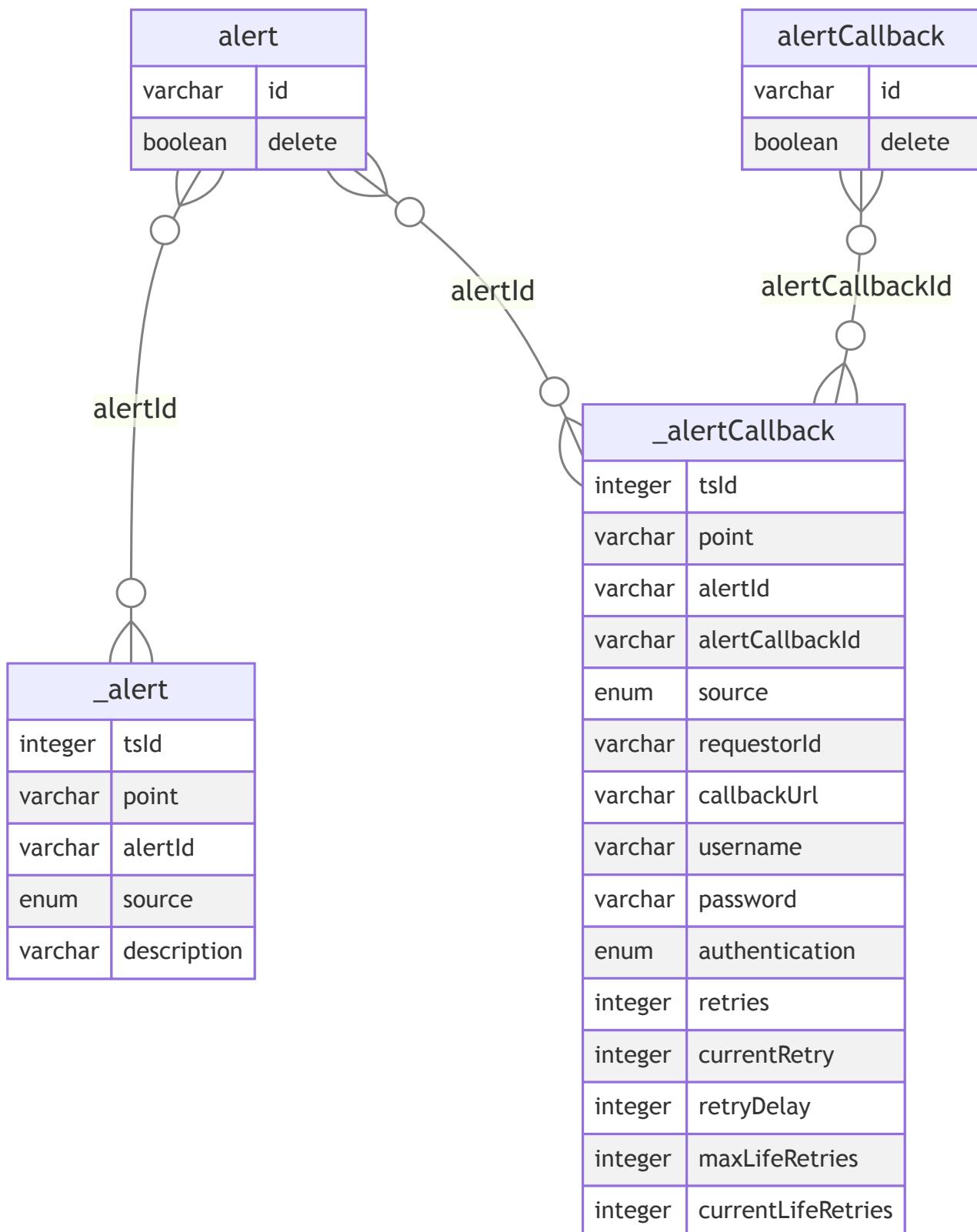
- Unique Identifier (`UUIDv4`) as the primary key
- `delete`

The child table consist of at least three fields before the real fields are included:

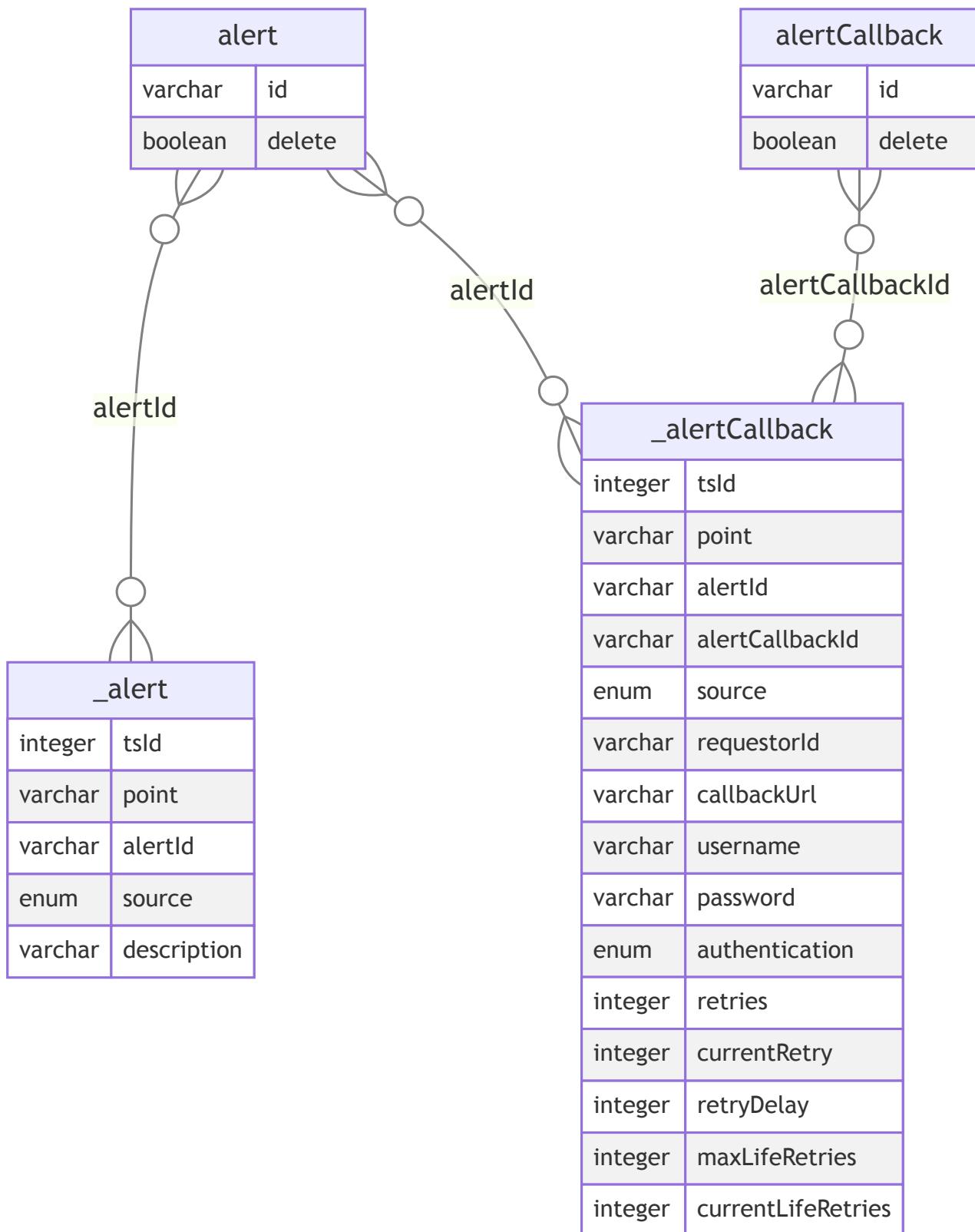
- `tsID` an auto-populated integer based sequential Identifier (sequence) as the primary key
- `point` to hold the timestamp
- `source`

- Additional identifier (`UUIDv4`) as the referential key(s) value to link back to the parent table(s) primary key

In entity relationship terms, this looks like:



Any other table that requires to have referential integrity will include the necessary identifier in the child table but referencing the parent table. This allows all held data to be held as time-based versions which reflects the creation, modification and deleting that occurs over normal operation time. In entity relationship terms, this looks like:



## 8.3.1. Bulk Create (Import)

- dataset presented via API will include point values
- dataset presented via API will optionally include internal identifiers (`UUIDv4`)
- dataset (JSON array) will be unpackaged into individual records
- each bulk (JSON array) import will have a maximum size of 1000 records

### Note:

- it is the responsibility of the client to control the quantity of records sent
- MNI API Server will return HTTP status code 413 if the internal queues are currently full or the entire request payload will be discarded if it exceeds the maximum size
- If the request is invalid, HTTP status code 400 will be returned
- For non-administrative datasets:
  - When record is stored, the metadata fields are automatically populated with:
    - `point` - extracted from supplied dataset record
    - `source` - extracted from supplied dataset record
    - `delete` - extracted from supplied dataset record
- For all datasets:
  - the internal identifiers (`UUIDv4`), created if not supplied or used if supplied
- HTTP status code 202 will be returned

### Note:

- during bulk imports MNI Alert, Fetch and Predict services are effectively suspended until the API Server queues are empty

## 8.4. Bulk Read (Export)

- if no such record(s) exists, HTTP status code 404 will be returned
- If the request is invalid, HTTP status code 400 will be returned
- All held records value will be packaged as dataset (JSON array) including all internally held identifiers (`UUIDv4`) and for non-administrative datasets the metadata fields will be included:
  - point field value (date format in compressed ISO-8601, YYYYMMDDThhmmss)
  - source field value (string enum)
  - delete field value (boolean)
- HTTP status code 200 will be returned

### 8.4.1. Create

- record presented via API will NOT include point value
- If the request is invalid, HTTP status code 400 will be returned

- When record is stored, two metadata fields are automatically populated with:
  - `point` = current system date/time
  - `source` = historical
  - `delete` = false
- the internal identifier (`UUIDv4`) will be returned along with HTTP status code 200

## 8.4.2. Read

- if no such record(s) exists, HTTP status code 404 will be returned
- If the request is invalid, HTTP status code 400 will be returned
- if no point value or range is requested via API query parameters then the latest historical record(s) (i.e. state = historical) will be returned with HTTP status code 200
- If point value or is requested via API query parameter that extends into the future, record(s) with either a source of historical or predicted and not marked for deletion (i.e. `delete` = false) will be returned with HTTP status code 200

## 8.4.3. Update

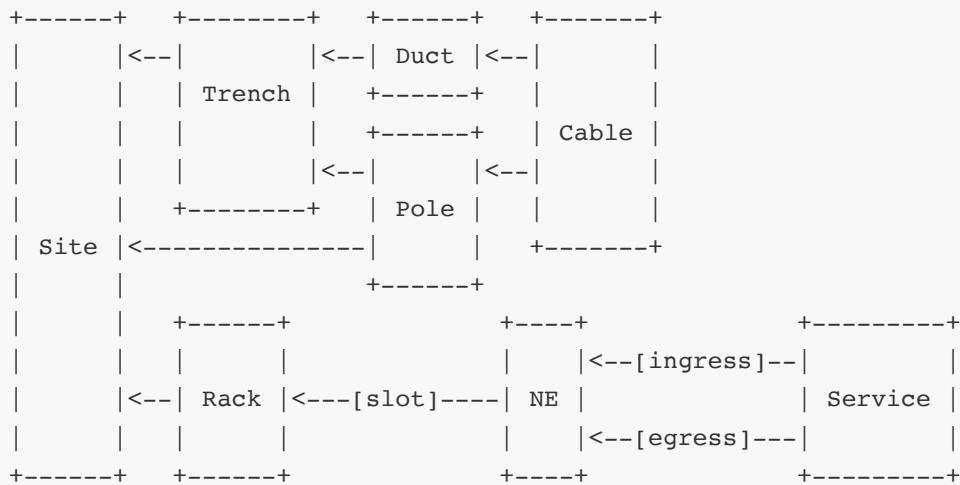
- If specified record does not exist then HTTP status code 404 will be returned
- If specified record exists but is marked for deletion (`delete` = true) then HTTP status code 404 will be returned
- If the request is invalid, HTTP status code 400 will be returned
- The existing record is **NOT** updated or altered in any fashion
- If a partial update (HTTP PATCH) then:
  - the existing record is read, the update applied but the resulting record stored as a new record with the metadata fields automatically populated with:
    - `point` = current system date/time
    - `source` = historical
    - `delete` = false
  - HTTP status code 204 will be returned
- If a full update (HTTP PUT) then:
  - the supplied record stored as a new record with the metadata fields automatically populated as:
    - `point` = current system date/time
    - `source` = historical
    - `delete` = false
  - HTTP status code 204 will be returned

## 8.4.4. Delete

- If specified record does not exist then HTTP status code 404 will be returned

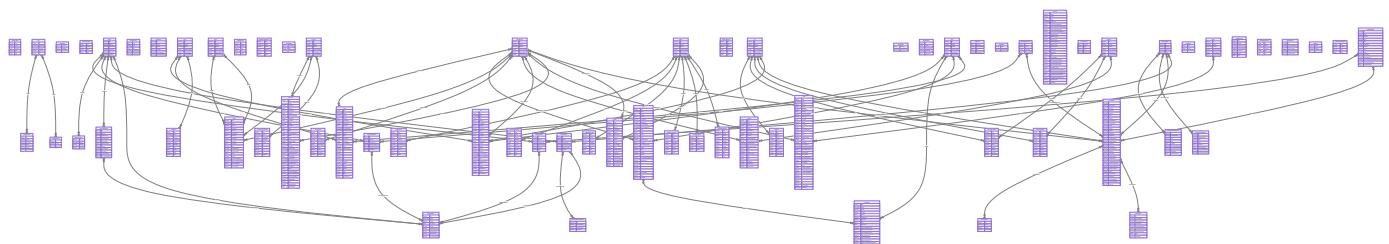
- If the request is invalid, HTTP status code 400 will be returned
- The existing record is **NOT** deleted but the existing record is read, but the resulting record stored as a new record with the two metadata fields automatically populated with:
  - `point` = current system date/time
  - `source` - as per existing record value
  - `delete` = true
- HTTP status code 204 will be returned

## 9. Resource Related Dependencies



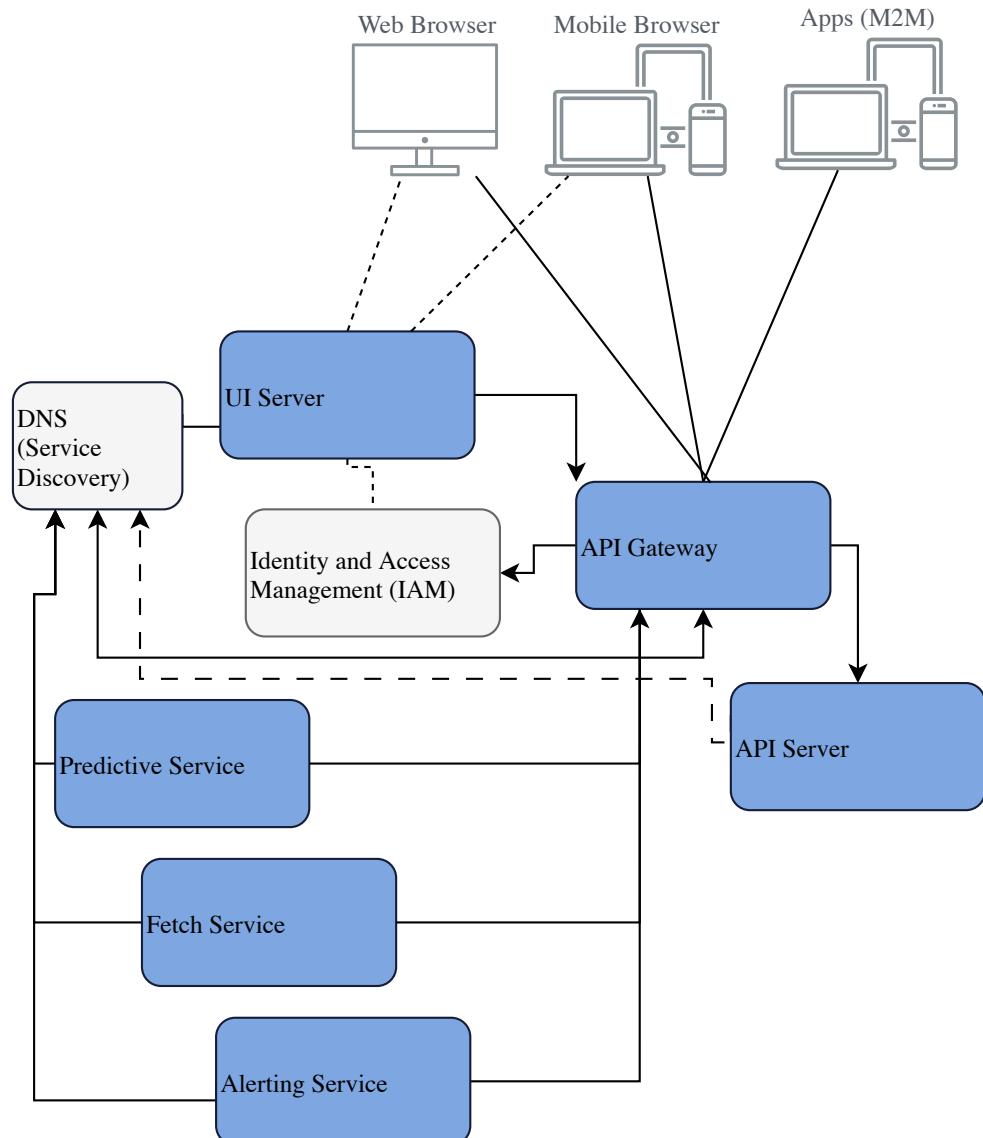
## 10. Database Schema

As part of the CI/CD process, the MNI database schema is automatically documented as a Entity Relationship Diagram (ERD) and the latest version is shown below



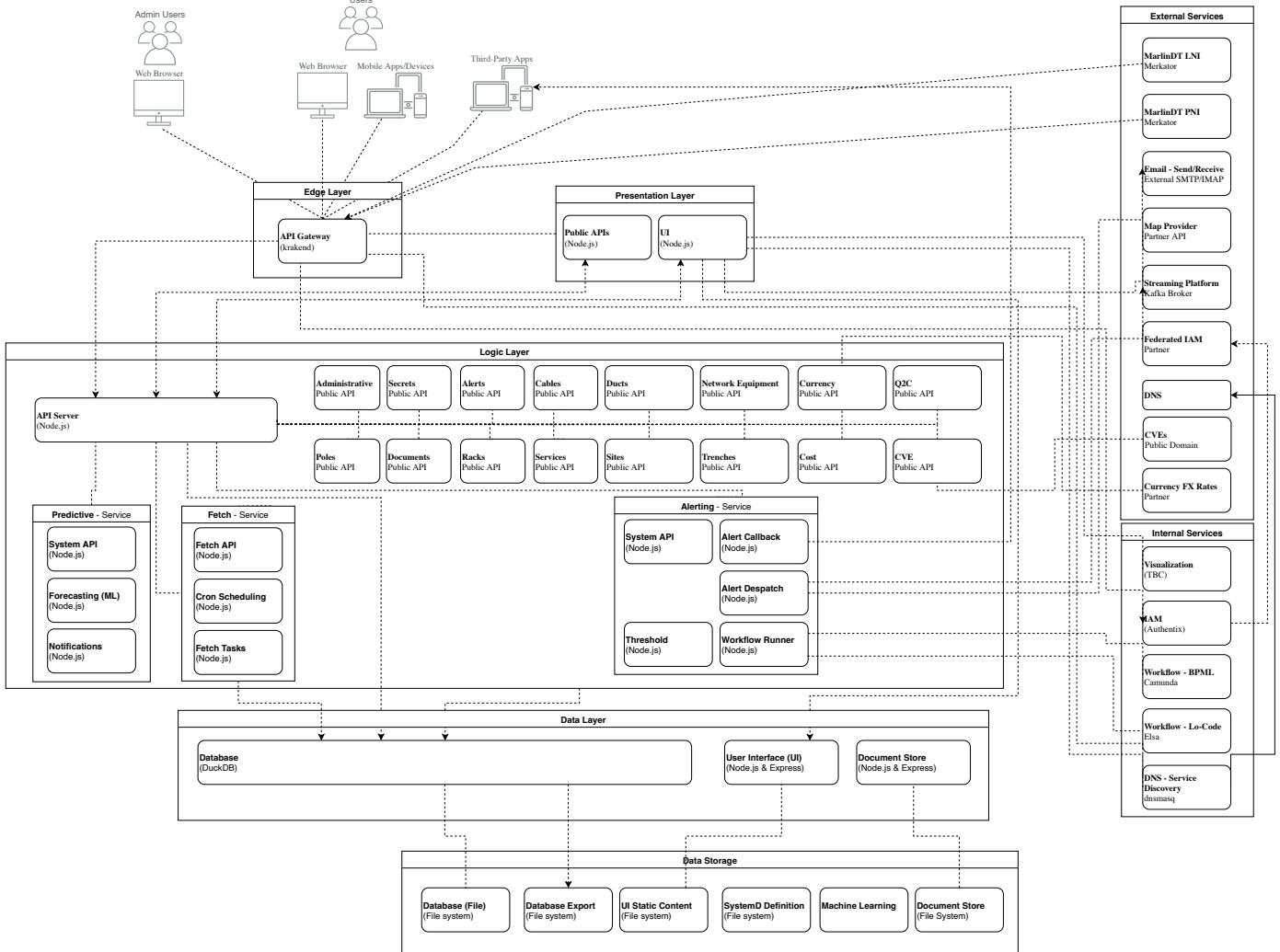
Schema Model

## 11. Architectures



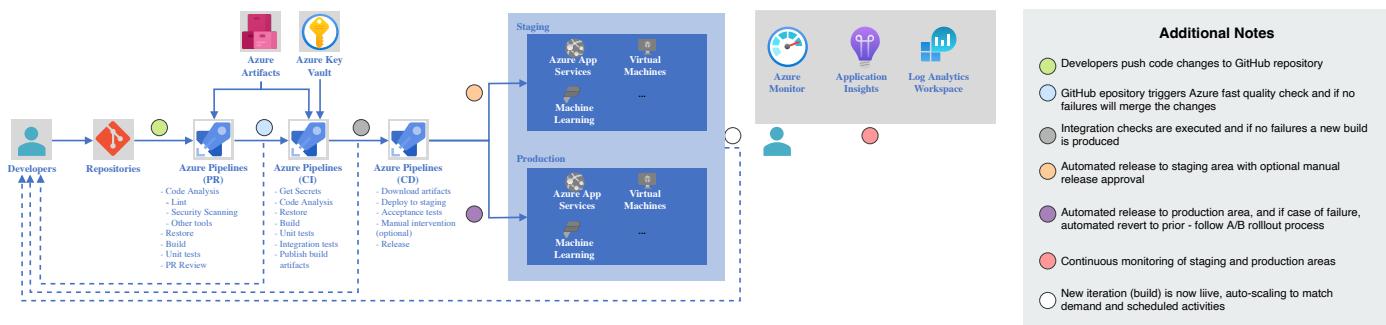
Simplified

### 11.0.1. Application



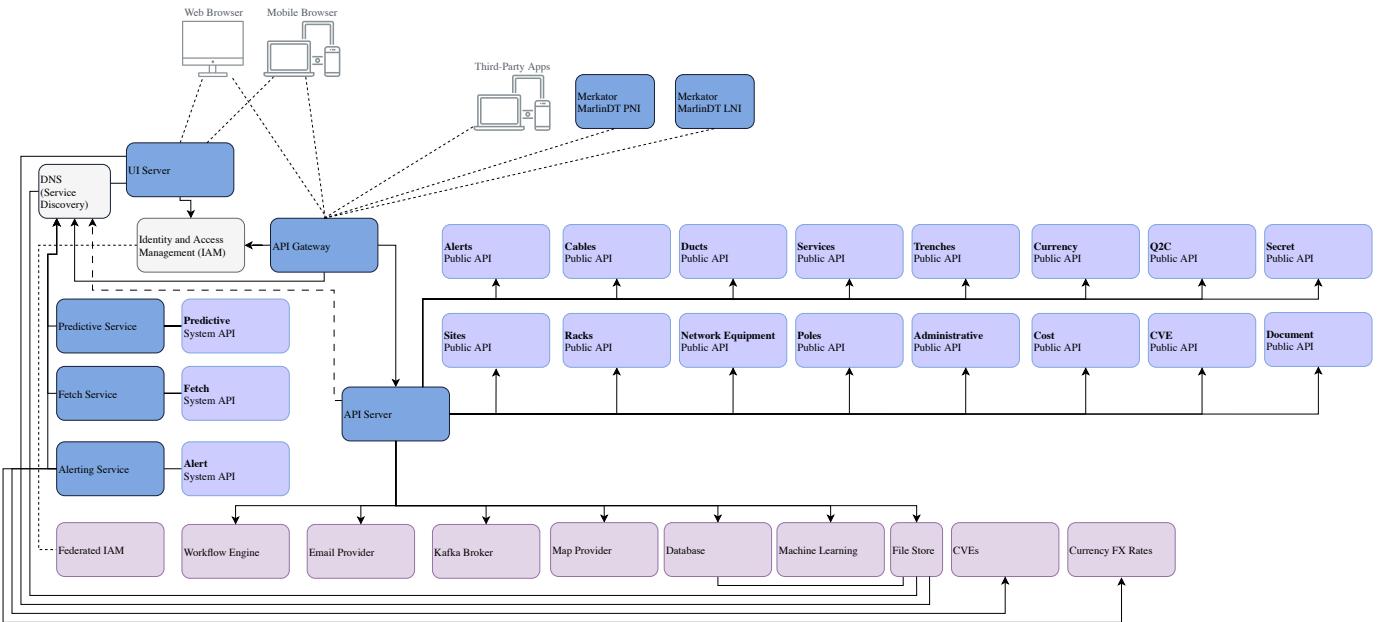
## Application

### 11.0.2. DevOps CI/CD Pipeline



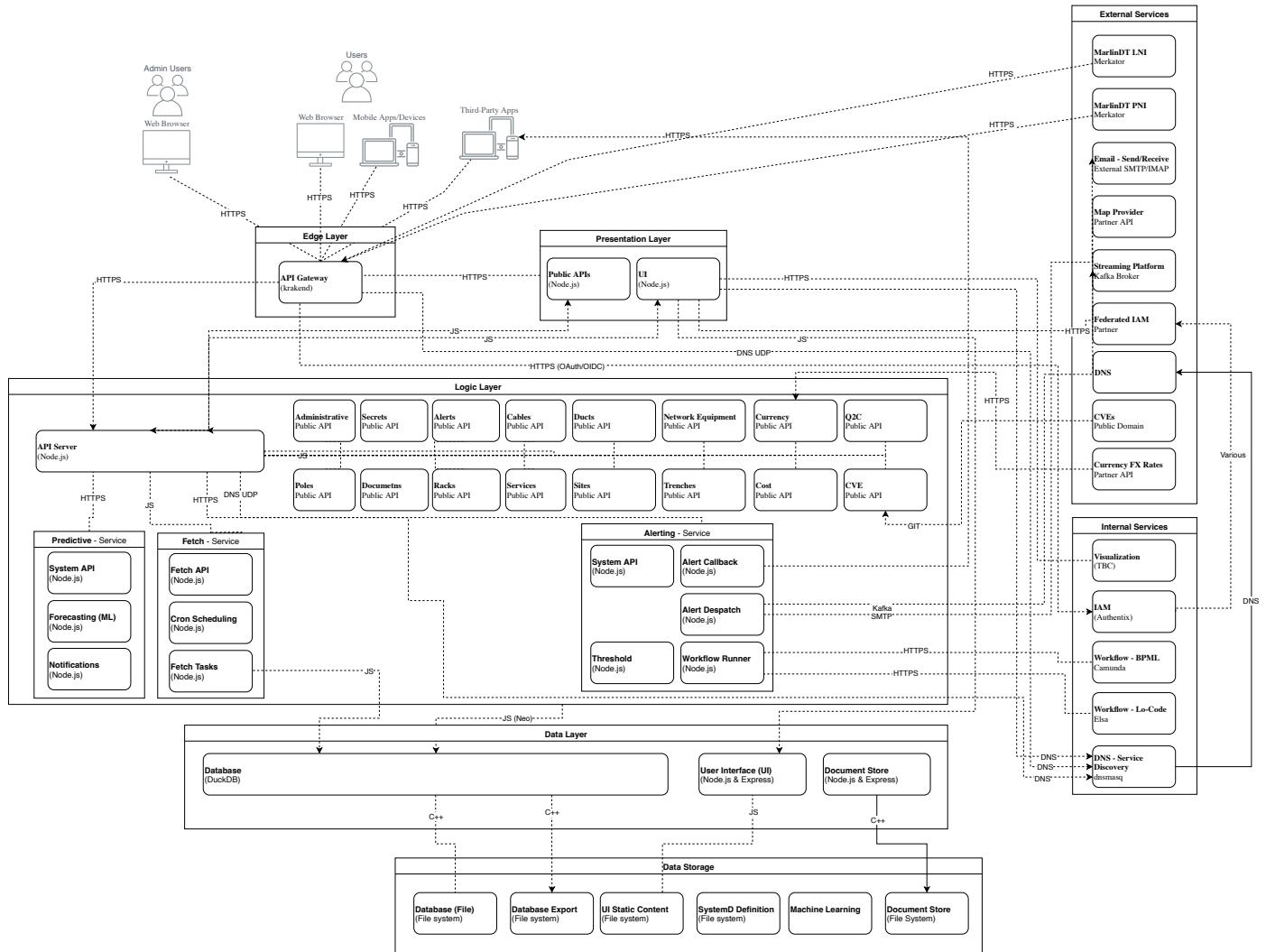
## DevOps

### 11.0.3. Functional



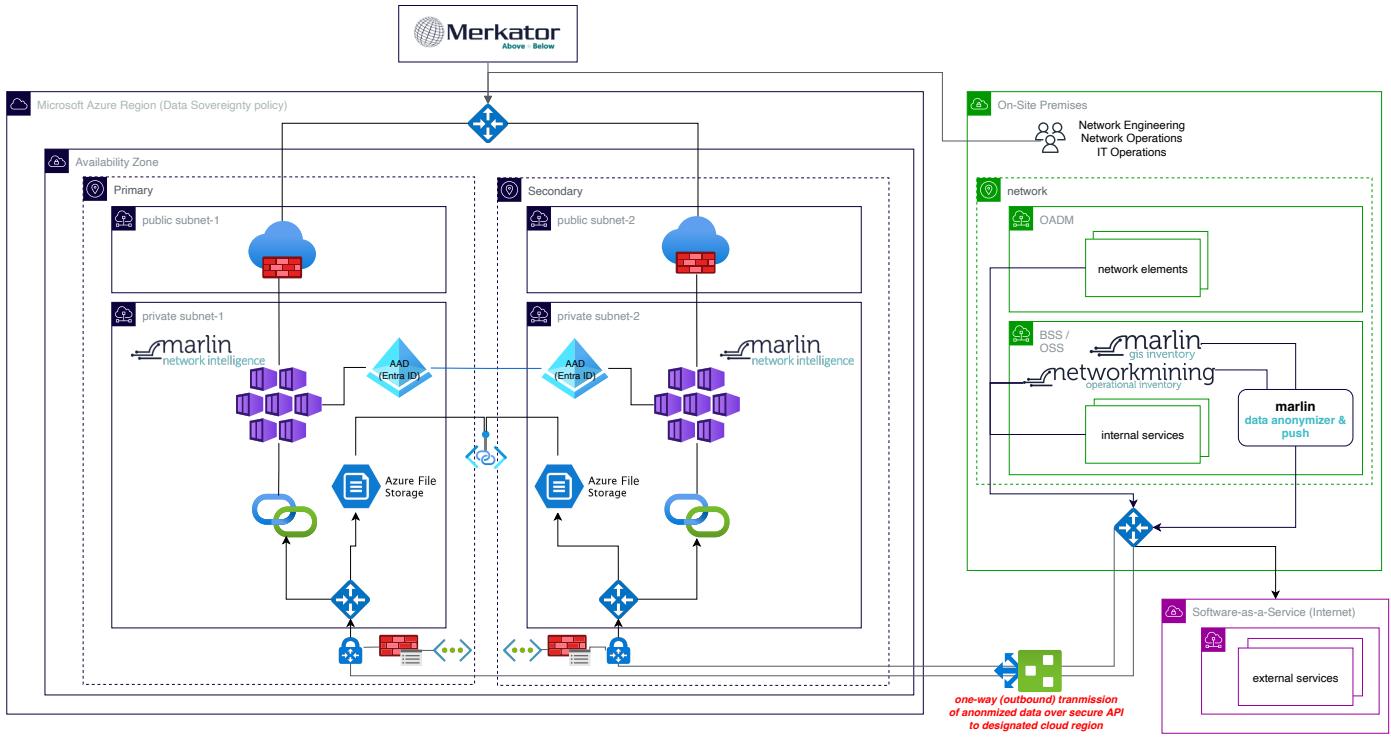
Functional

## 11.0.4. Integration



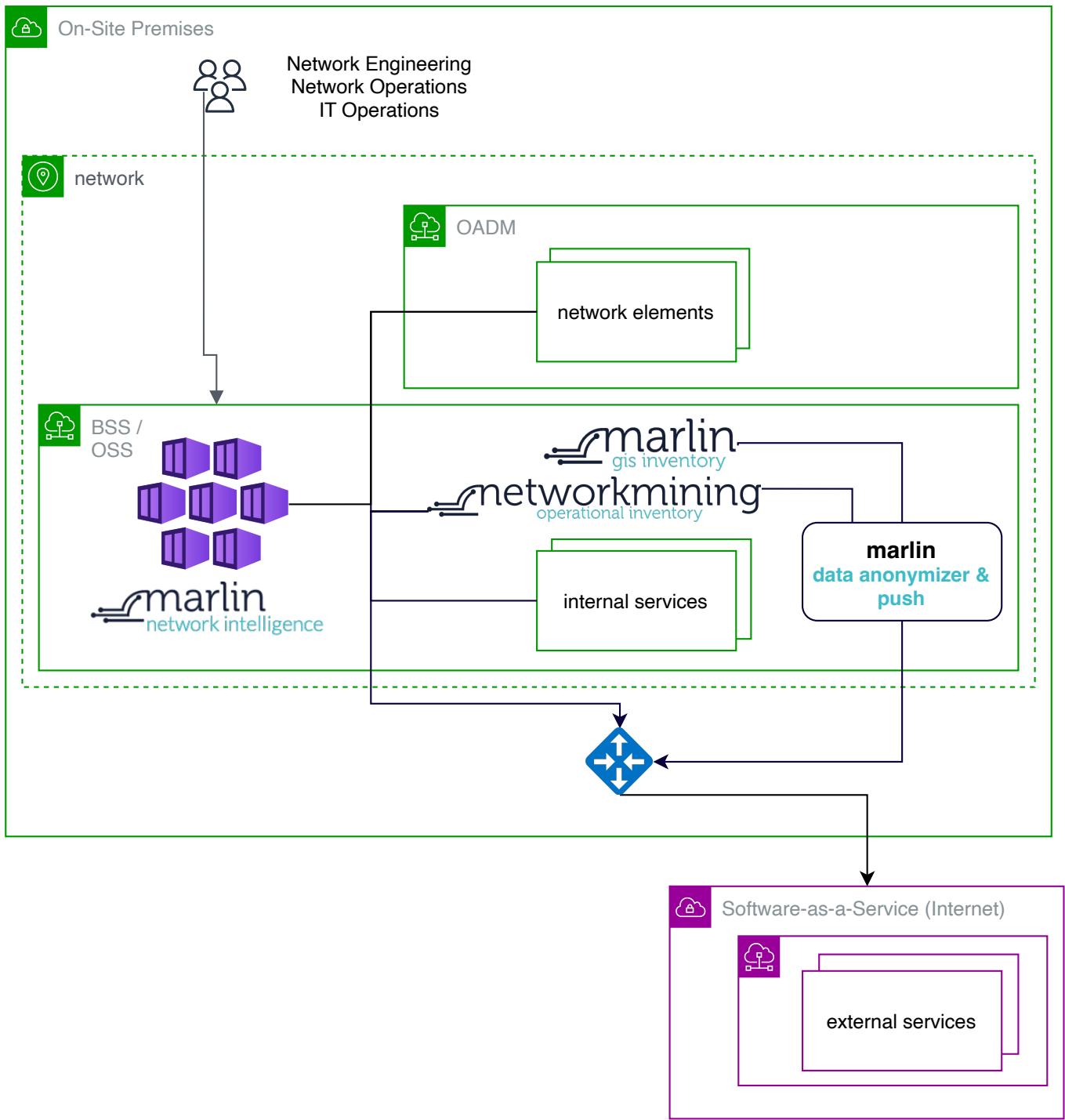
Integration

## 11.0.5. Logical (Cloud)



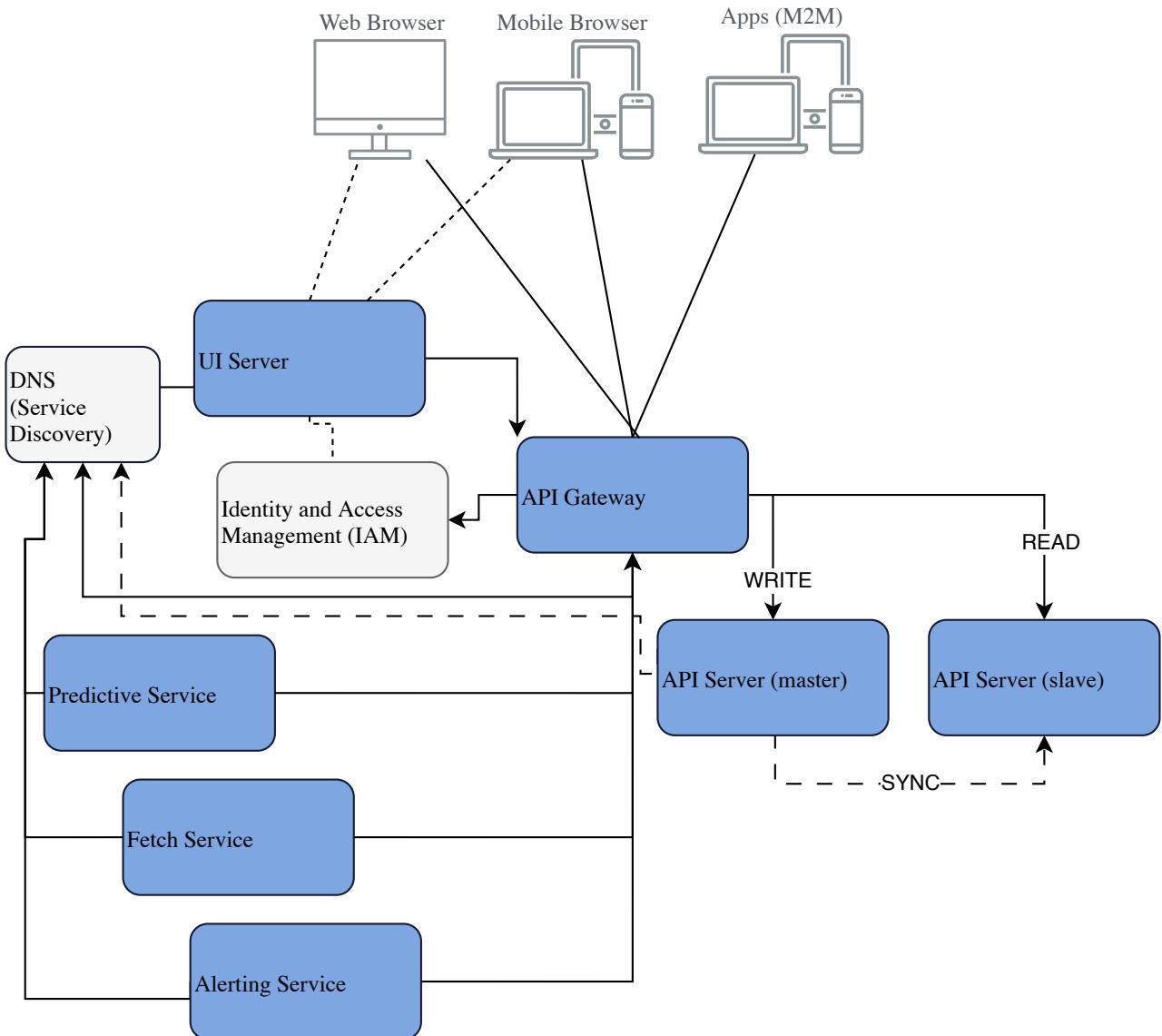
Logical Cloud

## 11.0.6. Logical (On-Premise)



Logical On-Premise/Self Hosted

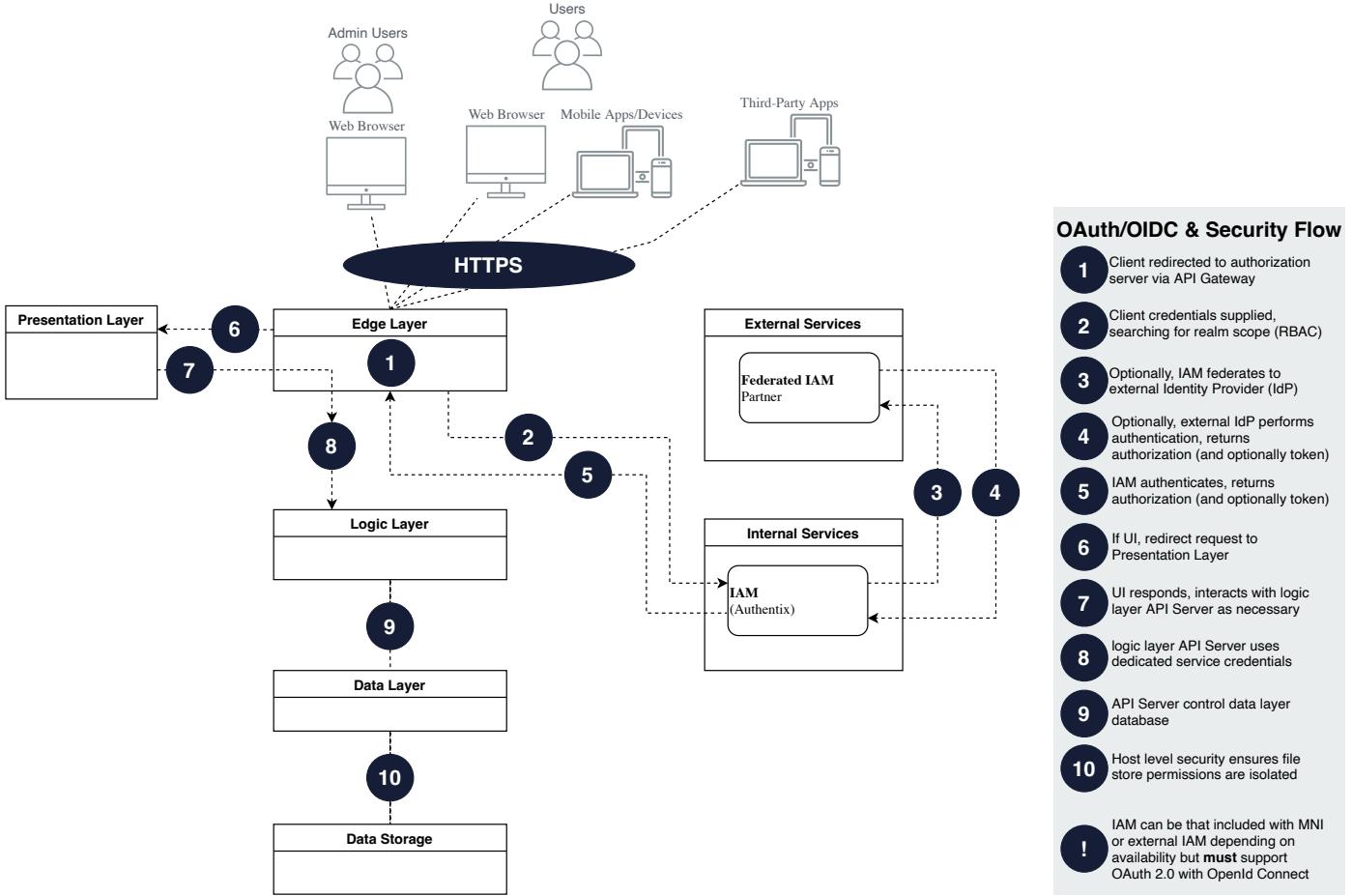
## 11.0.7. Cluster Replication



### API Server Clustering

- API Server embeds DuckDB. Due to horizontal scaling limitations of DuckDB, backend database replication needs is used
- MNI already tracks internal changes for prediction purposes (Predict Service)
- Resource changes pulled from master API Server by slave API Server(s) and updated locally
- API Gateway, through existing DNS Service Discovery, sends writes to master and reads to slave(s) with failover back to master for scaling purposes
- Replication communication via the encrypted multicast stream **does not** occur through the API Gateway
- UI Server, API Gateway, Predictive Service, Fetch Service and Alerting Service are already fully capable of horizontal and vertical scaling

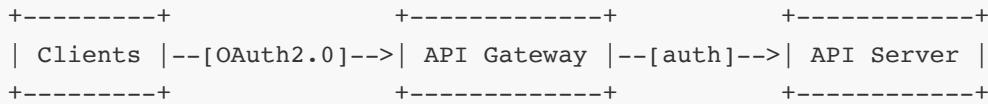
## 11.0.8. Security



## Authentication and Authorization

### 11.0.8.1. Through the API Gateway

Browser clients and machine-to-machine (M2M) clients operate through the API Gateway and are required to use OAuth2.0/OpenID Connect authentication, between the API Gateway and the API Server a different authentication scheme is used.



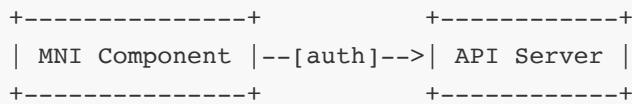
The API Gateway has the responsibility of undertaking both authentication and authorization (fine-grain) for Role-Based Access Control (RBAC).

#### Note:

- MNI contains the ability to disable and ignore RBAC based authorization, however authentication can not be disabled.
- To ignore authorization change `APIGW_IGNORE_AUTH_VALIDATOR` to `true` prior to installation of the API Gateway.
- Authorization can be re-enabled by changing `APIGW_IGNORE_AUTH_VALIDATOR` to `false` and performing an update or upgrade of the API Gateway

### 11.0.9. Internally

MNI Alert Service, Fetch Service and Predict Service, as well as the API Gateway use service authentication.



**Note:**

- MNI Alert Service, Fetch Service and Predict Service can be configured to route through the API Gateway, but will need to authenticate with the same IAM
- The service authentication credentials are generated during installation of the API Server and are specific to that installation.
- When deploying API Server for cluster operations, the service authentication credentials **MUST** match across all environment configuration files (`/etc/mni/mni.ini`), specifically `MNI_SERVICE_USERNAME` and `MNI_SERVICE_KEY`.

### 11.0.9.1. API Server Encrypted Storage

MNI API Server makes use of AES encryption, key size 256 bits for storing credentials within its secret store.

**Note:**

- The AES encryptions credentials are generated during installation of the API Server and are specific to that installation.
- When deploying API Server for cluster operations, the AES encryptions credentials **DO NOT NEED** to match across all environment configuration files (`/etc/mni/mni.ini`), specifically `APISERV_ENCRYPTION_KEY` and `APISERV_ENCRYPTION_IV`.

### 11.0.9.2. Role-Based Access Control (RBAC)

The MNI OpenAPIs already includes the fine-grain RBAC markers against each and every API, they are:

- `read:mni_admin`
- `read:mni_cable`
- `read:mni_duct`
- `read:mni_ne`
- `read:mni_pole`
- `read:mni_rack`
- `read:mni_service`
- `read:mni_site`
- `read:mni_trench`
- `read:mni_ui`
- `write:mni_admin`
- `write:mni_cable`
- `write:mni_duct`

- `write:mni_ne`
- `write:mni_pole`
- `write:mni_rack`
- `write:mni_service`
- `write:mni_site`
- `write:mni_trench`
- `write:mni_ui`

MNI OpenAPIs include the use of multiple roles, but the client is only expected to require one - this allows multiple job or function roles (planning/design, procurement etc., API integration) to be supported.

#### Note:

- The mapping of job or function roles and roles to user credentials is handled by the Identity and Access Management (IAM) and is beyond the scope of this document.

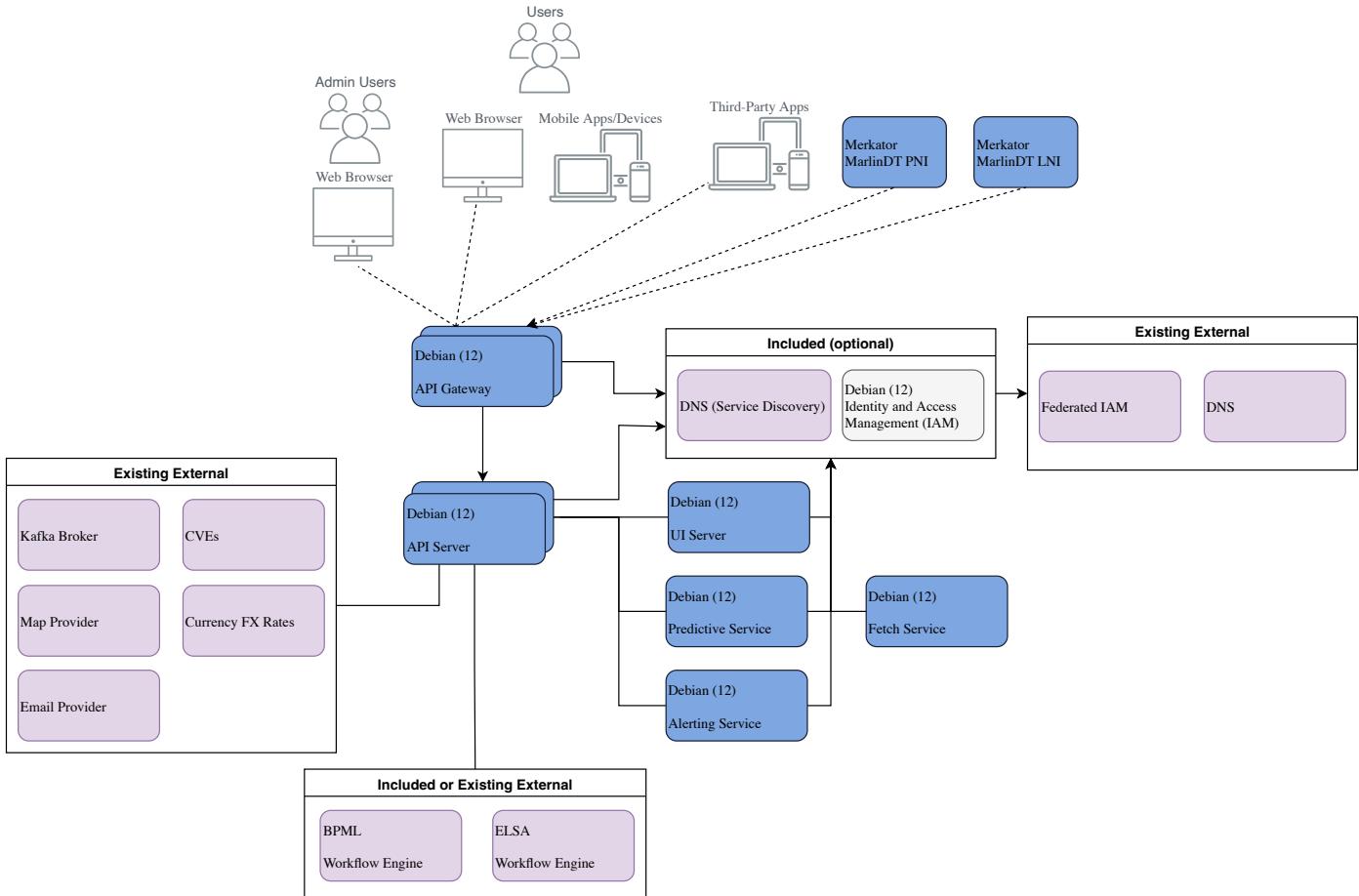
The following roles are reserved for internal use by MNI Alert Service, Fetch Service and Predict Service if they are required to operate through the API Gateway:

- `read:mni_alert`
- `read:mni_fetch`
- `read:mni_predict`
- `write:mni_alert`
- `write:mni_fetch`
- `write:mni_predict`

#### Note:

- `mni_alert`, `mni_fetch`, `mni_predict` roles are for MNI itself and **MUST NOT** be assigned to any external clients

## 11.0.10. Deployment



Suitable for standalone integration, lab, proof of concepts and low-yield production usage or clustered deployments.

#### Note:

- all Debian hosts shown can be combined into single host, either bare-metal or virtual

## 12. DNS Resolution

MNI makes use of service discovery via DNS server (`SRV`) records to locate MNI components, this is more commonly referred to as `SD-DNS`. This is used for both horizontal and vertical scaling and by the API Gateway for backend routing.

```
Type
Domain Record
  Data  TTL
A apiserver.{hostname}.{domain} 127.0.0.1 3600
A {hostname}.{domain} 192.168.1.16 3600
SRV _https._tcp.apiserver.{hostname}.{domain} 0 10 7443 apiserver.{hostname}.{domain} 3600
SRV _https._tcp.gateway.{hostname}.{domain} 0 10 8443 {hostname}.{domain} 3600
```

Change `{hostname}.{domain}` to whatever the host reported (or was told) its fully qualified hostname was before defining the DNS records. You can change the time-to-live (`TTL`) to a one day (`86400`) when you are happy DNS resolution is working.

You can verify via as follows:

```
$ dig +short _https._tcp.apiserver.{hostname}.{domain} SRV  
$ dig +short apiserver.{hostname}.{domain} A  
$ dig +short _https._tcp.gateway.{hostname}.{domain} SRV  
$ dig +short gateway.{hostname}.{domain} A
```

#### Note:

- The utility `dig` is part of the `bind9-dnsutils` package on Debian hosts.

Depending on the deployment architecture, single host vs. multiple hosts, the hostname may vary. For example, you should get something like this if all working correctly on a combined single host:

```
# hostname  
marlinintegration.merkator.com  
# cat /etc/hosts  
127.0.0.1      localhost  
127.0.1.1      MarlinMigration  
  
# The following lines are desirable for IPv6 capable hosts  
::1      localhost ip6-localhost ip6-loopback  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters  
  
$ dig +short _https._tcp.apiserver.marlinintegration.merkator.com SRV  
0 10 7443 apiserver.marlinintegration.merkator.com.  
$ dig +short apiserver.marlinintegration.merkator.com A  
127.0.0.1  
$ dig +short _https._tcp.gateway.marlinintegration.merkator.com SRV  
0 10 8443 marlinintegration.merkator.com.  
$ dig +short marlinintegration.merkator.com A  
192.168.1.16
```

For Round-Robin DNS, the DNS Server would need something similar to the zone extract shown below:

```
marlinintegration IN A 192.168.1.16  
                  IN A 192.168.1.17  
                  IN A 192.168.1.18
```

#### Note:

- When using Round-Robin DNS, the DNS `SRV` records do not need any changes as the contained `A` host record itself for `marlinintegration.merkator.com` will resolve to the listed IP addresses
- Most modern Web Browsers support Round-Robin DNS, however any applications operating in a machine-to-machine fashion may require additional configuration or customisation
- It is outside the scope of this document for configuration of DNS Servers

# 13. MNI Host Configuration

---

Each host environment hosting MNI components must contain a MNI configuration environment file. This file can be found at `/etc/mni/mni.ini` and is automatically created if not found during installation.

**Note:**

- The contents of the configuration environment file should be manually updated and synchronised between all hosts that form MNI as MNI internal components will read some of the configuration of the other components.

## 13.1. MNI Wildcard SSL Certificates

---

MNI requires either a wildcard SSL certificate, or suitable SSL certificate per host as below:

- API Gateway (`APIGW_*`)
- API Server (`APISERV_*`)
- UI Server (`UISERV_*`)
- Identity and Access Management (`IAM_*`)

## 13.2. MNI Environment File

---

The file `/etc/mni/mni.ini` deployment environment file supports the following MNI functional components:

- Alert Service (`ALERTSRV_*`)
- API Gateway (`APIGW_*`)
- API Server (`APISERV_*`)
- Fetch Service (`FETCHSRV_*`)
- UI Server (`UISERV_*`)
- Identity and Access Management (`IAM_*`)
- Local DNS Server (`DNSSERV_*`)

**Note:**

- Some of the configuration items are automatically determined during installation, these include hostname/address and service credentials.
- Some items are automatically set during the CI/CD build process
- This deployment environment file must be manually synchronised if using more than one host

### 13.2.1. Common

```

#
# Common
#
CONFIG_DIRECTORY="/etc/mni"
HOST_SERVICE_GROUP="mni"
MNI_NAME="MarlinDT Network Intelligence"
MNI_BUILD="1763499309"
MNI_VERSION="1.0.0"
MNI_LICENSE_PRIVATE_KEY=""
MNI_LICENSE_PUBLIC_KEY=""
MNI_DEFAULT_COUNTRY_CODE="BEL"
MNI_SERVICE_USERNAME=""
MNI_SERVICE_KEY=""

```

- `CONFIG_DIRECTORY` value should be left as is
- `HOST_SERVICE_GROUP` can be changed
- `MNI_NAME` can be changed
- `MNI_BUILD` is automatically populated during CI/CD build process
- `MNI_VERSION` is automatically populated during CI/CD build process
- `MNI_LICENSE_PRIVATE_KEY` value should be generated via the MNI License Generator tool and the resulting value included here for the API Server - no other component requires this
- `MNI_LICENSE_PUBLIC_KEY` is automatically populated during CI/CD build process and should **not** be changed
- `MNI_DEFAULT_COUNTRY_CODE` can be changed to one of the following values: ABW, AFG, AGO, AIA, ALB, AND, ARE, ARG, ARM, ASM, ATA, ATF, ATG, AUS, AUT, AZE, BDI, BEL, BEN, BES, BFA, BGD, BGR, BHR, BHS, BIH, BLM, BLR, BLZ, BMU, BOL, BRA, BRB, BRN, BTN, BWA, CAF, CAN, CHE, CHL, CHN, CIV, CMR, COD, COG, COK, COL, COM, CPV, CRI, CUB, CUW, CYM, CYP, CZE, DEU, DJI, DMA, DNK, DOM, DZA, ECU, EGY, ERI, ESP, EST, ETH, FIN, FJI, FLK, FRA, FRO, FSM, GAB, GBR, GEO, GGY, GHA, GIB, GIN, GLP, GMB, GNB, GNQ, GRC, GRD, GRL, GTM, GUF, GUM, GUY, HND, HRV, HTI, HUN, IDN, IMN, IND, IRL, IRN, IRQ, ISL, ISR, ITA, JAM, JOR, JPN, KAZ, KEN, KGZ, KHM, KIR, KNA, KOR, KWT, LAO, LBN, LBR, LBY, LCA, LIE, LKA, LSO, LTU, LUX, LVA, MAR, MCO, MDA, MDG, MDV, MEX, MHL, MKD, MLT, MMR, MNE, MNG, MNP, MOZ, MRT, MSR, MTQ, MUS, MWI, MYS, MYT, NAM, NCL, NER, NGA, NIC, NIU, NLD, NOR, NPL, NRU, NZL, OMN, PAK, PAN, PCN, PER, PHL, PLW, PNG, POL, PRK, PRT, PRY, PSE, PYF, QAT, REU, ROU, RUS, RWA, SAU, SDN, SEN, SGP, SHN, SLB, SLE, SLV, SMR, SOM, SRB, SSD, STP, SUR, SVK, SVN, SWE, SWZ, SYC, SYR, TCA, TCD, TGO, THA, TJK, TKL, TKM, TLS, TON, TTO, TUN, TUR, TUV, TWN, TZA, UGA, UKR, URY, USA, UZB, VAT, VCT, VEN, VGB, VIR, VNM, VUT, WLF, WSM, YEM, ZAF, ZMB, ZWE
- `MNI_SERVICE_USERNAME` is automatically set during the installation of the API Server - other components do require this
- `MNI_SERVICE_KEY` is automatically set during the installation of the API Server - other components do require this

## 13.2.2. OAuth2.0 / OpenID Connect

```

#
# OAuth2.0 / OpenID Connect
#
#OAUTH=true
OAUTH_CLIENT_ID="MNI"
OAUTH_CLIENT_SECRET="mQGuleEcwodREynXpDqBbt1Xe4yzikN"
OAUTH_DISCOVERY_URL="https://mni.merkator.local:6443/realm/MNI/.well-known/openid-configuration"

```

- `OAUTH` is required to have a value of `true`
- `OAUTH_CLIENT_ID` default value matches that used by the included IAM Server
- `OAUTH_CLIENT_SECRET` default value matches that used by the included IAM Server
- `OAUTH_DISCOVERY_URL` default value should be automatically changed during installation, but it is recommended this value is manually verified

If you were to use Microsoft Azure Active Directory for authentication, then the values would be something similar to:

```

OAUTH=true
OAUTH_CLIENT_ID="MNf3ce2c1d-abcd-41ee-cowd-reydd5489d741I"
OAUTH_CLIENT_SECRET="Mni8Q~RWkJ3xDeadLuckOC5a6k11G_CUfoqJGdn."
OAUTH_DISCOVERY_URL="https://login.microsoftonline.com/a17ec843-b999-41ee-cowd-rey03a903e14/v2.0/.well-known/openid-configuration"

```

#### Note:

- You would need to change the Microsoft tenant id `a17ec843-b999-41ee-cowd-rey03a903e14` and potentially Discovery URL to match your Azure account.

### 13.2.3. Alert Service

```

#
# alertService
#
ALERTSRV_CRONTIME="0 0 * * *"
ALERTSRV_DEBUG=true
ALERTSRV_ENDPOINT_KEEPALIVE_INTERVAL_MS=120000
ALERTSRV_ENDPOINT_RETRY_INTERVAL_MS=15000
ALERTSRV_HOST_SERVICE_LOG_FILE="/var/log/mni/alert.log"
ALERTSRV_HOST_SERVICE_SYSTEMD="mnialert.service"
ALERTSRV_HOST_SERVICE_USERNAME="mnialert"
ALERTSRV_TLS_INSECURE_CONNECTIONS=true
ALERTSRV_WORKING_DIRECTORY="/usr/local/mni/alert"

```

- `ALERTSRV_CRONTIME` can be changed, adhere to rules of Linux CronTab for specifying the execution time
- `ALERTSRV_ENDPOINT_KEEPALIVE_INTERVAL_MS` can be changed if required

- `ALERTSRV_ENDPOINT_RETRY_INTERVAL_MS` can be changed if required

### 13.2.4. API Gateway

```

#
# apiGateway
#
APIGW_ADDRESS=
APIGW_HOST_SERVICE_LOG_FILE="/var/log/mni/apigw.log"
APIGW_HOST_SERVICE_SYSTEMD="mniapigw.service"
APIGW_HOST_SERVICE_USERNAME="mniapigw"
APIGW_IGNORE_AUTH_VALIDATOR=true
APIGW_PORT=8443
APIGW_PROXY_RATE_LIMIT_REQUESTS=32767
APIGW_PROXY_CAPACITY_REQUESTS=32767
APIGW_PROXY_RATE_LIMIT_EVERY="1m"
APIGW_TLS_INSECURE_CONNECTIONS=true
APIGW_SSL_CERT="apigw.crt"
APIGW_SSL_CSR="apigw.csr"
APIGW_SSL_DAYS=90
APIGW_SSL_KEY="apigw.key"
APIGW_SSL_SIZE=4096
APIGW_KRAKEND_VERSION="2.12.0"
APIGW_WORKING_DIRECTORY="/usr/local/mni/apigw"

```

- `APIGW_ADDRESS` is automatically set to the Linux network interface providing the default route
- `APIGW_IGNORE_AUTH_VALIDATOR` is set to `true` the Role Based Access Controls (RBAC) used for authorization are disabled and only authentication is used instead
- `APIGW_PORT` if changed, ensure DNS records match
- `APIGW_PROXY_RATE_LIMIT_REQUESTS` for large installations, adjust accordingly - refer to KrakenD documentation
- `APIGW_PROXY_CAPACITY_REQUESTS` for large installations, adjust accordingly - refer to KrakenD documentation
- `APIGW_PROXY_RATE_LIMIT_EVERY` for large installations, adjust accordingly - refer to KrakenD documentation
- `APIGW_KRAKEND_VERSION` which version of KrakenD to download and use - this should not be changed

### 13.2.5. API Server

```

#
# apiServer
#
APISERV_ADDRESS="0.0.0.0"
APISERV_API_DIRECTORY="/usr/local/mni/api/openapi"
APISERV_BACKUP_DIRECTORY="/usr/local/mni/api/backup"
APISERV_DEBUG=true

```

```

APISERV_DUCKDB_VERSION="v1.4.1"
APISERV_DUCKDB_BACKUP_CRONTIME="0 2 * * *"
APISERV_DUCKDB_BACKUP=true
APISERV_DUCKDB_FILE="/usr/local/mnri/api/mnri.duckdb"
APISERV_DUCKDB_MAX_MEMORY="20GB"
APISERV_DUCKDB_THREADS=4
APISERV_ENCRYPTION_KEY="qBU4K9XBw2JtYx7puSGiSig8LbxxDpSaprK7+u+EJI8="
APISERV_ENCRYPTION_IV="U5uKpGIE3cIXh5kuhFZXuQ=="
APISERV_KEEPALIVE=true
APISERV_HOST_SERVICE_LOG_FILE="/var/log/mnri/api.log"
APISERV_HOST_SERVICE_SYSTEMD="mnriapi.service"
APISERV_HOST_SERVICE_USERNAME="mnriapi"
APISERV_MULTICAST_ADDRESS="230.185.184.183"
APISERV_MULTICAST_PORT=54321
APISERV_PORT=7443
APISERV_PREMISES_PASSED_BOUNDARY_DISTANCE=35
APISERV_ROLE="standalone"
APISERV_SPATIAL_UPDATE_GEOMETRY_CRONTIME="*/5 * * * *"
APISERV_SSL_CERT="api.crt"
APISERV_SSL_CSR="api.csr"
APISERV_SSL_DAYS=90
APISERV_SSL_KEY="api.key"
APISERV_SSL_SIZE=4096
APISERV_TIMEOUT_KEEPALIVE=5000
APISERV_TIMEOUT_REQUEST=300000
APISERV_DOCUMENT_DIRECTORY="/usr/local/mnri/api/document"
APISERV_UPLOAD_DIRECTORY="/usr/local/mnri/api/upload"
APISERV_URL_PREFIX="/mnri"
APISERV_URL_VERSION="/v1"
APISERV_USE_DNS_SD=false
APISERV_WORKING_DIRECTORY="/usr/local/mnri/api"

```

- `APISERV_ADDRESS` if using single host, this should be loopback address `127.0.0.1`, otherwise use the IP address which is reachable from the API Gateway and other MNI components
- `APISERV_API_DIRECTORY` the directory on the host to hold the OpenAPI definition
- `APISERV_BACKUP_DIRECTORY` the directory to hold the database backup files
- `APISERV_DUCKDB_VERSION` which of DuckDB to download and use - this should not be changed
- `APISERV_DUCKDB_BACKUP_CRONTIME` can be changed, adhere to rules of Linux CronTab for specifying the execution time
- `APISERV_DUCKDB_BACKUP` toggle performing backups
- `APISERV_DUCKDB_FILE` database file representing the DuckDB database
- `APISERV_DUCKDB_MAX_MEMORY` maximum memory to use for caching
- `APISERV_DUCKDB_THREADS` maximum number threads to use
- `APISERV_ENCRYPTION_KEY` is automatically set during the installation of the API Server - no other component requires this
- `APISERV_ENCRYPTION_IV` is automatically set during the installation of the API Server - no other component requires this

- `APISERV_KEEPALIVE` use HTTP keep-alive for connections, either `true` or `false`
- `APISERV_MULTICAST_ADDRESS` multicast address to use replication for clustering of API Server
- `APISERV_MULTICAST_PORT` multicast port to use replication for clustering of API Server
- `APISERV_PORT` if changed, ensure DNS records match
- `APISERV_PREMISES_PASSED_BOUNDARY_DISTANCE` distance in metres between pairs of latitude (X), longitude (Y) coordinates to determine if counted towards nearest match
- `APISERV_ROLE` used to enable replication for the clustering of API Server, possible values are below. All replication occurs through encrypted multicast streams which notify other members of a change, which then request the change and process the change as if they were the master itself. For DNS purposes, the master should be the host which receives create, update or delete requests, and the slaves should handle majority of the read requests.
  - `standalone` single host with no replication, or
  - `master` primary lead host within an API Server cluster, **Note:** only one deployed host should be `master`
  - `slave` multiple hosts within an API Server cluster
- `APISERV_SPATIAL_UPDATE_GEOMETRY_CRONTIME` can be changed, adhere to rules of Linux CronTab for specifying the execution time
- `APISERV_TIMEOUT_KEEPALIVE` timeout in milliseconds for the HTTP keep-alive connections
- `APISERV_TIMEOUT_REQUEST` request lifetime in milliseconds for the HTTP keep-alive connections
- `APISERV_DOCUMENT_DIRECTORY` the directory to hold the previously uploaded documents - this can be an external network share or another filesystem as dependant on expected storage requirements
- `APISERV_UPLOAD_DIRECTORY` the directory to hold an uploaded document before it is moved to the document directory
- `APISERV_URL_PREFIX` the API Server endpoint URL path prefix - this should not be changed
- `APISERV_URL_VERSION` the API server endpoint URL path version prefix - this should not be changed
- `APISERV_USE_DNS_SD` toggle API Server using DNS Service Discovery to determine which host/port to expose its endpoint against - this should not be changed unless DNS `SRV` records have been previously verified (i.e. from API Gateway)

### 13.2.6. Fetch Service

```

#
# fetchService
#
FETCHSRV_CVE_SCAN=true
FETCHSRV_CVE_DIRECTORY="/usr/local/mni/cvelistV5"
FETCHSRV_DEBUG=true
FETCHSRV_ENDPOINT_KEEPALIVE_INTERVAL_MS=120000
FETCHSRV_ENDPOINT_RETRY_INTERVAL_MS=15000
FETCHSRV_FX_UPDATE=true
FETCHSRV_HOST_SERVICE_LOG_FILE="/var/log/mni/fetch.log"
FETCHSRV_HOST_SERVICE_SYSTEMD="mnifetch.service"
FETCHSRV_HOST_SERVICE_USERNAME="mnifetch"

```

```
FETCHSRV_TLS_INSECURE_CONNECTIONS=true  
FETCHSRV_WORKING_DIRECTORY="/usr/local/mni/fetch"
```

- `FETCHSRV_CVE_SCAN` toggle fetching on CVE repository from the Internet
- `FETCHSRV_CVE_DIRECTORY` if used, where to cache the CVE repository - this can be an external network share or another filesystem as dependant on expected storage requirements
- `FETCHSRV_FX_UPDATE` toggle updating foreign currency exchange rates from the Internet - if disabled FX rates can be updated through API Server APIs

### 13.2.7. User Interface (UI) Server

```
#  
# uiServer  
#  
UISERV_ADDRESS="192.168.1.16"  
UISERV_DIST_DIRECTORY="/usr/local/mni/ui/dist"  
UISERV_DEBUG=true  
UISERV_DNS_RESOLVE=300000  
UISERV_HOST_SERVICE_LOG_FILE="/var/log/mni/ui.log"  
UISERV_HOST_SERVICE_SYSTEMD="mniui.service"  
UISERV_HOST_SERVICE_USERNAME="mniui"  
UISERV_PORT=4443  
UISERV_SSL_CERT="ui.crt"  
UISERV_SSL_CSR="ui.csr"  
UISERV_SSL_DAYS=90  
UISERV_SSL_KEY="ui.key"  
UISERV_SSL_SIZE=4096  
UISERV_TIMEOUT_REQUEST=120000  
UISERV_TICK_INTERVAL_MS=60000  
UISERV_TLS_INSECURE_CONNECTIONS=true  
UISERV_URL_PREFIX="/mni"  
UISERV_USE_DNS_SD=false  
UISERV_WORKING_DIRECTORY="/usr/local/mni/ui"
```

- `UISERV_ADDRESS` is automatically set to the Linux network interface providing the default route
- `UISERV_DIST_DIRECTORY` this should not be changed
- `UISERV_DNS_RESOLVE` timeout in milliseconds for DNS lookups
- `UISERV_PORT` if changed, ensure DNS records match
- `UISERV_TICK_INTERVAL_MS` obsolete
- `UISERV_URL_PREFIX` this should match the API Server endpoint URL path prefix - this should not be changed
- `UISERV_USE_DNS_SD` toggle UI Server using DNS Service Discovery to determine which host/port to expose its endpoint against and this should not be changed unless DNS `SRV` records have been previously verified

### 13.2.8. Identity and Access Management (IAM) Server

```

#
# IAM
#
IAM_ADDRESS="192.168.1.16"
IAM_BOOTSTRAP_ADMIN_USERNAME="admin"
IAM_BOOTSTRAP_ADMIN_PASSWORD="admin"
IAM_HOST_SERVICE_LOG_FILE="/var/log/mni/iam.log"
IAM_HOST_SERVICE_SYSTEMD="mniam.service"
IAM_HOST_SERVICE_USERNAME="mniam"
IAM_KEYCLOAK_VERSION="26.4.4"
IAM_PORT_HTTPS=6443
IAM_SSL_CERT="iam.crt"
IAM_SSL_CSR="iam.csr"
IAM_SSL_DAYS=90
IAM_SSL_KEY="iam.key"
IAM_SSL_SIZE=4096
IAM_WORKING_DIRECTORY="/usr/local/mni/iam"

```

- `IAM_ADDRESS` is automatically set to the Linux network interface providing the default route
- `IAM_BOOTSTRAP_ADMIN_USERNAME` the default username of the administrative account used in Keycloak
- `IAM_BOOTSTRAP_ADMIN_PASSWORD` the default password of the administrative account used in Keycloak - this password should be changed after installation via the Keycloak Management console ([https://{:IAM\\_ADDRESS}:{IAM\\_PORT}/](https://{:IAM_ADDRESS}:{IAM_PORT}/))
- `IAM_KEYCLOAK_VERSION` version of Keycloak to use - this should not be changed
- `IAM_PORT_HTTPS` if changed, ensure DNS records match

### 13.2.9. Predict Service

```

#
# predictService
#
PREDICTSRV_CRONTIME="0 0 * * *"
PREDICTSRV_DEBUG=true
PREDICTSRV_ENDPOINT_RETRY_INTERVAL_MS=15000
PREDICTSRV_ENDPOINT_KEEPALIVE_INTERVAL_MS=60000
PREDICTSRV_HOST_SERVICE_LOG_FILE="/var/log/mni/predict.log"
PREDICTSRV_HOST_SERVICE_SYSTEMD="mnipredict.service"
PREDICTSRV_HOST_SERVICE_USERNAME="mnipredict"
PREDICTSRV_TLS_INSECURE_CONNECTIONS=true
PREDICTSRV_WORKING_DIRECTORY="/usr/local/mni/predict"

```

- `PREDICTSRV_CRONTIME` can be changed, adhere to rules of Linux CronTab for specifying the execution time

### 13.2.10. DNS Server

**Note:**

- while some deployments will NOT include the embedded DNS Server (dnsmasq), this configuration is still required for the `HOST` and `DOMAIN NAME` configuration

```

#
# dnsServer
#
DNSSERV_INSTALL=false
DNSSERV_ADDRESS=
DNSSERV_VERSION="2.70"
DNSSERV_HOST_SERVICE_LOG_FILE="/var/log/mni/dns.log"
DNSSERV_HOST_SERVICE_SYSTEMD="mnidns.service"
DNSSERV_HOST_SERVICE_USERNAME="mnidns"
DNSSERV_PORT=53
DNSSERV_HOST="marlinintegration"
DNSSERV_DOMAIN="merkator.com"
DNSSERV_WORKING_DIRECTORY="/usr/local/mni/dns"
DNSSERV_SYSTEMD_RESOLVE=1
DNSSERV_RESOLVCONF=1

```

- `DNSSERV_ADDRESS` is automatically set to the Linux network interface providing the default route
- `DNSSERV_VERSION` the version of dnsmasq to download and use - this should not be changed
- `DNSSERV_PORT` the TCP/UDP port to use DNS services against - this should not be changed
- `DNSSERV_HOST` the hostname used during installation of many components
- `DNSSERV_DOMAIN` the domain name used during installation of many components
- `DNSSERV_SYSTEMD_RESOLVE` records if systemd resolve was deployed - this should not be changed and is set during installation
- `DNSSERV_RESOLVCONF` records if resolve-conf was deployed - this should not be changed and is set during installation

### 13.2.11. Non-Unique per Component

The following are defined per component and are treated in the same manner:

- `{MNI_COMPONENT}_DEBUG` can be either `false` or `true` to toggle addition verbose logging
- `{MNI_COMPONENT}_TLS_INSECURE_CONNECTIONS` if struggling with SSL certificates, set this to `true` for the respective component
- `{MNI_COMPONENT}_HOST_SERVICE_LOG_FILE` can be changed but recommended any changes are consistently applied
- `{MNI_COMPONENT}_HOST_SERVICE_SYSTEMD` value should be left as is
- `{MNI_COMPONENT}_HOST_SERVICE_USERNAME` can be changed and is used as the dedicated component username
- `{MNI_COMPONENT}_WORKING_DIRECTORY` can be changed and it used as the home directory for the dedicated component user
- `{MNI_COMPONENT}_ENDPOINT_RETRY_INTERVAL_MS` used by service components and can be changed
- `{MNI_COMPONENT}_ENDPOINT_KEEPALIVE_INTERVAL_MS` used by service components and can be

changed

## 13.2.12. SSL Certificates

MNI will unless configured otherwise generate and use Self-Signed SSL certificates, if using signed certificates then the number of days and key size values can be ignored. If using same wildcard domain SSL certificate all components can use the same certificate chain, otherwise one certificate per Server component is suitable.

- `{MNI_COMPONENT}_SSL_CERT` - `PEM` formatted certificate including any authority certificates
- `{MNI_COMPONENT}_SSL_CSR` - Self-Signed certificate request filename
- `{MNI_COMPONENT}_SSL_DAYS` - Self-Signed certificate lifetime in days
- `{MNI_COMPONENT}_SSL_KEY` - `PEM` formatted key certificate filename
- `{MNI_COMPONENT}_SSL_SIZE` - Self-Signed certificate key size

## 13.2.13. Example

### Note:

- It is recommended that the deployment environment file be configured for the deployment before installation is started.

A fully populated deployment environment file would be similar to:

```
#=====
# MarlinDT Network Intelligence (MNI) - Deployment Environment File
#
# Corporate Headquarters:
# Merkator • Vliegwezenlaan 48 • 1731 Zellik • Belgium • T:+3223092112
# https://www.merkator.com/
#
# © 2024-2025 Merkator nv/sa. All rights reserved.
#=====
#
# Common
#
CONFIG_DIRECTORY="/etc/mni"
HOST_SERVICE_GROUP="mni"
MNI_NAME="MarlinDT Network Intelligence"
MNI_BUILD="1745876813"
MNI_VERSION="1.0.0"
```

```

MNI_LICENSE_PRIVATE_KEY="PT09PUJFR010IEjQ0VOU0UgS0VZPT09PQpLSFhrZ3JtRFCxYTkvUy8zL3ZhUVB1T
mpQTTRQSUsVnV3bEtLZEdkVlhuW1zIWEldjh0UGNMN1pmYXZHampObEw5zZJvTmdrazhyR2pJT1cyS05lelN5MW1
3cjRER1hBYkv5TwdUNjbjWxhQd3SzS3BMOTz1RUpqR01SRXBByNmpwdUVUVUUxNFcyQ3ZSdmlTeFNDcFJKT21HeEV5T
mxVMWhmZDJrLzJ3RV1Kc1FTR1FWcS9MZ0hIeWhUTFg3d0wwUjZkeXJJd0pDbk5zUGwvYy9YU1hramdKU0hWdTUxL0Y
xM1NvdFMxMlp1YXM0ajFTaDZpRDF1RjMvK053U0dvWnJzzkxYTNIUjdnUm1KMmdKRTE2ek16Q0RDdy92WDhIOFFSB
1poNVdaWXBRQ2Z5cG9JUC9kcTFqTGxkYWhwaHduZmNXNvtc2JUSWhWZ3dLcldxb1JsnGVFL2VJMXZPbDNEl2lxVkr
aaK1ZNThqcXVSQXJCd3VJcVE0UFJL3FOZ0phSGtzVGxCYtzeFNBeUrQkZTY3FTamtTSz1vdnFra016SStrVkf6R
1RrajZ6N2xISV1hZEZqNGZKaTBWL25KZwtiZ1BvdktYTzBDQmVITUZkt2pKMHh6b3BscHFGejBnbEhuTUJ4Smlpb1Z
PSkwrMWJvbGY5R111V2E2NUH2K3M1OWY3TU5OaEFOSnZ3bkdyK01QaXVrMwd1K21PRWp4alVTR31zcFzUng0U1JQd
m04NkpsZ11vS05CYXMzQU5rc3pzUDBmcmhvL1V4YVlobjVhcmlvbGzoUk1VbGtKN1c0NVIvTkpacHEzcCt6Zi9ZRT1
hSpnOF1WVG5UOHdZVVIwczQ5a21rL3FiaVAwTUtjc3hzYmtXeWxIT1AzWW82a0NybfhRcz18ffUyRnNkR1ZrWDEvR
itINLeecowDrey1NEFsQkRKZEVEUTF0d3BPTGUwTHp0UW12bTBDDUUCydEJmu055dkvxdVhGz1FrUG5tRXR1RU11a1g
0L0Y3TDc3YnUxVSt0cXBpQ0Z1Qm9QdmxNWGpBbjQ0ZHCSR1MrbmR0b3VXYmFVbXVzRHplejhDNHQyMWx2Q282WFRse
npvcU1Nzn1xz0J1YjROTnpKZ01CSGZYSVbWTF4aTBuSUdpA3o5sjNjZ2RCZgpXn2VGYWVMWTVjQmwvU1ZzSE9kN11
qZGJrvThXKzVldjVuY1UwYTVWY3fjdlhTYUtnZ3FpTGtrs3JDMGszWnZ4VER2QW13Sk15bjFtz1YwY1VtN3F3b3o1Q
zFFRzV3aElyTm1Rb3dXdFY5M0pnL0dGSwpTUWpyaThMSy8wZUVtYTzrYmZ6Y1ZZYj1HYURnZStQL3N3ZEdRaUxydmh
1T2FQSGxVmKFKMw9ack9nbk0rzktzVUVoaw1rUEphclh0S1Fqd0NiWly4RVhVWng5cjdKaGVTdEVDeGJRB014Zmwxt
ncyzlQ0alJsbjzHenk4UnpHNWpsS31zck9ibUd3NE1abzJQVjdzVi84N31IqjMwYmgvRW4rUWNqOUwrdFlYZ24ywuh
UQWQ4ZnFsaU0rUkFtbFdssKvKzHxms80dXFPTws1SzVmaFFFejNws2NkQmZ0z1VvNC91eEkyUFpJV1k5ZCtLQZse
HFKV5jN31BYTc5SVNaT0FNUU16K0M1WWFyNGNIRElsYWFQXQI2Q1ptTm10TlpOalh6aUhLcFRDNXRIaU5ycm81b1V
qYnlwTjBsQjh0RWFDWkdFv0ZkTWMrVm9rQ09DZW1FN3pGMC8xQ0w3UUVTQytBeTNScXBkNkNPuFRVYUxtUTlQZnNLR
3I4bkRwZFZwVldCbXpCaHFzzXJMzUxDY2tETC9pY2VRbmFpcmJhQ3NBYjb2RghObitFM05hYTzKN28yR1hDd0tzUC9
6T3IxQTlWZm9xV1pYdkFmK1poWFNOenRYL1RFbnlobitaRTBnWHNRQ0cvaG55YkRPN1QzcWFoa01XVGh0eUUxVWJhz
jz1VHBrYm12R2h1WENkZ1U2S2RlaEhqTjVYK0M2eXzTVE43L3V4a0Jps3p5NW9GYnYraFBJaz0KPT09PUVORCBMSUN
FT1NFIetFWT09PT0"

MNI_LICENSE_PUBLIC_KEY="LS0tLS1CRUdJTibQVUJMSUMgS0VZLS0tLS0KTU1JQ01qQU5CZ2txaGtpRz13MEJBuu
VGQUFP0FnOEFSu1DQ2dLQ0FnRUFwc2Q4eW9KN2FpMFHWeU8rWDNwYwphNXIXvFhBSUE5MFVjVVJERCtsdmM4OU1V
akdkcUd20U9LZWZxGxLaHM1Wnd6WjZsK09va111L2JjOGpYOUpLCmVmZHQzYkVHWW0cnYrVEpyZWR1S1VqTDkrR2
1JQu81eFppeUd2WT1kR1NtajRaN2JaNXdobD1TLzhmSERPR2MKNFZITmozV0f1t3NEOVUrMmxoMnozNHRVT2w1OG1n
cDYyQkUvR0YrZmjkwHR6T2Vtc3Q3K0k4ZTRkUFgveGFnbQpNeENHRvhEK2UreUJreT1JcmYzeVVTY2RNdHRSVVNhbV
BIeGt5bFFidnVCsy9sd1NsY0lteGEvWjvMYTcwRHZBCKzpSGJBTw1TYVIwQXoyMGh4cFpncytwwU5NMctPeVQ0TXU5
bUZrRC9keEgzR11ZS0pjCTNnTTJOM296ZzdiTXcKTEhHNmxYbkpteVvhS3dUT0pDzzlmM2tBOV10SXhydC9VZGphYk
NOY3huLeecowDreyStNchFCc0RFR1c5TW5DSgpNWHU4eVc0eUN2Uks3TEhjVm15WD15Sk9DT1hrTTZKeWo4Q2R2S2V
nYkhSNWNiN1Q1NjVKVXZJN19oNFVmOGcrCmU0Z0luM0VTRndhdzdTNGdUdkNOC3poZ09tQ0xsN2dKWDd0cmV4dTFWk
3NvamFYSFc1WG5oSXc2TzFaaDNYQmEKU1YweVFoa3pNajhJQXUzNWJZWHR3UzJUNDRQeERLUwtGYzZsZfhoK0ZscWd
zYjJxK0NJbnJNzvKaV1Iq3dpQgpONTdWT3FRUnRyS1VLYXU5MHPNa2JkTHNYNwtCM0pxTHBldW1pOGhITmJRZHEXN
FBEMFVWbGtpTG5TV1pr0WdPCi9jMERrZ3AwK0zoRn12VXJ4VD1kSmFzQ0F3RUFbUT09Ci0tLS0tRU5EIFBVQkxJQyB
LRVktLS0tLQo="

MNI_DEFAULT_COUNTRY_CODE="GBR"
MNI_SERVICE_USERNAME="af06f2fe515b91efbLeeCowdreyb937343cdb54849c148c2"
MNI_SERVICE_KEY="BFmZLMn2VPc+120ekJL0c+Sx6IF9uLeeCowdreyMGyhQ/sssh1z0X1uCYKTqJjzUM1iAw9OHX
++MI6CPvB"
#
# OAuth2.0 / OpenID Connect
#
# local iamServer (Keycloak)
OAUTH=true
OAUTH_CLIENT_ID="MNI"
OAUTH_CLIENT_SECRET="mQGuleEcwodREynXpDqBbt1Xe4yzikNs"
OAUTH_DISCOVERY_URL="https://marlinintegration.merkator.com:6443/realms/MNI/.well-
known/openid-configuration"
# remote Microsoft Azure Active Directory (EntraID)

```

```
#OAUTH_CLIENT_ID="Mnf3ce2c1d-b03c-4lee-4cowd-reydd5489d741I"
#OAUTH_CLIENT_SECRET="Nok8Q~RWkLeeCowdreykl1G_CUfoqJGdn."
#OAUTH_DISCOVERY_URL="https://login.microsoftonline.com/a17ec843-alee-4cowd-reyc-
3e803a903e14/v2.0/.well-known/openid-configuration"
#
# alertService
#
ALERTSRV_CRONTIME="0 0 * * *"
ALERTSRV_DEBUG=true
ALERTSRV_ENDPOINT_KEEPALIVE_INTERVAL_MS=120000
ALERTSRV_ENDPOINT_RETRY_INTERVAL_MS=15000
ALERTSRV_HOST_SERVICE_LOG_FILE="/var/log/mni/alert.log"
ALERTSRV_HOST_SERVICE_SYSTEMD="mnialert.service"
ALERTSRV_HOST_SERVICE_USERNAME="mnialert"
ALERTSRV_TLS_INSECURE_CONNECTIONS=true
ALERTSRV_WORKING_DIRECTORY="/usr/local/mni/alert"
#
# apiGateway
#
APIGW_ADDRESS="192.168.1.16"
APIGW_HOST_SERVICE_LOG_FILE="/var/log/mni/apigw.log"
APIGW_HOST_SERVICE_SYSTEMD="mniapigw.service"
APIGW_HOST_SERVICE_USERNAME="mniapigw"
APIGW_IGNORE_AUTH_VALIDATOR=true
APIGW_PORT=8443
APIGW_PROXY_RATE_LIMIT_REQUESTS=32767
APIGW_PROXY_CAPACITY_REQUESTS=32767
APIGW_PROXY_RATE_LIMIT_EVERY="1m"
APIGW_TLS_INSECURE_CONNECTIONS=true
APIGW_SSL_CERT="apigw.crt"
APIGW_SSL_CSR="apigw.csr"
APIGW_SSL_DAYS=90
APIGW_SSL_KEY="apigw.key"
APIGW_SSL_SIZE=4096
APIGW_KRAKEND_VERSION="2.12.0"
APIGW_WORKING_DIRECTORY="/usr/local/mni/apigw"
#
# apiServer
#
APISERV_ADDRESS="0.0.0.0"
APISERV_API_DIRECTORY="/usr/local/mni/api/openapi"
APISERV_BACKUP_DIRECTORY="/usr/local/mni/api/backup"
APISERV_DEBUG=true
APISERV_DUCKDB_VERSION="v1.4.1"
APISERV_DUCKDB_BACKUP_CRONTIME="0 2 * * *"
APISERV_DUCKDB_BACKUP=true
APISERV_DUCKDB_FILE="/usr/local/mni/api/mni.duckdb"
APISERV_DUCKDB_MAX_MEMORY="20GB"
APISERV_DUCKDB_THREADS=4
APISERV_ENCRYPTION_KEY="qBU4K9XBw2JtYx7puSGiSig8LbxxDpSaprK7+u+EJI8="
APISERV_ENCRYPTION_IV="U5uKpGIE3cIXh5kuhFZXuQ=="
APISERV_KEEPALIVE=true
```

```
APISERV_HOST_SERVICE_LOG_FILE="/var/log/mni/api.log"
APISERV_HOST_SERVICE_SYSTEMD="mniapi.service"
APISERV_HOST_SERVICE_USERNAME="mniapi"
APISERV_MULTICAST_ADDRESS="230.185.184.183"
APISERV_MULTICAST_PORT=54321
APISERV_PORT=7443
APISERV_PREMISES_PASSED_BOUNDARY_DISTANCE=35
APISERV_ROLE="standalone"
APISERV_SPATIAL_UPDATE_GEOMETRY_CRONTIME="*/5 * * * *"
APISERV_SSL_CERT="api.crt"
APISERV_SSL_CSR="api.csr"
APISERV_SSL_DAYS=90
APISERV_SSL_KEY="api.key"
APISERV_SSL_SIZE=4096
APISERV_TIMEOUT_KEEPALIVE=5000
APISERV_TIMEOUT_REQUEST=300000
APISERV_DOCUMENT_DIRECTORY="/usr/local/mni/api/document"
APISERV_UPLOAD_DIRECTORY="/usr/local/mni/api/upload"
APISERV_URL_PREFIX="/mni"
APISERV_URL_VERSION="/v1"
APISERV_USE_DNS_SD=false
APISERV_WORKING_DIRECTORY="/usr/local/mni/api"
#
# fetchService
#
FETCHSRV_CVE_SCAN=true
FETCHSRV_CVE_DIRECTORY="/usr/local/mni/cvelistV5"
FETCHSRV_DEBUG=true
FETCHSRV_ENDPOINT_KEEPALIVE_INTERVAL_MS=120000
FETCHSRV_ENDPOINT_RETRY_INTERVAL_MS=15000
FETCHSRV_FX_UPDATE=true
FETCHSRV_HOST_SERVICE_LOG_FILE="/var/log/mni/fetch.log"
FETCHSRV_HOST_SERVICE_SYSTEMD="mnifetch.service"
FETCHSRV_HOST_SERVICE_USERNAME="mnifetch"
FETCHSRV_TLS_INSECURE_CONNECTIONS=true
FETCHSRV_WORKING_DIRECTORY="/usr/local/mni/fetch"
#
# uiServer
#
UISERV_ADDRESS="192.168.1.16"
UISERV_DIST_DIRECTORY="/usr/local/mni/ui/dist"
UISERV_DEBUG=true
UISERV_DNS_RESOLVE=300000
UISERV_HOST_SERVICE_LOG_FILE="/var/log/mni/ui.log"
UISERV_HOST_SERVICE_SYSTEMD="mniui.service"
UISERV_HOST_SERVICE_USERNAME="mniui"
UISERV_PORT=4443
UISERV_SSL_CERT="ui.crt"
UISERV_SSL_CSR="ui.csr"
UISERV_SSL_DAYS=90
UISERV_SSL_KEY="ui.key"
UISERV_SSL_SIZE=4096
```

```
UISERV_TIMEOUT_REQUEST=120000
UISERV_TICK_INTERVAL_MS=60000
UISERV_TLS_INSECURE_CONNECTIONS=true
UISERV_URL_PREFIX="/mni"
UISERV_USE_DNS_SD=false
UISERV_WORKING_DIRECTORY="/usr/local/mni/ui"
#
# IAM
#
IAM_ADDRESS="192.168.1.16"
IAM_BOOTSTRAP_ADMIN_USERNAME="admin"
IAM_BOOTSTRAP_ADMIN_PASSWORD="admin"
IAM_HOST_SERVICE_LOG_FILE="/var/log/mni/iam.log"
IAM_HOST_SERVICE_SYSTEMD="mniam.service"
IAM_HOST_SERVICE_USERNAME="mniam"
IAM_KEYCLOAK_VERSION="26.4.4"
IAM_PORT_HTTPS=6443
IAM_SSL_CERT="iam.crt"
IAM_SSL_CSR="iam.csr"
IAM_SSL_DAYS=90
IAM_SSL_KEY="iam.key"
IAM_SSL_SIZE=4096
IAM_WORKING_DIRECTORY="/usr/local/mni/iam"
#
# predictService
#
PREDICTSRV_CRONTIME="0 0 * * *"
PREDICTSRV_DEBUG=true
PREDICTSRV_ENDPOINT_RETRY_INTERVAL_MS=15000
PREDICTSRV_ENDPOINT_KEEPALIVE_INTERVAL_MS=60000
PREDICTSRV_HOST_SERVICE_LOG_FILE="/var/log/mni/predict.log"
PREDICTSRV_HOST_SERVICE_SYSTEMD="mnipredict.service"
PREDICTSRV_HOST_SERVICE_USERNAME="mnipredict"
PREDICTSRV_TLS_INSECURE_CONNECTIONS=true
PREDICTSRV_WORKING_DIRECTORY="/usr/local/mni/predict"
#
# dnsServer
#
DNSSERV_INSTALL=false
DNSSERV_ADDRESS=
DNSSERV_VERSION="2.70"
DNSSERV_HOST_SERVICE_LOG_FILE="/var/log/mni/dns.log"
DNSSERV_HOST_SERVICE_SYSTEMD="mnidns.service"
DNSSERV_HOST_SERVICE_USERNAME="mnidns"
DNSSERV_PORT=53
DNSSERV_HOST="marlinintegration"
DNSSERV_DOMAIN="merkator.com"
DNSSERV_WORKING_DIRECTORY="/usr/local/mni/dns"
DNSSERV_SYSTEMD_RESOLVE=1
DNSSERV_RESOLVCONF=1
#
```

# 14. Hosting Specifications

## 14.1. Standalone

### 14.1.1. Combined Single Host

Component	CPU	RAM	Disk
All Combined	4	16GB	64GB
API Server	2	10GB	8GB
Alert Service	1	1GB	1GB
API Gateway	1	1GB	1GB
Fetch Service	1	1GB	4GB
Predict Service	1	1GB	1GB
UI Server	1	1GB	1GB

### 14.1.2. Individual Hosts

Component	CPU	RAM	Disk
All Combined	4	16GB	64GB
API Server	2	10GB	8GB
Alert Service	1	1GB	1GB
API Gateway	1	1GB	1GB
Fetch Service	1	1GB	4GB
Predict Service	1	1GB	1GB
UI Server	1	1GB	1GB

## 14.2. Cluster

### 14.2.1. Per API Server Host

Component	CPU	RAM	Disk
API Server	2	10GB	8GB

### 14.2.2. Other

Component	CPU	RAM	Disk
Alert Service	1	1GB	1GB
API Gateway	1	1GB	1GB
Fetch Service	1	1GB	4GB
Predict Service	1	1GB	1GB
UI Server	1	1GB	1GB

**Note:**

- Both the API Gateway and UI Server can be deployed on to multiple hosts without additional configuration. The DNS A records used by MNI will need to be modified to support multiple resolved hosts/IP addresses (DNS Round-Robin)

## 14.3. Operating Systems

MNI is developed and supported on Debian based operating systems, this includes Ubuntu and other similar variants. The minimum supported version of Debian is version 12.

**Note:**

- RedHat Enterprise Linux (RHEL) and its variants are not supported but compatible and installation scripts will require adjustment to support the RHEL package manager. For demonstration systems any host supporting DuckDB, NodeJS 22.x can be used.

## 14.4. Host Firewalls

A generic example for an IPv4 `iptables / netfilter` host based firewall with MNI deployed on a single host supporting IPv4 is shown below:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p tcp -m tcp -m multiport --dports 4443,6443,7443,8443 -m conntrack --ctstate NEW,RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
-A INPUT -p icmp --icmp-type 8 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp --icmp-type 0 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
-A INPUT -s 8.8.8.8/32 -p udp -m udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
-A INPUT -s 8.8.4.4/32 -p udp -m udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -j DROP
-A FORWARD -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp --icmp-type 8 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -p icmp --icmp-type 0 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
```

```
-A OUTPUT -d 8.8.8.8/32 -p udp -m udp --dport 53 -m state --state NEW,ESTABLISHED,RELATED  
-j ACCEPT  
-A OUTPUT -d 8.8.4.4/32 -p udp -m udp --dport 53 -m state --state NEW,ESTABLISHED,RELATED  
-j ACCEPT  
-A OUTPUT -o eth0 -j ACCEPT  
-A OUTPUT -j ACCEPT  
COMMIT
```

#### Note:

- port numbers 4443, 6443, 7443 and 8443 should be updated to reflect deployment
- port number 22 is used for SSH connections on the host
- port number 53 is used for DNS operations and should be configured to reflect deployment
- IP addresses 8.8.8.8 and 8.8.4.4 are used for upstream DNS resolution and should be changed to reflect deployment

## 15. License Generator

MNI requires a valid license to be included within the environment deployment file and this license must be generated specifically for the host or hosts operating the MNI API Server.

#### Note:

- if deploying API Server to a container environment, such as Kubernetes, MNI automatically detects this and is less stringent on the hostname matching on the license but will still enforce the licensed domain name.
- The MNI License Generator is NOT included into the release archive and should **NOT** be provided to customers nor partners.
- The MNI License Generator includes cryptographic private and public keys, the public key is also embedded within `oasConstants.mjs` and the default `mni.ini`, all of these must use the **same** cryptographic keys

Within the `licensegen` source directory, deploy the required NodeJS libraries:

```
$ cd mni/src/licensegen  
$ npm install --omit=dev  
  
added 20 packages, and audited 21 packages in 590ms  
  
7 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

Test execution of `licensegen`:

```
$ node licenseGen.mjs
```

```
20251120T114910 error missing CLI options/parameters
```

```
20251120T114910 usage:
```

```
node licenseGen.mjs \
--generate      create a new encrypted key for inclusion within mni.ini
  [--out=mni.key]           filename to save encrypted key to
  [--start=YYYYMMDD[T]HHmmss]
  [--duration=13]           positive integer, no validation against unit is performed
  [--unit=month]            second,minute,hour,day,week,month,quarter or year
  [--role=standalone]       role of the API server, standalone, master or slave
  [--host=mni]              the hostname executing API Server
  [--domain=merkator.local] the domain name executing API Server

--verify        verify encrypted key contents
  [--in=mni.key]           filename to read encrypted key from
```

```
20251120T114910 example:
```

```
node licenseGen.mjs --generate --start=20250801T000000 --duration=1 --unit=month --
role=standalone --host=mdt --domain=marlindt.net --out=mdt.pem
```

To generate a standalone single host license key:

```

$ node licenseGen.mjs --generate --start=20250801T000000 --duration=1 --unit=month --
role=standalone --host=mni --domain=customer.com --out=customer.pem
20251120T115423 info {
  out: 'customer.pem',
  role: 'standalone',
  host: 'mni',
  domain: 'customer.com',
  start: '20250801T000000',
  expiry: '20250831T235959'
}
$ cat customer.pem
PT09PUJFR010IExJQ0VOU0UgS0VZPT09PQpBUEx4Vk5hQ2Y20Gd3ZEpmMmRzbHhFMTU1V21pZEt
wSzNTcjZJVXR1T1FMT05aaW0rOVJ1WDh2R1QyNWgrQTRwWE1pRTYvdVludzZEYjBtY054T053L3pUM3ZiOHAwVm53TnJMbTd0MXIrSWV2S2FTanY2alUydVpLVUK5THRsWFJ6NjVadXF1bTNHZ1hVNU41cmJPdVFxdHFvUDVSVS85T1hSdmxDL2ZscDhNNVZMR2NaMHBLNDJFVFJrUkNNV110Ti9DWnZ1WGc1Rzz0RXBXaDUzODdvK2RCQ3J3UjdldnhnMHZvOVVNN2FrSit5VTZ5RFWFndFU5eGzia2FUOHJPWktCMEYzVwlhWXFJbHVrazQ3SmFxUS95NE5VY09RRytsWFhWTGJQK0hxZFQwdG94bnFwMu1vQ0MwOFRMQ2ladkkwaVdIRC9NaVNPsgZZU1Z2dFZ2Y2N3Ym9qWVJUY0Y3bV1jaGpJM1R6amo3WHpQdkzRvVV6aWVjOHd1THptQ2NEcEFrNHpZWUhSYWdSVVR1Zm9QK3VMYWR5V2JuMnRUUHVWOUt1UzNXs28vdWEwb0R0NmxB0R31hZ2JyMHN4NG9kdtyaUFUOEfnMEhtcnA2ZEY4Ky9jREVaQWV5c21iTzAzUERiYnF2dDJ5NDhKK1FuZ2ZVR29oM0R2a2YxT3hMTFlzNHoxUE1nYW1RY0xrZDhNVU9iYmVSNjNFWWJyTU9oRlhisjN2VEtNdm5td0tuZHJvU1AxOGdQSE92dnpuExhqMENrT2V3Yk1qZnh4UXRmVG1hSkhHR1d5azA5dUh4S3F1bHBXSjBqTDV2bW9IU3BVSXRraDY4Z012V2pYYVVXR3cvdXBZNE01NG1xVDNka1ZkSmhCU214amMyZDR2ZU9nNENsWUpIR21CWT18ffUyRnNkR1zrWDE5RCtrVG1M01RsTkx6SmUzV2toCTNjcjBONERTajAzVUNIZzJrc3FVajRLY0NjN3Rik3ZIR3VoA3o0U2JvWlZTeCtyWDFNN0lnQ3RTUGR6U3FYt3ZvZFFWNkRydVpWOGRUZTQxN2FJMVdIV2h0aVNaTWV4U3dTn5aSXfpdF1iOTd1QWdaZk5uVHM5YU8xNTZQTU1MTVE4bmJXv0xCsNjwaz0KV1lRVFpjbuDsdlIxL05zR3JRaHVJcVh4HNPNu5zc3lqR2VNQk42QVhMbWdNalBPMFc3Rwp0L0hUYmthY0N1UDAwNEDQc jFxc0M5Q2RNVG5uS1NvNDRBMHkvZGs0MndaSVp6Z1VaK0FGZ29vZUR0MVE1bngyaHMvZ1BHbXhuRjZITEhUSk1LazJJQWQrQkzxWl1jZEFKNF1vQ1FOOXBlemdEbGRzQjFQODvhTjZiYWRIR3hFTW90SSs4Vm0yUVMwVzFaTT1NYURFb3ZvM3kwcUtBY2JabUFEQX1qbDJOMnZ5MkR1VXpvQjVvYk11UkU1MHNXS1ErWjgzQUowZz1mRG5vaFM5RU1PdEx6Zno3SxhOMEpTRHRA1VtbFRYbmdDY9BU29Ve1VkfWFluWEFwVWJ2NVUrUm4wRzdvNHUXRUZYTHjyMGIzVjdXZ0ZLYkRnOHUrMmVnTV1WejBhMkJieStKZWPodTBpZVQ1N1ZxT1c2V05zQ1hPTWRLT1NKSWhrzUtVWFpmaDZvcml0UjFua3NwTkxKb3JhNVd6L01MNkVOMFixazIvV3RBVzF5Mhh3UzB6dxZUcGt4MxEySwNkbzdxWGRZVVXYj1nbnFYUWVoMS90dWNVMzVoQVFIdlBsVT1wVjhVWD1FMms2S2p6cDkxV1k1TVZ0YXZueFh3R2YydGNkTDlZv3FaS11JdVFrcFhPMHVEZm5HdkM2R2tGV1RvUnkyA3d1R3owVDUxZTRhbU55MGZSWGt5UGw2ejlyb3pIUGhhenpHTwt1cy9kTTZKZ28vUmFiQ0N2UVhXUDh2aXNtQWNqZkFJQk1zeE53NxFiaU9uU3RoC1pkUW9kak9xZzJKV3RaS2pBUkc2MWVYR0E1TmxKUENFZjIoY25VeGxoN1JtOHYxams9Cj09PT1FTkQgTELDRU5TRSBLRVk9PT09

```

The contents of the generated license key file, in this example `customer.pem` should be placed within the respective `mni.ini` variable `MNI_LICENSE_PRIVATE_KEY` prior to deployment, for example:

```

#
# Common
#
...
MNI_LICENSE_PRIVATE_KEY="PT09PUJFR01OIExJQ0VOU0UgS0VZPT09PQpBUEx4Vk5hQ2Y20Gd3ZEpmMmRzbHhFM
TU1V21pZEtWszNtCjzJVXR1T1FMT05aaW0rOVJ1WDh2R1QyNWgrQTRwWE1pRTYvdVludzZEYjBY054T053L3pUM3Z
iOHAwVm53TnJMbTd0MXIrSWv2S2FTanY2alUydVpLVUk5THRswFJ6NjVadXF1bTNHz1hVNU41cmJPdVFXdHFvUDVSv
S85T1hSdmxDL2ZscDhNNVZMR2NaMHBLNDJFVFJrUkNNV110Ti9DWnZ1WGc1RzZoRXBXaDUzODdvK2RCQ3J3Ujdldnh
nMHZvOVVNN2FrSit5VTz5RWFndFU5eGzia2FUOHJPWktCMEYzVwlhWXFJBhVrazQ3SmFxUS95NE5VY09RRytsWFhWT
GJQK0hxZFQwdG94bnFwMULvQ0MwOFRMq21adkkwaVdIRC9NaVNpSGZZUlZ2dFZ2Y2N3Ym9qWVJUY0Y3bV1jaGpJM1R
6amo3WHpQdkzRVVV6aWVjOHd1THptQ2NEcEfrNhpZWUhSYwdSVVR1Zm9QK3VMYWR5V2JuMnRUUHVWOut1uzNXs28vd
WEwb0R0Nmx0R31hz2JyMHN4NG9kdytyaUFUOEfnMEhtcnA2ZEY4Ky9jREVaQWV5c21iTzAzUERiYnF2dDJ5NDhKK1F
uZ2ZVR29oM0R2a2YxT3hMTFlzNHoxUE1nYW1RY0xrZDhNVU9iYmVSNjNFWWJyTU9oRlhIsjN2VEtNdm5td0tuZHJvU
1AxOGdQSE92dnpUeXhqMENrT2V3Yk1qZnh4UXRmVG1hSkhHR1d5azA5dUh4S3F1bHBXSjBqTDV2bW9IU3BVSXRraDY
4Z012V2pYYVVXR3cvdXBZNE01NG1xVDNka1ZkSmhCU214amMyZDR2ZU9nNENsWUpIR21CWT18ffUyRnNkR1ZrWDE5R
CtrVG1Mq1RsTkx6SmUzV2toCTNjcjBONERTajAzVUNIZzJrc3FVa jRLY0NJN3RiK3ZIR3VOa3o0U2JvW1ZTeCtyWDF
NN01nQ3RTUGR6U3FYt3ZvZFFWNkRydVpWOGRUZTQxN2FJMvdIV2h0aVNaTWV4U3dTNk5aSXFpdfliotd1Qwdazk5uv
HM5YU8xNTZQTU1MTVE4bmJXV0xCsnJwaz0KV11RVFpjbdUdSd1IxL05zR3JRaHVJcVh4NHNPNU5zc3lqR2VNQk42Qvh
MbWdNa1BPMFc3RWpoL0hUYmtHY0N1UDAwNEdQcjFxc0M5Q2RNvg5uS1NvNDRBMHkvZGs0MndaSVp6Z1VaK0FGZ29vZ
UR0MVE1bngyaHMvz1BhbXhuRjZITEhUSk1LazJJQWQrQkZxyWljZEFKNFIVq1FOOXBLEmdEbGrzQjFQODVhtjziYWR
IR3hFTW90SSs4Vm0yUVMwVzFaTT1NYURFb3ZvM3kwcUtBY2JabUFEQXlqbDjOMnZ5MkR1VXpVqjVvYk11ukU1MHNxs
1ErWjgzQUowZz1mRG5vaFM5R1PdEx6Zno3SXhOMEpTRHRaL1VtbFRYbmdDYw9BU29VelVkwFluWEFwVWJ2NVUrUm4
wRzdvNHUXRUZYTHJyMG1zVjdxZ0ZLYkRnOHUrMmVnTV1WejBhMkJieStKZwpodTBpZVQ1N1ZxT1c2V05zQ1hPTWRLT
1NKSWhrzUtvWFpmaDZvcml0Ujfua3NwTkxB3jhNvd6L01MNkVOMFixazIvv3RBVzF5Mhh3UzB6dXZUcGt4MXEysWN
kbzdxWGRzWVvVXYj1nbnFYUWVoMS90dWNVMzVoQVFIdlBsVTlwVjhVWDlFMms2S2p6cDkxV1k1TVZ0YXZueFh3R2Yyd
GNkTD1zV3FaS11JdVFrcFhPMHVEzm5HdkM2R2tGV1RvUnkyA3d1R3owVDUxZTRhbU55MGZSWGt5UGw2ejlyb3pIUGH
henpHTWt1cy9kTTZKZ28vUmFiQ0N2UVhXUDh2aXntQWNqZkFJQk1zeE53NXFiaU9uU3ROc1pkUW9kak9xzZJKV3Ras
2pBUkc2MWVYR0E1TmxKUENFZjloY25VeGxoN1jtOHYxams9Cj09PT1FTkQgTE1DRU5TRSBLRVk9PT09"
...

```

#### Note:

- For clustering, use `--role=master` and `--role=slave` respectively and generate multiple license keys for each MNI API Server host

## 16. Building the release

The MNI CI/CD pipeline can be kickstarted manually by executing the pipeline script. This will perform the required steps to produce a version release archive and commit any changes back to the [MNI repository](#) hosted within Microsoft Azure DevOps.

```

$ ./build/pipeline.sh
[play] Pipeline build 1763643251 [12:54:11]
[play] Checking for YAML Query (yq) [12:54:11]
[checkmark] - ok [12:54:11]
[play] Checking for release version [12:54:11]
[checkmark] - ok [12:54:11]
[play] Updating build tag [12:54:11]
[checkmark] - ok [12:54:11]

```

```
[PLAY] Checking for JSON Query (jq) [12:54:11]
[CHECK] - ok [12:54:11]
[PLAY] Checking for NodeJS [12:54:11]
[CHECK] - ok [12:54:11]
[PLAY] Scanning: apiServer (NodeJS) [12:54:11]
[CHECK] - ok [12:54:12]
[PLAY] Scanning: uiServer (NodeJS) [12:54:12]
[CHECK] - ok [12:54:13]
[PLAY] Scanning: predictService (NodeJS) [12:54:13]
[CHECK] - ok [12:54:13]
[PLAY] Scanning: alertService (NodeJS) [12:54:13]
[CHECK] - ok [12:54:14]
[PLAY] Scanning: fetchService (NodeJS) [12:54:14]
[CHECK] - ok [12:54:14]
[PLAY] Scanning: licenseGen (NodeJS) [12:54:14]
[CHECK] - ok [12:54:15]
[PLAY] Checking for JSON Query (jq) [12:54:15]
[CHECK] - ok [12:54:15]
[PLAY] Checking for NodeJS [12:54:15]
[CHECK] - ok [12:54:15]
[PLAY] Checking for release version [12:54:15]
[CHECK] - ok [12:54:15]
[PLAY] Documenting: apiServer (NodeJS) [12:54:15]
[CHECK] - ok [12:54:22]
[PLAY] Documenting: uiServer (NodeJS) [12:54:22]
[CHECK] - ok [12:54:26]
[PLAY] Documenting: predictService (NodeJS) [12:54:26]
[CHECK] - ok [12:54:28]
[PLAY] Documenting: alertService (NodeJS) [12:54:28]
[CHECK] - ok [12:54:30]
[PLAY] Documenting: fetchService (NodeJS) [12:54:30]
[CHECK] - ok [12:54:33]
[PLAY] Documenting: licenseGen (NodeJS) [12:54:33]
[CHECK] - ok [12:54:35]
[PLAY] Checking for YQ [12:54:35]
[CHECK] - ok [12:54:35]
[PLAY] Checking for release version [12:54:35]
[CHECK] - ok [12:54:35]
[PLAY] Updating OpenAPI version and build tag [12:54:35]
[CHECK] - ok [12:54:35]
[PLAY] OpenAPI Generate JSON [12:54:35]
[CHECK] - ok [12:54:36]
[PLAY] OpenAPI Generate HTML [12:54:36]
[CHECK] - ok [12:54:39]
[PLAY] Checking for NodeJS [12:54:39]
[CHECK] - ok [12:54:39]
[PLAY] Mermaid ERD to SVG [12:54:39]
[CHECK] - ok [12:54:41]
[PLAY] Checking for NodeJS [12:54:41]
[CHECK] - ok [12:54:41]
[PLAY] Packing UI [12:54:41]
[CHECK] - ok [12:54:43]
```

▶	Preparing environment	[12:54:43]
✓	- ok	[12:54:44]
▶	Checking for NodeJS	[12:54:44]
✓	- ok	[12:54:44]
✓	- ok	[12:54:44]
▶	Copying source	[12:54:44]
✓	- ok	[12:54:45]
▶	Updating license public key	[12:54:45]
✓	- ok	[12:54:45]
▶	Obfuscating JavaScript	[12:54:45]
✓	- ok	[12:54:51]
▶	Creating distribution archive	[12:54:51]
✓	- ok	[12:54:54]
✓	- ok	[12:54:54]
▶	Updating upstream repository	[12:54:54]
✓	- ok	[12:54:56]

The build steps contained within the pipeline script are executed sequentially in the following order:

1. `buildtag.sh`
2. `cvescan.sh`
3. `builddeps.sh`
4. `openapi.sh`
5. `erd2svg.sh`
6. `buildui.sh`
7. `release.sh`

## 16.1. JavaScript Code Scanning

As part of the CI/CD process, MNI is scanned for vulnerabilities that may exist in the MNI source code. This is undertaken via.

Third-Party Package	Used Version	License
<code>auditjs</code>	4.0.47	Apache 2.0

### Note:

- Please refer to [auditjs](#) for further details.

## 16.2. Individual Build Scripts

If required, the individual build scripts can be executed in isolation - examine contents of `build` directory for further details but these include:

- `builddeps.sh` generate `nodejs-dependencies.md` in the documentation directory
- `buildtag.sh` update the MNI version specific details in OpenAPI definitions and other areas
- `buildui.sh` webpack the MNI UI to perform obfuscation

- `cvescan.sh` perform CVE analysis on MNI JavaScript source code files
- `erd2svg.sh` produced SVG diagram from the Entity Relationship Diagram definition extracted from the database schema
- `openapi.sh` generate JSON/YAML definitions for the OpenAPI which are embedded within the API Server
- `pipeline.sh` wrapper script to run all build steps
- `release.sh` wrapper script to produce bundle of compressed TAR files containing each MNI component and MD5 verification file

## 16.3. Release artifacts

Within the `release` directory two files will be generated:

1. `mni_{version}.md5` this contains the MD5 hash signature of the main compressed TAR file
2. `mni_{version}.tar.gz` the version specific release of MNI

for example:

```
$ ls -1
mni_1.0.0.md5
mni_1.0.0.tar.gz
```

The version specific release archive can be verified as not corrupted or altered since the checksum was originally generated:

```
$ md5sum -c mni_1.0.0.md5
mni_1.0.0.tar.gz: OK
```

Unpacking the specific release of MNI, the contents include:

```
$ tar -zxf mni_1.0.0.tar.gz
$ ls -1
mni_1.0.0.md5
mni_1.0.0.tar.gz
mni_alertService_1.0.0.md5
mni_alertService_1.0.0.tar.gz
mni_apiGateway_1.0.0.md5
mni_apiGateway_1.0.0.tar.gz
mni_apiServer_1.0.0.md5
mni_apiServer_1.0.0.tar.gz
mni_dnsServer_1.0.0.md5
mni_dnsServer_1.0.0.tar.gz
mni_fetchService_1.0.0.md5
mni_fetchService_1.0.0.tar.gz
mni_iamServer_1.0.0.md5
mni_iamServer_1.0.0.tar.gz
mni_predictService_1.0.0.md5
mni_predictService_1.0.0.tar.gz
```

```
mni_uiServer_1.0.0.md5  
mni_uiServer_1.0.0.tar.gz
```

The component archives can also be verified as not corrupted or altered since the checksum was originally generated:

```
$ find . -type f -name "mni_*?.?.?.md5" -exec md5sum -c {} \; && md5sum -c mni_?.?.?.md5  
mni_fetchService_1.0.0.tar.gz: OK  
mni_uiServer_1.0.0.tar.gz: OK  
mni_alertService_1.0.0.tar.gz: OK  
mni_dnsServer_1.0.0.tar.gz: OK  
mni_iamServer_1.0.0.tar.gz: OK  
mni_apiGateway_1.0.0.tar.gz: OK  
mni_apiServer_1.0.0.tar.gz: OK  
mni_predictService_1.0.0.tar.gz: OK  
mni_1.0.0.tar.gz: OK
```

## 17. Deploying the release

Each MNI component archive can be unpacked for installation purposes.

The structure of all unpacked archives will be similar to:

```
└── alertService  
└── apiGateway  
└── apiServer  
    ├── sample  
    └── world  
└── dnsServer  
└── fetchService  
└── iamServer  
└── predictService  
└── uiServer  
    └── dist  
        ├── css  
        ├── fonts  
        ├── img  
        └── js  
            └── vendor  
                └── chartjs
```

Each component directory will contain its own `mni.ini` - if using a single host, update the copy within the API Server directory `apiServer` first, install the API Server, and all other components will defer to the installed environment configuration file.

Each component contains the following scripts:

- `mni.ini` this is a shared file that is duplicated into the release archive
- `common.sh` this is a shared file that is duplicated into the release archive

- `install.sh` this script is used for undertaking a new installation and is specific to the component
- `uninstall.sh` this script is used for uninstalling an previous installation and is specific to the component
- `update.sh` this script is used for updating an existing installation, typically this is just updating MNI itself but is specific to the component
- `upgrade.sh` this script is used for updating an existing installation, both any third-party and MNI itself will be upgrading but is specific to the component

## 17.1. Example Component Installation

```
# ./install.sh
✋ MNI UI Server Install
[17:56:30]
▶ Adding MNI group
[17:56:30]
✓ - ok
[17:56:30]
▶ Adding user account
[17:56:30]
✓ - ok
[17:56:30]
▶ Adding config directory
[17:56:30]
✓ - ok
[17:56:30]
▶ Adding log directory
[17:56:30]
✓ - ok
[17:56:30]
▶ Adding working directory
[17:56:30]
✓ - ok
[17:56:30]
▶ Adding working dist directory
[17:56:30]
✓ - ok
[17:56:30]
▶ Adding Config to config directory
[17:56:30]
✓ - ok
[17:56:30]
▶ Setting directory permissions
[17:56:30]
✓ - ok
[17:56:30]
▶ Preparing environment
[17:56:30]
✓ - ok
[17:56:31]
```

```

[PLAY] Checking/Updating for valid address binding (uiServer)
[17:56:31]
[OK] - ok
[17:56:31]
[PLAY] Creating (if not exists) SSL certificate
[17:56:31]
[OK] - ok
[17:56:31]
[PLAY] Updating NodeJS package for target
[17:56:31]
[OK] - ok
[17:56:31]
[PLAY] Adding NodeJS libraries
[17:56:31]
[OK] - ok
[17:56:47]
[PLAY] Adding service log file
[17:56:47]
[OK] - ok
[17:56:47]
[PLAY] Adding SystemD service
[17:56:47]
[OK] - ok
[17:56:48]
[OK] - completed
[17:56:48]
# cat /var/log/mni/ui.log

```

-----  
| -- \v -- | \_\_\_\_ - \_ [ | \_\_\_\_ - [ | \_ | \_\_\_\_ - - - | - \_\_\_\_ | / / / | -- -  
	\ /		/ \_ \	\_		/ / \_ \	\_ / \_ \		' \_ \ \ / / / / \_	/ \_				
				\_ /		< ( \_		( \_ )					\ v / \ \ \_ \ ( \_	
\_		\_	\ \_		\_	\ \ \_ , \_	\ \_ \ \_ /	\_		\_		\_ / /	\_	\ \_

```

20251120T175648 info {
  event: 'starting',
  mni: {
    name: 'MarlinDT Network Intelligence',
    version: '1.0.0',
    build: '1745876813'
  },
  endpoint: {
    host: 'marlinintegration',
    domain: 'merkator.com',
    fqdn: 'marlinintegration.merkator.com',
    address: '192.168.1.16',
    port: 4443,
    urlPrefix: '/mni',
    urlVersion: '/v1'
  },
  environment: {
    configDirectory: '/etc/mni',
    country: 'GBR',
  }
}
```

```

distDirectory: '/usr/local/mni/ui/dist',
tickInterval: 60000,
timestamp: 'YYYYMMDD[T]HHmmss',
ignoreTlsSsl: true
},
apiGateway: {
  dns: 'https://marlinintegration.merkator.com:8443/mni/v1',
  ip: 'https://192.168.1.16:8443/mni/v1'
},
ouauth: {
  enabled: true,
  discovery: 'https://marlinintegration.merkator.com:6443/realms/MNI/.well-known/openid-configuration',
  credentials: { id: '***', secret: '*****' }
},
ssl: { key: 'ui.key', cert: 'ui.crt' },
timeouts: { request: 120000 }
}
(node:11788) Warning: Setting the NODE_TLS_REJECT_UNAUTHORIZED environment variable to '0' makes TLS connections and HTTPS requests insecure by disabling certificate verification.
(Use `node --trace-warnings ...` to show where the warning was created)
20251120T175649 debug oauthDiscovery {
  authorize: 'https://192.168.1.16:6443/realms/MNI/protocol/openid-connect/auth',
  logout: 'https://192.168.1.16:6443/realms/MNI/protocol/openid-connect/logout',
  token: 'https://192.168.1.16:6443/realms/MNI/protocol/openid-connect/token'
}
20251120T175649 debug dynamicPaths { urlPath: '/mni', osPath: '/usr/local/mni/ui/dist' }
20251120T175649 debug dynamicPaths { urlPath: '/mni/css', osPath: '/usr/local/mni/ui/dist/css' }
20251120T175649 debug dynamicPaths { urlPath: '/mni/fonts', osPath: '/usr/local/mni/ui/dist/fonts' }
20251120T175649 debug dynamicPaths { urlPath: '/mni/img', osPath: '/usr/local/mni/ui/dist/img' }
20251120T175649 debug dynamicPaths { urlPath: '/mni/js', osPath: '/usr/local/mni/ui/dist/js' }
20251120T175649 debug dynamicPaths {
  urlPath: '/mni/js/vendor',
  osPath: '/usr/local/mni/ui/dist/js/vendor'
}
20251120T175649 debug dynamicPaths {
  urlPath: '/mni/js/vendor/chartjs',
  osPath: '/usr/local/mni/ui/dist/js/vendor/chartjs'
}

```

#### Note:

- this will use any existing deployed `/etc/mni/mni.ini` environment configuration file, otherwise the existing copy within the component release will be deployed

## 17.2. Example Component Update

```
# ./update.sh
👉 MNI UI Server Upgrade
[18:02:05]
▶ Stopping SystemD service
[18:02:05]
✓ - ok
[18:02:05]
▶ Updating environment
[18:02:05]
✓ - ok
[18:02:05]
▶ Restarting SystemD service[
[18:02:05]
✓ - ok
[18:02:06]
✓ - completed
[18:02:06]
```

#### Note:

- this expects to use the existing deployed `/etc/mni/mni.ini` environment configuration file

## 17.3. Example Component Upgrade

```
# ./upgrade.sh
👉 MNI UI Server Upgrade
[17:59:24]
▶ Stopping SystemD service
[17:59:24]
✓ - ok
[17:59:24]
▶ Updating environment
[17:59:24]
✓ - ok
[17:59:24]
▶ Updating NodeJS package for target
[17:59:24]
✓ - ok
[17:59:24]
▶ Updating NodeJS libraries
[17:59:24]
✓ - ok
[17:59:25]
▶ Updating & Restarting SystemD service
[17:59:25]
✓ - ok
[17:59:26]
✓ - completed
[17:59:26]
```

#### Note:

- this expects to use the existing deployed `/etc/mni/mni.ini` environment configuration file

To upgrade third-party component adjust the respective `APIGW_KRAKEND_VERSION`, `APISERV_DUCKDB_VERSION`, `IAM_KEYCLOAK_VERSION` variable with the required version number within the deployed environment configuration file (`/etc/mni/mni.ini`) prior to executing the upgrade script:

```
APIGW_KRAKEND_VERSION="2.12.0"
APISERV_DUCKDB_VERSION="v1.4.2"
IAM_KEYCLOAK_VERSION="26.4.4"
```

## 17.4. Example Component Removal

```
# ./uninstall.sh
👉 MNI UI Server Uninstall
[17:55:18]
▶ Removing systemd service
[17:55:18]
✓ - ok
[17:55:19]
▶ Removing NodeJS libraries
[17:55:19]
✓ - ok
[17:55:19]
▶ Removing working directory
[17:55:19]
✓ - ok
[17:55:20]
▶ Removing user account
[17:55:20]
✓ - ok
[17:55:20]
✓ - completed
```

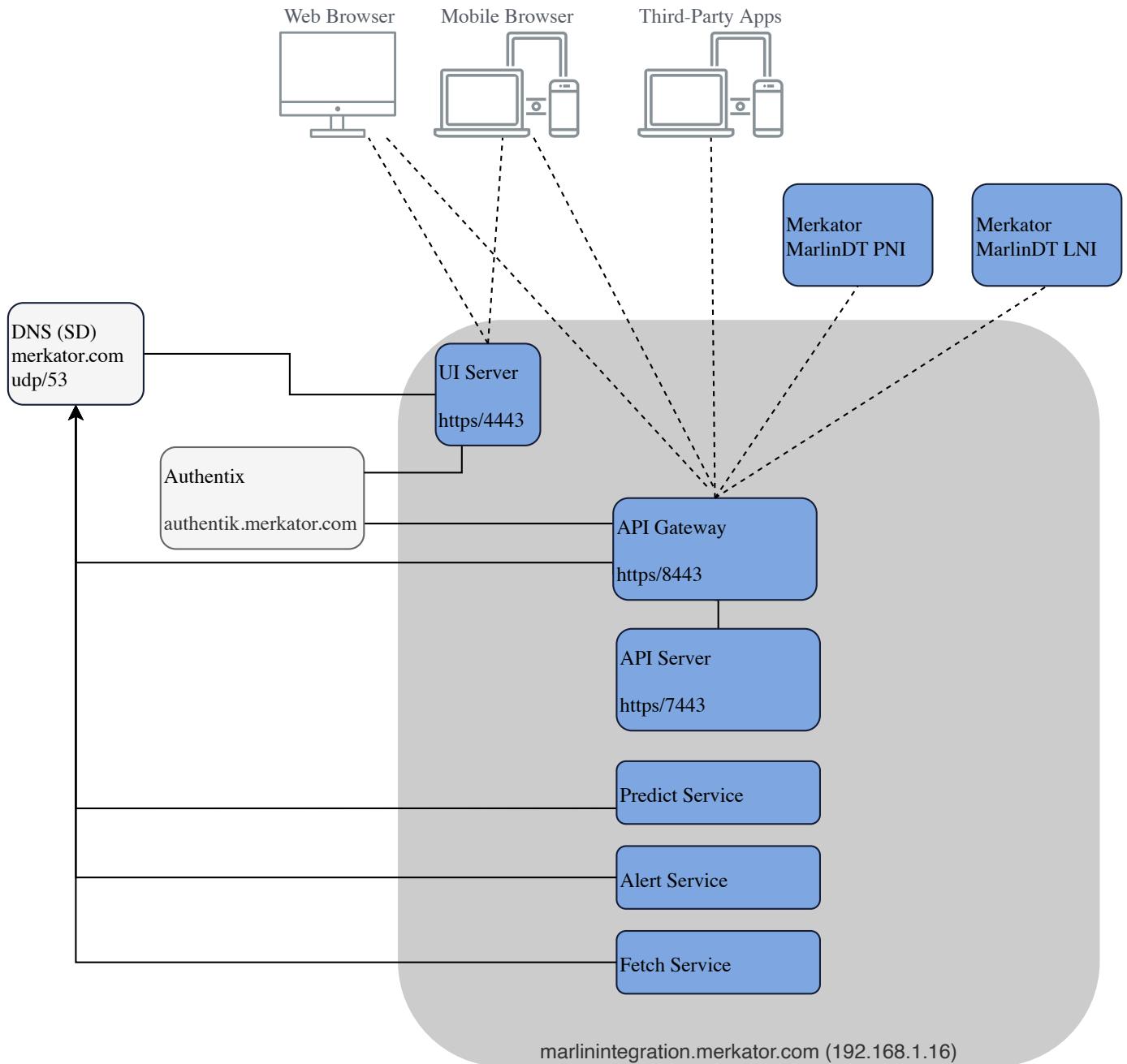
### Note:

- This expects to use the existing deployed `/etc/mni/mni.ini` environment configuration file
- Any deployed SSL certificates and the environment configuration file `/etc/mni/mni.ini` are not automatically removed as may be required for a later installation.

The following directories should be manually removed if MNI is no longer to exist upon a host

- `/etc/mni`
- `/var/log/mni`
- `/usr/local/mni`

## 18. Integration Lab



Integration Lab

```
ssh admin@192.168.1.16
```

**Note:**

- all MNI created host accounts are automatically created, secured and by default have shell access disabled, use `systemctl` to control the running service from a privileged, i.e. `root`, account

## 18.1. Host

- IP: `192.168.1.16`
- Host name: `marlinintegration`
- Domain name: `merkator.com`
- username: `admin`

- password: available from IT Support
- root username: `root`
- root password: available from IT Support

### 18.1.1. DNS Records

Component	DNS Type	DNS Record Name	DNS Record Value
API Server	SRV	<code>_https._tcp.apisrrver.marlinintegration.com</code>	<code>0 10 7443 _https._tcp.apiserver.marlinintegration.merkator.com</code>
API Server	A	<code>apiserver.marlinintegration.merkator.com</code>	<code>127.0.0.1</code>
API Gateway	SRV	<code>_https._tcp.gateway.marlinintegration.com</code>	<code>0 10 8443 _https._tcp.gateway.marlinintegration.merkator.com</code>
API Gateway	A	<code>marlinintegration.merkator.com</code>	<code>192.168.1.16</code>

The following are neither defined and/or not used within this environment:

Component	DNS Type	DNS Record Name	DNS Record Value
IAM Server	SRV	<code>_https._tcp.iam.marlinintegration.com</code>	<code>0 10 5443 _https._tcp.iam.marlinintegration.merkator.com</code>
IAM Server	A	<code>marlinintegration.merkator.com</code>	<code>192.168.1.16</code>
UI Server	SRV	<code>_https._tcp.ui.marlinintegration.com</code>	<code>0 10 4443 _https._tcp.ui.marlinintegration.merkator.com</code>
API Gateway	A	<code>marlinintegration.merkator.com</code>	<code>192.168.1.16</code>

#### Note:

- DNS records for this host are hosted externally from the lab environment, please contact IT Support for further details

### 18.2. User Interface

- Browser URL: [\[URL\]\(https://marlinintegration.merkator.com:4443/\)](https://marlinintegration.merkator.com:4443/)
- username: `lee.cowdrey@merkator.com`
- password: `merkator`
- OS username: `mniui`
- OS group: `mni`
- OS deployed location: `/usr/local/mni/ui`
- OS log location: `/var/log/mni/ui.log`

### 18.3. API Server

- Endpoint Prefix: <https://marlinintegration.merkator.com:7443/mni/v1>
- service username: `af06f2fe515b91efb5f9a6ceddc937343cdb54849c148c2`
- service key:  
`BFmZLMn2VPc+l20ekJL0c+Sx6IF9uhM2nAp34TMGyhQ/shhlz0X1uCYKTqJjzUM1iAw9OHX++MI6CPvB`

- OS username: `mniapi`
- OS group: `mni`
- OS deployed location: `/usr/local/mni/api`
- OS log location: `/var/log/mni/api.log`

## 18.4. API Gateway

---

- Endpoint Prefix: <https://marlinintegration.merkator.com:8443/mni/v1>
- OS username: `mniapigw`
- OS group: `mni`
- OS deployed location: `/usr/local/mni/apigw`
- OS log location: `/var/log/mni/apigw.log`

## 18.5. Alert Service

---

- OS username: `mniaalert`
- OS group: `mni`
- OS deployed location: `/usr/local/mni/alert`
- OS log location: `/var/log/mni/alert.log`

## 18.6. Fetch Service

---

- OS username: `mnifetch`
- OS group: `mni`
- OS deployed location: `/usr/local/mni/fetch`
- OS log location: `/var/log/mni/api.log`

## 18.7. Predict Service

---

- OS username: `mnipredict`
- OS group: `mni`
- OS deployed location: `/usr/local/mni/predict`
- OS log location: `/var/log/mni/predict.log`

---

# 19. Third-Party

MNI is based on open source software and does not include any embedded commercial components that require separate commercial agreements.

## 19.1. Map Rendering

---

By default MNI User Interface makes use of Google Maps and as such requires a Google Maps API key - the credentials can be configured via the MNI Secrets API or the default changed in the API Server database schema definition prior to installation. Without this key the MNI User Interface will **NOT** visualise poles, sites or trenches over topological terrain - cables, ducts, network equipment are not impacted by this.

## 19.2. Foreign Exchange Rates

If enabled, in MNI Fetch makes use of a free service from [exchangeratesapi.io](https://exchangeratesapi.io) which is limited to 100 requests per calendar month. The credentials can be configured via the MNI Secrets API or the default changed in the API Server database schema definition prior to installation.

### Note:

- This can be enabled or disabled by setting the environment configuration file variable `FETCHSRV_FX_UPDATE` to `true` or `false` regardless of what MNI Fetch job tasks may exist
  - MNI uses the same free credentials, therefore multiple installations may cause limits imposed by [exchangeratesapi.io](#) to be reached
  - For each deployment, it is recommended that the customer obtains their own free credentials from [exchangeratesapi.io](#) to avoid hitting any limits.

## **19.3. Common Vulnerabilities and Exposures (CVE) Repository**

If enabled, there are no commercial restrictions for caching the CVE repository. Any further questions or concerns should be directed to [cve.org](http://cve.org).

**Note:**

- Refer to the environment configuration file variable `FETCHSRV_CVE_DIRECTORY` for the cache location.

## 19.4. Sample Data

The API Server includes sample data covering cables, ducts, utility poles, network equipment, racks, sites, services and trenches for the multiple countries.

This data can be added by executing the `sample.sh` script from **within** the API Server installation directory once the API Server has finished installation.

```
# cd mni/apiServer/
# ./sample.sh

█ MNI API Server Sample Data Load [17:35:16]
█ DNS service discovery [17:35:17]
✓ - ok [17:35:17]
█ Checking for API Readiness [17:35:17]
✓ - ok [17:35:17]
█ Adding secret - fx [17:35:17]
✓ - ok [17:35:17]
█ Adding bulk email providers [17:35:17]
✓ - ok [17:35:17]
```

▶	Adding bulk kafka providers	[17:35:17]
✓	- ok	[17:35:17]
▶	Adding bulk map providers	[17:35:17]
✓	- ok	[17:35:17]
▶	Adding bulk workflow engines	[17:35:17]
✓	- ok	[17:35:17]
▶	Adding CVEs	[17:35:17]
i	Loading CVE 2025-21601	[17:35:17]
✓	- ok	[17:35:18]
i	Loading CVE 2025-30649	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding bulk sites	[17:35:18]
i	Loading bulk sites for GBR	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding bulk racks	[17:35:18]
i	Loading bulk racks for GBR	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding bulk trenches	[17:35:18]
i	Loading bulk trenches for GBR	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding bulk poles	[17:35:18]
i	Loading bulk poles for GBR	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding bulk ducts	[17:35:18]
i	Loading bulk ducts for GBR	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding bulk cables	[17:35:18]
i	Loading bulk cables for GBR-1	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding bulk network equipment	[17:35:18]
i	Loading bulk nes for GBR	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding bulk services	[17:35:18]
i	Loading bulk services for GBR	[17:35:18]
✓	- ok	[17:35:18]
▶	Adding document - favicon.ico	[17:35:18]
✓	- ok	[17:35:18]
✓	- completed	[17:35:18]

### Note:

- the `sample.sh` script makes use of `SD-DNS` to locate the MNI API Server
- the `sample.sh` script uses the deployed environment configuration file to retrieve the API Server service credentials
- sample data per country is identified by the filename containing the country identifier, i.e. `cable-GBR-1.json` containing `GBR`
- Any resource dependencies, i.e. a rack requires a site to already exist, should be taken into consideration when loading sample data.
- any large JSON files should be limited to 100 records of the respective resource per file, this is to allow the data to be bulk loaded to the MNI API Server queues without data loss

- the following files are **not** country specific and should **removed** aftet the first execution of the sample script:
  - `CVE-2025-21601.json`
  - `CVE-2025-30649.json`
  - `admin_data.json`
  - `admin_email.json`
  - `admin_kafka.json`
  - `admin_map.json`
  - `admin_workflow.json`
  - `favicon.ico`
  - `secret-fx.json`
- the master copy of the sample files can be found within the repository within `/mni/sample` and the country specific files are:
  - `cable-ARE-1.json`
  - `cable-BEL-1.json`
  - `cable-BEL-2.json`
  - `cable-BEL-3.json`
  - `cable-BEL-4.json`
  - `cable-BEL-5.json`
  - `cable-BEL-6.json`
  - `cable-EGY-1.json`
  - `cable-EGY-2.json`
  - `cable-EGY-3.json`
  - `cable-EGY-4.json`
  - `cable-EGY-5.json`
  - `cable-ESP-1.json`
  - `cable-ESP-2.json`
  - `cable-ESP-3.json`
  - `cable-ESP-4.json`
  - `cable-ESP-5.json`
  - `cable-ESP-6.json`
  - `cable-FRA-1.json`
  - `cable-FRA-2.json`
  - `cable-GBR-1.json`
  - `cable-IRL-1.json`
  - `cable-ISL-1.json`

- [cable-JOR-1.json](#)
- [cable-JOR-2.json](#)
- [cable-JOR-3.json](#)
- [cable-JOR-4.json](#)
- [cable-LUX-1.json](#)
- [cable-PRT-1.json](#)
- [cable-PRT-2.json](#)
- [cable-PRT-3.json](#)
- [cable-SUR-1.json](#)
- [cable-TUR-1.json](#)
- [cable-USA-1.json](#)
- [cable-USA-2.json](#)
- [cable-USA-3.json](#)
- [cable-USA-4.json](#)
- [cable-ZAF-1.json](#)
- [cable-ZAF-2.json](#)
- [duct-ARE.json](#)
- [duct-BEL-1.json](#)
- [duct-BEL-2.json](#)
- [duct-BEL-3.json](#)
- [duct-BEL-4.json](#)
- [duct-BEL-5.json](#)
- [duct-BEL-6.json](#)
- [duct-EGY-1.json](#)
- [duct-EGY-2.json](#)
- [duct-EGY-3.json](#)
- [duct-EGY-4.json](#)
- [duct-EGY-5.json](#)
- [duct-ESP-1.json](#)
- [duct-ESP-2.json](#)
- [duct-ESP-3.json](#)
- [duct-ESP-4.json](#)
- [duct-ESP-5.json](#)
- [duct-ESP-6.json](#)
- [duct-FRA-1.json](#)

- duct-FRA-2.json
- duct-GBR.json
- duct-IRL.json
- duct-ISL.json
- duct-JOR-1.json
- duct-JOR-2.json
- duct-JOR-3.json
- duct-JOR-4.json
- duct-LUX-1.json
- duct-PRT-1.json
- duct-PRT-2.json
- duct-PRT-3.json
- duct-SUR-1.json
- duct-TUR-1.json
- duct-USA-1.json
- duct-USA-2.json
- duct-USA-3.json
- duct-USA-4.json
- duct-ZAF-1.json
- duct-ZAF-2.json
- ne-ESP.json
- ne-GBR.json
- pole-BEL.json
- pole-GBR.json
- rack-BEL.json
- rack-ESP.json
- rack-GBR.json
- service-GBR.json
- site-BEL.json
- site-ESP.json
- site-GBR.json
- trench-ARE.json
- trench-BEL-1.json
- trench-BEL-2.json
- trench-BEL-3.json

- trench-EGY-1.json
- trench-EGY-2.json
- trench-EGY-3.json
- trench-ESP-1.json
- trench-ESP-2.json
- trench-ESP-3.json
- trench-ESP-4.json
- trench-ESP-5.json
- trench-FRA.json
- trench-GBR.json
- trench-IRL.json
- trench-ISL.json
- trench-JOR-1.json
- trench-JOR-2.json
- trench-LUX.json
- trench-MYS.json
- trench-PRT-1.json
- trench-PRT-2.json
- trench-SAU.json
- trench-SUR.json
- trench-TUR.json
- trench-USA-1.json
- trench-USA-2.json
- trench-ZAF.json

## 19.5. World Boundaries

---

The API Server includes the world geometry boundaries of 230 countries, allowing longitude (`x`) and latitude (`y`) coordinates to be matched to a single geographic country - this data is loaded during installation, summarised within the API Server and the raw data subsequently removed.

This data is sourced from [geoBoundaries - an open database of political administrative boundaries](#).

## 19.6. Third-Party Elements & Libraries

---

MNI makes use of third-party JavaScript libraries, all referenced libraries and elements are currently open source.

**Note:** Some of the third-party elements are available under separate commercial terms if additional Enterprise functionality or support is required for a specific deployment.

A highly detailed list is automatically generated through the CI/CD process and is available at [dependencies](#), a summary shown below for reference.

## 19.6.1. API Gateway

Third-Party Package	Used Version	License
krakenD	2.12.0	Apache 2.0

## 19.6.2. API Server

Third-Party Package	Used Version	License
basic-auth-connect	1.1.0	MIT
compare-versions	6.1.1	MIT
compression	1.8.1	MIT
duckdb	1.4.1	MIT
dayjs	1.11.18	MIT
dotenv	17.2.3	BSD-2-Clause
express	5.1.0	MIT
express-fileupload	1.5.2	MIT
express-validator	7.2.1	MIT
js-base64	3.7.8	BSD-3-Clause
morgan	1.10.1	MIT
nocache	4.0.0	MIT
node-cron	4.2.1	ISC
serve-favicon	2.5.1	MIT
software-license-key	1.0.2	MIT
uuid	13.0.0	MIT

## 19.6.3. DNS Server

Third-Party Package	Used Version	License
dnsmasq	2.7.0	GPL

## 19.6.4. UI Server

Third-Party Package	Used Version	License
dayjs	1.11.18	MIT
dotenv	17.2.3	BSD-2-Clause
ejs	3.1.10	Apache-2.0
express	5.1.0	MIT
express-cache-controller	1.1.0	MIT
morgan	1.10.1	MIT
serve-favicon	2.5.1	MIT
uuid	13.0.0	MIT

## 19.6.5. Predict Service

Third-Party Package	Used Version	License
dotenv	17.2.3	BSD-2-Clause
uuid	13.0.0	MIT

## 19.6.6. Alert Service

Third-Party Package	Used Version	License
dotenv	17.2.3	BSD-2-Clause
kafkajs	2.2.4	MIT
node-cron	4.2.1	ISC
nodemailer	7.0.9	MIT-0
uuid	11.0.5	MIT

## 19.6.7. Fetch Service

Third-Party Package	Used Version	License
<code>dotenv</code>	17.2.3	BSD-2-Clause
<code>glob</code>	11.0.3	ISC
<code>mysql2</code>	3.15.3	MIT
<code>node-cron</code>	4.2.1	ISC
<code>oracledb</code>	6.10.0	Apache-2.0 OR UPL-1.0
<code>uuid</code>	13.0.0	MIT

## 19.6.8. License Generator

Third-Party Package	Used Version	License
<code>dotenv</code>	17.2.3	BSD-2-Clause
<code>js-base64</code>	3.7.8	BSD-3-Clause
<code>software-license-key</code>	1.0.2	MIT
<code>yargs</code>	18.0.0	MIT

## 19.7. Development Tooling - EchoAPI

EchoAPI, much like PostMan and others, is a suite of API development tools with features for API design, debugging, testing, and documentation generation, available as a standalone application and extensions for platforms like VS Code, IntelliJ IDEA, and a Chrome extension.

MNI itself does not require nor use EchoAPI, but a collection of MNI OpenAPIs has been provided, and the [export](#) can be imported and used.

**Note:**

- Online help is available on the EchoAPI [product help pages](#)

## 20. MNI OpenAPIs

Are automatically included into the API Server release and made available both via API (`/mni/v1/api`) and also the MNI User Interface (<https://marlinintegration.merkator.com:4443/mni/openapi>).

The MNI OpenAPIs are available in multiple formats including:

- [HTML](#)
- [JSON](#)
- [YAML](#)

**Note:** During deployment of MNI API Server, the YAML definitions are updated to reflect the host and domain names hosting it, before a JSON variant is produced and also stored.

## 21. MNI User Interface (UI)

The MNI UI is loosely based on a tab design and includes:

### 21.1. Login

A typical login page is shown.

Login credentials are validated through the MNI API Server (`/mni/v1/login`) to the Identity and Access Management (IAM) server and if successful, the client is redirected to the readiness page.

The MNI API Server makes use of the included Identity and Access Management (IAM) server, powered by Keycloak, to perform authentication.

**Note:**

- Keycloak can be swapped for any other OAuth2.0/OpenID Connect platform and MNI configured through the environment configuration file, specifically `OAUTH_CLIENT_ID`, `OAUTH_CLIENT_SECRET` and `OAUTH_DISCOVERY_URL`.
- If the Login page displays the OpenID Connect logo, then MNI has been configured for OAuth2.0/OpenID Connect; if not then authentication for users has been effectively **disabled**

### 21.2. Readiness

The readiness page is a transition page, where the client itself performs a MNI API Readiness check through the API Gateway before automatically moving to the Dashboard

**Note:**

- if after 10 attempts, the client is not able to verify readiness, the client is redirected back to the login page.

### 21.3. Country Selector

Most pages contain the ability to filter resources by a specified country.

**Note:**

- the default country code (e.g. `BEL`) can be changed to another during installation or through the MNI OpenAPIs
- the displayed list of country codes depends on what resources exist, for example if a resource existed in `ESP` then both `BEL`, the default, and `ESP` would be in the drop down list

### 21.4. Dashboard

The resource dashboard reflects the quantity of data available within MNI. The shown stacked chart represents the total number of a specific resource but split into active (green) and inactive (orange) as a percentage. It also provides business summary of the resource assets presented in the default country/locale, for example:

`BEL` Belgium

```
Premises Passed: 78 | Trench Coverage: 2Mm | Cable Coverage: 0m  
Costs: (Euro) Cables €40,382 | Ducts €57,950 | Poles €1,170 | NE €34,449.61 | Racks  
€21,996 | Services €0 | Sites €158,000 | Trenches €101,187,243.55
```

`GBR` United Kingdom

```
Premises Passed: 13 | Trench Coverage: 742.55m | Cable Coverage: 0m  
Costs: (Sterling) Cables £132 | Ducts £300 | Poles £270 | NE £34,449.61 | Racks  
£1,476 | Services £0 | Sites £123,000 | Trenches £37,127.50
```

#### Note:

- the dashboard when displayed within the browser, automatically refreshes every 5 minutes

## 21.5. Cables

Attempts to provide a graphical representation of any selected cable incarnation. Supported cable types include:

- Copper
- Coax
- Ethernet
- Fiber
  - Single (multiple strands)
  - Multi (multiple ribbons, each containing multiple strands)

### 21.5.1. Common

- Reference and identifier
- Source:
  - `historical`
  - `predicted`
- Location of cable (duct, pole etc.) all of which can be navigated
- Usage summary:
  - Number of marked as `faulty`
  - Number of marked as `reserved`
  - Number of marked as `used`

- Number of marked as `free`

**Note:**

- in the case of fiber, each ribbon and strand is visualised with its usage shown
- when the incarnation of the resource is `predicted`, the outer border of the resource will be `yellow` in color

## 21.5.2. Coax

- Includes the frequency range of the coax cable and the number of channels that may be operated over it

## 21.5.3. Copper

- Includes the number of twisted pairs that make up the copper cable

## 21.5.4. Ethernet

`ethernet`, while actually a copper cable, is treated as its own cable type due to have fixed use and fixed number of twisted pairs within

- Includes the category supported by the cable:
  - `Cat3`
  - `Cat4`
  - `Cat5`
  - `Cat5e`
  - `Cat6`
  - `Cat6A`
  - `Cat7`
  - `Cat8`
- If originally supplied, includes the maximum supported transmission speed of the cable

## 21.5.5. Fiber

- Includes the supported optical mode of the cable, either single-mode `SF` or multi-mode `MMF`

### 21.5.5.1. Single-Fiber

- Includes the number strands

### 21.5.5.2. Multi-Fiber

- Includes the number of ribbons
- Includes the number of strands *per* ribbon

## 21.6. Ducts

---

Attempts to provide a graphical representation of any selected cable incarnation. Supported duct types include:

- `duct`
- `sub-duct`
- Reference and identifier
- Source:
  - `historical`
  - `predicted`
- Purpose of the duct:
  - `gas`
  - `power`
  - `cable`
  - `water`
- Location of duct (duct or trench), all of which can be navigated
- Configuration of duct 1/2/3/5/6/7/8/10/12/14/19/24-way up to 48-way
- List of the contained cables within the duct, all of which can be navigated
- Usage summary:
  - Number of marked as `faulty`
  - Number of marked as `reserved`
  - Number of marked as `used`
  - Number of marked as `free`

**Note:**

- in the case of fiber, each strand is visualised with its own usage shown
- when the incarnation of the resource is `predicted`, the outer border of the resource will be `yellow` in color

## 21.7. Poles

---

Attempts to provide a graphical representation of selected utility pole over a geographic terrain.

If supplied or derived, the height and area type classifier of the pole will be shown. The classifier would be one of:

- `residential`
- `rural`
- `industrial`
- `mixed` - combination of residential/industrial/commercial.
- `psz` - Public Safety Zone
- `urban`

- `unclassified` - not recorded or known
- `utility` - public service building, i.e. hospital, school

**Note:**

- Selected pole identifier appears as tooltip
- Zoom In/Out via mouse-wheel, scroll by dragging terrain map
- Right-clicking on the shown pole icon would display list of any connected trenches (navigatable)
- Left-clicking on the shown pole icon would display list of any connected sites (navigatable)

## 21.8. Network Equipment

---

Attempts to provide a graphical representation of any selected network equipment in a friendly vendor-agnostic fashion.

**Note:**

- network equipment types are constrained by the supplied data
- if costing is to be applied, a cost should be added via the MNI API Server to match the vendor and model of the equipment:
  - Vendor `Cisco Systems, Inc.`
  - Model `C8300-1N1S-4T2X`
  - Cost `7199.61`
- The cost does not include a currency as it will default to the system configured default locale.
- Graphic representation of each data port contained within this network equipment, each box will show:
  - port number (tooltip for interface name)
  - port status:
    - Number of marked as `faulty`
    - Number of marked as `reserved`
    - Number of marked as `used`
    - Number of marked as `free`
  - port transmission speed (tooltip)
  - error count of the port (tooltip)
  - error indicator:
    - `tick`: 0 errors
    - `exclamation`: between 1 and 99 errors
    - `cross`: 100 or more errors
- Reference and identifier
- Source:
  - `historical`

- `predicted`
- When it was commissioned
- Optionally, when it was decommissioned
- Management IP Address
- Vendor
- Model
- Running software (firmware) image
- Version of running software (firmware) image
- Location based on a specified site, rack and rack slot(s)
- Common vulnerabilities and Exposures (CVE) summary:
  - Number of CVEs that would **impact** this vendor model equipment running the known software (firmware) image
  - Number of CVEs that are known against this vendor model equipment running the known software (firmware) image
- Usage summary:
  - Number of ports marked as `faulty`
  - Number of ports marked as `reserved`
  - Number of ports marked as `used`
  - Number of ports marked as `free`
  - Total number of errors reported against this network equipment

**Note:**

- Out-of-band and other connectivity ports not used for data are not included
- CVE information will only be matched and displayed if enabled through MNI Fetch Service
- when the incarnation of the resource is `predicted`, the outer border of the resource will be `yellow` in color
- Like a Configuration Management Database (CMDB), MNI includes the ability and hold and store the network equipment configuration (any textual format) for the specific incarnation of the network equipment

## 21.9. Racks

---

Attempts to provide a graphical representation of any selected role in a agnostic fashion.

**Note:**

- network equipment types are constrained by the supplied data
- if costing is to be applied, a cost should added via the MNI API Server to match the number of slots:
  - Slots: `12`
  - Cost: `120`

- The cost does not include a currency as it will default to the system configurated default locale.
- Graphic representation of each slot contained within this rack, each slot will be shown:
  - Usage status:
    - Number of marked as `faulty`
    - Number of marked as `reserved`
    - Number of marked as `used`
    - Number of marked as `free`
  - If any network equipment is defined against this slot, it will be listed and navigatable
- Reference and identifier
- Source:
  - `historical`
  - `predicted`
- When it was commissioned
- Optionally, when it was decommissioned
- Dimensions (in units originally specified i.e. `cm`, `mm`, `m`, `km`, `Mm`, `inch`, `feet`)
  - Height
  - Weight
  - Depth
- Location based on a sepecified site, floor, row and coordinates

**Note:**

- when the incarnation of the resource is `predicted`, the outer border of the resource will be `yellow` in color

## 21.10. Sites

---

Attempts to provide a graphical representation of selected site over a geographic terrain.

**Note:**

- MNI supports `on-net` and `off-net` sites
- a toggle is included to change what sites are available to be visualised

If supplied or derieved, the site area and type classifiers will be shown. The area classifier would be one of:

- `residential`
- `rural`
- `industrial`
- `mixed` - combination of residential/industrial/commercial.
- `psz` - Public Safety Zone
- `urban`

- `unclassified` - not recorded or known
- `utility` - public service building, i.e. hospital, school

The site type classifier would be one of:

- `adu` - accessory dwelling units
- `colo` - colocation
- `commercial` - place of work, retail unit etc.
- `dc` - data center
- `exchange` - district/town exchange building etc.
- `mdu` - multi-dwelling unit
- `pop` - point of presence
- `sdu` - single dwelling unit
- `street` - various street furniture options
- `unclassified` - not recorded or known

#### **Note:**

- Selected site identifier appears as tooltip
- Zoom In/Out via mouse-wheel, scroll by dragging terrain map
- Right-clicking on the shown pole icon would display list of any connected trenches (navigatable)
- Left-clicking on the shown pole icon would display list of any connected poles (navigatable)
- Additional `off-net` sites can be loading during installation, added via MNI OpenAPIs or MNI Fetch used to retrieve from a third-party
  - Using additional site data allows further complex predictions to occur

## **21.11. Services**

---

Attempts to provide a graphical representation of selected service, including ingress/egress ports on which network equipment.

#### **Note:** incomplete

## **21.12. Trenches**

---

### **21.12.1. Detail**

Attempts to provide a graphical representation of selected trench over a geographic terrain.

If supplied or derived, the total length, number of premises passed (calculated) will be shown.

#### **Note:**

- Selected trench identifier appears as tooltip
- Zoom In/Out via mouse-wheel, scroll by dragging terrain map
- Right-clicking on the shown pole icon would display list of any connected ducts (navigatable)

- Left-clicking on the shown pole icon would display list of any connected poles (navigatable)

## 21.12.2. Lifetime

Attempts to provide a graphical representation of selected trench over a geographic terrain for its entire lifetime

If supplied or derived, the total length, number of premises passed (calculated) will be shown.

### Note:

- Selected trench identifier appears as tooltip
- Zoom In/Out via mouse-wheel, scroll by dragging terrain map
- When the incarnation of the resource is `predicted`, the predicted parts of the trench will be shown as a yellow dashed line
- Not implemented: Right-clicking on the shown pole icon would display list of any connected ducts (navigatable)
- Not fully implemented: Left-clicking on the shown pole icon would display list of any connected poles (navigatable)

## 21.12.3. Per Country

Attempts to provide a graphical representation of all trenches defined against a specific country over a geographic terrain.

If supplied or derived, the total length, number of premises passed (calculated) will be shown.

### Note:

- Selected trench identifier appears as tooltip
- Zoom In/Out via mouse-wheel, scroll by dragging terrain map
- Right-clicking on the shown pole icon would display list of any connected ducts (navigatable)
- Left-clicking on the shown pole icon would display list of any connected poles (navigatable)

## 21.13. Quote-to-Cash (Q2C)

Attempts to provide a graphical canvas that allows a new trench path to be drawn. Select existing trench to navigate to intended build area.

Costs exist to provide high level cost quotes for different trench uses, these include:

- `service/drop` to support typical cabling for site/endpoint connection
- `ring` to support typical cabling used in metro rings
- `backhaul` to support typical cabling used in backhaul purposes
- `pole` to support typical cabling required for a utility pole
- `tower` to support typical cabling required for a cell tower
- `unclassified`

**Note:**

- The selected currency can be changed at any time, click on the calculator icon if you change the selected currency
- The intended purpose of the trench can be changed at any time, click on the calcultor if you change the intended purpose
- The trash can icon can clear any previously drawn trench
- The save icon will only appear once the calculation has been performed but currently does nothing else
- The Forward icon will only appear once the calculation has been performed and will export the drawn trench and additional MNI attributes to the client browser, the export will be:
  - **KML** formatted
  - Have a filename of `mni-{new trench UUIDv4}.kml`
  - Can be verified by importing into Google Earth:
    - Open the Google Earth app on your computer or browser [Google Earth](#)
    - For a new: and click Open Local KML file > Import file to project.
    - For an existing project: Go to the home screen, click the project you want to import to, and then click File > Import file to {Project name}.
    - Choose the MNI exported file and Click Upload from device to select a file from your computer
    - Select the KML file you want to import. It will automatically be displayed
- Right-clicking will removed the last added point from the trench path
- Left-clicking will add each the selected point as the trench path
- Zoom In/Out via mouse-wheel, scroll by dragging terrain map

## 21.14. Providers

---

The provider dashboard reflects the number and type of providers available within MNI.

The provider dashboard sub icon bar will now make available further detailed provider pages for:

- Email
- Map
- Publish (Kafka)
- Workflow Engines

### 21.14.1. Email

Allows providers to be added, modified and deleted and only supports SMTP gateways.

### 21.14.2. Map

Allows providers to be added, modified and deleted.

## 21.14.3. Kafka

Allows Kafka buses to be added, modified and deleted.

## 21.14.4. Workflow Engines

Allows workflow engines to be added, modified and deleted and only supports Camunda (BPMN) and ELSA endpoints.

## 21.15. Alerts

---

The alerts dashboard reflects the number and type of defined alert capabilities available within MNI.

The alerts dashboard sub icon bar will now make available further detailed pages showing active alerts for:

- Callback
- Publish (Kafka)
- Email
- Workflow

**Note:** UI aspects have not been implemented

## 21.16. Settings

---

The settings page allows the `historical` and `predicted` durations to be managed, refer to [21.2. Data Longevity](#) for further details.

## 21.17. OpenAPIs

---

The embedded Swagger Editor, from the vendor [Smartbear](#) allows users and developers to interactively make use of the included MNI Open APIs. The Open API definition is loaded from the MNI API Server and once any authentication has occurred makes all APIs available.

**Note:**

- due to SSL Certificates issues within the Integration Lab, most clients will have to have previously accepted (trusted) the certificates used by Merkator.
- Perform a [HTTP GET /api/readiness](#) in a separate browser tab and explicitly accept the certificates prior to proceeding to continue
- At this point you should receive a JSON response with a timestamp indicating that MNI API Server is accepting connections

```
{"point": "20251121T152211"}
```

## 21.18. Help/About

---

Currently lists the included third-party JavaScript libraries.

## 21.19. Quit (Logout)

---

While there is not an actual page for this, the client performs a logout through the MNI API Server (`/mni/v1/logout`) and any cached authentication data is removed.

The client is then redirected to the login page.

## 22. MNI API Server Internals

### 22.1. Database

MNI API Server does not include a traditional time-series database (`TSDB`) and instead implements a time-series layer over a generic high performance analytical database engine, **DuckDB**.

The design rationale behind this was to allow the used database technology to be replaced at a later date or to meet specific deployment requirements without requiring any other architectural or API changes - this is referred to as **loosely-coupled** APIs.

#### Note:

- See [Why DuckDB](#) for details
- Any potential alternative database technology **must** include support for NodeJS/JavaScript
- MNI API Server also makes use of the following DuckDB extensions:
  - [Spatial Extension](#)

### 22.2. Geometry

MNI API Server geometries are represented using geometric constructs defined as lists of one or more two-dimensional `xy`, or three-dimensional `xyz` points, rather than other GIS Well-Known Text (`WKT`) geometric point systems. Support for two or three dimensional geometries is dependant on the specific need of each of the MNI OpenAPIs:

- `x` (longitude) typically a float value, is measured along east-west axis with a range of `-180` to `+180` with scale `3` and precision `6`
- `y` (latitude) typically a float value, is measured along north-south axis with a range of `-90` to `+90` with scale `2` and precision `6`
- `z` (altitude) typically a float value, is measured height or elevation, scale `8` and precision `2`
- `m` typically a string value, is extra axis of information not associated with the cartesian x/y/z space

#### Note:

- as long as the use of these values is consistent, MNI does not itself impose further restrictions or conditions but care should be taken with any implemented Map provider.

Some of the MNI OpenAPIs include the option to supply `WKT` geometries instead, if supplied these are converted to the above system during processing

#### Note:

- MNI API Server can convert WKT points from such objects containing `LINestring`, `POINT`, `MULTIPOINT`

- Supports EPSG4326 (PNI) to EPSG3857
- GIS WKT support should be considered experimental, see [ESP](#) sample trench data

## 22.3. Data Longevity

MNI API Server automatically prunes data outside of a defined rolling window. This window range can be defined through the MNI UI and APIs.

It covers `historical` and `predicted` data separately and be defined as a specific quantity of time:

- Number of
- Unit of duration being:
  - `second`
  - `minute`
  - `hour`
  - `day`
  - `week`
  - `month`
  - `quarter`
  - `year`

By default the rolling window is constructed from the current date and time with +/- of:

- `historical 1 year`
- `predicted 6 month`

### Note:

- any `historical` data older than the start of the rolling window is removed
- any `predicted` data newer then the end of the rolling window is removed
- any `predicted` data earlier than the last piece of `historical` data is removed
- any and all `planned` data is left untouched
- data removal occurs on a configurable but frequent basis
- a sensible unit of duration should be applied, nothing smaller a week is advised

## 22.4. Database Backup

Scheduled via the MNI environment configuration file, the API Server will export the internal database to the local filesystem hosting the MNI API Server. Any encrypted data, in the secret store or elsewhere is maintained as encrypted even in the export.

### Note:

- see `APISERV_BACKUP_DIRECTORY` or the API Server log file for directory selected
- see `APISERV_DUCKDB_BACKUP_CRONTIME` for the scheduled backup time

- see `APISERV_DUCKDB_BACKUP` for toggling backup operations
- Above the manual import procedures provided by DuckDB itself, MNI does not itself include the ability to recover from a previously generated backup

## 22.5. Tracking Changes

---

MNI components make use of queues maintained and/or provided through the API Server. This allows for distribution of activities as well as coordination.

All write-related data operations handled by MNI API Server are tracked through two queues:

1. `predictQueue`
2. `replicationQueue`

The queues record on a temporary basis, the resource type, resource identifier and the related state (CRUD). Once the respective MNI components has retrieved a queue entry and processed it **successfully**, the MNI component calls back through the Open API to mark the entry as complete (`delete = true`).

For MNI Predict Service, it reads the queue through the Open APIs and examines each resource for the purpose of prediction; this can be simple duplication for such resources that only have a single incarnation (i.e. a site), simple for those with sequentially incarnations (i.e. a rack slot usage) and complex for those with sequentially incarnations across multiple intersects.

If cluster replication is enabled, any slave MNI API Servers will subscribe to an encrypted multicast stream that the master MNI API Servers publishes into to determine what resources have changed.

**Note:** while the multicast streams and role handling (`master` vs. `slave`) are implemented, the additional logic to request the change has not yet been implemented.

## 22.6. MNI Fetch

---

In a similar fashion to tracking changes, MNI Fetch Service jobs are also maintained by the API Server. The MNI Fetch Service requests the list of jobs on startup and executes them on a scheduled basis. See `schema.sql` `INSERT INTO fetchJob...` for details.

The fetch job definitions include:

- CVE Pull
- CVE Impact List
- Foreign Currency (FX) Currency Update
- Importing from PNI (not implemented)
- Importing from LNI (not implemented)

**Note:**

- all fetch job tasks must exit in the MNI Fetch Service prior to being supported. MNI Fetch also include a no operation function (`noop`) as a default.
- the fetch job definition can includes dynamic parameters which allows reusability.
- any required secrets/credentials from the MNI API Server secret store

## 22.7. Alert Service

---

In a similar fashion to tracking changes, MNI Alert Service jobs are also maintained by the API Server. The MNI Alert Service requests the list of jobs on startup and executes them on a scheduled basis. See `schema.sql` `INSERT INTO alert...` for details.

The purpose of MNI Alert Service is to perform analysis on resources and if action is required to generate a notification than can be subsequently despatched through one or more multiple channels to the intended recipient.

**Note:**

- all alert job tasks must exit in the MNI Alert Service prior to being supported - this should at least be a *stub* function.
- MNI Alert also include a no operation function (`noop`) as a default.
- the alert job definition can include dynamic parameters which allows reusability.
- any required secrets/credentials from the MNI API Server secret store

---

## 23. MNI Alert Internals

---

MNI Alert retrieves its list of jobs from the MNI API Server using the `/alert` APIs.

**Note:**

- MNI Alert is therefore data driven

---

## 24. MNI Fetch Internals

---

MNI Fetch retrieves its list of jobs from the MNI API Server using the `/fetch` APIs.

**Note:**

- MNI Fetch is therefore data driven

---

## 25. MNI Predict Internals

---

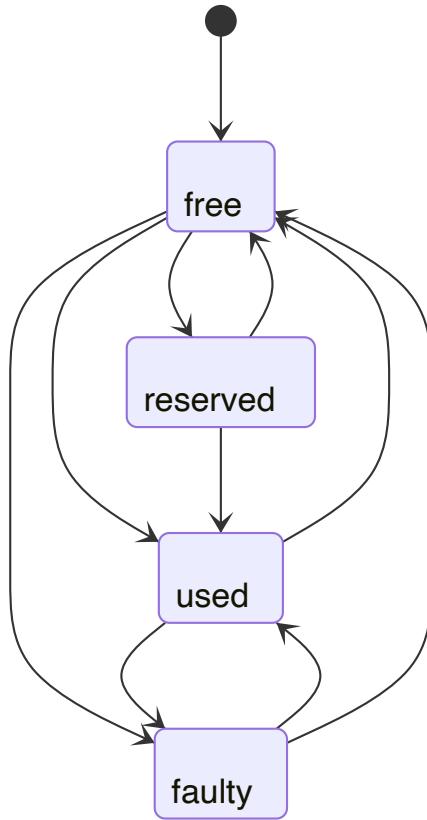
### 25.1. Duplication

Where resources only exist with a single incarnation, MNI simply produces a duplicate of the resource with the predicted marker and passes it back to MNI API Server for storage, as MNI can not predict any changes without external influence.

### 25.2. Simple Prediction

Where resources exist with multiple incarnations, such as rack slots, cables, network equipment ports, ducts, the resource is treated with simple prediction with regards to usage.

The historical state/usage of the resource is taken into consideration for what the predictive change will be:



For example, the probability a `faulty` state would transition to `reserved` without first transitioning to `free` is low in comparison to the other possibilities.

For the mathematical prediction to occur, the state/usage is converted to a numeric value between 0 and 1 to allow for Machine Learning (ML) practices to be followed and once the predicted value has been determined, transformed back to the original format representing the state/usage.

## 25.3. Complex Prediction

---

It expected to build upon the simple prediction by means of utilising Long Short-Term Memory (LSTM) networks.

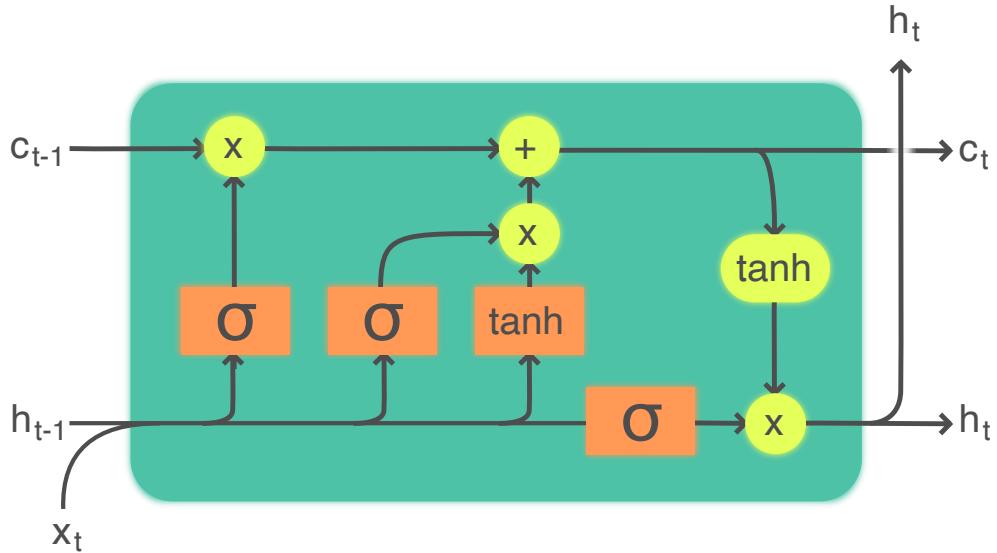
LSTM networks are a powerful type of recurrent neural network (RNN) specifically designed for time series prediction because they can learn and remember patterns over long sequences of data. This ability helps them overcome the vanishing gradient problem that traditional RNNs face, allowing them to capture complex, non-linear dependencies in historical data.

An LSTM network's core is the memory cell, which acts as a long-term storage unit for information, and several "gates" that regulate what information is stored, updated, or output at each time step.

- Forget Gate: Decides which information from the previous cell state is no longer relevant and should be discarded. A value close to 0 means "forget completely," while a value near 1 means "keep entirely".
- Input Gate: Determines which new information from the current input is important and should be added to the cell state. This helps the network decide what new data points are significant enough to be remembered.
- Output Gate: Controls which parts of the current cell state are exposed as the output (or hidden

state) at the current time step. This filtered information is used to make predictions and passed to the next unit.

- Cell State: The "memory" that runs through the entire sequence, selectively updated by the gates at each step to carry relevant information forward across potentially long time intervals.



## Legend:

Layer	Componentwise	Copy	Concatenate

### 25.3.1. Advantages for Time Series Prediction

- Captures Long-Term Dependencies: LSTMs are effective in scenarios where events in the distant past influence future outcomes, a key challenge for traditional models like ARIMA.
- Handles Non-Linearity: They can model complex, non-linear relationships and patterns often found in real-world time series data (e.g., financial markets, weather data).
- Learns from Raw Data: LSTMs automatically extract relevant features from raw sequential data, reducing the need for manual feature engineering and data stationarity requirements (as is needed with ARIMA models).
- Robustness: They are relatively robust to noise and irregularities in data due to their gating mechanisms which help filter out irrelevant information.

## 26. MNI User Interface Internals

### 26.1. Distribution Directories

MNI UI will dynamically detect and expose contained directories within the UI content

## 26.2. Metadata

---

Via the UI Server `/metadata` URL, additional metadata is passed to the client for caching. This includes:

- build: MNI build number
- country: MNI default country
- currency ISO Code: MNI default country currency code
- currency Name: MNI default country currency name
- currency Symbol: MNI default country currency symbol
- gateway URL: MNI API Gateway URL
- icon: MNI User Interface favourite icon filename
- name: MNI application name
- notificaton Icon: MNI User Interface notification icon filename
- readiness Attempts: MNI configured number of attempts for readiness checks
- refresh Ms: MNI configured refresh delay in milliseconds
- retry Ms: MNI configured general client refresh delay in milliseconds
- root URL: MNI User Interface root URL
- short Name: MNI short application name
- version: MNI version number

**Note:**

- the above allows for dynamic rebranding of the MNI UI to happen if required

---

## 27. Additional Q&A

### 27.1. Business / Functional / Sales Context

---

What are the business analyses, functional analyses, or sales requirements that formed the basis for this solution?

MNI's purpose was to provide analytical insight into both physical and logical data associated with service infrastructure and assist providers with the challenges that may be faced: "Any modern telecommunication business is constantly faced with finding current accurate information but also attempting to plan, coping with the ever changing "What if ?" questions from their own employees in addition to both customers and external market influences"

What was the initial question or request from sales that led to building the prediction functionality?

You can discuss this with the Sales team, but "MarlinDT Network Intelligence (MNI) does not replace the fine-grain capabilities of either MarlinDT PNI or LNI, but serves to help you make decisions or, at the very least, streamline your company's decision-making processes by collecting, visualising, and communicating your data associated with your telecommunications infrastructure and services. ". For high-level summary analysis, questions included "how many premises do we pass", "how much fiber is in the ground" etc.

Where are the analyses, insights, and capacity-planning features meant to be derived from this platform?

"MarlinDT Network Intelligence uses time-series data powering rich real-time visualizations and an interactive dashboard including the reporting of analytical trends and predictions."

The published APIs include the exposed features, the UI itself just attempts to visualise some of them. MNI is broken up into components, the API Server does the storage and related items, Alert undertakes things like capacity analysis and the despatches notifications, Fetch interacts with external systems, Predict undertakes the predictive actions

Data Integration (Marlin ↔ MNI)

How do you provide MNI with actual data from Marlin?

"Existing customers of MarlinDT can deploy an additional probe to push PNI and LNI metadata to MarlinDT Network Intelligence for processing. Those currently without MarlinDT PNI and/or LNI products, can generate and send their metadata to MarlinDT Network Intelligence for processing. MarlinDT Network Intelligence makes use of industry leading tmforum open APIs to allow rich, faster and more flexible integration to existing OSS/BSS". For the integration lab, Kris was kind enough to supply of marlin database (as has been mentioned multiple times), interesting data was manually extracted (no assistance was provided by the Marlin team so database structure had to be manually reversed) and prepared for MNI processing.

MNI is meant to be fed from Marlin and NetworkMining, or any other external system which our customers may use (unfortunately not everyone uses Marlin or NetworkMining), this work while added to Jira at the start of 2025 was never undertaken by the respective teams.

MNI Fetch was included instead to be able to fetch data from systems such as Marlin, however Marlin has NO APIs and requires direct database access (oracle) and while MNI Fetch can support this the requested goal at the time (from Luc) was to import and display all of the Spain (ESP) trench data. So for customer deployments it is safer/easier for Marlin and NetworkMining to push data to MNI. Marlin can send daily whatever is has, MNI will record any changes and store rather than just storing everything again.

How does MNI know which objects in the Marlin datamodel are cables, ducts, etc.?

the same way Marlin does, the necessary SQL statements include appropriate WHERE and JOIN clauses

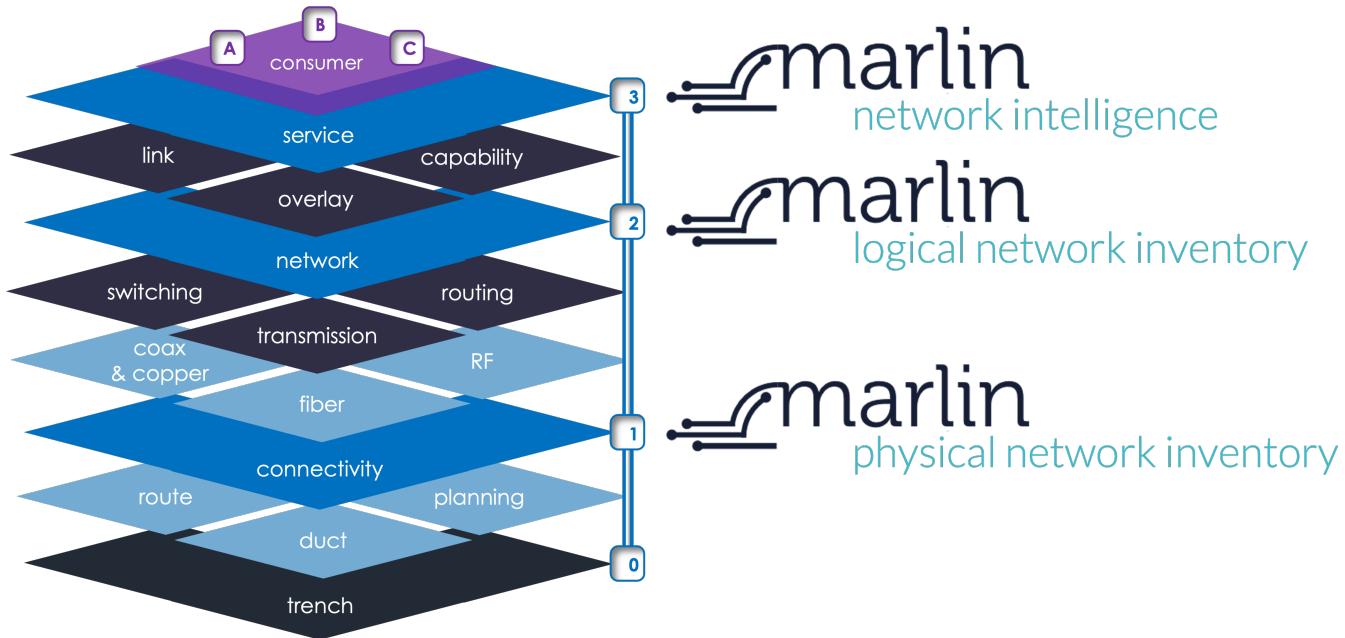
## 27.2. Feature Identification & Versioning

How does MNI identify the same feature across different time slices, given that MNI uses UUIDs and Marlin uses integers?

MNI is not Marlin nor is it NetworkMining.

MNI is a Business Support System (BSS), Marlin and NM are Operational Support Systems (OSS) - so MNI is few levels higher up in the solution space and that means it does not hold/process data at the same fine-grain level as Marlin/NM - THIS IS BY INTENT as intelligence and insights are from multiple sources of data which is correlated, aggregated and summarised to give a broader perspective. For example, Marlin knows about select physical items and it knows them very well but above and beyond that scope it knows nothing - this is referred to as fine-grain.

Perhaps you have previously seen the diagram I created representing the "Stack of Knowledge", working from layer 0 upwards:



In our industry no product directly uses the same identifiers as another, though for MNI it is part of the APIs on the resources includes a reference attribute which holds the external identity. MNI also has the ability internally to correlate data records, so system A has identifier 123 and B has identifier XYZ, MNI can hold this data against one of its own identifiers but that is a MNI Fetch capability.

MNI use of Universally unique identifier (UUID) was simply I needed something, that was universal. MNIs use of UUID can actually be removed so that any data can be used, so you could use Marlin integers but I would NOT recommend if you are using multiple external systems for obvious reasons

## 27.3. Data Relationships

How does MNI store relationships such as:

See above areas of this document, but summary answers below for completeness

- network connectivity
- infrastructure connectivity
- associations between objects?

MNI's APIs and data model are built around domain knowledge, e.g. a cable could be slung between utility poles, or running in a duct, which could be in a trench, or running around a rack

When you add a resource, like a rack, you have the option to not only give the rack a geometry location (X,Y,Z), row and/or column for a data centre floor BUT to also assign it to something else. In the case of a rack you can assign it to a site. A site can be anything/anywhere from a cabinet on the street, a box on top of public lighting, cell site and everything else including traditional buildings

So every MNI resource can have association for/to another resource but it is not mandatory - with MNI "the more you put in the the more value you can get out"

- Versioning, Deltas & Time Management

So you are referring to the time-series data, again this is in the readme and been covered many times including the fact we do not use on-the-shelf time-series database here

#### How are deltas detected between time slices?

There are no time slices, no forced snapshots, no duplication of data to force a time perspective, there is just time and when a resource is created, updated and even deleted (but never removed) that is recorded by means of literally creating a new incarnation of it - in the readme file you will see this listed as "Shadow Time-Series Tables". The resource identifier is separate from the resource detail, for example a trench, the trench id is what everything else interacts with, but the shadow table holds the detail of what/where (etc.) the trench is

#### How does MNI store snapshots of moments in time?

There are no snapshots, no moments in time, think of it the other way around, there is the resource and something happens to it over its lifetime, when something happens, and importantly, MNI is informed or fetches the change, then something is updated.

#### Does each time slice contain a complete record, or are only deltas stored?

The changed resource detail is held as new incarnation, the resource identifier is still the same single thing. For example, if you extended an existing trench, the trench resource detail include the new incarnations, the extra bit of trench and the existing trench, which are timestamped. For performance reasons the new incarnation is complete. When you just update a resource, e.g. network equipment port is used, you literally send in the network equipment identifier with port identifier and its new usage state (i.e. used), MNI itself takes the entire latest incarnation is knows, applies the change as a difference and then stores the result. This helps for performance and other logic

MNI is a BSS not OSS and does not hold the same fine-grain data for PNI as Marlin, so what you see documented in the APIs IS the data that is basically stored. MNI is meant to add value to Marlin and NetworkMining, NOT detract from them, we want customers to buy and use the entire product portfolio.

#### How does MNI handle the very large data volumes if regular snapshots (e.g., daily) of the full network are taken?

The system in the integration lab contains data from about 12 countries, 2 mega metres of trenches , costs and other stuff. The database is currently standing at 100Megabytes (yes Mb not Gb nor Tb). The MNI APIs were designed to hold just the data that is required and this answers the next part of this question MNI does not like Wyre Snowflake (and others?) literally extract and store **all** data every 24 hours to get fake a time-series but it just stores data WHEN it changes. So if a site is defined and nothing happens to it then it only ever exists once in time, **not** duplicated once per 24 hours etc. See below on more comments on the TSDB

The Wyre Snowflake architecture, a typical TSDB, extracts everything on a reoccurring basis and stores it. It then tries to work out what changed but generally just uses it to determine statistics, i.e. number of connections made in last 7 days. With MNI if you want to know about something call the API (or use the API), the API will return the last known change regardless of when and when there are multiple changes over the resource lifetime then you can specify the "point" in time to get the previous, current or predict incarnation of the resource. Take a rack, when its installed on a site, it starts empty, over time the rack slots are filled, when the rack eventually becomes full (no spare slots), it is either replaced with a new different bigger rack or additional racks are added. MNI allows you to literally scroll through the lifetime of that specific rack

usage changes, it is stored once and only ever once as it is the same physical rack, the changes (so the detail) is what is extended (or duplicated if that is easier to understand) over time. Where something refers to the rack, it refers to the single rack and not the rack detail.

## 27.4. Prediction Functionality

What does “prediction” mean in this context?

Where MNI holds the lifetime of a resource (past and current), it also attempts to guess (given a predict is literally an educated guess) what it will be at some point in the future.

Taking our simple rack, and the rack slots, if we see from the past a free rack slot is consumed every month, then if the rack had originally 24 slots then in 2 years the rack will be full. Life is not that simple of course, even for the simple rack. Slots can be consumed, equipment then removed and the slot becomes free only for a bigger piece of network equipment to be put in to the rack and double the number of slots consumed.

Taking a piece of network equipment, which has 24 ports, customer services are connected, backhaul connected, customer service disconnected etc. etc.

MNI Predict handles these scenarios currently and they are referred to as simple predictions. This is documented in the readme, but MNI Predict examines, say checks every slot on a rack, looks at its history and mathematically calculates what it could be. It then sends the change back to the MNI API Server literally like a normal update but now it contains a prediction marker with a timestamp of the future rather than the historical marker. The MNI UI (and APIs) understand these markers and visualise the results differently, below is the UI showing the usage of a historical fiber strands from a single fiber cable:

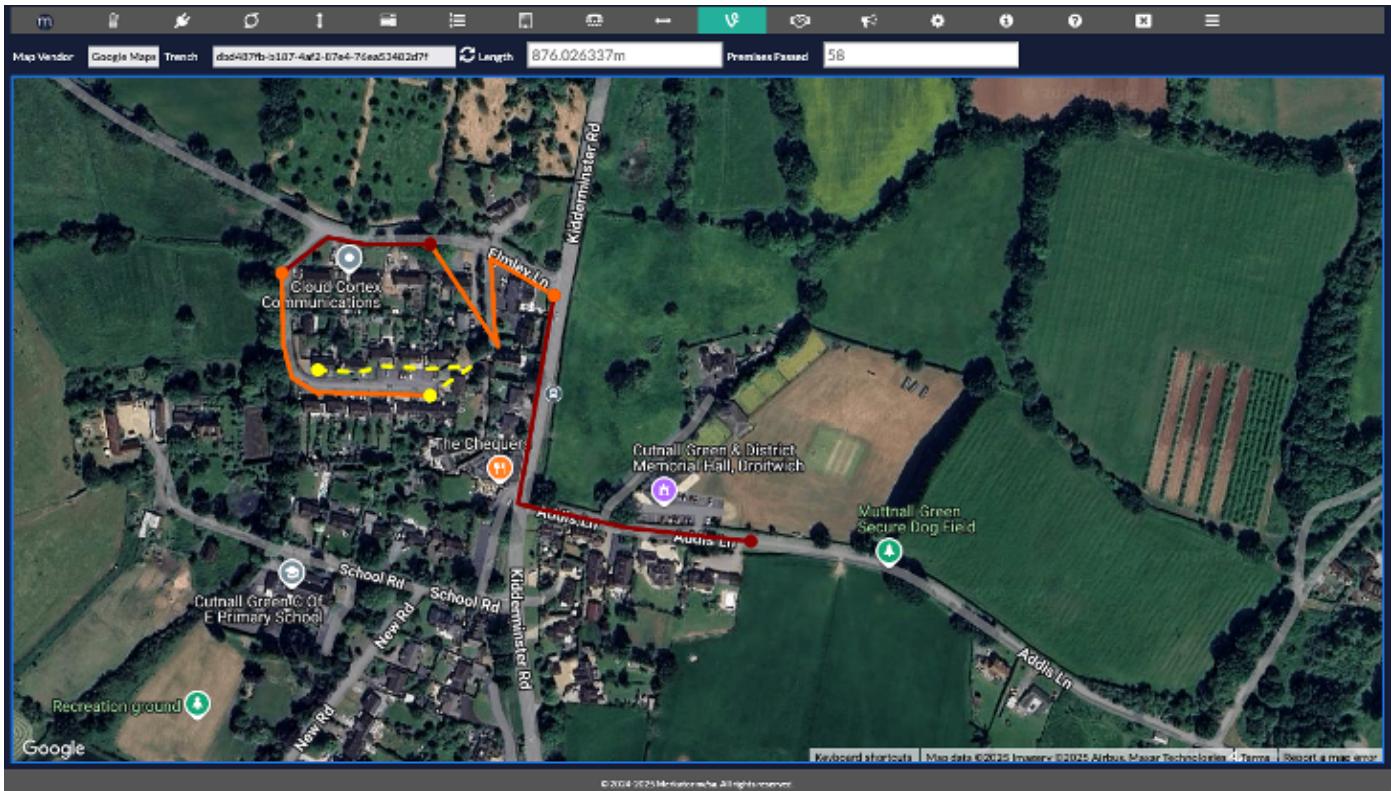


If we had selected the predicted version, the colour scheme would show the predicted changes, after prediction (yellow indicates predictive) note changes to strands 18, 28, 38, 42,43,48 etc:



The more changes MNI has been made aware of, the higher the accuracy of the prediction. When it comes to resources like sites, the predictive capabilities are limited, as a site does not typically (in MNI terms) change, but this is where the next type of prediction comes in.

**Complex** predictive capabilities are associated across multiple resources, so if a trench runs from point A to point B and MNI has been made aware that a new housing/business estate is being built (made aware by looking at off-net site data, either passed to MNI directly or MNI Fetch collecting from say BelMaps), then we can see our trench runs geographically close to the new estate and could predict the trench will need to be extended to reach the new estate. Here is an example of that via the UI, the yellow dashed line represents a predicted trench extension and the red and orange lines represent that the expansion (build out) of a given trench over its lifetime.



What is the logic or method behind the predictions?

Right now only **simple** predictions are implemented for cables, racks, network equipment ports, as never received the previously requested access to BelMaps in order to load in more off-net data to start working on the complex scenarios.

Every resource in MNI (cable, duct, pole, NE, rack, service, site and trench) has the lifetime capability, the predictive use cases require Merkator Marlin and NetworkMining teams coming together bringing their respective domain knowledge gathered from studying the patterns in the data. Collectively Merkator could identify many *what-if* questions that would form some of the complex predictions - Luc is good for this for business level *what-if's*.

Another note, MNI Predict is currently switched off in integration lab due to where development was but can be switched back on by enabling/starting the MNI Predict service

You had previously asked about capacity, MNI Alert needs further development to complete this but is simply taking a collective of a resource, adding up what is used and subtracting that from what the maximum is. All the functions in MNI Alert exist in stub form but this was suspended in favour of the predictive aspects of MNI (and lack of resources).

## 27.5. Architecture / Technology Choices

Could the functionality currently provided by MNI have been achieved more simply using a time-series database (TSDB)?

No.

An off the shelf TSDB requires larger infrastructure, commercial licenses (something MNI does not require for any of the technology it uses), requires vast preprocessing, forces times intervals across all data and to common time periods (hourly, daily), creates vast amounts of data it then has to store through complex data compression.

The database MNI uses is NOT a TSDB, the MNI API Server adds and maintains all the time-series data. Many reasons but it makes it lean, efficient and more importantly it separates the logic from the storage. So MNI could change the backend database technology to another database product **without** breaking anything else or forcing users or other applications to have to change - this is known as a loosely-coupled architecture and is core to a micro services/cloud-native architecture.

MNI data changes when the provider makes a change, extends or adds a trench, runs a new cable, connects a port. If nothing has happened to the resource it **does not** just duplicate it against a time period (i.e. a day), as it's the same resource incarnation, nothing has changed, so nothing is stored.

If your data was changing frequently, for example weather conditions, you can take a sample every second, or minute or hour or day and this is where a TSDB comes in to play - conversely MNI data is not constantly changing. This is where you hear the phrase *Data Lake*, vast amount data which very quickly turns in to a *Data Swamp* as the data value degrades due to the cost/time of processing it and more importantly its meaning in a relationship.

---

## 28. Merkator Technical Services

---

For technical support, you can contact Merkator by email, or on the web.

### 28.1. By email

---

Customer Service Management may be reached at: [support@marlindt.net](mailto:support@marlindt.net).

### 28.2. On the web

---

Please visit [support.marlindt.net](mailto:support.marlindt.net). To obtain access, you must provide your support contracted email address.

### 28.3. Technical documentation

---

The latest technical documentation for products is now available on the [Customer Service Management](#). Make sure you have an account to access the content.

---

## 29. Corporate Headquarters

---

Merkator nv/sa · Vliegwezenlaan 48 · 1731 Zellik · Belgium

[+3223092112](tel:+3223092112)

[www.merkator.com](http://www.merkator.com)

**MarlinDT Network Intelligence (MNI) © 2024-2025 Merkator nv/sa**

— — | — — — | — — — | — — — — — | — — — / / — — — |  
\	/ \_ \ ' \_	/ / \_ '	\_ / \_ \ ' \_	' \_ \ / / / \_	/ \_ \ '							
			\_ /		< (		( )					\ v / \ \_ \ (
\_	\_ \ \_	\_	\_ \ \_ ,	\_ \ \_ /	\_	\_	\_ \ /	\_	\_	\_ \ /	\_	\_ \ ,