

WHITEPAPER

How DDI Solutions Power Telco Data Center Operations



1. Executive Summary

Every phone call, video stream, and connected sensor in a mobile network eventually touches a building full of servers: the **telco data center**. Inside that building, before any application can run and before any subscriber can connect, three questions must be answered for every single device:

1

Who are you, and what network settings should you receive?

— answered by DHCP

2

Where is the service you are looking for?
— answered by DNS

3

Which addresses are in use, by whom, and for how long?

— answered by IPAM

And a fourth question, asked later and just as often: *who held which address, at exactly what time?* — answered by the record the other three leave behind. That record turns out to underpin security investigations, audits, and even billing.

Together these three are called **DDI** (DNS, DHCP, and IP Address Management). In most data centers they are run as three separate silos, by separate tools and sometimes separate teams. The result is drift, gaps, manual spreadsheets, and outages caused not by hardware failure but by two systems disagreeing about a single address.

ManageEngine DDI Central merges the three silos into one policy-driven control plane. In a telco context, that unlocks capabilities that are difficult or impossible with siloed tools: fully isolated tenants that can reuse identical address ranges without conflict; devices that are identified by their "fingerprint" before they are given an address; a DNS that gives different, correct answers to the same question depending on who is asking; 400 bare-metal servers that install themselves; new 5G network slices brought to life in under 60 seconds through an API call; and a four-layer security shield wrapped around it all — plus the timestamped ledger that billing, audit, and dispute resolution quietly depend on.

The core message: when IP, DNS, and DHCP are managed as one system, the network stops being something you configure by hand and becomes something you program. DDI Central is the control plane that makes that possible.

2. The World Behind Your Phone Signal

A story to begin with

It is 8:40 on a Monday morning in Chennai. Priya steps off her train, opens her phone, and starts a video call with her project team. The call connects in under a second. Video is crisp. Nobody thinks about it — and that is the point. An enormous amount of coordinated machinery just worked flawlessly, invisibly, in milliseconds.

Somewhere between the third ring of a call connecting and the first frame of video appearing, a device asked a question — *where do I belong, and what is my name here?* — and something in a rack of servers answered it in less time than it takes to blink. Most people never think about that moment, and they should not have to. For the engineers, architects, and operators who build the infrastructure underneath modern telecommunications, that moment is the job.

Her phone spoke to a nearby cell tower. The tower handed her traffic to a regional site, which carried it into a large facility on the edge of the city: the **telco data center** — the building where a telecom operator (a "telco") runs the brains of its network. If the mobile network were a railway, the towers would be the stations and the data center would be the central control room, the depot, and the headquarters rolled into one.

What is a telco data center?

A data center, in general, is a secure building full of servers — powerful computers that run software around the clock. A telco data center is a special kind: it does not just host websites or company applications; it runs the mobile network itself. The software that decides whether your SIM card is allowed on the network, that routes your video call, that meters your data plan — all of it lives here.

The easiest way to understand the building is to walk through it zone by zone, the way the operations team does. Think of it as a small, highly organized city with distinct districts:

How a Telco Data Center Is Set Up

A simple view of the systems that keep telecom services running

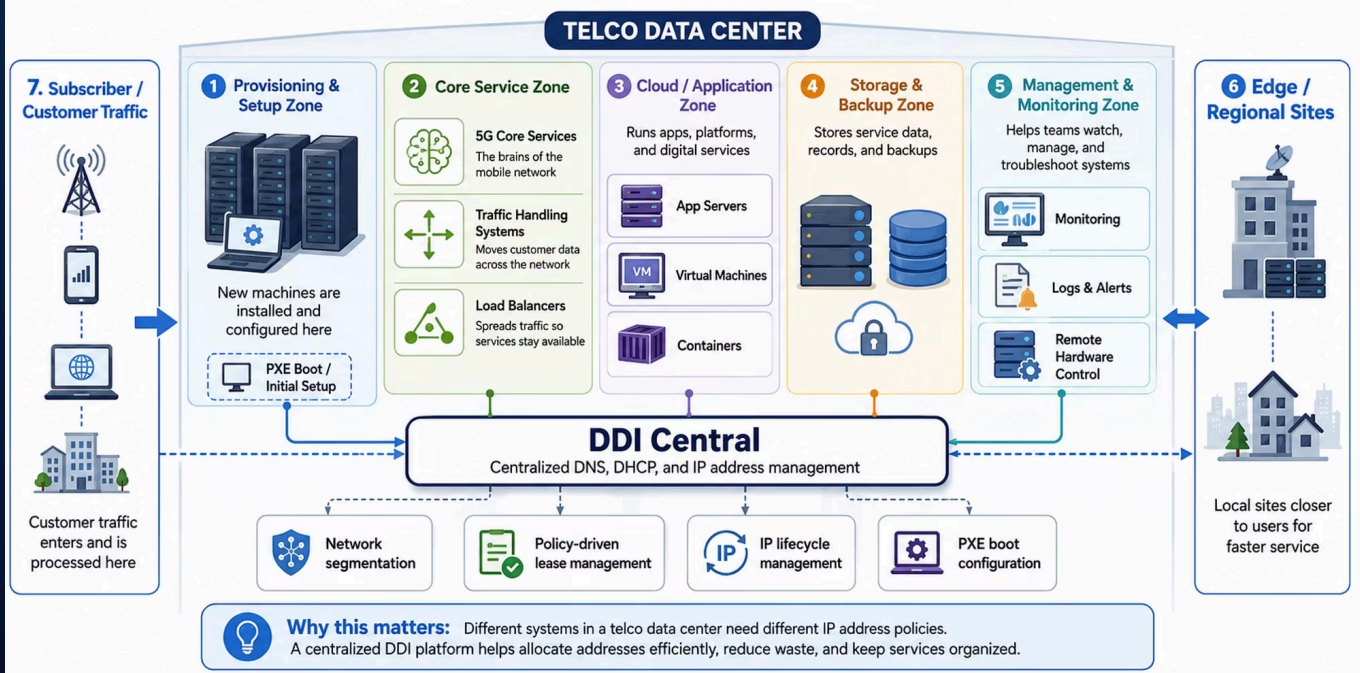


Figure 1 — A simple view of the systems that keep telecom services running — with DDI Central as the unified control plane beneath all of them.

Zone 1 — The Provisioning & Setup Zone (the loading dock and orientation room)

New machines arrive here, are racked, powered on, and configured for the first time. In modern telcos this happens with almost no human touch, using a mechanism called **PXE boot** (pronounced "pixie") — a way for a brand-new, empty server to ask the network, "What software should I install on myself?" We will see later that DDI Central is the system that answers that question.

Zone 2 — The Core Service Zone (the brain)

This is where the **5G core** runs — the software heart of the mobile network. It authenticates subscribers, manages their sessions, applies their data plans, and moves their traffic. It also houses traffic-handling systems and **load balancers** — traffic policemen that spread work across many servers so no single one is overwhelmed and services stay available.

Zone 3 — The Cloud / Application Zone (the office district)

Modern telco software is built like modern cloud software: as small pieces packed into **containers** (lightweight, standardized software boxes) running on platforms such as **Kubernetes** or **OpenStack** — orchestration systems that start, stop, and move those boxes automatically across hundreds of servers, alongside traditional **virtual machines** (software-defined computers running on shared hardware).

Zone 4 — The Storage & Backup Zone (the archive)

Subscriber records, service data, call detail records, and backups live here. If anything in the city burns down, this district holds the copies.

Zone 5 — The Management & Monitoring Zone (the control room)

Dashboards, logs, alerts, and remote hardware controls that let engineers watch and repair everything — often without walking to the racks. Telcos call this operational traffic **OAM** (Operations, Administration, and Maintenance). A crucial security principle: this district must be strictly separated from customer traffic. The keys to the city must never travel on the same roads as the public.

Zone 6 — Edge & Regional Sites (the suburbs)

Smaller sites placed physically closer to users so that latency-sensitive services respond faster. Not every request should travel to the main city; some are best answered around the corner.

Zone 7 — Subscriber Traffic (the citizens)

Finally, the reason it all exists: the phones, laptops, sensors, and machines of millions of subscribers, whose traffic enters, is processed, and leaves — every second of every day.

3. Telco Data Center — Architecture

The city walk is how you would describe the building to a visitor. Engineers draw the same building a second way — as a stack of horizontal bands, each resting on the one below. It is worth learning to read this diagram properly, because it is the picture that appears in nearly every telco architecture discussion. Once you can read it top to bottom, you can follow almost any conversation about where a problem lives.

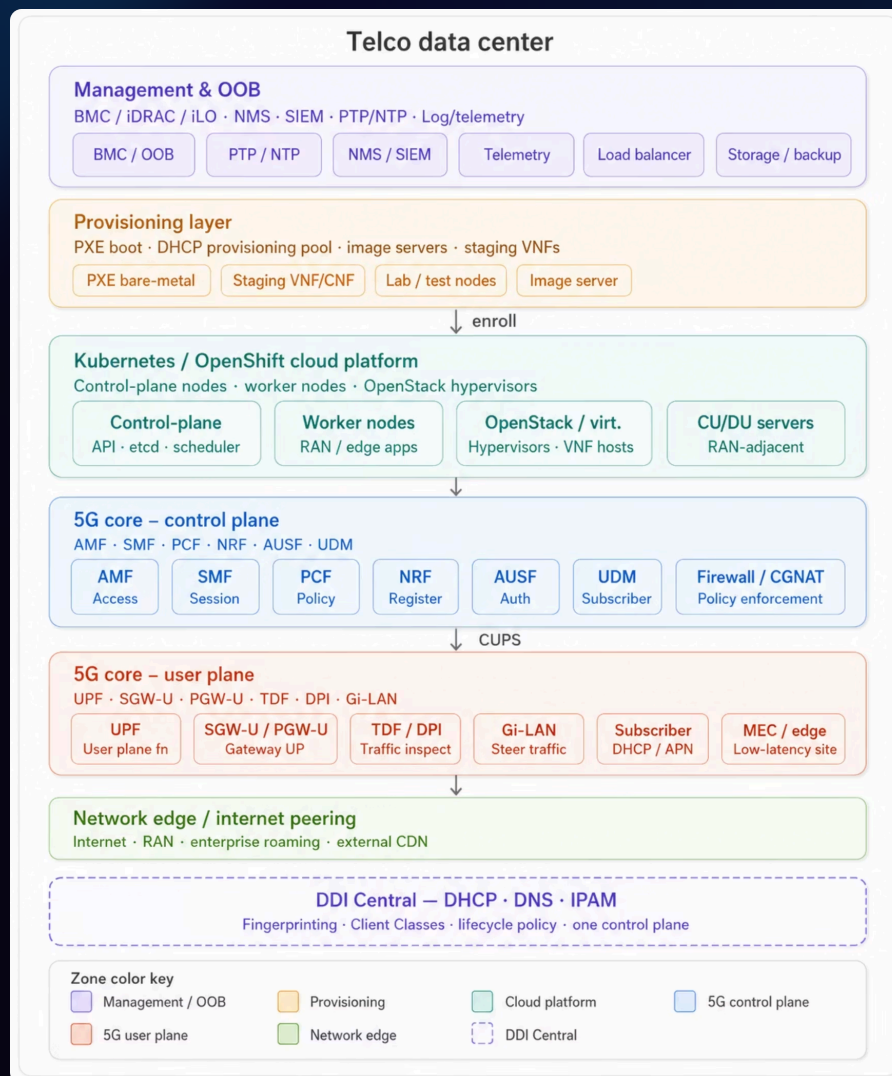


Figure 2 — The same building as an engineer sees it: management and out-of-band, provisioning, the cloud platform, the 5G control and user planes, and the network edge — with DDI Central underneath them all.

Band 1 — Management & OOB: the control room

OOB (out-of-band) is a separate management path that stays reachable even when the main network is broken. Think of the service lift that keeps running when the main lifts are down — you need a way in precisely when everything else has failed. Inside this band: **BMC** (baseboard management controller) and its vendor-branded equivalents **iDRAC** (integrated Dell Remote Access Controller) and **iLO** (integrated Lights-Out, HP's version) are the small always-on computers built into every server that let an engineer power-cycle or reinstall a machine remotely. **NMS** (network management system) watches equipment health; **SIEM** (security information and event management) collects and correlates security logs. **PTP** (Precision Time Protocol) and **NTP** (Network Time Protocol) are the two time-synchronisation protocols that keep every clock aligned — mobile networks are uniquely dependent on precise time, because radio equipment must agree on timing to within microseconds. Telemetry, load balancers, and storage or backup complete the band.

The operational rule for this band is simple and absolute: it must never share addressing with subscriber traffic. The keys to the building do not travel on the public roads.

Band 2 — Provisioning: the loading dock

Where new capacity is prepared before it is trusted with production work. **PXE** (Preboot Execution Environment) bare-metal is blank hardware installing itself over the network; the **DHCP** (Dynamic Host Configuration Protocol) provisioning pool is the temporary set of addresses those machines borrow while they do it; **image servers** hold the operating system images they fetch; and staging **VNF** (virtualised network function) and **CNF** (containerised network function) nodes — network software delivered as virtual machines and as containers respectively — sit alongside lab and test nodes, where that software is rehearsed before it goes live.

Notice the small arrow leaving this band, labelled **enroll**. That is the moment a machine stops being inventory and becomes part of the platform below.

Band 3 — Kubernetes / OpenShift cloud platform: the office district

That is the layer that actually hosts telco software. **Control-plane nodes** run the cluster's own brain — its **API** (application programming interface), its scheduler, and **etcd** (the cluster's distributed key-value store: the name comes from the Unix /etc configuration directory, made distributed), the small database holding the cluster's memory of what should be running where. **Worker nodes** are the machines that run the real workloads, including **RAN** (radio access network) and edge applications. **OpenStack hypervisors** host the older-style virtual machines that many network functions still ship as. And **CU** (centralised unit) and **DU** (distributed unit) servers are the compute that sits closest to the radio equipment, doing the digital signal work for cell sites.

Band 4 — 5G core, control plane: the decision-makers

This is the band that decides *whether* and *how* you may use the network. **AMF** (Access and Mobility Management Function) registers your device and follows it as it moves. **SMF** (Session Management Function) creates and manages your data session. **PCF** (Policy Control Function) applies policy: your plan, your speed, your entitlements. **NRF** (Network Repository Function) is the register of which network functions currently exist and how to reach them — a directory service for the core itself. **AUSF** (Authentication Server Function) authenticates you. **UDM** (Unified Data Management) holds the subscriber data your **SIM** (subscriber identity module) is checked against. And Firewall / **CGNAT** (carrier-grade network address translation, **NAT** being the rewriting of addresses as traffic crosses a boundary) enforces policy at the edge of the core; CGNAT is what lets very large numbers of subscribers share a much smaller pool of public internet addresses — a direct consequence of IPv4 scarcity, and one of the reasons disciplined address management matters so much here.

The arrow leaving this band carries a label worth knowing: **CUPS** (control and user plane separation). It is the architectural decision to split the deciding from the carrying, so each can be scaled — and placed — independently. It is precisely why the next band down can be pushed out to edge sites near users while the decision-makers stay centrally in the core.

Band 5 — 5G core, user plane: the carriers

This is the band that actually carries your data. **UPF** (User Plane Function) is the main mover. **SGW-U** (Serving Gateway — User plane) and **PGW-U** (Packet Data Network Gateway — User plane) are the gateway components inherited from 4G that still carry traffic in mixed networks. **TDF** (Traffic Detection Function) and **DPI** (deep packet inspection) identify what kind of traffic is flowing, which is how operators apply policy and charging correctly. **Gi-LAN** (the local area network sitting on the Gi interface — Gi being the standards-defined boundary between the mobile core and external packet networks) is where operators insert their own services, such as filtering or caching, to steer traffic. **DHCP** (across all bands) is the address engine — the named gateway a device asks for when it connects) where subscriber devices get their addresses. And **MEC** (multi-access edge computing) is the low-latency compute placed near users.

Band 6 — Network edge and internet peering: the front door

This is where the operator's network meets everyone else's: the public internet, the **RAN** (radio access network — the towers and antennas), enterprise and roaming partners, and external **CDNs** (content delivery networks — the caches that hold popular content close to users). **Peering** is simply the set of arrangements by which two networks agree to exchange traffic directly.

3.1. The Role of a DDI solution

Now look at how DDI solution is drawn into the network. It is not another band in the stack; it is a dashed strip running the full width beneath all of them. That is a deliberate piece of visual grammar, and it is the single most important thing on the diagram.

Every band above it needs addresses. Most of them need DHCP. All of them need DNS. And the operations team needs to know, at any moment, what is connected, where it is running, and which zone it belongs to. Management nodes, staging servers, cluster workers, core functions, user-plane components, edge sites — different jobs, different policies, different isolation requirements, but the same underlying need. Fingerprinting, client classes, and lifecycle policy are how DDI Central serves each band differently from one control plane.

Remember it this way: When you read the stack top to bottom and it answers six questions — who watches, who builds, who hosts, who decides, who carries, who connects. Then read the dashed band underneath: whatever the layer, whatever the decade, it still needs an address and a name. If the 5G core is the brain and the user plane is the highway, DDI Central is the address system and the map.

3.2. Zooming in: inside the 5G core, and the two places DDI controls

The stack view shows where DDI sits relative to everything else. Now zoom into the busiest floor of the building – the 5G core itself – and watch the actual conversations. This is the diagram people mean when they say "architecture deep dive," and it looks intimidating until you know the three things to look for.

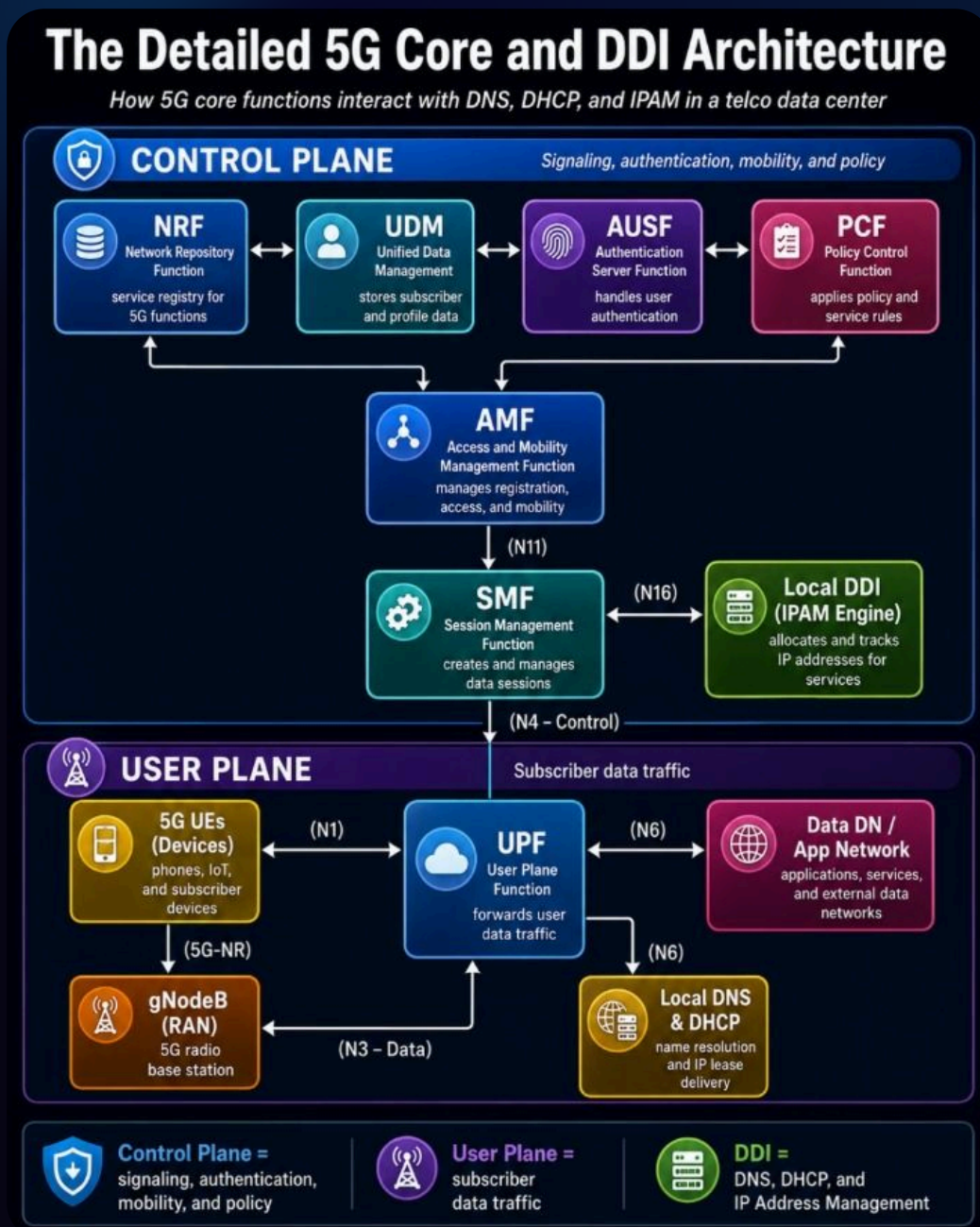


Figure 3 – How 5G core functions interact with DNS, DHCP, and IPAM. Blue is the control plane, purple is the user plane, and – the detail that matters – the two green boxes are DDI: one in each plane.

First thing to look for: the diagram has two halves

The top half, in blue, is the **control plane** — signalling, authentication, mobility, and policy. It is the half that decides. The bottom half, in purple, is the **user plane** — it carries actual data traffic. It is the half that carries things. That split is the CUPS principle from the previous diagram, drawn out in detail. Everything above the dividing line is deciding; everything below it is moving.

Second thing to look for: the **N-labels** are doors, not obstacles

Every arrow carries a label like N1, N3, N4, N6, N11, or N16. These are **reference points** — the standardised, named interfaces 3GPP defines for specific 5G functions. You do not need to memorise them. What matters is *why they exist*: because each door is standardised, a telco can buy the AMF from one vendor and the UPF from another and still have them interoperate. Three are worth recognising on sight:

- **N3** carries subscriber data from the radio base station up into the UPF (User Plane Function) — the pipe your video call actually travels through.
- **N4** is how the SMF (Session Management Function) in the control plane instructs the UPF in the user plane. It is the CUPS link made concrete: the decider telling the carrier what to do.
- **N6** connects the UPF outward to data networks, applications, and services — the exit to the wider world.

Third thing to look for: DDI appears twice, once in each plane

This is the whole point of the diagram, and it is easy to miss. Look for the two green boxes. In the control plane sits **Local DDI (IPAM Engine)**, which allocates and tracks IP addresses for services. In the user plane sits **Local DNS & DHCP**, which handles name resolution and IP lease delivery. Same platform, two placements, two different jobs — because the two planes need different things at different moments.

In the control plane, the question is *"what address should this new session get, and is it recorded?"* The SMF asks the IPAM engine at the moment a session is created, and the allocation is written to the master registry instantly. In the user plane, the questions are *"where is this name?"* and *"what is my lease?"* — asked constantly, by every device, and answered at wire speed. Note the word **Local** on both boxes: these engines are co-located with the core.

Now follow one phone through the whole diagram

Here is the entire picture as a single story. Everything below happens between you pressing a button and the screen responding:

1. The phone – a **UE** (user equipment) in the diagram's language – wakes up and reaches the **gNodeB** (the 5G radio base station) over **5G-NR** (5G New Radio, the radio link itself).
2. The **AMF** (Access and Mobility Management Function) takes registration. Behind it, the **AUSF** confirms the SIM is genuine, the **UDM** supplies the subscriber profile, the **PCF** says which plan and service rules apply, and the **NRF** is how all of these found each other in the first place – it is the registry of which functions exist and where.
3. The AMF hands over to the **SMF** to create a data session.
4. The SMF asks the **Local DDI IPAM engine** for an address. It allocates one, records the allocation, and the master registry is current from that instant – no reconciliation, no spreadsheet.
5. The SMF programs the **UPF** over N4 with the rules for that session.
6. Data flows: phone → gNodeB → UPF over N3 → out to the **Data DN** (data network) and application services over N6.
7. And every time an app on that phone looks up a service name, **Local DNS** answers from inside that slice's own view – sub-millisecond, in-building, and only ever returning addresses the slice is authorised to reach.

Steps 4 and 7 are DDI Central. Remove them and the diagram still has a brain and a highway – but nothing has an address, nothing has a name, and nobody can tell you what is connected where.

Remember it this way: the control plane decides, the user plane carries, and DDI shows up in both. Upstairs it answers *"what address does this session get?"*, downstairs it answers *"where is that name, and what is my lease?"* Two questions, two placements, one source of truth.

4. The Three Core Network Services That Run Everything

Every device in our city — every server, container, phone, and sensor — needs an **IP address**: a unique numeric label such as 10.10.14.82 that identifies it on the network, exactly the way a postal address identifies a house. No address, no communication. With 500,000 or more devices in a single facility, three questions must be answered continuously, correctly, and fast. Each question has a dedicated service.



Figure 4 — DDI as a stack: IPAM is the foundation (which addresses exist and who owns them), DHCP sits above it (who gets one, and with what settings), and DNS sits on top (what name points where).

DNS — the directory that knows where everything is

DNS (Domain Name System) decides *where users and workloads go*. Humans and applications use names — "video.example.com" or "upf.core.local" — but the network only understands numeric addresses. DNS is the directory service that translates names into addresses, like a phone book or directory assistance: you ask for a name, it returns the number. DNS organises names into **zones** (sections of the directory it is responsible for), stores them as **records** (individual entries), and can apply **policies** — rules about who receives which answer. Hold on to that last idea: it becomes one of DDI Central's superpowers.

DHCP — the front desk that hands out the keys

DHCP (Dynamic Host Configuration Protocol) decides *who gets on the network and what configuration they receive*. Picture a hotel front desk. A guest arrives; the receptionist checks them in, assigns a room, hands over the key, and explains where breakfast is served. DHCP does the same for devices: when a phone or server joins the network, DHCP assigns it an IP address (the room), plus its essential settings — the **gateway** (the exit that leads to the rest of the network), the **VLAN** (which "floor" of the network it belongs to), DNS server addresses, and even boot instructions for new machines. The assignment is a **lease**: temporary, with a defined duration, after which it must be renewed or returned — just like a hotel stay.

IPAM — the land registry that keeps the master ledger

IPAM (IP Address Management) maps *identity and ownership* of every address. If DHCP is the front desk and DNS is the directory, IPAM is the city's land registry: the authoritative ledger recording every plot — which addresses exist, which are allocated, reserved, or free, who owns each one, how heavily each neighbourhood (**subnet**, a contiguous block of addresses) is utilized, and the full life story of every address from allocation to retirement. Without IPAM, teams track addresses in spreadsheets — and spreadsheets drift, collide, and lie.

4.1. The problem: three services, three silos

In most data centers these three services run independently — different tools, different owners, different sources of truth. The front desk does not talk to the registry; the directory is updated by hand. The consequences are predictable: an address is leased by DHCP but marked free in the spreadsheet; a DNS record points at a server that was decommissioned last month; a new subnet overlaps an old one and traffic silently goes to the wrong place. Engineers spend their days reconciling systems instead of improving the network.

4.2. The answer: DDI — three services, one control system

DDI is simply the industry term for DNS + DHCP + IPAM operated as a single, integrated system. In an integrated DDI platform, one action updates all three views atomically: when DHCP grants a lease, IPAM records it instantly and DNS learns the name — automatically. Policies, automation, and audit apply across all three at once. There is one truth, not three approximations of it.

Remember it this way: DHCP is the front desk (who gets in, and what key), DNS is the directory (where things are), IPAM is the land registry (who owns what). DDI merges all three into one control tower. **ManageEngine DDI Central** is that control tower, purpose-built for telco scale.

The industry has standard names for the main slices — worth memorizing, because they appear in every telco conversation:

Slice	Full name	Plain meaning	Typical users
eMBB	Enhanced Mobile Broadband	The high-bandwidth lane — maximum throughput	Video streaming, AR/VR, consumer smartphones
URLLC	Ultra-Reliable Low-Latency Communications	The ambulance lane — tiny, guaranteed delays (down to ~1 ms)	Industrial robots, autonomous vehicles, remote control
mMTC	Massive Machine-Type Communications	The scooter lane — huge numbers of low-power devices	IoT sensors, smart meters, smart cities
Enterprise / Private	Private slice	The gated lane — isolated, policy-controlled	A factory, campus, or business customer

What is 5G Network Slicing?

One physical 5G network, multiple virtual slices optimized for different services.

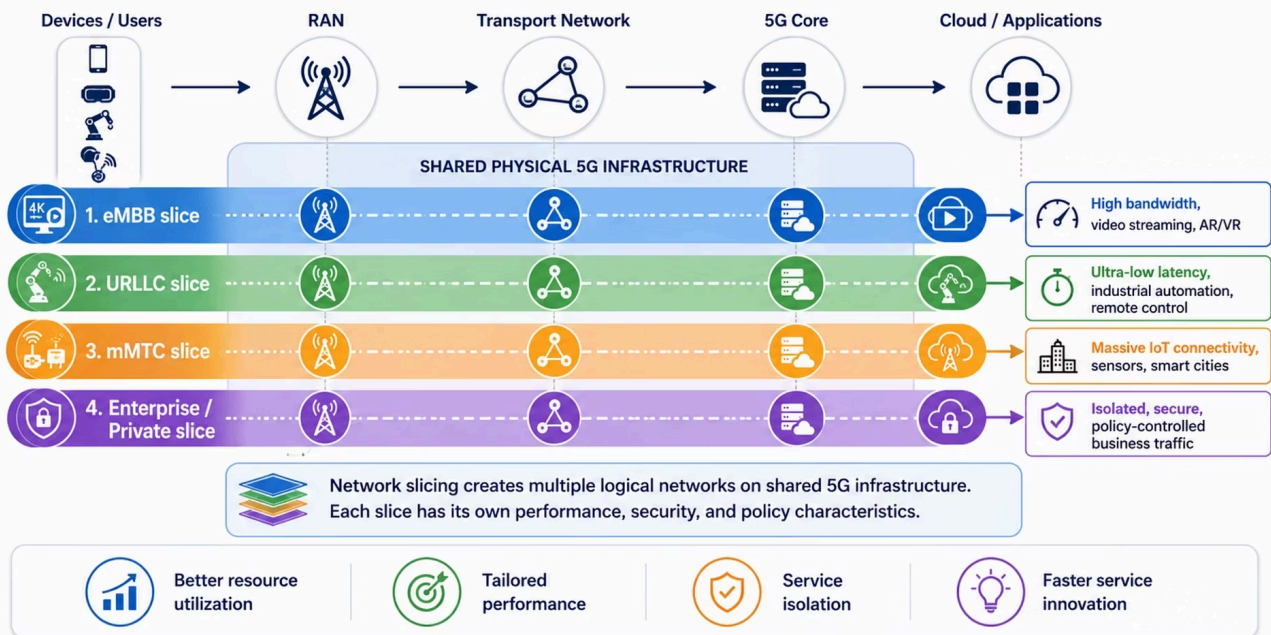


Figure 5 — One physical 5G network, multiple virtual slices — each running end to end across devices, radio, transport, core, and cloud, with its own performance, security, and policy profile.

One clarification worth carrying into customer conversations: eMBB, URLLC, and mMTC are the **standard 3GPP slice types** — 3GPP being the body that writes the mobile standards. The **management plane** referred to throughout this paper is *not* an official slice category; it is an operational grouping for OAM traffic — the infrastructure talking to itself for configuration and health checks. It is designed like a slice because it needs the same hard isolation from subscriber traffic, which is why it gets its own dedicated pool and DNS view.

Slicing gives operators better resource utilization, tailored performance per service, isolation between services, and faster innovation — a new slice is a new product.

4.3. Why slicing makes IP addressing hard — and DDI essential

Here is the catch. Each slice needs its own address pools, its own DNS view, its own lease policies, and hard isolation from every other slice — while all slices run on the same physical infrastructure. A sensor in the mMTC slice must never receive an address from the URLLC pool. A management query must never resolve to a subscriber-facing server. Multiply that by hundreds of thousands of endpoints and multiple tenants, and manual address management is not merely painful; it is impossible. This is precisely the problem DDI Central was built to solve.

Returning to our city. Every district we toured has an addressing need, and DDI Central serves all of them from a single platform:

- **Provisioning layer:** PXE boot and DHCP for bare-metal servers and the network functions that will run on them (VNFs and CNFs – virtualized and containerized network functions, i.e., network software delivered as VMs or containers).
- **Kubernetes / OpenStack platforms:** IP planning, subnet allocation, and lifecycle tracking for the constantly shifting population of containers and virtual machines.
- **5G control plane:** addressing and DNS for the core functions that manage signalling – with names like AMF, SMF, PCF, NRF, AUSF, and UDM. You do not need to memorize them; think of them as the departments of the brain: access management, session management, policy, the registry of functions, authentication, and the subscriber database.
- **5G user plane:** fast DHCP and policy segmentation for the components that actually carry subscriber data – chiefly the **UPF** (User Plane Function, the network's cargo mover), plus edge computing sites (**MEC**) placed near users for speed.
- **Management & OOB plane:** isolated addressing and DNS for the control room – monitoring systems, out-of-band hardware controllers, time servers, and logs.

The mechanics behind the third and fourth bullets are the ones we walked through in Part 1: the 5G core splits into a control plane that decides and a user plane that carries, and DDI Central appears in both – as the IPAM engine the SMF asks for an address when a session is created, and as the local DNS and DHCP that serve devices and edge functions at millisecond speed. Five layers, five different addressing needs, one platform underneath all of them.

Remember it this way: applications change, platforms change, radio generations change – but everything, always, needs an address and a name. DDI Central owns that layer.

Network-layer visibility into how 5G slicing works

DDI Central gives operators real-time, slice-aware visibility across DNS, DHCP and IPAM.

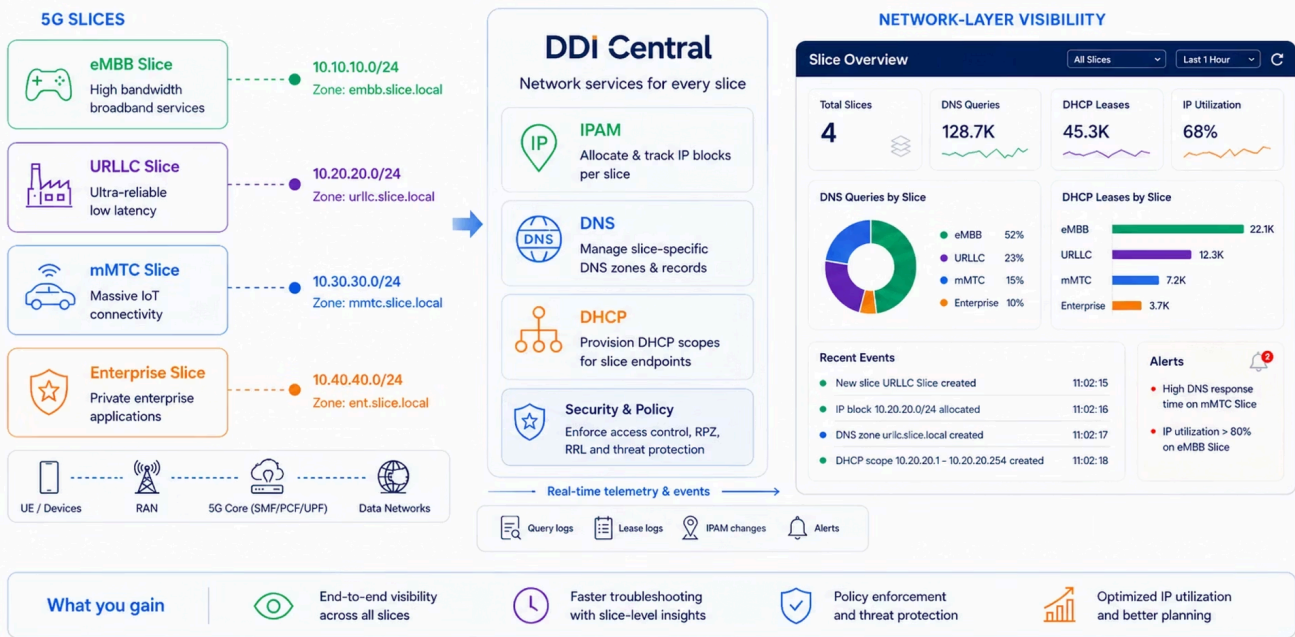


Figure 6 – Network-layer visibility into slicing: each slice gets its own IP block, DNS zone, and DHCP scope – and DDI Central tracks queries, leases, utilization, and alerts per slice, in real time.

Remember it this way: slicing turns one network into many lanes. Every lane needs its own front desk, directory, and land registry – kept perfectly separate, yet managed from one control tower.

5. Deep Dive Into A Sample Telco Data Center

A Day in the sample scenario: One Deployment, Everything Together

To set the context in a sample on-premise telco deployment. Hold this picture in mind as the 'reference customer':

- **500,000 endpoints** managed — every server, container, phone, sensor, and controller.
- **77,000 local subscribers** served through edge nodes and cell towers.
- **3 active 5G slices** (eMBB, URLLC, mMTC) plus a management plane — each with its own VRF and DHCP pool, e.g., 10.10.0.0/18 for eMBB, 10.20.0.0/20 for URLLC, 10.30.0.0/20 for mMTC, 10.40.0.0/24 for management.

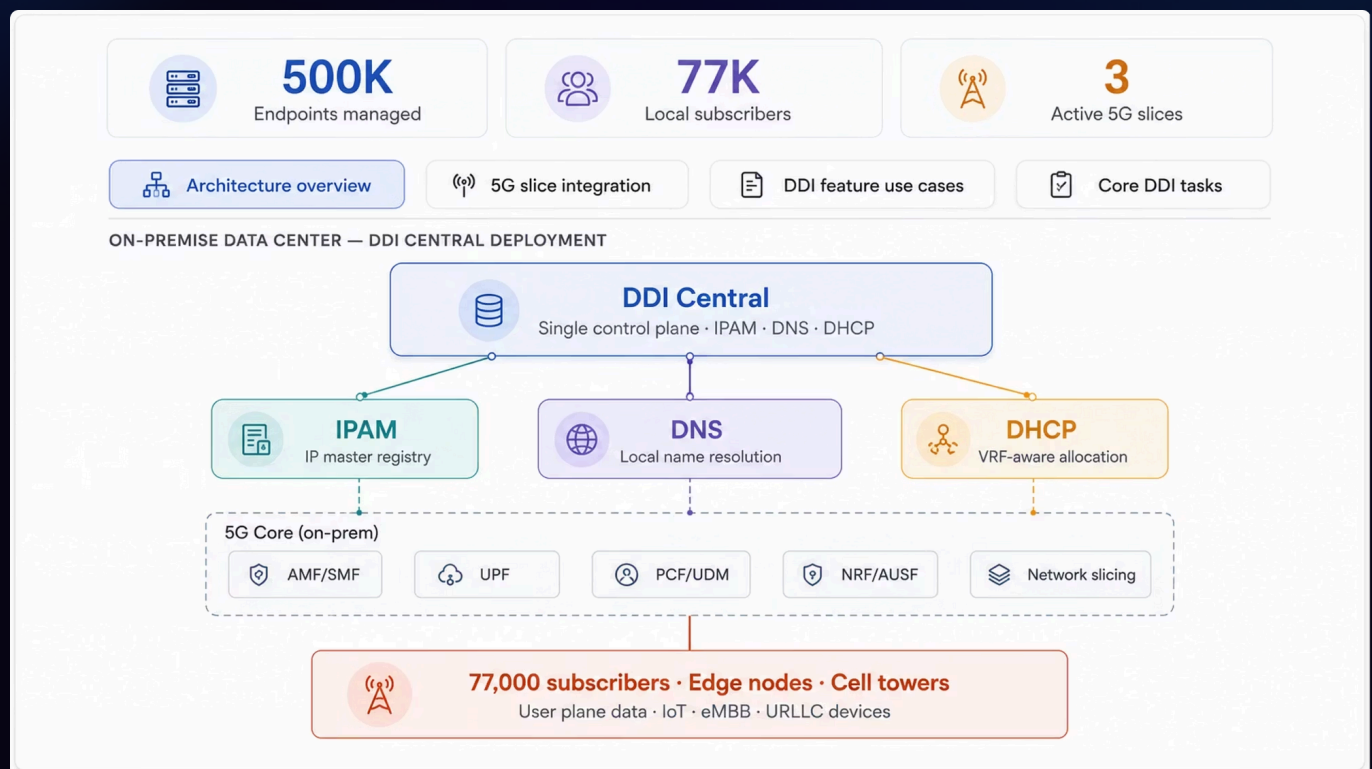


Figure 7 — The reference deployment: DDI Central as a single on-premise control plane over IPAM, DNS, and DHCP, feeding the 5G core and, through it, 77,000 subscribers across edge nodes and cell towers.

DDI Central runs as the single control plane across three engines — IPAM as the IP master registry, DNS for local name resolution, DHCP for VRF-aware allocation — feeding the on-prem 5G core and, through it, the subscribers. Three core routines keep the city running:

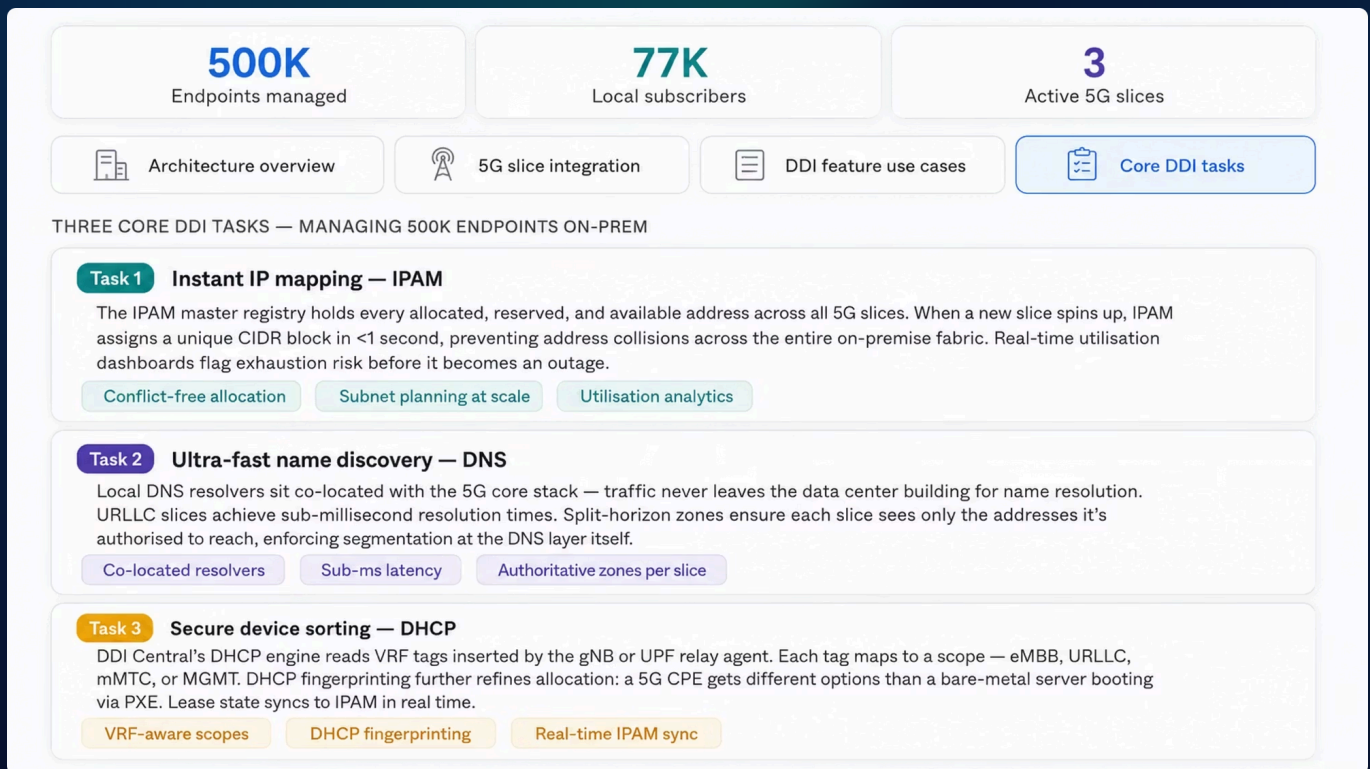


Figure 8— The three core DDI tasks that keep 500,000 endpoints running on-premise: instant IP mapping, ultra-fast name discovery, and secure device sorting.

Task 1 — Instant IP mapping (IPAM). The master registry holds every allocated, reserved, and available address across all slices. When a new slice spins up, IPAM assigns a unique CIDR block in under a second, preventing collisions across the entire fabric, while real-time utilization dashboards flag exhaustion risk before it becomes an outage.

Task 2 — Ultra-fast name discovery (DNS). Resolvers co-located with the 5G core keep resolution inside the building; URLLC slices see sub-millisecond answers; split-horizon zones ensure each slice sees only what it is authorized to reach.

Task 3 — Secure device sorting (DHCP). The DHCP engine reads the VRF tags inserted by the radio and relay layer, maps each to the correct scope, refines the decision with fingerprinting — a 5G handset is treated differently from a bare-metal server PXE-booting — and syncs every lease to IPAM in real time.

6. Flagship DDI Capabilities, Root to Shoot

This is the heart of the paper. Each capability is introduced with the problem it solves, the mechanism that solves it, the product screens that deliver it, and the story that makes it stick.

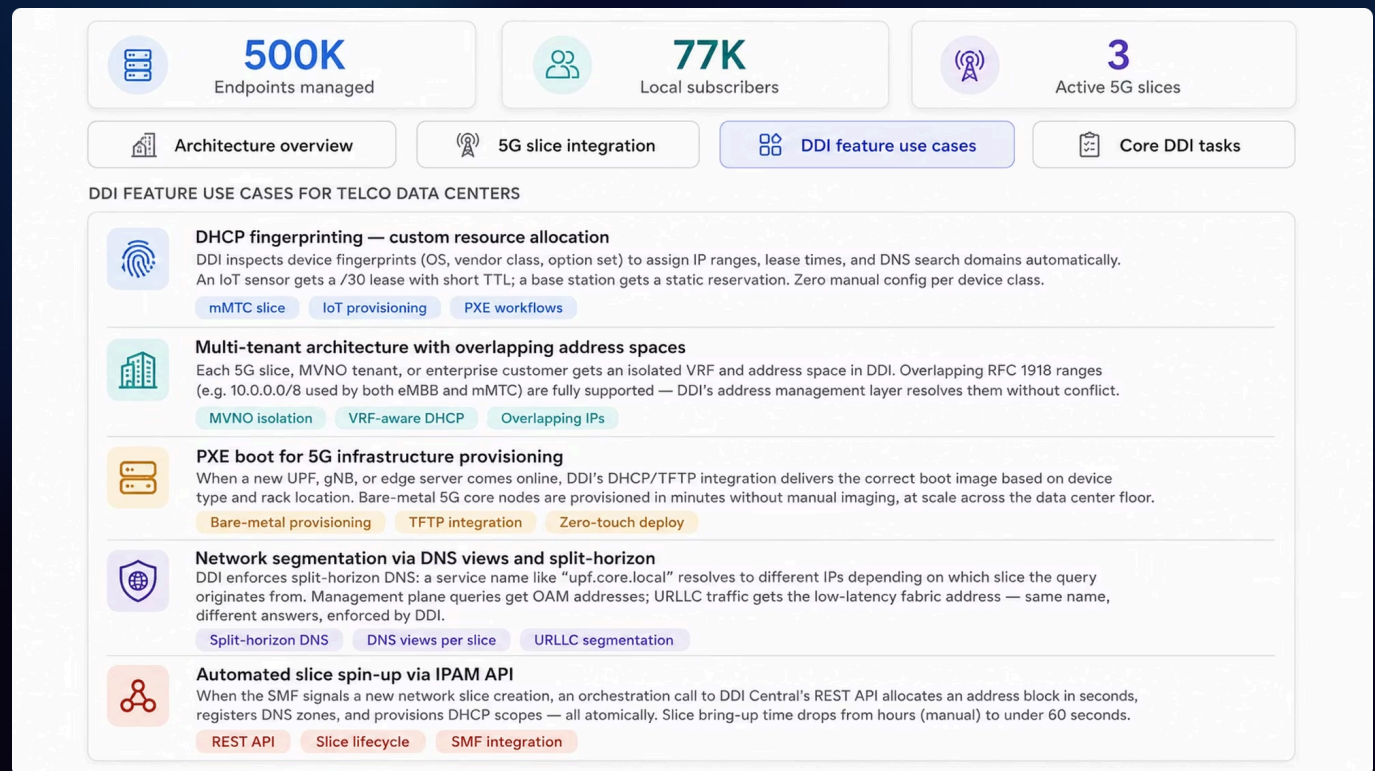


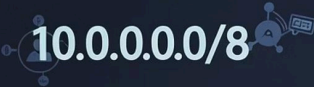
Figure 9 — The map for this section: the five DDI capability families that telco data centers depend on — fingerprinting, multi-tenancy, PXE provisioning, DNS-based segmentation, and API-driven slice automation.

6.1. Multi-Tenant Architecture: One Platform, Many Tenants — Isolation With Overlapping Addresses

The problem

Telco facilities are rarely single-occupant. A colocation or carrier-neutral data center hosts multiple operators; a host operator carries **MVNOs** (Mobile Virtual Network Operators — brands that sell mobile service using the host's infrastructure); enterprises buy private slices.

Every tenant tends to use the same familiar private address ranges — the **RFC 1918** ranges (10.x.x.x, 172.16.x.x, 192.168.x.x), the address blocks reserved worldwide for private networks, free for anyone to use internally. So tenant after tenant shows up wanting 10.10.0.0. In a traditional tool, that is a collision. Forcing every tenant to renumber is slow, political, and error-prone.

CIDR Prefix	IP Address Range	Typical User Case
10.0.0.0/8 	10.0.0.0 – 10,255,255.255 16,77,216 (24-bit block)	Large enterprise networks, data centers
172.16.0.0/12 	172.16.0.0 – 12.31.255.255 1,048,576 (20-bit block) 	Medium-sized organization networks
192.168.0.0/16 	192.18.0.0 – 122.-255,255 65,536 (16-bit block)	Home and small office networks

.Figure 10 — The three private address ranges every network in the world is free to reuse internally — which is exactly why tenants collide.

The DDI Central mechanism

DDI Central gives **each tenant or slice its own dedicated DHCP server instance inside its own cluster, with its own contained IP space, policies, lease tables, and DNS views**. Nothing is shared across the boundary — and **the server instance boundary is the isolation boundary**. Because each instance is a sealed world, identical address ranges can coexist across tenants with zero conflict, and there is no cross-tenant leakage even when ranges overlap completely.

6.1.1. Colocation and Carrier-Neutral Multi-Tenant Isolation

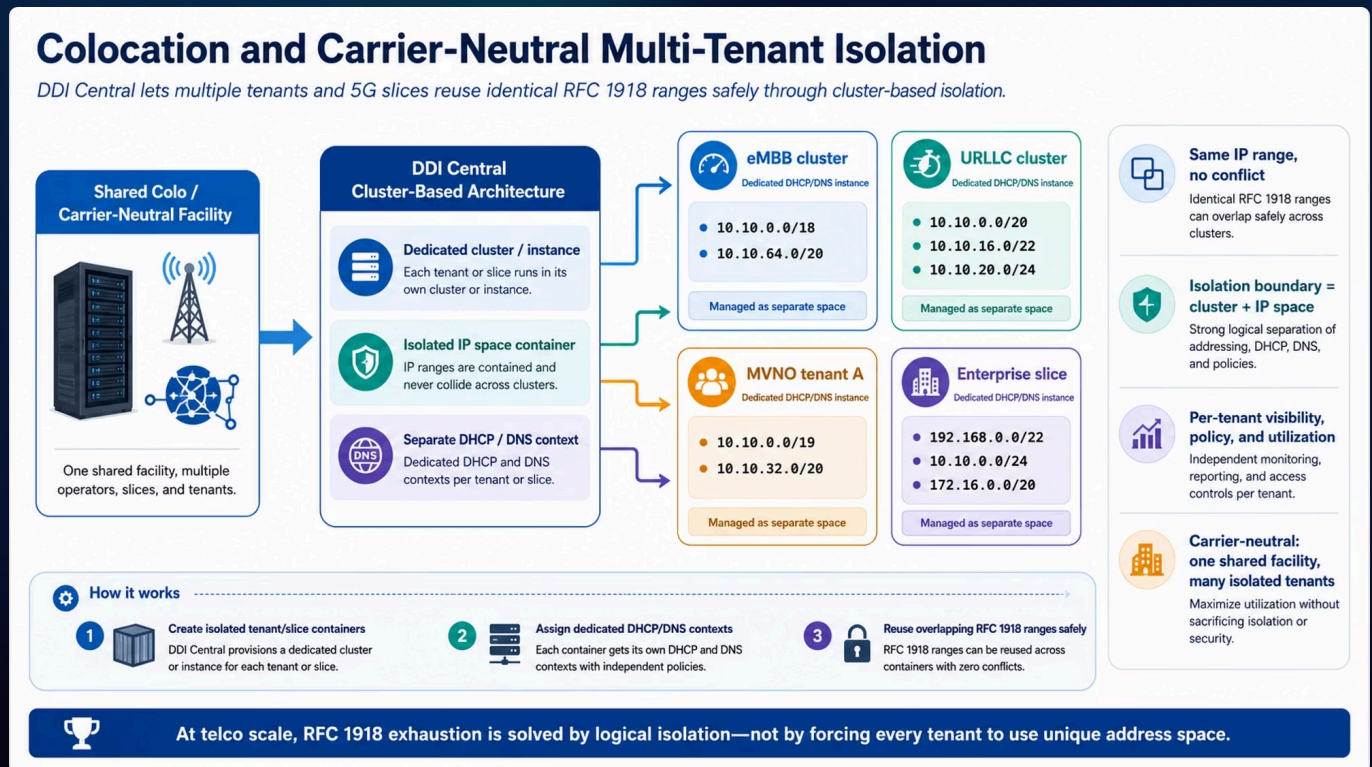


Figure 11— Cluster-based isolation in a shared colocation facility: every tenant or slice runs in its own cluster with its own DHCP/DNS context and contained IP space, so identical RFC 1918 ranges coexist without conflict.

The story that makes it stick

Think of four apartment towers on one plot of land. Every tower has a Room 402. No letter is ever misdelivered — because every tower has its own front desk, its own key cabinet, and its own resident register. The tower is the boundary. In DDI Central, the DHCP server instance is the tower.

In the example, four tenants coexist on one data center: an eMBB cluster using a 10.10.0.0/8 pool, a URLLC cluster on 10.10.0.0/20, an MVNO tenant on 10.10.0.0/19, and an enterprise slice on 10.10.0.0/24 — four overlapping ranges, four sealed instances, total isolation, all provisioned and governed from a single management pane. (The '/8, /20, /24' suffixes are CIDR notation — shorthand for how large an address block is: the smaller the number, the bigger the block.)

6.1.2. 5G Network Slice Integration— DDI Isolation Per Slice

How DDI Central enforces slice isolation

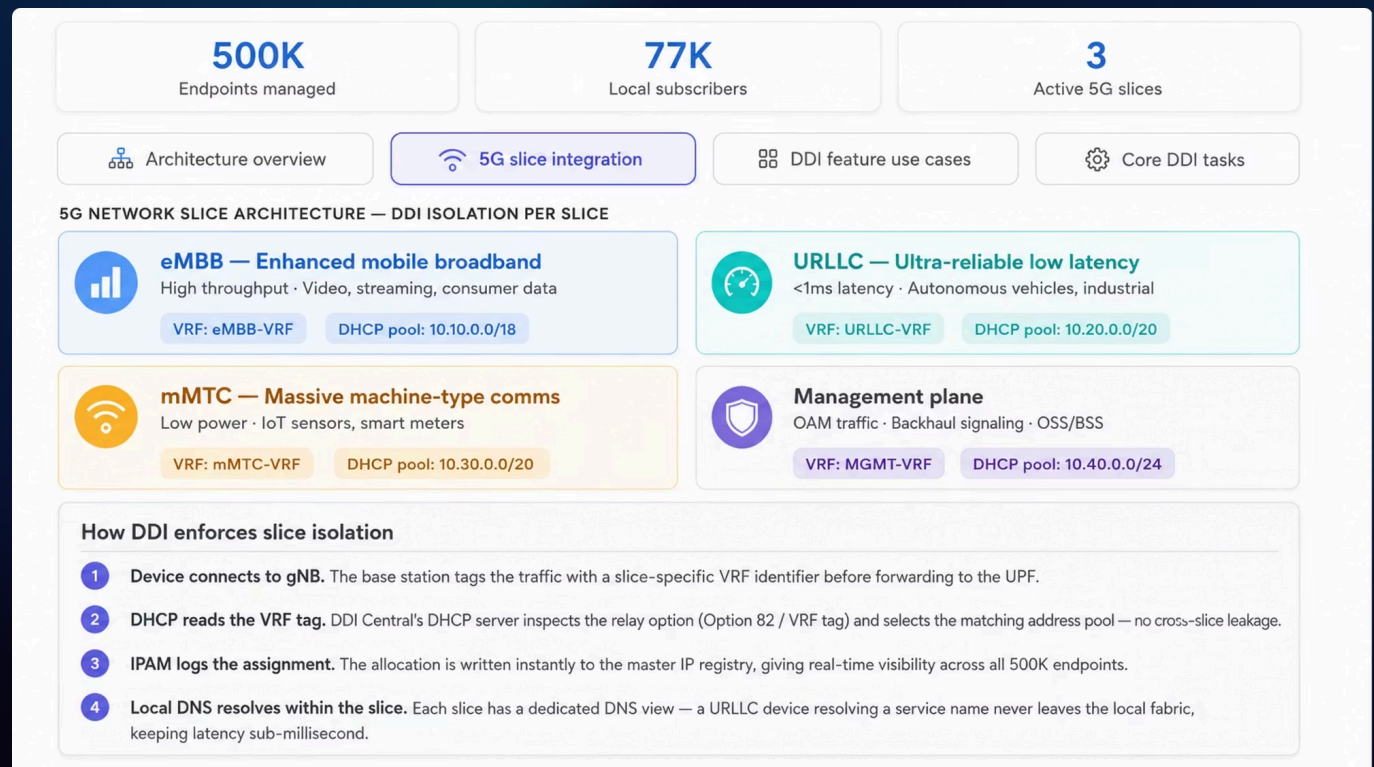
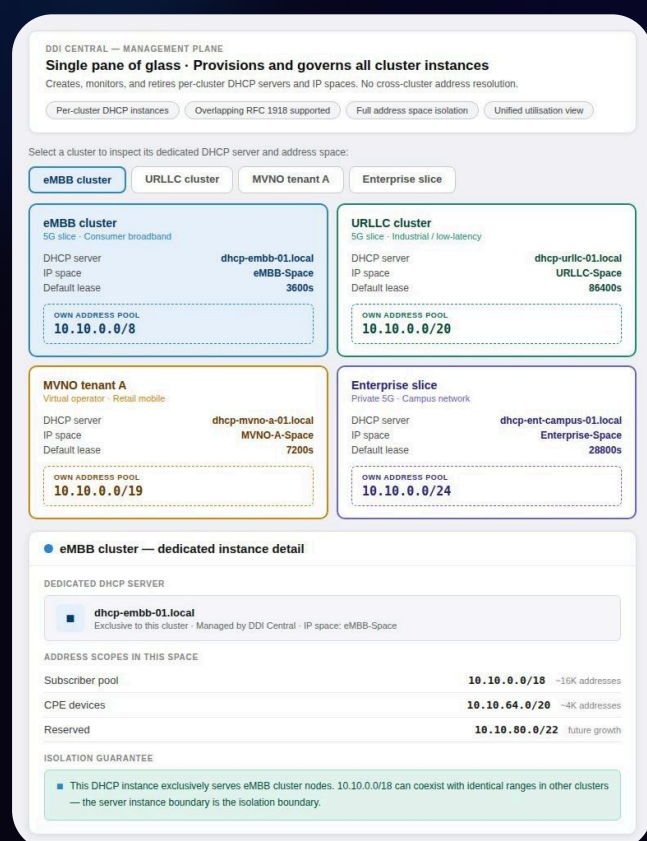


Figure 12— Slice integration in the same deployment: each slice has its own VRF and DHCP pool, and the four-step isolation flow — tag, read, log, resolve — runs on every single connection.

End to end automated loop: A device connects to a base station; the base station tags its traffic with the slice identifier; DHCP reads the tag and fingerprint and selects the matching pool — no cross-slice leakage; IPAM logs the allocation instantly across all 500,000 endpoints; and DNS resolves the device's service lookups inside its own slice view at sub-millisecond speed. Four steps, three services, one control plane, zero spreadsheets.

What it looks like in the product

DDI Central presents a single management pane that provisions, monitors, and retires per-cluster DNS servers, DHCP servers and IP spaces. Selecting any cluster opens its dedicated instance — its own server, its own address scopes, its own lease policy, and an explicit isolation guarantee.



DDI CENTRAL — MANAGEMENT PLANE
Single pane of glass · Provisions and governs all cluster instances
 Creates, monitors, and retires per-cluster DHCP servers and IP spaces. No cross-cluster address resolution.

Per-cluster DHCP instances Overlapping RFC 1918 supported Full address space isolation Unified utilisation view

Select a cluster to inspect its dedicated DHCP server and address space:

eMBB cluster URLLC cluster MVNO tenant A Enterprise slice

eMBB cluster
 5G slice · Consumer broadband
 DHCP server: dhcp-embb-01.local
 IP space: eMBB-Space
 Default lease: 3600s
 OWN ADDRESS POOL: 10.10.0.0/8

URLLC cluster
 5G slice · Industrial / low-latency
 DHCP server: dhcp-urllc-01.local
 IP space: URLLC-Space
 Default lease: 86400s
 OWN ADDRESS POOL: 10.10.0.0/20

MVNO tenant A
 Virtual operator · Retail mobile
 DHCP server: dhcp-mvno-a-01.local
 IP space: MVNO-A-Space
 Default lease: 7200s
 OWN ADDRESS POOL: 10.10.0.0/19

Enterprise slice
 Private 5G · Campus network
 DHCP server: dhcp-ent-campus-01.local
 IP space: Enterprise-Space
 Default lease: 28800s
 OWN ADDRESS POOL: 10.10.0.0/24

eMBB cluster — dedicated instance detail

DEDICATED DHCP SERVER

■ dhcp-embb-01.local
 Exclusive to this cluster · Managed by DDI Central · IP space: eMBB-Space

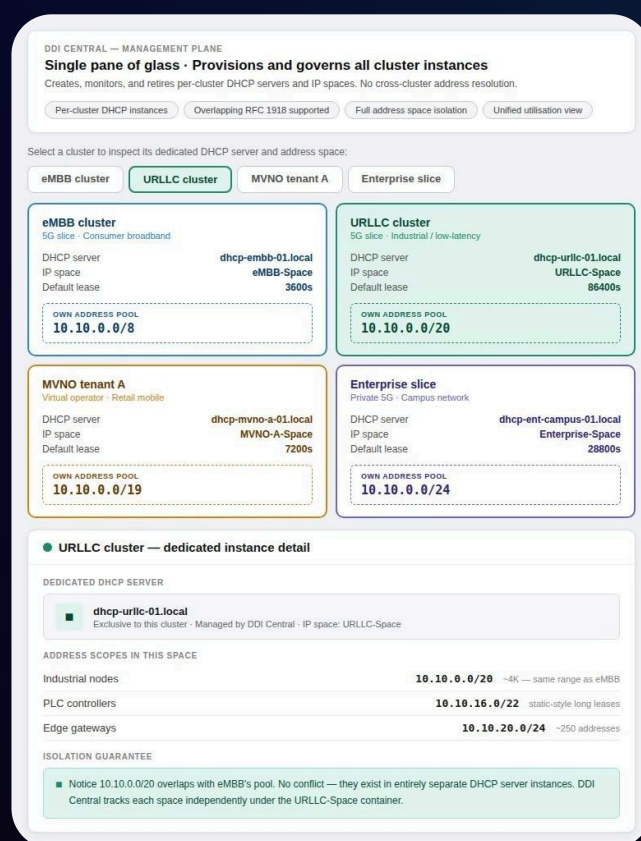
ADDRESS SCOPES IN THIS SPACE

Scope	Address Range	Count
Subscriber pool	10.10.0.0/18	~16K addresses
CPE devices	10.10.64.0/20	~4K addresses
Reserved	10.10.80.0/22	future growth

ISOLATION GUARANTEE

■ This DHCP instance exclusively serves eMBB cluster nodes. 10.10.0.0/18 can coexist with identical ranges in other clusters — the server instance boundary is the isolation boundary.

Figure 13 — eMBB cluster: a dedicated DHCP server (dhcp-embb-01.local) inside the eMBB-Space IP container, with a subscriber pool, a CPE pool, and reserved space for growth. This instance exclusively serves eMBB nodes.



DDI CENTRAL — MANAGEMENT PLANE
Single pane of glass · Provisions and governs all cluster instances
 Creates, monitors, and retires per-cluster DHCP servers and IP spaces. No cross-cluster address resolution.

Per-cluster DHCP instances Overlapping RFC 1918 supported Full address space isolation Unified utilisation view

Select a cluster to inspect its dedicated DHCP server and address space:

eMBB cluster URLLC cluster MVNO tenant A Enterprise slice

eMBB cluster
 5G slice · Consumer broadband
 DHCP server: dhcp-embb-01.local
 IP space: eMBB-Space
 Default lease: 3600s
 OWN ADDRESS POOL: 10.10.0.0/8

URLLC cluster
 5G slice · Industrial / low-latency
 DHCP server: dhcp-urllc-01.local
 IP space: URLLC-Space
 Default lease: 86400s
 OWN ADDRESS POOL: 10.10.0.0/20

MVNO tenant A
 Virtual operator · Retail mobile
 DHCP server: dhcp-mvno-a-01.local
 IP space: MVNO-A-Space
 Default lease: 7200s
 OWN ADDRESS POOL: 10.10.0.0/19

Enterprise slice
 Private 5G · Campus network
 DHCP server: dhcp-ent-campus-01.local
 IP space: Enterprise-Space
 Default lease: 28800s
 OWN ADDRESS POOL: 10.10.0.0/24

URLLC cluster — dedicated instance detail

DEDICATED DHCP SERVER

■ dhcp-urllc-01.local
 Exclusive to this cluster · Managed by DDI Central · IP space: URLLC-Space

ADDRESS SCOPES IN THIS SPACE

Scope	Address Range	Count
Industrial nodes	10.10.0.0/20	~4K — same range as eMBB
PLC controllers	10.10.16.0/22	static-style long leases
Edge gateways	10.10.20.0/24	~250 addresses

ISOLATION GUARANTEE

■ Notice 10.10.0.0/20 overlaps with eMBB's pool. No conflict — they exist in entirely separate DHCP server instances. DDI Central tracks each space independently under the URLLC-Space container.

Figure 14 — URLLC cluster: note that 10.10.0.0/20 overlaps the eMBB pool exactly — and there is no conflict, because the two live in entirely separate DHCP server instances, tracked independently.

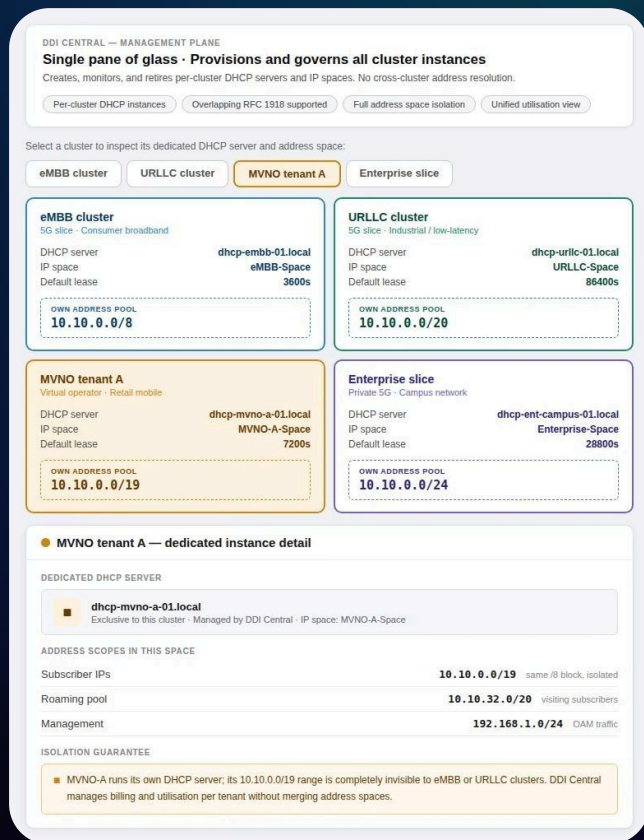


Figure 15 — MVNO tenant A: its own server, its own subscriber, roaming, and management pools. The tenant's 10.10.0.0/19 range is completely invisible to the eMBB and URLLC clusters, so utilization and billing are per tenant without ever merging address spaces.

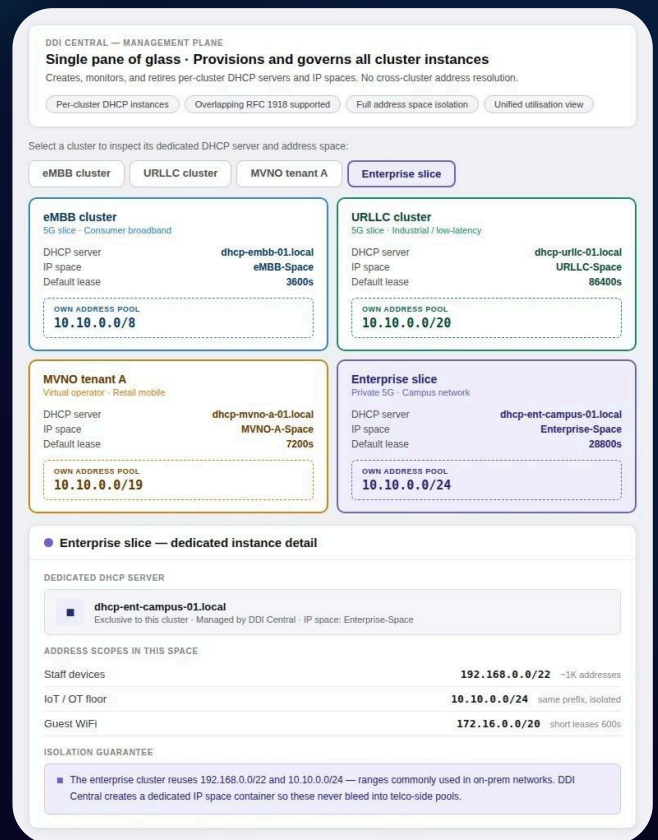


Figure 16 — Enterprise slice: reuses 192.168.0.0/22 and 10.10.0.0/24 — ranges any on-premise corporate network is likely to already use — inside a dedicated IP space container, so they never bleed into telco-side pools.

6.1.3. The Solution: Overlapping RFC 1918 IP Ranges — Zero Conflict

Four tenants, four identical ranges, **total isolation** — all managed from one control plane.

eMBB Cluster

10.10.0.0/8 pool

URLLC Cluster

10.10.0.0/20 — overlaps eMBB, no conflict

MVNO Tenant A

10.10.0.0/19 — same block, isolated

Enterprise Slice

10.10.0.0/24 — reuses common ranges

What this earns the operator

Conflict-Free colocation- and carrier-neutral-ready multi-tenancy; MVNOs that share infrastructure but never IP space; policy enforcement at the server boundary; no VLAN re-engineering to onboard a new tenant; and per-tenant visibility, utilization reporting, and even billing — without ever merging address spaces.



One-line takeaway: at telco scale, address exhaustion and conflicts are solved by logical isolation — not by forcing every tenant to be unique.

6.1.4. How DDI Central enforces boundaries between tenants

Each tenant in a shared telco data center gets a fully isolated DHCP server instance — with its own IP pools, policies, and DNS views. No shared state, no cross-tenant leakage, even when address ranges overlap.

- 1 — Dedicated DHCP server instance per tenant
- 2 — Separate DNS views per slice/tenant
- 3 — Policy enforcement at the server boundary

One server instance per slice — configured independently

Each 5G slice gets its own DHCP server instance in DDI Central. Configuration is scoped entirely within that instance — pools, options, policies, and lease times are all slice-specific and never shared across boundaries.

-  **Changes to one instance have zero impact on other slices — full operational independence.**

6.2. Fingerprint First, Allocate Second — the DHCP policy engine

The problem

Half a million devices ask the front desk for keys — but they are wildly different guests. A consumer smartphone should get a short lease from the broadband pool. An industrial robot controller must land in the ultra-low-latency pool. A battery-powered sensor needs a week-long lease so it can sleep. A management node must be locked away from subscriber traffic entirely. Configuring this per device, at 500,000 devices, is unthinkable.

The DDI Central mechanism

Devices introduce themselves. Every DHCP request carries telltale fields — **Option 60** (the vendor class, effectively "who made me and what am I"), **Option 55** (the list of settings the device asks for — its "shopping list"), and **Option 82** (a tag inserted by the network relay saying where the request came from — including which slice). Together these form a **DHCP fingerprint** as distinctive as a knock on the door. DDI Central's policy engine reads the fingerprint **before** any address is assigned, using rules called **client classes**. A client class is built from six components that together define *who gets what, and why*:

- **Class Type** — a reusable template for the fingerprint pattern.
- **Match Type** — how to inspect, e.g., extract a substring of bytes from an option.
- **Offset + Length** — exactly where to look inside the option field.
- **Match String / Value** — what the extracted bytes must equal.
- **Conditional logic** — AND/OR combinations for compound rules (e.g., right vendor AND right slice tag).
- **DHCP options applied** — the outcome on match: the pool, lease time, gateway, DNS server, and more.

Four guests at the front desk

Each of the four walkthroughs below follows the same four steps: (1) the device's **DHCPDISCOVER** arrives, (2) the fingerprint engine classifies it, (3) a policy rule matches, and (4) the **DHCPOFFER** goes out with the appropriate IP address *and* every secondary parameter — pool, lease time, gateway, DNS server, domain, and time server.

The 5G smartphone. An Android phone announces vendor class "android-dhcp" with a long parameter shopping list. The engine matches the consumer-devices policy – cross-checking the Option 82 relay tag to confirm the eMBB path – and issues an address from the eMBB pool (10.10.0.0/18, roughly 16,000 addresses) with a standard one-hour/ 3600 seconds lease.

Choose a device type to see the full policy allocation flow:

- DHCPDISCOVER arrives at DDI DHCP server**

Option 55 (param list)	1,3,6,15,26,28,51,58,59,43
Option 60 (vendor class)	android-dhcp-12
Option 12 (hostname)	android-device-01
Client MAC	AA:BB:CC:DD:EE:FF
- Fingerprint engine classifies the device**

Matched fingerprint	Android smartphone · consumer profile
Opt 55 param count	10 parameters
Opt 60 string	android-dhcp-12
- Policy rule matched**

MATCHED POLICY
Policy: consumer-devices

Match condition	Address pool
Matches Option 60 prefix "android-" and Option 55 length ≥ 9	10.10.0.0/18
- DHCPOFFER — IP + secondary parameters allocated**

DHCPOFFER PAYLOAD	
Assigned IP	10.10.14.82
Subnet / pool	10.10.0.0/18
Lease time	3600s
Default gateway	10.10.0.1
DNS servers	8.8.8.8, 8.8.4.4
Domain name	consumer.local
NTP server	ntp.consumer.local

Figure 17 – Device type: 5G smartphone → eMBB slice. Option 60 "android-dhcp" plus a 10-parameter request matches the consumer-devices policy; the offer carries a 3,600-second lease and consumer-grade DNS and domain settings.

The industrial PLC. A Siemens factory controller announces "SIMATIC-S7-1500" and requests boot-server options. It matches the industrial-controllers policy and lands in the URLLC pool with a day-long lease (86400 seconds) and ultra-low-latency parameters — enforced at the moment of address assignment.

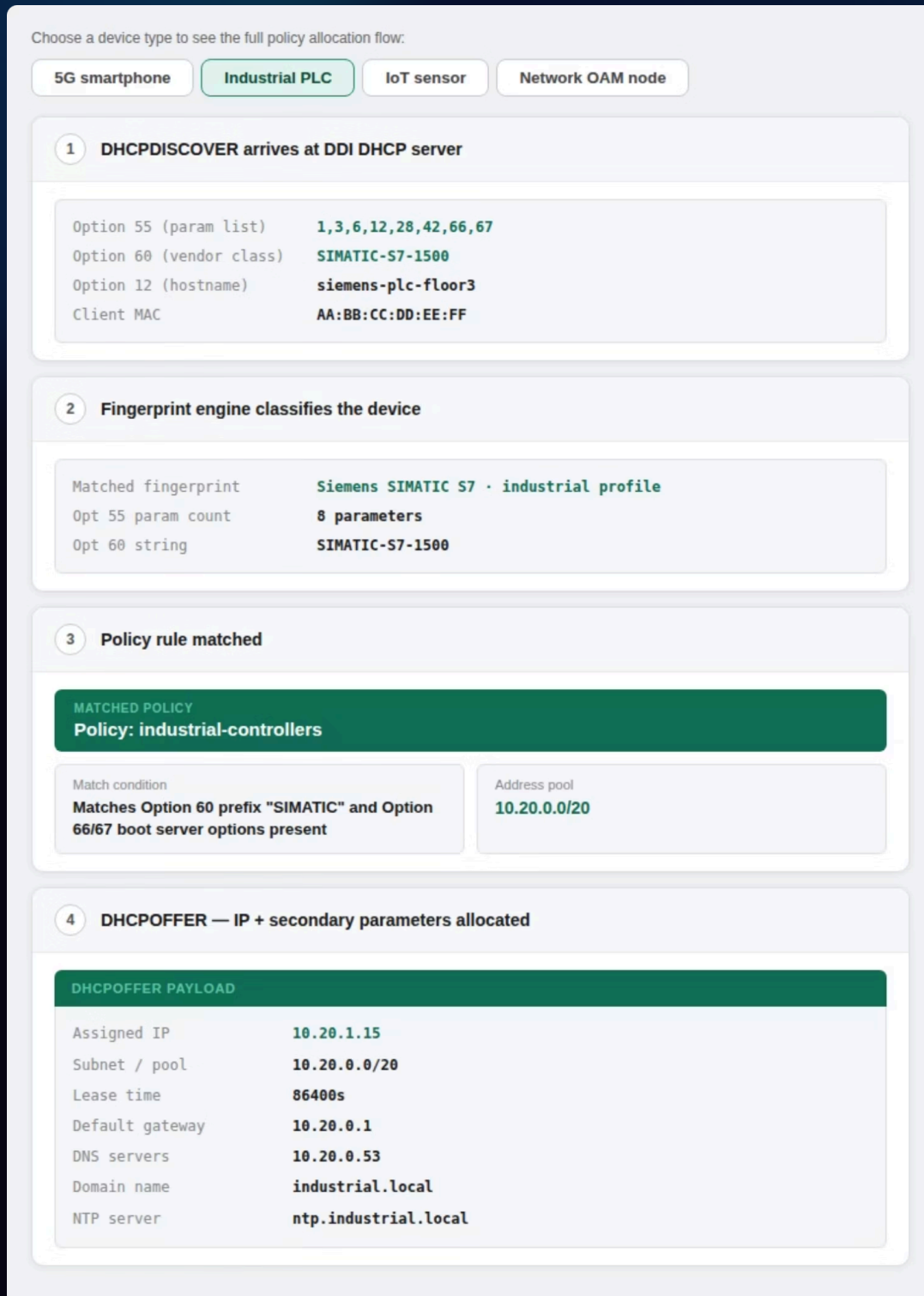


Figure 18 — Device type: industrial PLC → URLLC slice. The "SIMATIC" vendor string plus the presence of boot-server options 66/67 selects the industrial pool, a 86,400-second lease, and the local industrial DNS.

The IoT sensor. A narrowband sensor announces via Option 60 prefix "NBIoT-LPWA" with a minimal shopping list i.e. minimal DHCP option sets (≤ 6 parameters) as these devices are deliberately simple requiring light configurations. It matches the predefined IoT policy and receives an IP address deliberately for a *short*-lease period (300 seconds) from the mMTC pool (optimised for low-power, high-density deployments)— the sensor wakes, reports, and goes back to sleep, so the address returns to the pool almost immediately, conserving IP space across what may be millions of identical devices.

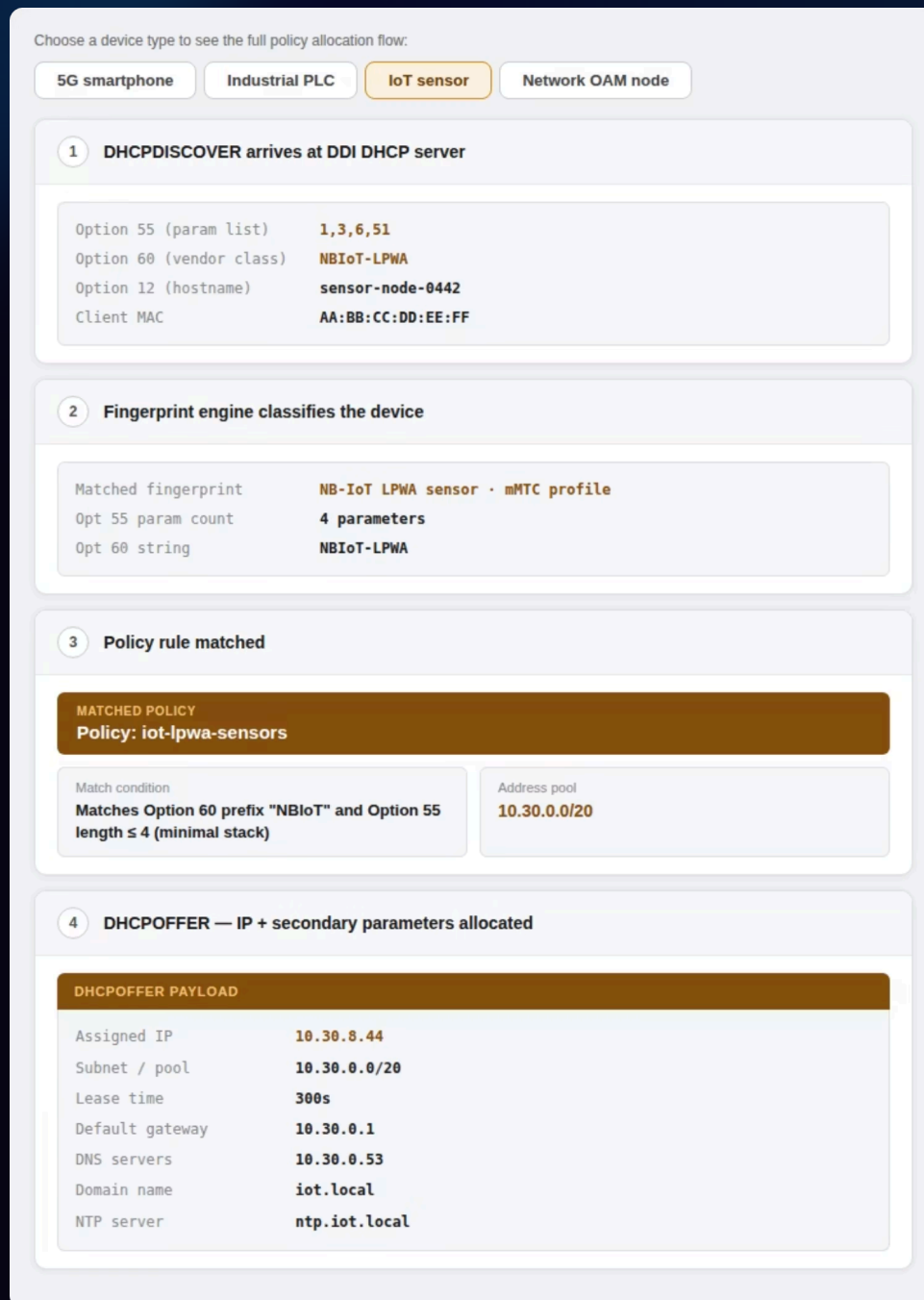


Figure 19 – Device type: IoT sensor → mMTC slice. A four-parameter minimal option set is itself part of the fingerprint – simple devices ask for little, and that simplicity identifies them. Note the short 300-second lease.

The OAM node. A Nokia AirFrame management node is fingerprinted straight into the small, tightly held management pool – with long leases for stable operations and complete isolation from subscriber paths.

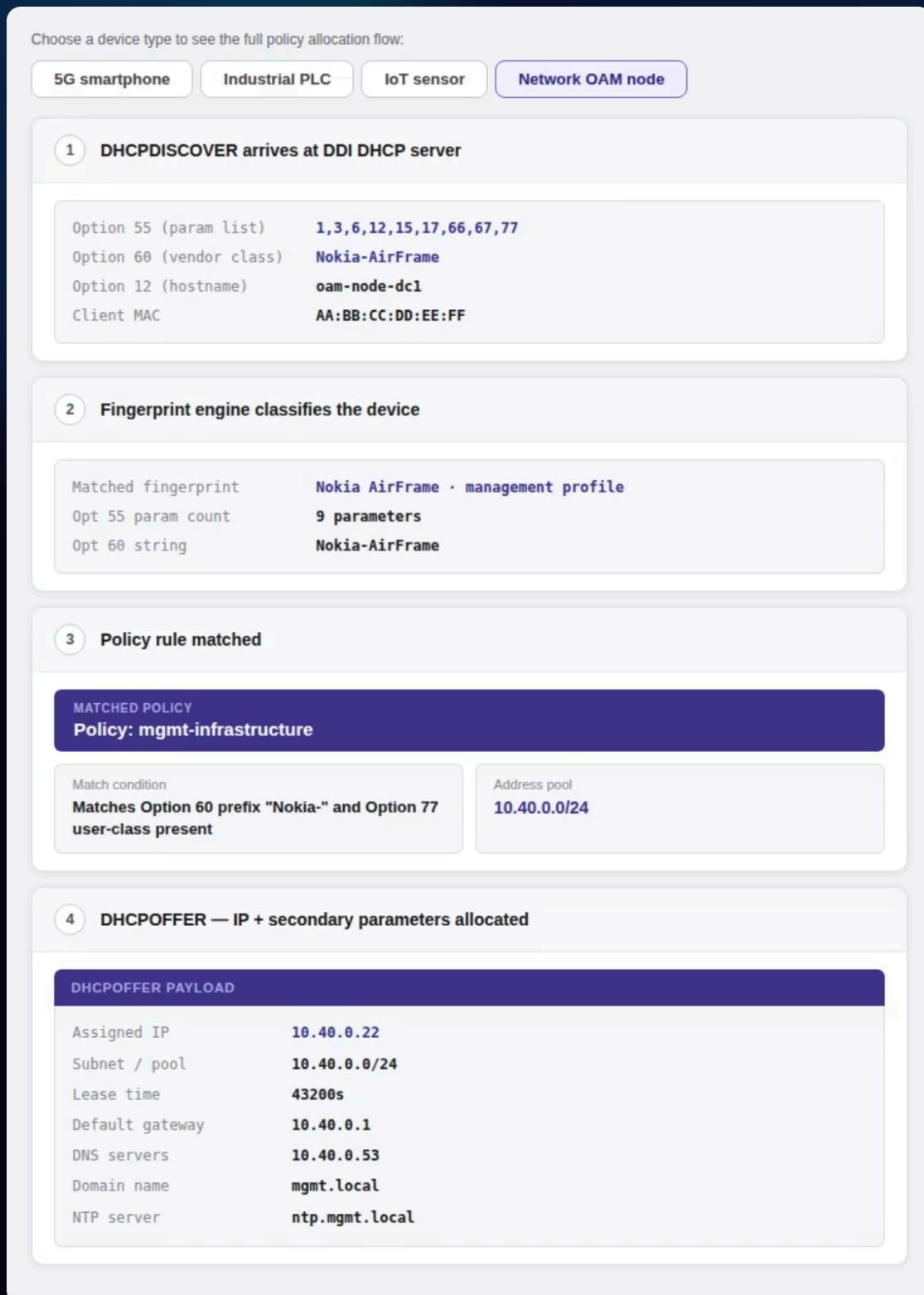


Figure 20— Device type: network OAM node → management plane. The “Nokia-” vendor prefix plus a user-class option routes the node into the deliberately small 10.40.0.0/24 management pool – isolated from subscriber and user plane traffic, with dedicated DNS views and IPAM policies.

Reading the offer like an operator: why the lease times differ

The four offers differ in far more than the address. The lease time in particular is a deliberate operational decision, and it is the fastest way to show a customer that DDI Central is tuning to the device's job rather than to the network diagram:

Device class	Lease	Why that number
5G smartphone (eMBB)	3,600s – 1 hour	Consumers arrive and leave constantly. A short lease keeps the large /18 pool recycling efficiently.
Industrial PLC (URLLC)	86,400s – 24 hours	A controller cannot lose its identity mid-operation; a renewal during a real-time control loop is a risk. A day-long, stable assignment removes it.
IoT sensor (mMTC)	300s – 5 minutes	Sensors wake, report, and sleep. Returning the address immediately conserves space across millions of devices.
OAM node (management)	43,200s – 12 hours	Keeps SSH sessions, SNMP polling, and management traffic stable across a maintenance window without holding addresses indefinitely.

Look at the DNS servers too. The consumer offer points at public resolvers; the industrial offer points at 10.20.0.53 on the industrial fabric – internal, low-latency, never leaving the building. And look at the size of the management pool: a deliberate /24, just 254 addresses. That is a built-in sanity check. If more than 254 devices ever match the management policy, something is wrong and the operator wants to know immediately.

Zero manual configuration per device class. The OEM-set identity strings carry the intelligence; the policy engine does the sorting; every decision is logged.

One more point that matters to operations teams: the policy is written once and then applies to every device that ever sends a **DHCPDISCOVER** on that network. Nobody touches it again unless the device taxonomy changes.

DHCP Fingerprinting policy for 5G slice management

Device sorting into the right slices using client class configuration

What is a client class?

A policy rule inside a DHCP server that matches device fingerprints to IP pools

What does it inspect?

Option 60 (vendor class),
Option 82 (relay tag), Option
55 (parameter list)

What does it deliver?

The right IP pool, gateway,
DNS server, lease time — per
device class, per slice

A client class in DDI Central is built from six components that together define **who gets what, and why**:

Class Type Template (reusable pattern)	Match Type Substring extraction
Offset + Length Byte position in option	Match String Expected value
Conditional AND/OR logic	DHCP Options Applied on match

Configuring it: the client class screens, slice by slice

The four screens below are the actual configuration behind the four stories. Read each one the same way: **Offset** and **Length** say where to look inside Option 60 and how many characters to read; **Match Value** says what those characters must equal; and the **Conditional Statement** adds the second test — the Option 82 relay tag — so a device must be both *the right kind* and *on the right path* to match. The Options row at the bottom is the reward: pool, lease, DNS, and gateway.

The Enhanced Mobile BroadBand Slice (eMBB slice)

Select a slice to see its client class configuration:

DHCPv4

[Client Class](#) > Create Client Class

CLASS TYPE * Template

IMPORT VALUES ☐

MATCH TYPE * substring

OFFSET * 0

LENGTH * 12

MATCH STRING * option[60].text

CONDITIONAL STATEMENT

```

if (substring(option[60].text, 0, 12) == "android-
dhcp"
    and substring(option[82].hex, 4, 8) == "eMBB-VRF")
{
    class "embb-policy";
}
        
```

MATCH VALUE * android-dhcp

DHCP OPTIONS **CUSTOM OPTIONS**

OPTIONS pool: 10.10.0.0/18 · lease: 3600s · dns: 8.8.8.8
· gw: 10.10.0.1

Figure 21 – eMBB slice: read the first 12 characters of Option 60, require "android-dhcp", and cross-check the eMBB-VRF relay tag – then allocate from the /18 pool with a one-hour lease.

How it works: Offset 0, Length 12 extracts the **first 12 characters of** Option 60 and checks for the android-dhcp vendor class prefix identifying consumer 5G smartphones. The Conditional Statement cross-checks Option 82 for the eMBB-VRF relay tag inserted by the gNB. Both conditions must be true for the eMBB pool to be allocated – the large /18 pool supports up to 16,384 concurrent subscriber addresses with a standard 1-hour lease.

Ultra-Reliable Low-Latency Communication (URLLC) Slice

Select a slice to see its client class configuration:

DHCPv4

[Client Class](#) > Create Client Class

CLASS TYPE * Template

IMPORT VALUES ☐

MATCH TYPE * substring

OFFSET * 0

LENGTH * 7

MATCH STRING * option[60].text

CONDITIONAL STATEMENT

```

if (substring(option[60].text, 0, 7) == "SIMATIC"
    and substring(option[82].hex, 4, 9) == "URLLC-
VRF") {
    class "urllic-policy";
}
        
```

MATCH VALUE * SIMATIC

DHCP OPTIONS **CUSTOM OPTIONS**

OPTIONS pool: 10.20.0.0/20 · lease: 86400s · dns: 10.20.0.53 · gw: 10.20.0.1

Figure 22— URLLC slice: the first 7 bytes must read "SIMATIC" and the relay tag must be URLLC-VRF. Only industrial controllers arriving on the URLLC path match — anything else is unmatched.

How it works: Offset 0, Length 7 extracts the first 7 bytes of Option 60 and checks for the Siemens "SIMATIC" prefix. The Conditional Statement cross-checks Option 82 for the "URLLC-VRF" relay tag. Only industrial controllers arriving on the URLLC relay path will match — anything else is unmatched.

Massive Machine-Type Communications (mMTC) Slice

Select a slice to see its client class configuration:

DHCPv4

[Client Class](#) > Create Client Class

CLASS TYPE * Template

IMPORT VALUES ☐

MATCH TYPE * substring

OFFSET * 0

LENGTH * 9

MATCH STRING * option[60].text

CONDITIONAL STATEMENT

```

if (substring(option[60].text, 0, 9) == "NBloT-LPW"
    and substring(option[82].hex, 4, 8) == "mMTC-VRF")
{
    class "mmtc-policy";
}
        
```

MATCH VALUE * NBloT-LPW

DHCP OPTIONS **CUSTOM OPTIONS**

OPTIONS pool: 10.30.0.0/20 · lease: 300s · dns: 10.30.0.53
· gw: 10.30.0.1

Figure 23— mMTC / IoT slice: "NBloT-LPW" plus the mMTC-VRF tag. The short lease conserves address space across millions of sensors that sleep most of the time.

How it works: Offset 0, Length 9 extracts the first 9 bytes of Option 60 and checks for the NBloT-LPW LPWA vendor class prefix. The Conditional Statement cross-checks Option 82 for the mMTC-VRF relay tag. The short 300s lease time conserves address space across millions of low-power IoT sensors that sleep most of the time — only devices on the mMTC access path will match this class.

Management Plane (MGMT Slice)

Select a slice to see its client class configuration:

eMBB slice URLLC slice mMTC / IoT slice **MGMT plane**

DHCPv4

[Client Class](#) > Create Client Class

CLASS TYPE * **Template**

IMPORT VALUES ☐

MATCH TYPE * **substring**

OFFSET * **0**

LENGTH * **6**

MATCH STRING * **option[60].text**

CONDITIONAL STATEMENT

```
if (substring(option[60].text, 0, 6) == "Nokia-"
    and substring(option[82].hex, 4, 8) == "MGMT-VRF")
{
    class "mgmt-policy";
}
```

MATCH VALUE * **Nokia-**

DHCP OPTIONS CUSTOM OPTIONS

OPTIONS

```
pool: 10.40.0.0/24   lease: 43200s   dns:
10.40.0.53   gw: 10.40.0.1
```

Cancel **Save**

Figure 24 – Management plane: "Nokia-" plus the MGMT-VRF tag, restricted to an intentionally small /24 of 254 addresses – the size itself signals that only a handful of infrastructure nodes should ever match.

How it works: Offset 0, Length 6 extracts the first 6 bytes of Option 60 and checks for the Nokia AirFrame vendor class prefix Nokia-. The Conditional Statement cross-checks Option 82 for the MGMT.VRF relay tag, restricting this pool exclusively to OAM infrastructure nodes. The intentionally small /24 pool (254 addresses) signals that only a handful of management-plane nodes should ever match this class – the 12-hour lease ensures stable addresses for long-running OAM sessions.

One-line takeaway: DDI Central is the experienced receptionist who recognizes every guest by how they knock – and hands each one exactly the right key, on the right floor, with the right rules, automatically.

6.3. Hierarchical IPAM — order from core to customer

The problem

A telco's network does not stop at the data center walls. Fiber backhaul and transport networks stretch from core sites to regional hubs, to edge sites, and finally to customer premises and cell sites. Without a plan, address space becomes a junk drawer: overlaps, waste, and troubleshooting archaeology.

Precise Network Segmentation In A Telco Data Center With DDI Central

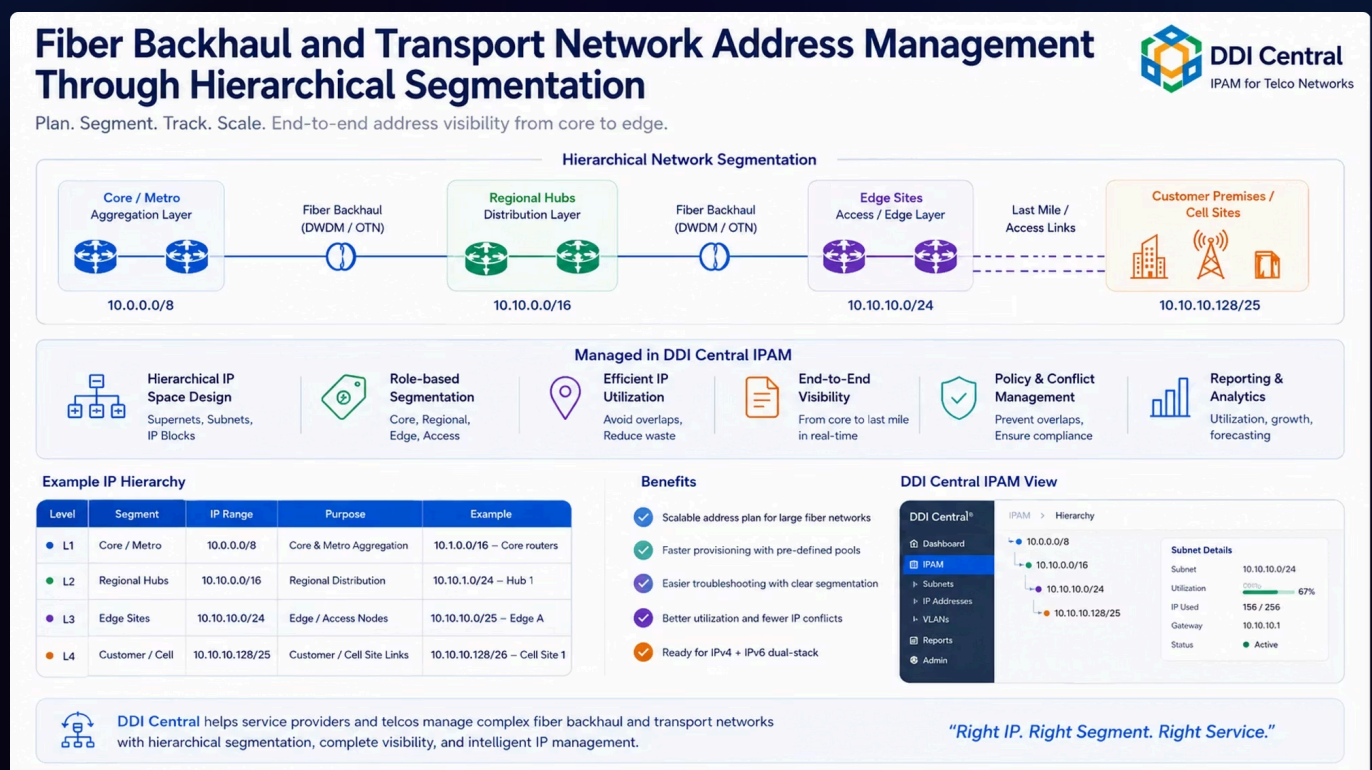


Figure 25 — Hierarchical segmentation from core to cell site: each level of the network receives a block carved from its parent, so the address itself tells you where you are — and DDI Central tracks utilisation and conflicts across every level.

The DDI Central mechanism

DDI Central's IPAM Tower supports **hierarchical segmentation**: a deliberate pyramid of address blocks in which each network layer receives a block carved from its parent – core/metro aggregation at the top (e.g., a /8), regional hubs beneath it (/16), edge sites below that (/24), down to customer and cell-site links (/25 and smaller). The address itself tells you where you are, like a postal system: country, state, city, street. On top of this hierarchy the platform provides role-based segmentation, real-time utilization tracking, conflict prevention, end-to-end visibility from core to last mile, and reporting for growth forecasting – ready for IPv4 and IPv6 dual-stack (running the current and next-generation address systems side by side).

Level	Segment	IP range	Purpose	Example allocation
L1	Core / metro	10.0.0.0/8	Core and metro aggregation	10.1.0.0/16 – core routers
L2	Regional hubs	10.10.0.0/16	Regional distribution	10.10.1.0/24 – Hub 1
L3	Edge sites	10.10.10.0/24	Edge / access nodes	10.10.10.0/25 – Edge A
L4	Customer / cell	10.10.10.128/25	Customer and cell-site links	10.10.10.128/26 – Cell Site 1

The payoff is practical. When a new edge site or cell site is added, nobody hunts through a spreadsheet for a free block or worries about overlapping an existing one: DDI Central identifies the right segment, assigns the right subnet, and keeps utilization and status visible across every level from core to last mile.

One-line takeaway: "Right IP. Right segment. Right service." – a scalable address plan where the structure itself is documentation.

6.4. Split-Horizon DNS — same queries, tailored destinations

The problem

Applications across all slices ask for the same service names — say, **upf.core.local** (the address of the User Plane Function). But the right answer differs by asker. A broadband phone should reach the broadband UPF; a factory robot must reach the local low-latency UPF; a monitoring tool needs the management endpoint. One name, four correct answers — and giving the wrong one either leaks traffic across slice boundaries or ruins latency.

The DDI Central mechanism

Split-horizon DNS (also called per-slice DNS views) means the DNS server inspects **where a query comes from** — the source slice, VRF, or view — and returns the answer appropriate to that context. (A **VRF**, Virtual Routing and Forwarding instance, is essentially a private routing table — one lane's private road map.)

For example, the same query for **upf.core.local** returns 10.10.1.20 to the eMBB slice, 10.20.1.20 to URLLC, 10.30.1.20 to mMTC, and 10.40.1.20 to the management plane. Each slice also gets its own authoritative zones, so it can only ever see the addresses it is authorized to reach — segmentation enforced at the DNS layer itself. And because resolvers sit co-located with the 5G core, answers come back in under a millisecond; queries never leave the building.

How DDI split-horizon DNS works for 5G slice segmentation

Same FQDN, different answers based on the source slice or segment

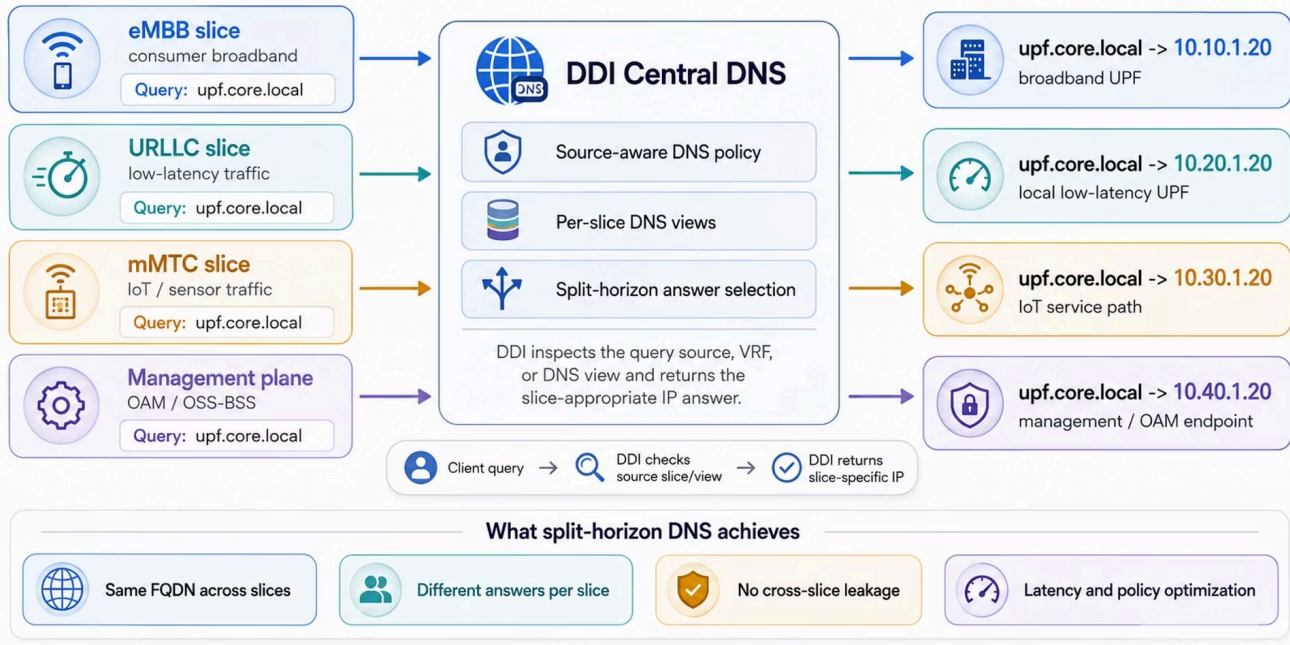


Figure 26 – Split-horizon DNS in action: four slices ask for the same name, `upf.core.local`, and each receives the address that is correct for it – broadband, low-latency, IoT, or management.

The story that makes it stick

Ask any employee in a large campus "where is the cafeteria?" and the correct answer depends on which building they are standing in. Split-horizon DNS is a directory service smart enough to know which building you are calling from.

One-line takeaway: same name everywhere, right answer every time, no cross-slice leakage – and sub-millisecond fast.

6.5. Automated Slice Spin-Up via REST API — A network slice in under 60 seconds

The problem

Slices are products. When a customer orders a private slice — say, an enterprise AR/VR service — the network must allocate an IP block, create DNS zones, and provision DHCP scopes for it. Done by hand across three tools, that is hours of tickets and a fertile field for typos.

The DDI Central mechanism

DDI Central exposes a full **REST API** — a standard web interface through which other software can drive it. When the 5G core's session manager (SMF) signals a new slice, the orchestrator makes one API call, and DDI Central performs the whole addressing lifecycle **atomically** (all-or-nothing, so no half-configured slices): it allocates the IP block, creates and registers the DNS zone, and provisions the DHCP scope with the right lease policy. The deck's example is an enterprise AR/VR slice at a Chennai edge data center: block 10.45.20.0/24 allocated, zone arvr.enterprise5g.local registered, scope 10.45.20.1-254 created with short edge-device leases — **slice bring-up in under 60 seconds**, versus hours manually. Every lifecycle event is triggered and tracked without manual intervention, and admins can investigate and validate calmly instead of firefighting.



Automated Slice Spin-Up via REST APIs

Provision IPAM, DNS, and DHCP automatically when the SMF triggers a new network slice.

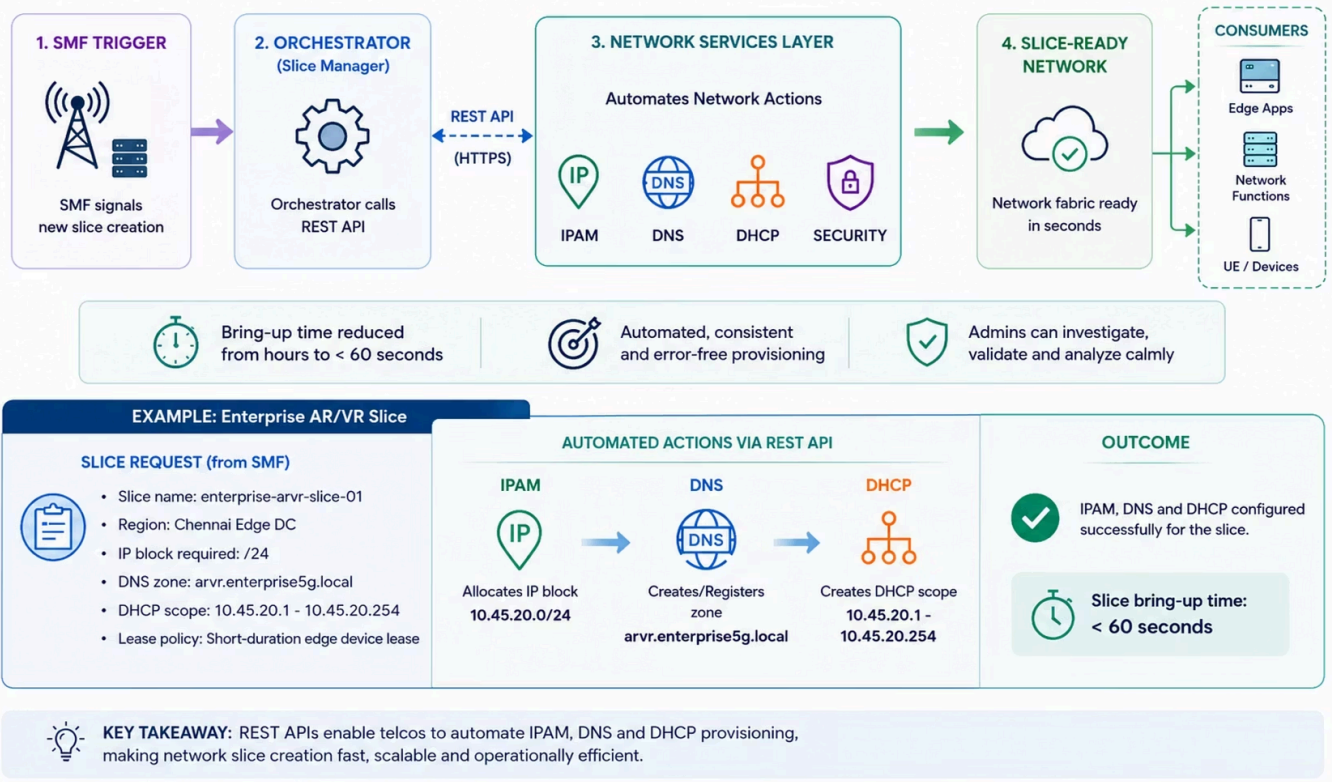


Figure 27 – Automated slice spin-up: the SMF signals a new slice, the orchestrator makes one REST call, and IPAM, DNS, and DHCP are provisioned in sequence – bring-up time falls from hours to under 60 seconds.

One-line takeaway: this is what "programming the network" literally looks like – a slice ordered like a product and delivered like a software deployment.

6.6. Zero-Touch Provisioning with PXE Boot — 400 servers install themselves

The Scenario

A 5G core capacity expansion: 400 bare-metal servers arriving over four weeks at one regional data center, destined for four different roles — UPF workers, AMF/SMF control-plane nodes, gNB baseband units (the compute behind radio base stations), and MEC edge nodes. One boot server holds the software images. Marketing has already announced the capacity uplift: the deadline is 30 days. Manually imaging 400 machines would consume the entire window.

The DDI Central mechanism

PXE boot lets an empty server, on first power-up, ask the network for its operating system. The DHCP response tells it where the boot server is and **which image file to fetch**. DDI Central turns this into a policy: **one DHCP template** with conditional logic that reads each server's factory-set vendor-class identifier (Option 60 again — e.g., "5GCORE-UPF-WORKER") and serves the matching boot path from the same boot server: UPF machines get the UPF image, control-plane machines the control-plane image, baseband units a realtime OS image, edge nodes the edge image. And critically, there is an **else** branch: any unrecognized device is booted into a quarantine image and the security team is alerted — an unknown machine plugged into the data center never reaches production.

The 25-minute autonomous boot loop

Root to shoot, this is the entire sequence — and no human appears in it after step 1:

1. The server is racked and powered on. **Its firmware sends a DHCPDISCOVER** carrying Option 60: 5GCORE-UPF-WORKER.
2. DDI Central matches that string and answers with the boot server address (10.10.50.4) and the exact file to fetch (/boot/upf/rhcos-dpdk.efi).
3. The server fetches the **bootloader** over TFTP. iPXE then pulls the kernel, the initial RAM disk, and the configuration over HTTP. The operating system installs, configures its data-plane acceleration, and joins the Kubernetes cluster.
4. The machine is a production UPF — total elapsed time under 25 minutes, with nobody touching a keyboard.

The one thing that makes the whole loop work is **Option 60**. The hardware vendor ships a manifest stating which MAC address carries which vendor-class string, so the DHCPDISCOVER a server sends already announces what it is. DDI Central never has to guess. The work moves upstream to the factory, where it is a single line of configuration, instead of downstream to the data center floor, where it is manual labour multiplied by four hundred.

Templates > Create Template DHCPv4

[Page Tips?](#)

NAME * 5G-Core-ZTP-PXE

TYPE * PXE boot

MATCH SUBSTRING

MATCH STRING * option vendor-class-identifier

OFFSET * 0

LENGTH * 17

MATCH VALUE * 5GCORE-UPF-WORKER

NEXT SERVER * 10.10.50.4

CONDITIONAL STATEMENT ☒

CONDITIONAL STATEMENT

if option vendor-class-identifier = 5GCORE-AMF-SMF-CP { filename "/boot/cnf-cp/rhcos-cnf.efi"; } 

else if option vendor-class-identifier = 5GCORE-GNB-RADIO1 { filename "/boot/gnb/ubuntu-rt.efi"; } 

else if option vendor-class-identifier = 5GCORE-MEC-EDGE01 { filename "/boot/mec/rhcos-edge.efi"; } 

else { filename "/boot/rescue/quarantine.efi"; } 

Cancel Save

Figure 28 — One PXE boot template with conditional logic: the vendor-class identifier selects the boot image — UPF, control plane, gNB, or MEC edge — all from the same boot server at 10.10.50.4, with an else branch that sends anything unrecognized to a quarantine image.

The PXE Boot template, field by field

- **Name** — 5G-Core-ZTP-PXE. A label for your records.
- **Type** — PXE boot. Tells DDI Central this template controls boot parameters in DHCP responses.
- **Match substring** — the Option 60 fingerprint check: read from offset 0, take 17 characters, and compare them to 5GCORE-UPF-WORKER. That is the default match, because UPF workers are the majority of the fleet.
- **Next server** — 10.10.50.4. One boot server, one address, for all four roles; only the filename changes.
- **Conditional statement** — switched on. Three branches cover the control-plane, gNB, and MEC roles, each returning a different boot file from that same server.
- **The else branch** — the safety net. Anything that matches none of the four known roles — a misconfigured BMC, a contractor's laptop in the wrong jack, hardware nobody expected — is routed to a quarantine image and flagged. The network does not answer strangers with production access.

Why this scales

- **No per-server configuration** — the OEM-set identity string carries the role.
- **No DHCP scope sprawl** — one scope plus one template covers all four roles.
- **No race conditions** — IPAM pre-stages reservations, so leases bind cleanly as machines arrive.
- **Auditable by default** — every DHCP transaction is logged with the branch it matched.
- **Fail-safe** — unknown devices land in quarantine, never in production.

The story that makes it stick

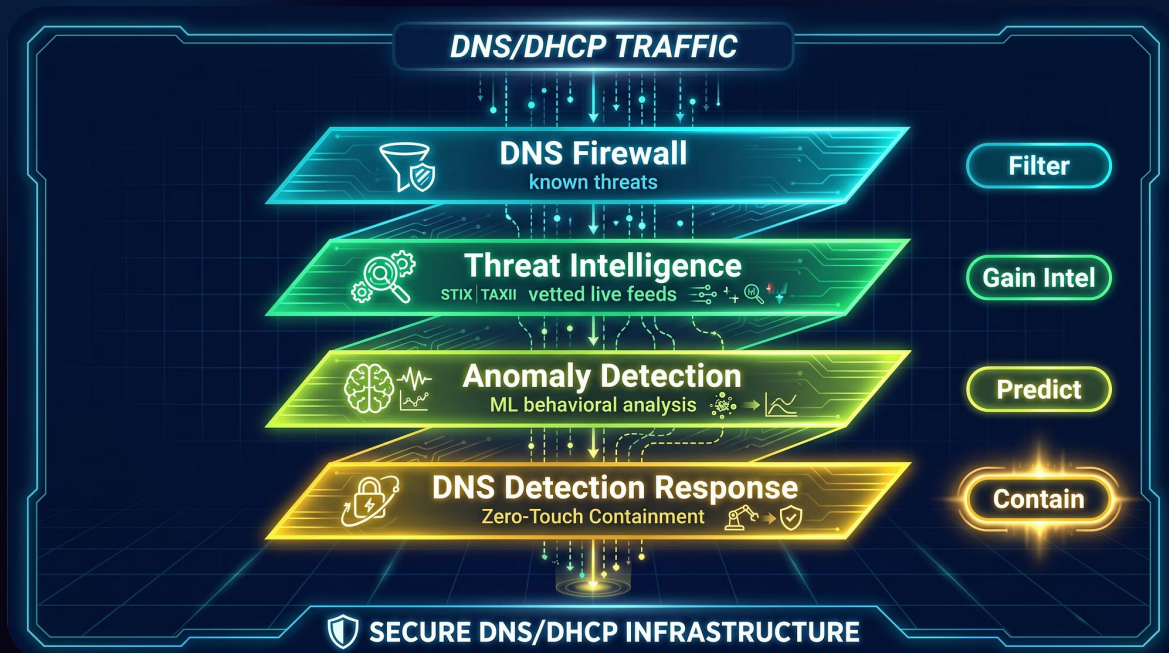
It is a company where every new employee onboards themselves: each arrives wearing a factory-issued badge stating their role, the front desk reads the badge and hands over the exact starter kit for that role — and anyone with an unrecognized badge is escorted to a waiting room while security is notified. The loading dock becomes a live 5G node with no per-server touch.

One-line takeaway: one template, 400 servers, zero conflicts — and a built-in immune system for strangers.

6.7. Defense-in-Depth DDI Security

Why attackers love DNS and DHCP

Everything must use them. Malware reaches out to home through DNS; data is smuggled out disguised as DNS queries ("DNS tunnelling"); roguedevices grab leases; flooding attacks starve address pools. Guarding the DNS/DHCP layer means guarding the whole network — so **DDI Central offers Defense-in-Depth security wrapped up in four layers**. If a threat bypasses one, it meets the next.



Layer 1 — DNS Firewall: known bad, gone in milliseconds

Using **Response Policy Zones (RPZ)** — blocklists enforced inside the DNS server itself — queries to known malicious domains are blocked or redirected at the source. **Response Rate Limiting (RRL)** caps how many answers the server sends per second (attackers and botnets use open servers to multiply their traffic at a victim). Full security analytics log every DNS event centrally, so the SOC (Security Operations Centre) has data — evidence — in real time.

Layer 2 — Threat Intelligence: if the world knows the threat, you know it too

Curated, vetted feeds from trusted vendors update the platform in real time to blunt zero-day campaigns; operators can also bring their own intelligence via the **STIX/TAXII** open standards for describing and delivering threat data. Connect once, defend continuously — with a forensic trail of DNS query patterns for threat hunting and compliance.

Layer 3 — Anomaly Detection: Catching the unknown enemies

Machine-learning behavioral screening watches DNS and DHCP for patterns no signature list knows yet: **DGA domains** (algorithm-generated, random-looking names malware uses to find its controllers), tunnelling behaviour, fast-flux infrastructure, and volumetric amplification attacks. Baselines tighten over time, exposing genuinely weak signals early — before threat feeds catch up. Unknown threats leave footprints; DDI Central reads them before they detonate.

Layer 4 — DNS Detection & Response: automated containment

When something is confirmed bad, response is automatic: **DNS quarantine** instantly blocks queries to flagged domains; **DHCP quarantine** flags offending MAC addresses and blocks their leases in real time — the network refuses to hand the intruder a key in. Forensic activity trails are captured automatically, and auto-triage lets analysts investigate without rushing — cutting attacker dwell time by up to 90%.

6.7.1. What only a DNS-DHCP integrated IPAM (DDI) can see

Two capabilities deserve special mention because they are impossible anywhere else in the building. The first is **rogue DHCP detection**: an unauthorized server handing out addresses is the digital equivalent of a stranger handing out keys — DDI Central spots it the moment it answers a probe, before any device accepts a rogue lease. The second is **IPAM**: every address is supposed to exist, so an impostor is visible the moment it starts operating. The second is the

Forensic timeline: because every lease, renewal, and release is recorded, DDI Central can answer the question every investigation eventually reaches — which device held which address, at which precise moment, on which part of the network.

This is not DDI doing the security team's job. It is DDI giving the security team a kind of visibility that didn't exist before DDI's architecture, because nothing else sees every device, every time, the way the addressing layer does.

One-line takeaway: four concentric shields — block the known, learn the global, detect the unknown, contain the confirmed — wrapped around the two services every device must use.

6.8. The Ledger That Powers Every Bill

The capability nobody expects from an addressing platform

Every bill a telco sends, every usage record, and every subscriber dispute traces back to the same underlying fact: a particular device held a particular address for a particular window of time. That fact is not created by the billing system. It is created by DDI, as an ordinary byproduct of doing its job.

- **Every lease is a timestamp.** When an address is handed out and when it is returned are both recorded, automatically.
- **Every slice move is a record.** If a device shifts between slices, there is a record of which network it belonged to, and when.
- **The address already encodes the service tier.** Because each slice draws from its own dedicated range, a premium low-latency subscriber and an ordinary consumer subscriber are distinguishable from their addresses alone — no extra instrumentation required.
- **Tenant usage was never commingled.** Isolated address spaces mean per-tenant utilization is a fact on file rather than an estimate to be apportioned.

Where it pays for itself: the moment of dispute

A subscriber who insists they were not connected at the time they were billed can be answered definitively, because the addressing system kept an exact, timestamped record of who held what and when. A wholesale partner questioning a capacity charge can be shown a per-tenant record rather than a calculation. What looks from the outside like a billing team doing careful detective work is, underneath, DDI Central's ordinary records simply being read.

One-line takeaway: the addressing layer is also the metering layer. If you can prove who held what and when, you can bill it, defend it, and audit it.

7. The Outcomes

A modern telco data center becomes faster, more efficient, and more secure when IP, DNS, and DHCP are managed as one system. Concretely:

- **Accelerate** provisioning and service delivery with automation – slices in seconds, servers that install themselves, faster deployments.
- **Reduce effort** by unifying IP, DNS, and DHCP operations – less manual work, fewer reconciliation errors, lower operational overhead.
- **Strengthen security** with policy-based control and visibility – reduced risk, reduced errors, and defense in depth around the most abused services.
- **Improve utilization** – overlapping-range multi-tenancy and hierarchical planning squeeze far more value from the same infrastructure.



Figure 29 – The outcome in one picture: a modern telco data center becomes faster, more efficient, and more secure when IP, DNS, and DHCP are managed as one system.

7.1. Why this matters to more people than you might think

It would be easy to assume this is a story only network engineers care about. It is not. The way a telco manages its addressing infrastructure determines how quickly it can launch a service, how confidently it can promise a customer dedicated isolated capacity, how credibly it can commit to five-nines uptime, and how defensible its security posture is when regulators or auditors come asking. When you walk into a room, these are the people in it — and this is what each of them gets:

Stakeholder	What DDI Central changes for them
Network operations	One source of truth instead of three; fewer manual steps; troubleshooting that starts with an answer rather than a reconciliation.
Architects	Isolation and segmentation designed into the foundation rather than bolted on afterwards.
Security leads	Visibility no other system has, enforcement at the first hop, and a forensic timeline that already exists when it is needed.
Product teams	A slice is a product, and it ships in under a minute through an API call rather than a week of tickets.
Commercial and SLA teams	Dedicated, isolated capacity that can be promised, metered, evidenced, and defended in a dispute.
Leadership and finance	Faster launches, lower operational overhead, fewer incidents, and better utilization of infrastructure already paid for.

DDI is the thread running underneath nearly everything else. It is rarely the headline — and almost always the reason the headline was possible.

7.2. Conversation starters that map to capabilities

When the customer says...	You now know to talk about...
"Onboarding a new tenant/MVNO takes weeks of renumbering."	Cluster-based modular isolation with overlapping RFC 1918 IP ranges (Refer Section 6.1)
"We hand-configure every device class and it never ends."	DHCP fingerprinting and client classes (Refer Section 6.2)
"Traffic keeps crossing slice boundaries / latency budgets slip."	Split-horizon DNS views and VRF-aware DHCP (Refer Section 6.4)
"Slice creation is a week of tickets across three teams."	REST-API slice spin-up in under 60 seconds (Refer Section 6.5)
"Capacity expansions blow through our maintenance windows."	PXE boot provisioning from one template (Refer Section 6.6)
"Our address plan is a spreadsheet nobody trusts."	Hierarchical IPAM from core to last mile (Refer Section 6.3)
"We can't see or stop DNS-borne threats."	The four-layer DDI security stack: DNS FireWall (DNS FRW) Policies (DNS SinkHoling, DNS Response Policy Zones, DNS Rate Limiting, Threat intelligence, ML driven anomaly detection, DNS Detection & Response (DDR) (Refer Section 6.7)

8. The Closing Idea — From Configuring To Programming IP Networks

Here is the thought to leave with every stakeholder. Historically, IP networks were configured: humans typed addresses into boxes, and the network was only as consistent as the humans were careful. With DDI Central as the control plane, the network is programmed: identity drives policy, policy drives allocation, allocation drives records, and every step is automatic, auditable, and reversible. A new slice is an API call. A new tenant is a new sealed instance. A new server is a badge read at the door. A threat is a lease refused.

When IP, DNS, and DHCP behave as one system, programming the IP network becomes easy — and DDI Central is the system that makes them one.

See It For Yourself

Reading about a control plane is one thing. Watching a network slice spin up in under a minute, a device get fingerprinted and policy-matched in real time, and a single name resolve four different ways depending on who is asking is an entirely different kind of understanding.

- [Start a 30-day free trial](#) of ManageEngine DDI Central and explore the platform inside your own environment, at your own pace, with no commitment attached.
- [Book a personalized demo](#) and have the team walk through your specific telco architecture, your specific scale, and your specific questions, live.