Description

Title

Plug-and-Play Infusion Pump Onboarding and Compliance Management System

Field of the Invention

The present invention relates to infusion pump integration and management in clinical environments. In particular, it concerns systems and methods for automatically detecting and onboarding infusion pumps of various models and firmware versions without manual configuration, and for maintaining a live inventory of such devices to facilitate regulatory compliance (e.g., Joint Commission requirements) and firmware/library upgrade planning.

Background

Healthcare facilities often deploy **smart infusion pumps** from multiple manufacturers, each with proprietary communication protocols or HL7 messaging formats for transmitting pump status and alerts[1][2]. Historically, integrating these pumps with hospital information systems has required vendor-specific gateways or bespoke interface configurations, as each manufacturer's device speaks a slightly different "language"[3]. For example, one pump model might output HL7 v2 alarm messages over an Ethernet or serial link, while another uses a custom serial data stream for events and vitals[1]. This heterogeneity makes *plug-and-play* interoperability challenging; biomedical engineers must often perform manual setup or write custom parsers for each new pump type, incurring significant labor and risk of error.

Moreover, maintaining **compliance** with safety standards is an ongoing challenge. The Joint Commission and other bodies expect hospitals to use **Dose Error Reduction Software (DERS)** and keep drug libraries up-to-date on all pumps[4]. In practice, however, compliance gaps occur: some pumps may be running outdated drug libraries or firmware, or staff may bypass the safety software if the library is incomplete[5]. Tracking the configuration of each pump (model, software version, library checksum, etc.) is typically done via periodic manual audits or static spreadsheets, which can quickly become outdated. A need exists for an automated solution to *continuously discover, inventory, and assess infusion pumps* across a facility, ensuring that outdated configurations are flagged and guiding timely upgrades to improve patient safety.

In summary, current systems lack a unified **plug-and-play** onboarding mechanism for diverse infusion pumps and an integrated **compliance management** tool. The present invention addresses these gaps by leveraging real-time data "sniffing" and adaptive protocol parsing to auto-detect devices, combined with a cloud-backed inventory that highlights compliance issues and recommends upgrade paths.

Summary of the Invention

Embodiments of the invention provide a **two-part system** for infusion pump integration and management. In **Embodiment 1**, a plug-and-play edge gateway automatically identifies an infusion pump's make, model, firmware version, and communication protocol by *sniffing* the pump's data output in real time. This gateway device requires zero manual configuration – as soon as a pump is connected (physically or over the network), the system's adaptive parsing engine analyzes incoming data streams (e.g. HL7 v2 messages or proprietary serial packets) to recognize protocol patterns and device identifiers. For example, the gateway can detect an HL7 message header or a vendor-specific frame format and thereby infer the pump model and firmware revision encoded in the data[2]. Once identified, the pump is automatically onboarded into the system's device registry without human intervention, **enabling true zero-touch onboarding** of new infusion pumps.

In **Embodiment 2**, a cloud-based inventory and upgrade planning platform aggregates data from all such gateway devices to build a live, evolving map of the infusion pump fleet in the hospital. This inventory records each pump's model, software/firmware version, network or serial communication characteristics, and **compliance-relevant attributes** like the drug library version or checksum currently in use. The system continuously updates this map as pumps are added, moved, or updated. Using these live records, the platform automatically **analyzes compliance gaps** by comparing the deployed configurations against latest safety requirements and manufacturer updates. For example, if a pump's drug library file is outdated or a firmware version is known to be non-compliant with the hospital's policies, the system flags it. The platform then generates **staged upgrade proposals** – a prioritized plan to bring all pumps into compliance. This plan can recommend, for instance, upgrading firmware on a batch of pumps in a staged manner (to minimize clinical disruption), or replacing legacy pumps that lack DERS with newer models, all mapped out with scheduling and grouping suggestions.

Both embodiments are designed with robust security and safety measures. The plug-and-play gateway uses **token-based authentication** (e.g. JWT or OAuth2 tokens) to securely connect to the cloud service, ensuring that only authorized devices send data to the inventory system[6]. Communication between the edge and cloud can occur via lightweight IoT protocols like **MQTT** or via HTTPS, providing flexibility in hospital IT environments while maintaining encryption and reliability. The gateway operates in a **fail-safe, non-intrusive manner**: it passively taps into the pump's data stream or sits in-line with a fail-open bypass, so any malfunction of the gateway will not disrupt the pump's normal operation. All critical events – such as detected pump identities, status messages, and compliance alerts – are logged with cryptographic signatures, producing an **immutable audit trail** of device data and system decisions[7]. This audit trail, which can be anchored to a blockchain or secure ledger, ensures tamper-evident records of infusion pump activity and any configuration changes, thereby supporting both cybersecurity and regulatory requirements for traceability.

In essence, the invention provides a *comprehensive infusion pump management solution*: from **automatic device onboarding** at the bedside (or wherever the pump is connected) to **centralized oversight** of device status and safety compliance across the healthcare enterprise. Clinically, this means biomedical and nursing staff can trust that any pump plugged into the system will immediately be recognized and monitored, and administrators can continuously verify that all pumps meet the latest safety standards (for example, 100% of pumps using the

active drug library, fulfilling the Joint Commission's expected practice of DERS usage[4]). Technically, the system's adaptive, extensible design allows new pump models or protocols to be accommodated via software updates or cloud-distributed parsing rules, "future-proofing" the hospital's infrastructure against device obsolescence. By combining plug-and-play connectivity with intelligent compliance analytics, the invention greatly reduces manual effort, helps prevent medication errors due to outdated pump settings, and maintains a secure audit-ready log of all infusion device activities.

Brief Description of the Drawings

- **FIG. 1** is a schematic architecture diagram of the plug-and-play infusion pump monitoring system, illustrating an edge gateway connected to multiple infusion pumps and a cloud platform that maintains the centralized inventory and compliance engine.

- **FIG. 2A** shows a flowchart of the **automatic pump detection process** in the edge gateway. Incoming data from a newly connected pump is sniffed and run through multiple protocol detectors to identify the communication format (HL7, proprietary serial, etc.), after which the model and firmware are determined and the device is onboarded.

- **FIG. 2B** is a continuation of the flowchart of FIG. 2A, detailing the subsequent steps of initiating secure communication. Once a pump is identified, the gateway generates an authentication token, establishes an MQTT/HTTP session with the cloud, and begins streaming the pump's data. Safety failover steps (such as buffering data during network outages and ensuring passive data tap) are also illustrated.

- **FIG. 3A** depicts an example **inventory dashboard or data structure** maintained in the cloud for all detected pumps. It highlights entries for each pump (identified by unique ID or network address), along with fields for model, firmware version, drug library checksum, last seen time, and compliance status (e.g., flags for outdated library or required firmware update).

- **FIG. 3B** illustrates a **compliance analysis and upgrade planning workflow**. This flow (or diagram) demonstrates how the system aggregates the inventory data, compares it against compliance rules and update availability, and then generates prioritized upgrade tasks. FIG. 3B also shows an example output, such as a timeline or report that recommends upgrading certain pumps in stages (e.g., ICU pumps first) and provides cryptographically signed validation once upgrades are completed.

(It will be understood that these figures are for illustrative purposes. Like-numbered elements in different figures represent the same or analogous components of the system.)

Detailed Description of Embodiments

Embodiment 1: Plug-and-Play Auto-Detection of Infusion Pumps

System Overview: As shown in FIG. 1, the system includes one or more **edge gateway devices** (100) deployed in the clinical setting, and a cloud-based central management server (150). Each edge gateway 100 is a small computing device (e.g., an IoT appliance or embedded controller)

positioned to intercept or tap into the data streams of one or more infusion pumps (10). In a typical use case, an infusion pump 10 may be connected to a hospital network or send data to a clinical workstation. The gateway 100 is interposed in this path or connected in parallel (non-intrusively) to sniff the data. For example, the gateway can plug into the serial port or network switch that the pump uses, thereby receiving a copy of all communication. Crucially, the gateway's design is **non-blocking** – it does not interfere with the pump's own connections. In one implementation, the gateway uses a hardware **splitter or mirror port** so that all data packets from the pump are duplicated to the gateway while still reaching their primary destination (e.g., the hospital EMR interface). This ensures the pump's normal operation and any critical alarm pathways remain unaffected by the presence of the monitoring system[8]. Even in in-line configurations (for pumps that must connect *through* the gateway), the gateway includes a fail-safe relay or pass-through circuit that closes if the device loses power, thereby allowing pump data to flow unimpeded in the event of gateway failure. In all cases, the pump's clinical function remains **fail-safe** and isolated from any single-point failure of the monitoring system.

Automatic Protocol Detection: FIG. 2A illustrates the process flow for **real-time sniffing and protocol detection** executed by the gateway (100). When a pump is first connected or powered on (step 201 in FIG. 2A), the gateway's sniffing module (102) passively monitors the data line for any traffic. This data could be, for instance, an HL7 message stream, ASCII or binary proprietary messages, or other medical device communication. The system employs an **adaptive parsing engine** that iteratively tries to interpret the incoming bytes under various known protocols or formats (step 203). For example, if the data contains human-readable segments starting with the characters "MSH|^~\&", the engine recognizes the HL7 v2.x header of a message[2]. It will then parse the message according to HL7 standards (step 205) to extract key fields – such as a device identifier, message type, manufacturer, and firmware revision if provided. Many infusion pumps following the IHE Infusion Pump Event Communication (IPEC) profile will send periodic **ORU** (Observation Result) or **RAS** messages that include the pump's identity and status[9]. The gateway can capture these and decode fields indicating, for example, the pump model name or unique device ID embedded in a dedicated segment.

If the incoming data does **not** match HL7 structure, the engine moves through a library of known proprietary formats (step 207). The system can store templated "fingerprints" of common pump protocols – for instance, a particular vendor's pump may begin each message with a fixed byte sequence or delimiter (such as 0x02 0xFD for start-of-text, etc.), or the data may be in a fixed-length record format unique to that model. The gateway tries each pattern/rule set in turn. Upon finding a match (decision 209), it selects the corresponding protocol plugin to fully parse the stream. This adaptive approach allows the system to **plug-and-play with multiple manufacturers**: the parsing logic for each supported protocol can be updated or added via software, without changing the hardware. In effect, the gateway acts like a "universal translator" that listens to the pump's native tongue and translates it into a common data representation internally[10]. In some embodiments, machine learning or heuristics may assist the detection (e.g., classifying the protocol by message entropy, timing, or character frequency), but rule-based matching of known signatures is typically used for reliability and transparency.

Device Identification and Onboarding: Once the protocol is identified and the data is being parsed correctly, the gateway extracts the **device identity information** (step 211 in FIG. 2A). Depending on the protocol, this could be found in different ways. For an HL7 message, the **MSH** (Message Header) segment might contain a field for sending device or application which

encodes the pump model/ID. Some systems use a **Z-segment** (custom segment) to convey device details like serial number or software version. In proprietary protocols, an initial handshake packet or a status message often contains a device type code and firmware version string. For example, a pump might periodically send a text line like "Device: XYZ Pump v3.4" or a binary code that the gateway's parser knows corresponds to a certain model. The invention includes a **model recognition database** (stored locally on the gateway or fetched from the cloud) that maps known identifiers to specific pump models and firmware versions. By consulting this database, the gateway can definitively determine, *for instance*, "Pump #1234 is an **Alaris GX model running firmware v1.2.3**" based on the parsed data.

With the model and version identified, the gateway 100 then automatically **onboards** the device into the system (step 213). Onboarding entails creating a record for the pump in the gateway's memory and preparing an announcement to the central server (150). This is a zero-touch process – no user input is required. The system may log a message like "Detected new device: Alaris GX, FW 1.2.3, ID 1234" and mark it as active. Importantly, the gateway can also detect *when a pump disconnects* or is powered off by noticing an absence of data or an explicit disconnect message; it will then mark the pump as offline in its records and inform the cloud, ensuring the inventory is always up to date in near-real-time.

Secure Communication Initialization: FIG. 2B shows the sequence where, after identifying the pump, the gateway establishes a secure channel to transmit data to the cloud server. In step 221, the gateway's network module (104) generates a **token-based authentication credential** for this session. In one preferred approach, the gateway obtains a JSON Web Token (JWT) or similar from an authentication service, proving its identity and permissions to the cloud[6]. The token may embed the gateway's ID and an expiration time, and is signed by a secret or private key known to the hospital's system, preventing unauthorized devices from injecting data. Using this token, the gateway opens a connection to the cloud's message broker (step 223). The connection could be an **MQTT** publish/subscribe channel over TLS or an HTTPS RESTful API, depending on implementation. MQTT is advantageous for streaming device data efficiently and in near-real-time; the gateway can publish a topic for each pump or each message type. For example, it might publish a JSON payload like {"pump_id": "1234", "model": "AlarisGX", "infusion_rate": 5.0, "alarm": "occlusion"} to a topic such as hospital/pumps/1234/events. The use of token-based auth with MQTT ensures that only clients with a valid token (the edge gateways) can publish to the broker, and all traffic is encrypted[11].

Once the secure channel is up, the gateway begins **dispatching messages** (step 225). Every piece of data parsed from the pump – be it periodic status, alarm events, or infusion parameters – is sent upward. The system normalizes these messages into a standard format (for instance, converting proprietary fields into a unified schema). This normalization allows the cloud platform to ingest data from different pump models in a uniform way[12]. In parallel, the gateway implements any **local safety failovers**: if network connectivity is lost or the cloud is unreachable, the gateway buffers the incoming pump data locally (e.g., in a circular memory buffer or on disk). It can continue to store hours or days of data if needed. When connectivity is restored, the buffered data is forwarded to ensure no information was lost during the outage. This design means that even if the hospital network has downtime, the pump's data and the audit logs remain complete (the system's cloud database will simply get a delayed transmission). Additionally, because the gateway's sniffing is *non-blocking and very low-latency*, it does not introduce any meaningful delay into the pump's communications. In practice, the time to copy

and process a message is on the order of milliseconds – e.g., replication latency under 2 ms so as not to **back-pressure** the real-time bus[13]. The pump and any connected clinical system see no difference in performance, satisfying a key safety requirement.

Adaptive and Extensible Parsing: It should be noted that the gateway's protocol detection module (102) is **extensible**. New infusion pump protocols can be added via software updates. In some embodiments, the gateway periodically checks the cloud (or a manufacturer's repository) for updates to its protocol library. These updates might include new pattern signatures, parsing rules, or even code modules for entirely new models. This enables **per-model customization** without needing to replace or manually reconfigure the gateway. For example, if a hospital acquires a new pump model that was not previously seen, the system can download a profile for that model (perhaps provided by the pump vendor or crowdsourced from a library of devices). The next time such a pump is connected, the gateway will recognize it using the updated rules – achieving plug-and-play support even for devices introduced after the system's deployment. This adaptive approach is far more scalable than having to pre-program every possible device at installation. It contrasts with existing "connector" approaches where each new device type might require writing a custom interface module[14]. Here, the heavy lifting is done by the system's ability to learn new protocols in the field, minimizing human intervention.

Local Processing and Alerts: While the primary role of the gateway is identification and forwarding, it can also perform certain local analytics or safety checks. For instance, if a pump issues a critical alarm (e.g., an overdose alert or pump failure), the gateway can immediately raise a local alert (such as an LED indicator or audible alarm) in addition to sending it to the cloud. This provides an extra layer of assurance that urgent events are noticed, even if network latency is present. The gateway could also filter or rate-limit some data if configured (though generally it forwards everything to ensure the cloud has the full picture). In essence, the gateway is a smart conduit with enough processing to ensure real-time, reliable data capture and preliminary interpretation, but the heavier tasks of compliance analysis are left to the cloud side in this design.

Embodiment 2: Live Inventory Mapping and Compliance Upgrade Planning

Once the edge gateways have auto-onboarded pumps and started streaming data, the **cloud platform** or central server (150 in FIG. 1) aggregates this information to maintain a comprehensive, live **inventory** of all infusion pumps in the system. FIG. 3A schematically represents this inventory as a database or dashboard. Each newly onboarded pump results in a new entry (or an update to an existing entry if the pump was already known). The entry typically includes: the unique device identifier (which could be the pump's serial number or a network address), the **model and manufacturer**, the currently running firmware or software version, the communication protocol (e.g., "HL7/IPEC over IP" vs "RS-232 proprietary"), and **compliance-related fields**. One important field is the **drug library version or checksum** on the pump. Many smart infusion pumps have a drug library file loaded (containing medication safety limits); this library often has a version number or checksum that can be retrieved from the device's data output or via a query. For example, some HL7 pump messages include an identifier for the active drug library profile on the pump. The system captures this, storing a shorthand like "Library Ver: 2025-01, Checksum XYZ123". Other relevant attributes could include the pump's last calibration date, its network security status (e.g., whether it's communicating over an encrypted channel or not), and any recent error codes. By **visualizing the pump fleet** in one

place, the hospital's biomedical engineering and IT staff can see, at a glance, the state of all devices – something that previously might require walking around with clipboards or logging into each pump's vendor software individually.

Real-Time Updates and Tracking: The inventory map is not static. It updates in real time. If a pump is physically moved and plugs into a different gateway, the cloud recognizes it (by the device ID) and can update the location or gateway association. If a pump's firmware is upgraded (manually by technicians or via the vendor), the next message from that pump would reveal a new version number – the system will automatically update the inventory entry to reflect the change. Similarly, if a new drug library file is deployed to pumps, the checksum field will change on the next data report, and the system captures that. This continuous synchronization means the inventory is **always current** within minutes; it obsoletes the periodic manual audits for these data points. Additionally, the platform can integrate with other hospital databases – for instance, pulling in purchase or maintenance records – to enrich each device entry with info like purchase date, warranty status, or location/department assignment. (Such integrations are optional but enhance the planning tool aspect.)

Compliance Analysis Engine: FIG. 3B shows the workflow of the compliance analysis and upgrade planning engine (160) that operates on the inventory data. The system first ingests the raw inventory (step 301) – essentially reading all pump records and their attributes. It then cross-references each record against a set of **compliance rules and reference data** (step 303). These rules may be derived from hospital policy, industry standards, and manufacturer recommendations. For example, one rule might be: *"Drug library must be within one version of latest approved library for that care area"*. Another might be: *"Firmware version must be the latest security-patched version as per vendor bulletin (or no more than one version behind unless documented)"*. The Joint Commission's guidance that DERS (the drug library) should be fully utilized[4] effectively means pumps without the current library or with DERS disabled are out of compliance. The engine will flag such cases[5]. It also checks for **fleet uniformity** issues: if, say, out of 100 pumps of Model X, 20 are still on an old firmware that has known issues, those 20 will be flagged. Compliance gaps are thus identified (step 305) for each device or group of devices, producing a list of issues such as "Pump #1234: Library out-of-date", "Pump #5678: Firmware below minimum required version", or "5 pumps (Model ABC) have a security patch pending".

Beyond simply flagging issues, the system synthesizes an **action plan**. In step 307 (FIG. 3B), the engine prioritizes and groups the needed upgrades. It takes into account factors like criticality (ICU devices might be upgraded first due to higher risk), availability of replacements or firmware files, and scheduling (pumps currently in use on patients might be scheduled for maintenance at a later shift or when idle). The output is a **staged upgrade path** – for example, it may recommend *"Week 1: Upgrade firmware on 10 pumps in ICU (list serials ...); Week 2: Update drug library on all ER pumps; Week 3: Decommission 5 outdated pumps in storage and replace with new pumps Model Z"*. These recommendations can be presented via a dashboard, or even automated into work orders for the biomedical team. The stages are designed to incrementally reach full compliance without interrupting clinical operations. The system could also simulate the compliance status after each stage, showing that by the end of the plan, all gaps will be closed.

Joint Commission and Safety Standards Alignment: The compliance engine's knowledge base can be kept current with evolving standards. For instance, if new guidelines come out requiring a certain safety feature in pumps, the rules can be updated such that any pump lacking that feature is flagged. The system effectively acts as a continuous **audit assistant**, encoding regulations into automated checks[15][4]. By identifying issues proactively, it helps the hospital avoid the scenario of failing an external audit or, worse, discovering too late that an infusion error occurred on a pump that hadn't been updated. In one scenario, suppose the Joint Commission issues an alert that a certain older pump model has led to sentinel events unless a specific software patch is applied. The inventory tool can immediately search the database to find if any of those pumps are in use and if their software is at the vulnerable version. If yes, those are highlighted with highest priority to remove or patch, and the plan might be "replace or update these specific pumps *immediately* (Stage 0)." This kind of agility in response is a major advantage of the live inventory approach.

Cryptographically Signed Audit Trails: Every action and data point in the system is recorded in a secure, tamper-evident log. In some embodiments, the cloud platform maintains a **ledger** (170) that receives event records from both gateways and the compliance engine. Each record might include a timestamp, the device/pump ID, the data or event (e.g., "Pump 1234 detected, firmware 1.2.3" or "Pump 1234 flagged for library update"), and a digital signature or hash. As noted earlier, gateways can sign events at the source using HMAC-SHA256 or ECDSA signatures[7]. These signatures are verified and recorded in the cloud ledger. The ledger can be implemented on a blockchain network or an immutable database with append-only entries[16]. For example, an internal Hyperledger Fabric blockchain may be used to log each event, with smart contracts enforcing that entries cannot be altered or deleted[17]. The result is an **audit trail** that is cryptographically verifiable. Months or years later, the hospital can prove exactly which pumps were in use, what their settings were, and what upgrades were done, with a chain-of-custody for each record. If a compliance report needs to be generated for regulators, the system can output a signed report summarizing compliance status and referencing the underlying immutable records.

This audit mechanism is also crucial for **cybersecurity**. By hashing and chaining events, any attempt to fabricate or modify device data (for example, by a malicious actor) would be evident. Additionally, the system's use of cryptographic techniques extends to software updates themselves: when the compliance plan is executed and, say, new firmware is uploaded to a pump, the system can store the cryptographic checksum of the firmware and a signed confirmation that the pump accepted the update. This builds a trustworthy log of maintenance actions, which is part of maintaining 21 CFR Part 11 electronic records compliance in medical devices (tamper-evident records of changes)[7].

User Interface and Alerts: The cloud platform provides interfaces for users (biomedical engineers, IT admins, or clinical engineers) to interact with the inventory and plan. In one embodiment, a **dashboard** (FIG. 3A) presents a tabular or graphical view of all pumps with color-coded status (green for up-to-date, yellow for warning, red for non-compliant). The user can drill down into each device to see details like "Firmware 3.0 (update available: 3.2), Library XYZ (expires soon or outdated), Last calibration 15 months ago, etc." The system can generate **alerts or notifications** for significant issues: e.g., an email or message if a pump goes out of compliance or if a scheduled upgrade is overdue. There is also an **upgrade planning interface** (FIG. 3B could be conceptually represented as a Gantt chart or checklist) where the

recommended stages are listed. Users may adjust the plan (e.g. change scheduling or mark certain pumps as exceptions) through this interface. The system will log any such user decisions as well, maintaining a record of who approved or deferred each action.

Integration with Device Management Systems: The invention can function alongside existing clinical engineering systems. For instance, hospitals often use computerized maintenance management systems (CMMS) to track equipment inventory and maintenance schedules. The platform can export its inventory and compliance data to such systems or import data from them (like device location or responsible owner info). Likewise, it can integrate with **EHR** systems or device integration engines – for example, if a pump is not just monitored but also remotely controllable, the system could potentially facilitate safe updates to pump configuration. However, the primary focus of this invention is on *monitoring and advisory*, not remote control, thereby keeping it simpler and safer (no risk of altering pump behavior directly, which would require stringent validation).

Safety Failovers and Redundancy: On the cloud side, safety means ensuring high availability and consistency of the inventory data. The cloud server (or cluster) 150 is preferably deployed with redundancy so that if one node fails, another takes over, and no data is lost (especially since gateways buffer data if needed). The system could employ transactional integrity – for example, using a database that commits each pump record update atomically and perhaps even stores copies off-site. The cryptographic ledger may be mirrored to an off-site node or to the pump manufacturer's server in a consortium model[18], creating an extra backup of the audit data. In case of a major outage, gateways will cache data and later sync, as described, which is a form of failover. If a gateway itself fails, as noted, the pumps connected to it are unaffected (they continue to function and can be manually observed by staff). When that gateway is replaced, the new unit can re-discover the pumps and fill in the gap.

Example Use Case: To illustrate the system's operation across both embodiments, consider a hospital that has just installed the system. A nurse brings a **smart infusion pump** to a patient's bedside, plugs it into the network (which is connected to a gateway device 100 under the hood). Immediately, the gateway sniffing module sees HL7 messages from the pump. It identifies the pump as a **BrandX Model Y** infusion pump with firmware 2.1. The gateway creates a device entry and sends a secure MQTT message to the cloud: "New device: BrandX Model Y, FW 2.1, Library checksum ABCD1234". The cloud inventory (FIG. 3A) adds this pump. A compliance rule check runs and finds that firmware 2.1 is two versions behind the latest 2.3 (which, say, fixed a known dosing bug), and the drug library ABCD1234 is outdated compared to the current approved library checksum XYZ9999. The system flags this pump with two issues (one critical, one moderate). An alert is sent to the biomedical team's console highlighting that a pump in use is out-of-date. However, instead of just alarming, the system's plan engine (FIG. 3B) notes that **15 other pumps** of the same model in the hospital are also on firmware 2.1. It generates a coordinated plan: *Stage 1:* obtain firmware 2.3 from vendor and schedule update for these 16 pumps over the next 3 days (split among ICU, OR, etc. as appropriate); *Stage 2:* push new drug library XYZ9999 to all pumps by end of week. The biomed manager uses the dashboard to approve this plan. As each pump gets updated (either via the hospital's maintenance process or potentially via an automated update function if supported), the gateway detects the change (new firmware number or new checksum) and the inventory status for that pump turns green. The audit log records "Pump 1234 firmware updated to 2.3 on 2025-11-01 10:00 by tech ABC (confirmed via device report)", cryptographically signing this entry. At the end of the week, the

manager can print a compliance report showing 100% of pumps are now on the correct library and firmware, with evidence of each action – a powerful document for both internal quality assurance and external accreditation audits.

In sum, **Embodiment 1 and 2 work in tandem**: the former **automates connectivity and data acquisition** from a diverse set of infusion pumps (making the system easy to deploy and scale), and the latter **transforms the raw data into actionable insights** for safety and compliance (making the system immediately valuable for improving clinical practice and meeting regulatory demands). This holistic approach ensures that a hospital's infusion pump fleet is not a black box but rather a transparent, managed part of the health IT ecosystem – one that can evolve with new technology while continuously safeguarding patient care.