Description

Title

SmartStop-Based Pause-Directive Compiler for Infusion Management

Field of the Invention

This invention relates to clinical decision support systems and infusion management in healthcare. In particular, it concerns a compiler and control system for infusion “pause directives” that automatically converts standing-order templates into machine-readable instructions, enabling real-time pausing of infusions based on patient-specific lab results. The system ensures that automated pause actions remain within nursing scope-of-practice and comply with medical regulations, while integrating seamlessly with electronic health records (EHRs) and device workflows.

Background

Intravenous infusion therapies often require prompt adjustments when patient conditions change (e.g. critical lab values). Traditionally, hospitals use standing orders or protocols authorizing nurses to take certain actions (such as pausing or holding a medication) when predefined criteria are met. For example, a physician’s standing order might instruct: “If patient’s INR > X, hold Warfarin infusion and notify provider.” These are manually executed by nurses and documented after the fact. While effective, manual protocols introduce delays and reliance on humans to catch abnormal labs, and they burden staff with documentation tasks.

Modern “smart” infusion pumps with Dose Error Reduction Software (DERS) provide safety by catching programming mistakes (wrong dose/rate) before infusion starts . They enforce preset dosage limits to prevent extreme over- or under-dosing . However, current smart pumps and EHR alerts generally do not handle dynamic therapy adjustments based on real-time clinical data (like lab results) during the infusion. They focus on preventing user entry errors , but do not automatically pause or adjust an ongoing infusion if a patient’s lab crosses a dangerous threshold. It remains the nurse’s responsibility to monitor labs and intervene, which can lead to delays or missed events.

Furthermore, nurses can only act within their scope of practice. Nurse Practice Acts and hospital policies forbid nurses from independently changing infusion rates or therapies beyond physician

orders. Pausing or stopping an infusion under a physician-approved protocol is generally allowed, but more complex titrations often are not. Current systems lack automated checks to ensure that any suggested action (e.g. auto titration) stays within the nurse's authorized scope. This poses safety and legal concerns if automation oversteps what a nurse is permitted to do without direct physician instruction.

Another challenge is lab result reliability. Lab values that are physiologically implausible or marked as contaminated (e.g. hemolyzed blood samples) can trigger false alarms. For instance, a hemolyzed blood sample can spuriously elevate potassium levels; laboratories often hide or suppress such results due to the risk of overestimation . An automated infusion-pausing system must be smart enough to ignore or verify dubious lab results to avoid unnecessary therapy interruptions.

There is also a need for robust documentation and compliance. When a standing order protocol is executed (such as an infusion hold), regulations (e.g. CMS Hospital Conditions of Participation) require that it be recorded as an order in the medical record and later authenticated (signed) by the responsible physician . The nurse executing the order should not be falsely documented as the "prescriber" – credit must go to the physician-of-record who authorized the standing order. Keeping an audit trail of all such automated events (what triggered the pause, who acknowledged it, when infusion was resumed, etc.) is critical for quality control and regulatory audit. Hospitals also demand that any software performing clinical functions comply with FDA guidance on Software as a Medical Device (SaMD) (ensuring transparency, validation, and patient safety) and adhere to nurse practice regulations and other guardrails.

Finally, system uptime and fail-safes are crucial. In the event the clinical decision support (CDS) system or network is down (whether due to software failure or planned EHR downtime), clinicians must revert to the manual protocol. The solution must detect downtime and gracefully fall back to standard manual infusion management so that patient care continues safely without the SmartStop automation.

In summary, there is a need for an integrated solution that (1) compiles physician-approved standing order templates into executable infusion pause rules; (2) monitors live patient data and automatically pauses infusions when trigger criteria are met, within the bounds of nursing practice; (3) filters out invalid inputs (bad labs) to reduce false alerts; (4) alerts and guides clinicians in real time with appropriate escalation if needed; (5) documents all actions attributing them to the ordering physician; and (6) provides robust failover and compliance with all healthcare regulations.

Summary of the Invention

The present invention provides a SmartStop pause-directive compiler and infusion management system that addresses the above needs. In one aspect, the invention comprises a compiler module that transforms high-level standing order templates into machine-readable rules

("SmartStop blocks") which can be executed by a clinical decision support engine in real time. These SmartStop rules encapsulate specific instructions to pause an infusion when certain lab value thresholds or conditions are met, under predefined circumstances.

Key features and novel aspects include:

- Standing-Order Template Compilation: The system ingests templated standing orders (e.g., a protocol saying "If lab result L exceeds threshold T, pause infusion of medication M"). The SmartStop compiler parses and converts these into executable logic units (e.g., FHIR Clinical Reasoning artifacts or rule engine entries) that are patient-specific and actionable. This ensures legally binding directives: once compiled and activated, the pause-directive is treated as a direct extension of the physician's order, ready for automatic execution.

- Real-Time Infusion Pause Logic: A monitoring engine continuously tracks live data (lab results, vital signs, device data) and active infusions. When a rule's condition is satisfied – for example, a lab value crosses the threshold – the system triggers an immediate pause command for the associated infusion pump. The infusion is halted (or put on hold) in real time, preventing potential harm from continuing the infusion under adverse conditions. This logic is deterministic and fast, operating within seconds of the data update.

- Scope-of-Practice Validation: Every pause-directive is validated against a knowledgebase of scope-of-practice rules before activation. The compiler and runtime ensure that the directive's action (pausing an infusion, in this case) is one that a nurse or other non-physician clinician is allowed to perform under standing orders. Certain actions like dose increases or complex titrations may be disallowed without physician intervention; the system flags or refuses to compile any template that demands an out-of-scope action. For example, if a template attempted an automatic dose titration beyond what nursing protocols permit, the compiler would reject or modify it to a simple pause-action and notification, thus enforcing nurse practice act guardrails. The runtime also double-checks context – for instance, if a situation arises that would require a new medication or a dosage change, the system will not autonomously do so, instead notifying a provider.

- Lab Result Quality Control (QC) Gate: The system interfaces with laboratory data and applies a verification filter before triggering any pause. Lab results marked as "hemolyzed" or otherwise contaminated/low-confidence are either ignored or require confirmation. For example, if a potassium result comes back flagged for hemolysis, the system will not issue a pause (or will issue an alert indicating uncertainty) because such a result may be erroneously high . This QC gate cross-checks for lab analyte flags, extreme improbable values, or repeat confirmations. It can also delay action until a result is verified (e.g., waiting for lab confirmation or re-run). This prevents unnecessary interruptions due to spurious data.

- Alert Routing and Escalation: When a SmartStop rule triggers an infusion pause, the system generates contextual alerts to notify clinical staff. The invention supports flexible routing rules: it can present an interruptive alert on the nurse's EHR interface or infusion pump screen, trigger audible bedside alarms, and send notifications to mobile devices (e.g., smartphones or wearable communicators like Vocera badges). Alerts are tiered by severity – for a critical pause (e.g., life-threatening lab value), the alert may be accompanied by a distinctive alarm tone and flashing indicators. The system also implements an escalation ladder: if the primary nurse does not acknowledge the alert within a configured time window, it automatically escalates – notifying the charge nurse next, and then the covering provider (physician) if needed. In extreme cases (no response and patient safety at risk), it can alert a Rapid Response Team. This ensures that a pause event prompting clinical assessment will not go unnoticed. A flowchart (see FIG. 5) illustrates the escalation sequence: from bedside RN up to physician and emergency team if acknowledgments are delayed.

- EHR Integration via FHIR and CDS Hooks: The SmartStop system is designed to integrate with hospital EHR workflows using standard interoperability frameworks. For instance, it can use CDS Hooks (a standardized mechanism for EHRs to call external decision support) to trigger the evaluation of pause rules at appropriate times – such as when new lab results are posted or during medication administration events. On detecting a trigger condition, the CDS service can return an order suggestion or decision to pause the infusion. Additionally, the system can leverage FHIR Task resources to coordinate actions: when a pause is needed, it creates or updates a Task in the EHR indicating that medication M should be paused, assigned to the appropriate nursing staff or device integration service. This allows the EHR to orchestrate the workflow – e.g., automatically halting a smart pump (if device integration is available) or prompting a nurse to pause it and document the hold. The use of FHIR and CDS standards ensures the solution works across different EHR platforms and that it can record the pause action as part of the patient's records.

- Order-Set Templating Logic: The compiler supports order-set templates comprising multiple related standing orders. For example, a sepsis protocol order set may include several SmartStop rules (for pausing vasopressor infusion if blood pressure too high, or holding dialysis if certain electrolyte imbalances occur, etc.). The compiler can take a base template (with placeholders for drug names, thresholds, timing, etc.) and instantiate it for a specific patient or clinical scenario. It pulls in patient-specific parameters (such as the exact threshold values ordered by the physician, which might be customized per patient) and generates concrete standing order entries. The output might be a set of discrete, active rules in a standing order registry (FIG. 6 shows a relational schema for storing these templates and compiled instances). Each entry in the registry links to the physician order authorization, the trigger condition, and the specific action (pause directive). This templating capability accelerates configuration of protocols and ensures consistency: changes to a base template can be recompiled to update all derivative

orders.

- Downtime Fallback Behavior: In the event that the SmartStop system or its integration points become unavailable (for example, if the CDS service is down or the hospital network/EHR is in downtime mode), the system automatically shifts to a failsafe mode. FIG. 4 depicts the downtime workflow: the SmartStop engine will notify users that it is offline and cannot monitor automatically, and it will revert control to standard manual protocols. The system is designed to fail safely – meaning that loss of automation will not halt care. Nurses are expected to follow the pre-existing manual standing order procedure (e.g., manually monitor labs and pause the infusion if criteria met, per hospital policy) until the system is restored. The SmartStop compiler may also output human-readable versions of the standing order rules (e.g., printouts or on-screen instructions) as a reference during downtime. Additionally, if network connectivity to infusion pumps is lost, the pumps themselves may alarm for manual intervention. The fallback ensures continuity of care and compliance with hospital downtime procedures.

- Automated Documentation and Attribution: All pause events and related actions are automatically documented in the patient's medical record. When the system pauses an infusion under a standing order, it logs an order entry or event note indicating the infusion was paused, including the reason (e.g., "SmartStop: Drug M paused due to Lab L = value exceeding threshold"). Crucially, the documentation is configured to attribute the action to the physician-of-record (or the ordering provider of the standing order set), not to the bedside nurse or system generically. This aligns the record with reality that the nurse was executing a physician-authorized protocol, thereby meeting regulatory requirements that standing orders be entered and later authenticated by a practitioner . For example, the system might enter an order, "Hold Drug M per Dr. Smith's standing order; Dr. Smith to sign," thereby providing immediate record of the action and flagging it for physician signature. This automation relieves nurses from extra paperwork and ensures compliance with CMS §482.24(c)(3), which mandates proper documentation and physician authentication of protocol-based orders. It also ties into hospital policy that standing orders must be approved by medical staff leadership and based on current guidelines (the templates compiled by SmartStop are assumed to have gone through such approval, and the system can store metadata indicating the approval date and committee).

- Comprehensive Audit Trail: The system maintains a detailed audit log for each SmartStop event. Every step – from the initial compilation of the directive, any modifications or validations, to each real-time trigger evaluation – is recorded. When a pause actually occurs, the system logs the triggering data (e.g., lab result value and timestamp), the identity of the rule that fired, and all notifications sent out. It also logs user responses: which clinician acknowledged the alert, any overrides or delays, and when the infusion was resumed. If a physician was contacted or had to provide input (say to restart or adjust therapy), that is noted as well. This audit trail can be stored in an immutable ledger for compliance (for example, using cryptographic hashing to

time-stamp events, akin to techniques for traceability in FDA-regulated software). The audit log not only supports internal quality improvement and forensic analysis of incidents, but also aids in meeting FDA SaMD guidelines that emphasize transparency and the ability to reconstruct software decisions for validation. FIG. 5's illustration of the alert routing also includes an "audit chain" showing how each notification and action is logged sequentially.

Overall, the SmartStop-based system provides a safety net for infusion therapy by automating pause decisions that were previously manual, while rigorously adhering to clinical and legal boundaries. It improves patient safety (by responding faster to dangerous conditions), reduces nurse workload (by automating monitoring and documentation), and ensures that advanced clinical decision support can be deployed without violating regulations or professional scopes.

Brief Description of the Drawings

- FIG. 1 illustrates the SmartStop compiler architecture. It shows how standing-order templates are input into the compiler, which includes a parser and rule generator. The output is a set of executable SmartStop blocks (rules) that interface with the EHR and infusion devices. The figure depicts components such as the template library, the compilation engine, and the registry of active pause-directive rules.

- FIG. 2 is a block diagram of the scope-of-practice validation module. This diagram highlights how a compiled directive is checked against a database of allowed actions. It shows logic gates distinguishing "pause" actions (allowed for nurses under protocol) versus more complex titration or medication changes (which are either converted to notifications or require physician approval). The figure also indicates inputs like the nurse's role and applicable nurse practice regulations.

- FIG. 3 illustrates the lab result QC rejection filter. This flowchart demonstrates how incoming lab results feed into the SmartStop logic. It shows a decision diamond checking for lab flags (e.g., hemolysis, error codes) and extreme outliers. Contaminated or unverified results are routed to a "no-action" path (or a confirmation sub-process), whereas clean results proceed to trigger evaluation. An example in the figure is a potassium result marked as hemolyzed being filtered out to avoid a false pause trigger.

- FIG. 4 depicts the downtime fallback mechanism. It outlines the system's behavior when the CDS or connectivity fails. In the illustration, a network-status monitor detects loss of connectivity or system fault, then signals all active SmartStop rules to suspend automation. The workflow shows notifications to users that automation is offline and instructs them to use the manual protocol. It also shows that, during downtime, the infusion pumps operate under standard limits and nurses document any actions manually as per usual policy.

- FIG. 5 shows the alert routing and escalation chain in the event of an infusion pause alert. This diagram (which can be a swimlane chart) has lanes for the bedside nurse, the charge nurse, and the physician. It demonstrates the initial alert delivery to the bedside nurse (via EHR pop-up, audible alarm, etc.), the timeline for escalation if the nurse does not acknowledge (e.g., after 2 minutes escalate to charge nurse via phone or pager; after 5 minutes escalate to physician's mobile alert). It also integrates the audit logging, marking each alert and response with timestamps. Additionally, a small sub-diagram in this figure depicts how the alert can interface with mobile notification systems (such as a smartphone app or Vocera).

- FIG. 6 is a schematic of the standing order registry database schema. It illustrates how compiled pause-directive rules are stored and related to other data. Key entities in the figure include: a Template table (with fields for condition, action, approval info), a Patient-Specific Order table (with links to the Template and the specific patient's parameters and physician-of-record), and an Audit Log table (with foreign keys to the order and events). The relationships between these tables show that each active directive is tied back to an approved template and to the authorizing provider.

- FIG. 7 provides an EHR-to-device orchestration overview. This high-level system diagram shows integration among the EHR system, the SmartStop logic server, and the infusion pump devices. It depicts how the EHR (via CDS Hooks or FHIR) communicates triggers (like new lab results) to the SmartStop engine, which then issues a pause command to the infusion pump (either directly if pumps are networked or via a Task for a nurse to carry out). The figure highlights bidirectional flow: EHR provides patient data to SmartStop, and SmartStop provides order updates or instructions back to the EHR and device. It emphasizes reliability features such as acknowledgment messages from the pump and how the system updates the EHR record.

Detailed Description of Embodiments

System Architecture and Compiler Workflow (FIG. 1)

Referring to FIG. 1, the SmartStop system architecture comprises a Standing Order Template Library (100), a SmartStop Compiler Engine (110), a Rule Execution Engine (150), and integration interfaces to the EHR (180) and infusion devices (190). The Template Library (100) stores generic standing-order protocols authored and approved by the hospital's clinical leadership (including physicians, nursing, and pharmacy directors). Each template defines a set of conditions and corresponding pause actions in human-readable form. For example, a template might state: "For medication heparin infusion: If aPTT > 80 seconds, pause infusion and alert provider." Templates also include metadata such as version, approval date, and references to evidence-based guidelines (ensuring they adhere to CMS and hospital policy requiring evidence-based, approved protocols).

When the system is deployed for a specific patient or order set, the SmartStop Compiler Engine (110) takes the relevant templates and compiles them into executable rules. The compilation process involves parsing the logic conditions and actions from the template and translating them into a machine-executable format such as a decision table, a set of if-then rules in a rules engine, or a Clinical Quality Language (CQL) artifact that can be utilized via CDS Hooks. During compilation, patient-specific parameters are substituted in. For instance, if the physician-of-record specifies a slightly different threshold for a given patient (maybe the template threshold is 80, but the doctor orders pause at 75 for a high-risk patient), the compiler will incorporate the modified threshold.

The Scope-of-Practice Validator (120, corresponding to FIG. 2) is invoked during compilation (and can also run during execution as a safety check). The validator checks each rule's intended action against an allowable actions table (125). This table encodes institutional and legal constraints – for example: "nurse may pause or stop an infusion with a valid standing order; nurse may not increase infusion rate beyond what was ordered without additional order; nurse may not substitute medications on their own," etc. If a template includes any action outside these bounds, the compiler either rejects that rule or modifies it to a compliant form. In one embodiment, if a template contained a dosage titration suggestion, the compiler could transform it into a recommendation alert rather than an automatic action, thereby keeping the automated portion within a pause/hold scope. The compiled output thus consists only of pause directives and associated notifications that are permissible. Each compiled rule is tagged with the role authorized to execute it (here, typically an RN for pause actions) and the supervising role (MD who authorized it).

Once compiled, the rules are stored in the Standing Order Registry database (130, see FIG. 6 for schema). Each entry links to the original template ID, the patient or context, the current status (active, disabled), and references to the responsible physician. At runtime, the Rule Execution Engine (150) continuously evaluates these rules against incoming data. It subscribes or listens to relevant data feeds: lab results from the hospital LIS (Laboratory Information System) via the EHR (for example, through a FHIR subscription or an HL7 lab interface), vital sign monitors, and infusion pump status information. The Rule Engine may evaluate conditions in real-time or be event-driven (e.g., triggered when a new lab result arrives).

Real-Time Pause Logic and Execution

When a condition is met, the Rule Engine initiates the pause logic sequence. Suppose a rule states: "If serum potassium > 5.5 mEq/L, pause potassium chloride infusion." Once a new lab result for potassium comes in at 5.8, the engine matches it to the rule. Before firing the rule, the system engages the Lab Result QC Gate (explained below, FIG. 3) to ensure the result is trustworthy. If the result passes QC, the engine proceeds to execute the action: it generates a pause command for the specified infusion.

The mechanism of pausing can vary depending on integration capabilities:

- If the infusion pump (190) is integrated and supports remote commands (many modern pumps on clinical networks do), the system sends a command via the device integration interface (perhaps using a vendor API or an HL7 message to an infusion management service) to halt the infusion. This can be orchestrated through a FHIR Task resource with an instruction like "pause medication administration" assigned to the pump or an intermediary.

- In parallel, or if direct device control is not available, the system creates an actionable alert for the nurse. For example, through a CDS Hooks trigger on "laboratory result received," it could pop up a card in the EHR's interface for that patient, stating: "SmartStop Alert: Potassium = 5.8. KCl infusion paused per protocol." If the system hasn't already directly paused the pump, the alert will instruct: "Please pause the infusion pump now." The nurse can acknowledge this, and upon doing so, the system can record that the nurse carried it out (or if auto-paused, that the system did so).

Crucially, the logic only pauses the infusion – it does not automatically resume it without appropriate conditions being met. Resumption might occur either automatically when the lab returns to normal (if the protocol and physician allow), or more typically, upon physician or nurse confirmation that it's safe. The rule can include a secondary condition like "resume if potassium falls below X after at least Y minutes" or it might simply require physician evaluation to restart. The SmartStop system can assist by continuously monitoring even after pause: for instance, it can keep checking labs and advise when criteria are back within range, sending a notification such as "Potassium now normal; infusion can be restarted per protocol." However, actual restart may be left to a manual action to ensure clinical oversight.

This design choice – focusing on pause-only logic – is intentional to stay within nursing autonomy. By halting infusions that may be dangerous while not independently starting or upping any infusion, the system respects the boundary that nurses implement orders but do not prescribe therapy. It essentially acts as a supervisory safety brake. This is a more acceptable form of automation under current regulations than a fully autonomous dosing system, and it complements existing smart pump dose-limit safeguards with a new dynamic condition safeguard.

Lab Result Verification and Quality Control (FIG. 3)

FIG. 3 illustrates the QC filtering of lab inputs. When the Rule Engine receives a new data point (e.g., lab result message), it passes through a Verification Module (310). This module checks for:

- Flags on the lab result: Most lab systems flag results if the sample was problematic (hemolyzed, insufficient sample, delayed processing) or if the result is outside analyzer linearity, etc. The SmartStop QC module will recognize such flags (via standard codes or text in the lab feed) and mark the data as unreliable.

- Value plausibility checks: Each rule knows the normal range or expected range of its triggering lab. If a value is extreme beyond physiological possibility (e.g., a potassium of 12 mEq/L which is likely an error since such a level would usually be incompatible with life unless a sample issue), the system can be configured to require confirmation. It might wait for a repeat result or cross-check with another measurement.

- Time continuity checks: If multiple lab results come in sequence, the module can compare them. For example, if one reading is drastically different from the previous one in a short time (beyond what's clinically reasonable), it might suspect an error.

- Optionally, the system might also incorporate a redundancy query: e.g., if the lab is critical, it could prompt a point-of-care test or blood gas verification (if available) before pausing, unless the risk of waiting is too high. This is a configurable aspect of the QC logic.

If the QC module determines the result is unreliable, the rule action is suppressed or deferred. The system could issue a different alert in that case: e.g., "Critical K+ result received but marked hemolyzed – system did not pause infusion. Repeat lab recommended." This behavior prevents knee-jerk reactions to false data, aligning with lab practices where such results may be hidden or require manual validation . Only if a result passes these checks (meaning it's both flagged as valid and clinically plausible) will the SmartStop rule trigger a pause.

Integration with EHR and Alert Mechanisms

Integration is achieved via both push and pull methods:

- The EHR can push relevant events to SmartStop. For instance, using a CDS Hook configured for "lab-observation-trigger," the EHR will send necessary context (patient, the new lab result) to the SmartStop service whenever a lab result is finalized. The SmartStop service processes it and responds with any needed actions (like an order to pause medication or an alert card). This real-time hook means the system doesn't need to constantly poll – it reacts to events.

- Alternatively or additionally, the SmartStop engine can poll/pull data via FHIR subscriptions or periodic queries (e.g., check the latest labs for patients on monitored infusions every X minutes). In some embodiments, it subscribes to a FHIR Observation resource feed filtered by patients and lab types of interest.

- For executing the pause, as noted, the system might post a FHIR Task with an instruction. For example, a Task resource could be created with status=requested, code=pause-infusion, and references to the specific Medication Administration or Device. The EHR or a connected middleware that supervises devices would pick this up and carry out the action (possibly marking the Task in-progress then completed once the

pump is actually paused).

- The system updates the patient's Medication Administration Record (MAR) or orders in the EHR. It may, for instance, append a note to the active infusion order: "On hold as of 14:30 per standing order due to lab result." Many EHRs have a function to put an order on hold; the integration can leverage that to reflect the paused state in the chart.

Alert display: The nurse's view in the EHR could show an alert pop-up requiring acknowledgment (an interruptive alert). Simultaneously, secondary devices can alert: The system may send an alert message via a secure messaging app or pager system configured for the unit. If the hospital uses an alarm middleware or aggregator (like a clinical communication system), SmartStop can send alerts into that system with appropriate priority tags (e.g., a "critical lab pause" alert might be marked as high priority to bypass silent settings on devices).

Escalation (FIG. 5): The SmartStop alert manager monitors whether the alert has been acknowledged or the Task completed. If not, after a preset interval (which may be configured per protocol severity, say 2 minutes), it will escalate. In one embodiment, escalation means sending a duplicate alert to the charge nurse of the unit (with text like "Escalation: Bed 5 infusion pause not acknowledged"). If still no response after another interval, the attending physician or on-call provider gets an urgent notification. The escalation ladder and timing can be adjusted to clinical context – e.g., a chemotherapy pause might escalate more slowly than a vasopressor pause, depending on how critical timing is. The design ensures failsafe notification so that if an on-duty clinician is unavailable (busy or a device malfunction), someone else will be informed to take action promptly.

All alert and escalation actions are logged in the Audit Trail (with timestamps and the identity of recipients/acknowledgers).

Downtime and Fallback Operations (FIG. 4)

Reliability is enhanced by planning for partial system failures:

- Detection: A watchdog component (perhaps within SmartStop or a separate monitoring service) pings the EHR and the SmartStop engine periodically. If it detects loss of connectivity to the EHR (e.g., the hospital enters downtime mode) or the rule engine itself encounters an error, it triggers downtime mode.

- User Notification: In downtime mode, active users (nurses, physicians) might receive a message via the EHR (if still up in read-only) or via other channels: "SmartStop system is currently unavailable. Automated infusion monitoring paused." This message is also posted in a central dashboard if available.

- Protocol Reversion: The system is designed such that every SmartStop rule corresponds to an existing manual protocol step. During downtime, nurses and physicians revert to

those manual protocols. For example, if the system isn't monitoring labs, the responsibility falls back to the bedside team to keep an eye on lab results as they come in and manually hold infusions as needed. The hospital's policy (which existed prior to automation) covers this scenario, and staff are trained to follow it whenever the CDS tools are not functional. The SmartStop system documentation even includes guidance or quick-reference for these protocols to support staff during outages.

- No Black-Box Effects: Importantly, the system fails open rather than closed. That is, a failure in SmartStop will never cause an infusion to stop arbitrarily or continue dangerously – it simply ceases to intervene, thereby leaving the status quo (nurse and pump alone) to manage. Because nurses are still present and pumps still have their built-in safety limits, patient safety is still maintained, just at a slightly reduced level of automation.

- Logging and Catch-up: When the system comes back online, it can attempt to "catch up" by scanning if any relevant events occurred during downtime (e.g., lab results that would have triggered a pause). It might then prompt users with retrospective advisories ("During downtime, X result was high; ensure infusion was handled appropriately"). This can be part of the audit and safety review.

FIG. 4's flowchart shows the branching: normal operation vs. downtime path, and indicates clearly that in downtime the pathway leads to manual protocol execution.

Compliance and Regulatory Safeguards

The SmartStop system is engineered to comply with multiple regulatory frameworks:

- CMS §482.24(c)(3) (Medical Records requirements for standing orders): The compiled directives are effectively standing orders, so the system ensures that each one meets CMS requirements. As described, each template is approved by the medical staff and is based on current guidelines (this approval occurs outside the software but is a prerequisite). The system's auto-documentation of each use of a standing order means the patient record reflects the order execution and the responsible practitioner (physician) sign-off is facilitated . By doing so, the hospital remains in compliance – it can use standing orders to improve care as allowed, without violating documentation rules.

- FDA Software as a Medical Device (SaMD) guidance: This software function (automatic pausing of infusion based on clinical input) qualifies as a clinical decision support tool with direct action on therapy – likely a regulated SaMD of moderate risk. The system is therefore developed under quality systems ensuring validation and traceability of requirements. Features like the complete audit log, the ability for clinicians to review the basis of alerts (e.g., showing which lab and threshold triggered a pause), and the fail-safe modes all align with FDA's guidance for CDS that is transparent and safe. For

instance, FDA's CDS guidelines emphasize the importance of the clinician being able to understand the rationale for a recommendation. SmartStop alerts are generated with an explanation ("paused because lab X = Y which is above threshold Z"). This transparency allows the clinician to override if they have additional context, thus maintaining human control. Additionally, the system could be designed to undergo FDA 510(k) clearance or equivalent, and complies with relevant device standards (alarm safety, interoperability standards like HL7/FHIR, etc.).

- Nurse Practice Act guardrails: The configuration of the system requires mapping to state or regional nurse practice regulations. In practice, most states allow RNs to execute protocols like holding meds for certain conditions as long as an authorized prescriber's order exists. The system effectively enforces that an order exists (because we compile from an order template) and does not stray beyond it. It does not, for example, let a nurse start a new medication that wasn't ordered just because a lab is abnormal – that would generally require an NP/PA or physician order. If a certain jurisdiction had a particular quirk (say, even holding a medication requires physician co-sign at time of event), the system can be configured to immediately send a co-sign request to a physician in tandem with the action. The scope-of-practice validation module (FIG. 2) encodes these rules and ensures no automation goes beyond what a nurse could lawfully do in that context. In the event a rule does need a physician's active involvement (e.g., "if event, give an extra medication dose" – which is not a pause and likely out of scope), the system would instead convert that into a suggested order for physician approval rather than an automatic action by the nurse.

- Other Standards: The system also addresses relevant standards like Joint Commission requirements for alarms (ensuring critical alarms aren't silenced without acknowledgement), and it supports 21 CFR Part 11 electronic records integrity by maintaining secure, time-stamped logs of all actions (especially since it enters orders in the EHR, those become part of the legal medical record with proper attribution). Additionally, for audit logging, techniques such as hashing each event record and storing an audit trail that can be externally verified can be used (similar to concepts disclosed in prior TraceLoop systems for traceability).

Example Use Case

To illustrate an end-to-end scenario, consider a patient on an IV heparin infusion with a standing order set for anticoagulation management:

1. Compilation: The physician's order set includes: "Check aPTT every 6 hours. If aPTT > 100 seconds, pause heparin infusion and notify provider. If aPTT < 60, call provider (titration needed)." The SmartStop compiler compiles the "pause if >100" rule as an automated directive, and the "<60 call provider" as an alert-only (since increasing rate is

not allowed for nurse alone, it just tells nurse to call, no automatic pump action).

2. Monitoring: Lab results flow in. The patient's 6-hour aPTT comes back at 105 seconds. This triggers the rule. The QC module sees the lab is valid (no flags).

3. Action: SmartStop sends a pause command to the infusion pump (which acknowledges receipt and stops the infusion). Simultaneously, the nurse's EHR screen shows a high-priority alert: "Heparin infusion automatically paused due to aPTT 105 (>100). Standing order protocol executed." It provides an OK/Acknowledge button.

4. Notification: Because this is critical, the system also sends a text page to the on-call physician: "SmartStop Alert: Patient X's heparin paused (aPTT 105). Review needed." The nurse also gets an audible alarm at the pump.

5. Acknowledgment: The nurse acknowledges the EHR alert, confirming she is aware. She assesses the patient, perhaps draws a confirmatory lab if needed, and contacts the physician if protocol dictates (the protocol said notify provider, so presumably either the system or nurse has already effectively done so).

6. Documentation: In the MAR, the heparin infusion entry now shows "On Hold at 14:35 per Dr. Jones' protocol (aPTT 105)." The system queued a co-sign task for Dr. Jones to later sign this entry. All these entries are time-stamped.

7. Resume: Two hours later, the physician orders to resume heparin at a lower dose. The nurse restarts the infusion (which either automatically clears the hold in EHR or the nurse documents "resume"). The SmartStop system logs that the infusion was restarted and ends that pause event in its audit log (total pause duration, etc. recorded).

8. Audit: Hospital quality staff later can review a report from SmartStop showing that the protocol worked as intended: infusion was paused within 1 minute of the lab result, alerts went out, and the provider was notified. All actions matched the standing order, and the physician authenticated the order in the chart next day per policy.

Industrial or Non-Hospital Variant

While the foregoing description focuses on hospital infusion use, the underlying SmartStop compiler and pause logic could be applied in industrial process control or other domains. The general pattern is: monitoring of parameters and pausing a process when thresholds are crossed under pre-authorized rules. For example, in a chemical plant, an engineer might set standing instructions for an operator: "If pressure > X psi, pause pump Y and alert supervisor." The SmartStop system could compile such instructions and automatically pause machinery when sensor readings exceed limits. The roles in that case (operator vs. supervising engineer) mirror the nurse vs. physician dynamic. The system would similarly log events and escalate

alerts (to shift supervisor, safety officer, etc.) if not addressed. Compliance in an industrial context would align with safety regulations (e.g., OSHA rules) rather than medical ones, but the core design – compiler, real-time threshold monitoring, scope check (operators allowed to act under certain SOPs), QC of sensor data, fallback (manual control if system fails), and audit logging – all carry over. The claims of this invention therefore encompass such variants beyond healthcare, ensuring a broad applicability wherever automated pause-directive control is beneficial.

In all embodiments, whether clinical or industrial, the SmartStop-based system provides a novel compiler-driven solution for translating supervisory protocols into automated, real-time safeguards on continuous processes, with careful attention to role-based authority, data reliability, and comprehensive record-keeping of every intervention.