SmartStop Sentinel Connect™: Hospital-Wide Sentinel-Event Automation and Reporting Framework

System Overview

SmartStop **Sentinel Connect™** is a hospital-wide sentinel event automation and reporting system designed to capture and respond to serious patient safety incidents in real time. A **sentinel event** is defined by The Joint Commission as a patient safety event that reaches a patient and results in death, permanent harm, or severe harm requiring immediate intervention[1]. Such events (e.g. fatal medication errors, wrong-site surgeries, inpatient suicides) demand prompt investigation and often mandatory reporting to regulatory bodies. The Sentinel Connect system functions as a **federated sentinel-event registry and decision pipeline** that spans the entire hospital, ingesting critical event data from diverse clinical sources and automating the workflow of triage, compliance checking, and reporting.

Key Functions: Sentinel Connect receives heterogeneous event inputs from smart infusion pumps (e.g. overdose alerts or pump failure alarms), electronic health record (EHR) medication administration modules (e.g. eMAR discrepancies), surgical site tracking systems (e.g. wrong-patient or wrong-site alerts), fall detection sensors, implantable device monitors, and behavioral health risk tools. Each incoming event is automatically **normalized into a unified schema**, ranked by **severity and time-to-harm** (priority), checked against state-specific reporting thresholds, and tagged for **Patient Safety Organization (PSO) safe-harbor** inclusion if eligible. Finally, the system auto-populates the event into the appropriate regulatory reporting format or incident-management record, ensuring no critical step is missed. This end-to-end automation accelerates incident response while maintaining strict compliance with healthcare regulations.

In summary, Sentinel Connect transforms ad-hoc and manual incident reporting processes into a streamlined, **audit-proof** pipeline. It encodes complex rules (state statutes, accreditation requirements, PSQIA guidelines) into software logic, so that clinicians and risk managers are immediately notified of high-priority events along with guidance on reporting obligations. The system's hospital-wide scope and interoperability allow it to break down data silos—aggregating sentinel events from multiple sources into one secure registry for analysis and follow-up. This leads to faster intervention (potentially preventing patient harm), comprehensive documentation, and standardized reporting that facilitates learning from every serious incident.

Architecture Overview (FIG. 1 & FIG. 2)

FIG. 1 illustrates the high-level architecture of the SmartStop Sentinel Connect system, which is composed of modular layers and components working in concert. At the input layer, **prebuilt connectors (101)** interface with each device type and vendor-specific system across the hospital. These connectors continuously ingest event data from smart pumps, EHRs, surgical trackers, etc., converting disparate data formats into a common internal representation. The connectors feed a central **Event Bus (110)** where all incoming signals are **normalized and mapped to a unified schema**. (In one embodiment, the system leverages AHRQ's Common Formats – a

standardized set of definitions and data elements for patient safety events[2] – to ensure uniform structuring of event information across all sources.)

Once normalized, events flow into a multi-stage **decision pipeline (120)** for analysis and routing. As shown in **FIG. 2**, the pipeline comprises several functional engines in sequence. First, a **Priority Triage Engine (121)** calculates a **risk priority score** for each event based on its severity (actual or potential harm) and urgency (time-to-harm if not addressed). For example, an infusion pump that has nearly overdosed a patient would be scored as extremely severe and time-critical, whereas a minor documentation discrepancy would rank lower. This risk scoring provides a quantitative basis for triaging events in real time – high-scoring events are flagged for immediate attention and response.

Next, the event is fed into a **Rules Engine (130)** that encodes regulatory and institutional logic. This engine compares the event attributes against **state reporting thresholds and statutes (131)** as well as **Patient Safety and Quality Improvement Act (PSQIA) guidelines**. Each U.S. state defines certain "reportable" sentinel events (often including occurrences like surgery on the wrong body part, unexpected death, or serious disability) that must be reported to the state Department of Health within a specified timeframe. The rules engine contains a knowledge base of these jurisdiction-specific criteria. It automatically determines, for instance, whether an event meets the legal definition of a reportable sentinel event in that state (e.g., a wrong-site surgery or a medication error resulting in serious injury)[3][4]. Simultaneously, the rules engine checks PSQIA/PSO rules – events that are being collected for quality improvement can be tagged as **Patient Safety Work Product (PSWP)** under the PSQIA safe harbor. PSWP is broadly defined to include any data or analysis assembled for reporting to a PSO[5]. If an event is not legally mandated to be reported externally (and thus can remain confidential), the system **tags it for PSO submission** to ensure the incident can be analyzed under privilege, encouraging internal reporting and learning without legal risk[6][7]. The rules engine essentially makes a compliance decision for each event: whether it requires external reporting (and to whom), whether it should be sent to the hospital's PSO for protective analysis, or both.

After rules evaluation, the event record (along with the triage results and rules-engine determinations) is passed to two parallel outputs. One output path goes to an **Export Engine (140)** which formats the event data for various external systems and destinations. This component houses multiple **export modules (141)** to auto-populate incident reports: for example, it can transmit a structured report to a state health department portal via API or secure email, populate the **AHRQ Common Formats** XML for PSO submission, or create an entry in internal risk management software (such as RLDatix) via integration. Notably, the export engine uses mapping templates to **seamlessly convert the unified event record into the required format of each target outlet**[8]. For instance, if a state requires a specific form, the system will fill the form fields; if an internal Root Cause Analysis (RCA) tool uses a database, the system will send the appropriate data fields. This automatic population of data eliminates duplicate data entry and ensures consistency across reports.

The second parallel output path feeds into the **Immutable Audit Ledger (150)**. Every event that enters the Sentinel Connect pipeline, and every decision or action taken on it, is recorded in an **append-only ledger** for audit and traceability. The ledger is implemented as a cryptographically chained log (for example, using blockchain or hash-linked records) to guarantee immutability. Each event entry is timestamped and digitally signed, creating a tamper-proof record of what

happened, when, and by whom. This means the system maintains an **audit-proof timeline** of the entire life cycle of a sentinel event: from initial detection, through triage scoring, rule evaluation, any human interventions, and final reporting. By anchoring each step in an **unalterable chain of transactions**, the system ensures authenticity and non-repudiation of the event records[9]. Even system administrators cannot retroactively modify or delete an event without leaving a cryptographic trace, which is crucial in investigations (it prevents after-the-fact alterations of incident data). The ledger not only helps with internal audits and compliance (e.g., demonstrating to regulators that required notifications were made on time) but also strengthens legal defensibility by showing a secure chain of evidence for each incident.

Finally, the architecture supports a **Clinician Override Workflow (160)**, shown in FIG. 2 as an optional feedback loop. While Sentinel Connect is highly automated, it allows designated clinicians or patient safety officers to intervene in the decision process when needed. For example, if the rules engine flags an event as reportable, an authorized clinician might review and add clinical context or even temporarily halt an automatic external report if new information suggests it doesn't meet criteria (for instance, if an apparent "medication error" was actually a charting issue with no patient impact). The system provides an **override interface (161)** for these purposes. Importantly, any **override action by a human is itself logged in the immutable ledger**, including the user identity, timestamp, and reason (if provided). The override mechanism has configurable safety limits – for critical events that unequivocally require statutory reporting (e.g., wrong-site surgery), the system may still require that the report be filed but could allow the clinician to append an explanatory note or additional documentation. In less clear-cut cases (like a near-miss that the algorithm marked for PSO only), clinicians might elevate the event for external reporting or vice versa. This human-in-the-loop design ensures that clinical judgment and context can be applied, preventing over-reporting of false positives and allowing escalation of events that algorithms might undervalue, all while preserving a full audit trail of these decisions.

In essence, the architecture (FIG. 1) is **modular and layered**: input connectors feed a normalizer; a triage engine prioritizes events; a rules engine enforces compliance logic; export modules output to required channels; and a ledger underpins the whole pipeline with security and traceability. FIG. 2 provides a conceptual flowchart of how an event travels through these layers from detection to outcome, depicting decision points (such as threshold checks and override opportunities) and parallel logging. The system can be deployed on-premises or in a cloud environment as needed, and it is designed to scale across multiple hospital sites (federation) if a healthcare network wants a centralized sentinel event registry collating data from all member facilities. Each component (connectors, rules, outputs, etc.) can be updated or expanded independently, which supports the addition of new device types or new reporting requirements over time without a redesign of the entire system.

Functional Layers and Components

The Sentinel Connect framework can be understood as a series of **functional layers**, each responsible for specific tasks in the sentinel event handling process. The primary layers/components are detailed below:

- **1. Input Connectors Layer:** This layer consists of **prebuilt connectors** for various medical devices and information systems. Each connector is essentially an adapter that

knows how to communicate with a particular source (e.g., an IV pump's event log, an EHR's alert feed, a bed alarm system). Connectors may use standard protocols (HL7/FHIR messages from EHRs, vendor APIs for pumps and monitors, or even direct database queries or IoT sensor feeds) to capture event data in real time. They perform initial data translation into a common event format. For example, a "smart pump connector" might listen for Dose Error Reduction System (DERS) alarms or occlusion alarms from infusion pumps; a "Pyxis connector" might capture medication dispensing override events; a "fall monitor connector" subscribes to bed exit alarms or wearable fall detector alerts. Each connector attaches source identifiers and timestamps to the raw event data and sends it into the pipeline. The system includes a library of connectors for major device vendors out-of-the-box, but also provides APIs for developing new adapters as needed, ensuring **plug-and-play integration** with virtually any system that can generate a patient safety event.

- **2. Normalization & Schema Mapping:** Upon ingestion, events enter a normalization module. This module parses each incoming event message and **maps it to a unified schema** that the rest of the system understands. The unified schema defines standard fields for sentinel events (patient ID, location, device ID, event type, severity, timestamp, etc., along with free-text descriptions and context data). By enforcing a common schema, Sentinel Connect can uniformly process events from different sources. For instance, whether an infusion pump reports an "air-in-line error" or the EHR flags a "medication administration error", both can be classified under a generic field like event_category = Medication Safety with appropriate sub-type codes. The normalization ensures apples-to-apples comparison of events. This layer may also enrich events with cross-references – for example, linking to patient records (through an MRN) to pull basic demographics or linking to the hospital's location hierarchy to identify the nursing unit or department where it occurred. It effectively creates a **consolidated event record** ready for analysis. (Using standardized data definitions from systems like AHRQ Common Formats helps in this mapping[2], allowing data aggregation for internal and external benchmarking.)

- **3. Priority Triage Engine:** This functional layer receives the normalized event and computes a **priority score or risk ranking**. The engine uses a combination of rules and weighted factors to assess how urgently the event needs addressing. Two primary dimensions influence the score: (a) **Harm Severity** – an estimate of actual or potential patient harm (e.g., on a scale from negligible to catastrophic), and (b) **Time-to-Harm (Urgency)** – how quickly intervention is required to prevent or mitigate harm. The product or composite of these (risk = severity × urgency) yields a priority level. For example, an event "infusion pump programming error – tenfold overdose averted by safety limit" would rate very high on severity (could cause death) but perhaps medium on urgency if caught in time; a "patient fall from bed with no injury" might be medium severity and low urgency (no ongoing threat). The triage engine can use hospital-configured parameters or scoring systems like the National Coordinating Council's index for medication errors, etc., to quantify severity. It may also consider **cascading effects** (if multiple events are happening together) or patient-specific factors (frailty, etc.). The output is a priority classification such as "Critical – Immediate Response", "High – prompt review within 24h", "Moderate", or "Low". This priority

drives downstream actions – a critical event triggers instant notifications (paging the risk manager or rapid response team), whereas a low priority event might simply queue for later aggregate analysis. The triage engine ensures **the most dangerous situations are escalated first**, addressing the "time-to-harm" aspect explicitly as an innovation (traditional incident reporting often lacks real-time prioritization).

- **4. Compliance Rules Engine:** The rules engine is the **brain of regulatory compliance** in Sentinel Connect. It houses a knowledge base of **state laws, federal regulations, and institutional policies** regarding event reporting. This includes:

- State-mandated **sentinel event definitions and reporting timelines**. (For example, many states enumerate "never events" or serious reportable events; if an event falls in those categories, the engine knows it must be reported within X days to the state authority[4].) The engine compares event details (e.g., outcome = death or severe harm, event type = surgery on wrong body part) against these definitions. It encodes things like: "IF event.type = wrong_site_surgery THEN reportable_to_state = TRUE" or "IF event.harm = death within 24h of treatment THEN trigger state notification". Each state's thresholds (like what constitutes "major permanent loss of function" or other legal criteria) are represented in the rules.
- **PSQIA (Patient Safety Act) guidelines** for PSOs. The engine determines if an event can be protected under the PSO safe harbor. If an event is not required by law to be disclosed outside the hospital, it can be categorized as **PSWP (Patient Safety Work Product)** and sent to a PSO for confidential aggregation[5]. The rules engine tags such events with a PSWP flag and routes them appropriately. Conversely, if an event *must* be reported to a state or accreditor, the engine marks it as **excluded from PSWP** for those particular details (since data mandated for external reporting cannot be kept confidential under PSQIA). In some cases, the engine will **split** the data: send non-identifiable trending data to PSO while also preparing identifiable info for regulators, maintaining compliance with the boundaries of privilege.
- **Accreditation requirements** (e.g., The Joint Commission encourages self-reporting of sentinel events[10]). The rules engine may flag that a report to The Joint Commission's Sentinel Event Database is recommended (even if not legally required) to avail of their expertise. This can be an advisory output.
- **Internal policy rules** set by the hospital's risk management. For instance, a hospital might require that any medication error reaching a patient, even if not harmful, be reviewed or sent to their corporate PSO. The engine is configurable to include these site-specific rules so that the automation aligns with local policy (e.g., "all falls with injury, regardless of severity, must undergo RCA within 48 hours" – the system can enforce this by scheduling an RCA task in the linked tool).

The rules engine outputs a **decision matrix** for each event: e.g., "Report to State = Yes (meets statute XYZ), Report to PSO = Yes (mark as PSWP), Notify Joint Commission = Optional, Internal RCA needed = Yes". These determinations then guide the next steps of export and notification. By encoding statutes and guidelines, the system ensures **no reportable event slips through the cracks**, and conversely prevents over-reporting of events that don't meet criteria (thus avoiding unnecessary disclosures).

- **5. Export and Integration Engine:** The export engine takes the decisions from the rules engine and carries out the actual generation of reports/notifications. It contains multiple sub-functions:
- **State Reporting Module:** If state reporting is required, this module prepares the report. Many states provide specific forms or online portals for sentinel events (often requiring details like patient info, event description, root cause analysis, and corrective actions). Sentinel Connect auto-fills the known information into the required format. If an electronic data interchange is available, it uses that (for example, some states might allow XML/HL7 submission). Otherwise, it can generate a Word or PDF form (like Maine's Sentinel Event form[11]) for quick review. It also tracks the regulatory deadline (e.g., "must submit within 3 business days"[12]) and can send reminders or escalate if a report isn't finalized by a human in time.
- **PSO Submission Module:** For events marked PSWP, the system packages the data according to the **AHRQ Common Formats for Event Reporting (CFER)** or other PSO-specified format[13]. It can anonymize certain fields if needed (the rules engine might instruct it to strip patient identifiers if sending to a PSO that is aggregating de-identified data). The module then transmits the data to the PSO's secure system. Many PSOs use the AHRQ's PSOPPC (Patient Safety Organization Privacy Protection Center) which accepts Common Format reports; Sentinel Connect can interface with that web service to upload the event automatically. This encourages robust reporting of near-misses and unsafe conditions under protection, since the effort required from staff is minimal – the data is captured and sent with one click or no human intervention.
- **Internal RCA/Incident Management Module:** If the hospital uses an internal software like RLDatix or Epic's incident module, the system can create an incident record there as well. For example, RLDatix's event reporting platform typically relies on staff to fill out forms[8]; with Sentinel Connect, when a sentinel event is detected, it can **push the event information directly into RLDatix** via an API or data feed, essentially pre-filling a report in that system. This integration means hospital risk managers find the event already logged in their usual management tool, with key details attached (device data, patient info, preliminary classification). Similarly, for RCA tracking, if the hospital uses a tool to manage corrective action plans, the system can initiate a case or task in that tool.
- **Alert/Notification Module:** Aside from formal reports, the export layer also sends **real-time alerts** to relevant personnel. Critical events can trigger SMS, pager, or email alerts to predefined roles (e.g., Chief Medical Officer, Patient Safety Officer, unit manager). The alert contains a summary and maybe a secure link to the full event record in Sentinel Connect's dashboard. This ensures immediate human awareness and response parallel to the automated reporting.

All these export sub-modules operate under a unified orchestration: the system keeps track of what was sent where, and when. If an external transmission fails or is pending human approval (say a draft report needs sign-off), the system logs that status and continues to remind users until completion. **FIG. 2** depicts these outputs branching from the main pipeline, emphasizing that multiple outputs can occur from one event (e.g., one event can generate both a state report and a PSO report, each in the proper format).

- **6. Immutable Ledger and Audit Trail:** A cornerstone of the system's design is the **immutable audit ledger** that underlies all other layers. Technically, this can be

implemented via blockchain or a cryptographic log service. Every significant action in the system produces a ledger transaction: event received, normalized, triaged (with the score value), rule outcome (with rationale, e.g., "matched rule ID 5: reportable sentinel – patient death"), report dispatched to state (with timestamp), user override action, etc. Each transaction references the ID of the previous one, forming a chronological chain. The ledger is **append-only** – data cannot be removed or altered, only new entries can be appended, each containing a hash of the prior entry. This yields a tamper-evident sequence[9]. For audit purposes, authorized users (or auditors) can query the ledger to reconstruct exactly what happened with any given event. It provides **end-to-end traceability**: for example, one can prove that an incident was detected at 3:00 PM, triaged as high risk by 3:01 PM, and reported to authorities by 3:30 PM, with all actions accounted for. The ledger can also store digital signatures (e.g., when a user approves a report, their digital cert is logged) to ensure accountability. This design not only satisfies regulatory requirements for audit trails (e.g., showing compliance with HIPAA's security rule on audit logs) but also fosters trust in the automation – clinicians and administrators know they can always verify the system's actions, and any deviations or delays are transparent.

- **7. Security and Access Control:** While not explicitly mentioned in the overview, it's worth noting that all these layers incorporate strong security measures. The system deals with sensitive patient data and safety information, so connectors use encrypted channels, the central registry is access-controlled, and the ledger (though distributed or replicated for safety) is permissioned so that only authorized parties (e.g., compliance officers, PSO staff) can view the sensitive content. Role-based access ensures that clinicians see only events relevant to their patients or unit, whereas system-wide patient safety staff can see aggregate data. This prevents inadvertent disclosure while enabling needed visibility. Additionally, the system has fail-safes: if it loses connectivity to an external service (like a state portal), it queues the report and flags it for manual follow-up, ensuring no data is lost.

Each of these functional components is modular. They communicate via well-defined interfaces (often messaging or API calls internally). This modularity means the hospital can customize or upgrade pieces without disrupting the whole. For example, if a new infusion pump model is acquired, one can deploy a new connector module for that pump vendor while the rest of the pipeline remains unchanged. Or if reporting rules change due to new laws, an update to the rules engine knowledge base will propagate to all relevant events automatically. The layered approach with separation of concerns (ingestion, normalization, triage, compliance, output, logging) yields a robust and adaptable system architecture.

Safety and Audit Modules

Safety and auditability are paramount in the Sentinel Connect system, given the high-stakes nature of sentinel events. The design includes multiple features to ensure **safe operation, data integrity, and accountability**:

- **Audit-Proof Timeline (Immutable Ledger):** As described, the immutable ledger provides a complete, **chronological timeline** of events and actions that cannot be altered retroactively. This serves as the ultimate audit trail for both internal QA and external

regulators. In practical terms, this means if a surveyor or investigator asks for evidence of compliance for a particular incident, the hospital can retrieve a ledger report showing every step taken (e.g., "Event detected at 14:03:22 by Pump#77; triaged as Critical; Dr. Smith reviewed at 14:10 and added note; auto-report sent to State system at 14:20 with confirmation ID XYZ"). Because the ledger is cryptographically secured, it **ensures the authenticity of the record** – entries are digitally signed and chained, preventing tampering[9]. This level of auditability is an advancement over prior systems where incident logs might be editable or scattered in emails and paper notes.

- **Clinician Override and Review Workflow:** Automation must be balanced with human oversight for safety. The **override workflow** allows clinicians to intervene when necessary, which is a crucial safety feature. For example, if the system mis-classifies an event or lacks context, a clinician (e.g., risk manager, nursing supervisor) can adjust the classification or prevent an erroneous external report. The system supports this through a secure dashboard where pending events are listed, along with the system's recommended actions. Authorized users can approve or modify these actions. *Fail-safe logic* is incorporated: overriding a mandatory report might require entering a justification and possibly a higher-level approval, to avoid suppressing legally required reporting without due consideration. All overrides are logged in the ledger, as noted, so that there is transparency about any deviations from automated decisions. This **dual-layer (automated + human) approach** greatly reduces the chance of both false negatives (missed reports) and false positives (unnecessary reports), thus enhancing overall safety.

- **Automated Alerts & Escalation:** The system's notification subsystem acts as a safety net by ensuring that critical events get human attention immediately. If a high-severity event is detected, the system doesn't only rely on automated reporting; it also directly alerts clinicians (for patient rescue) and administrators (for management). For instance, in an opioid overdose scenario, the system might trigger an alert to the rapid response team or call a "code blue" if the patient is in danger, independent of the reporting process. This real-time **clinical failsafe** ensures patient care actions are not lost in the focus on reporting. The escalations can be multi-tiered: if an assigned responder does not acknowledge within a certain time, the alert escalates to another person or level of authority.

- **Rules Engine Safeguards:** The rules engine includes **safety checks** to avoid errors in compliance. It might have redundancies such as double validation for critical conditions. For example, if an event is very close to a state threshold, the engine could flag it for human review rather than automatically dismissing it. It also logs how it reached each decision, which is useful for auditing its accuracy. The knowledge base for rules is maintained up-to-date (e.g., if a state changes a law effective on a future date, the engine can handle versioning of rules by date). This prevents non-compliance due to outdated criteria.

- **Immutable Evidence for Legal Protection:** By tagging events as PSWP where appropriate and capturing them in the secure database, the system helps the hospital maintain the **legal protections of the Patient Safety Act**. Documents or data prepared for the PSO are sequestered within the system's PSO module and labeled appropriately, which assists the hospital in demonstrating that analysis and discussions of those events

are privileged. If a legal discovery process occurs, the hospital can delineate which information is PSWP (thus confidential) and which was part of mandatory reports, with confidence grounded in the system's records[5]. The cryptographic logging of when data was sent to a PSO can prove that certain analyses were indeed part of that protected process.

- **Data Integrity and Validation:** The system performs validations at each stage to ensure data integrity. Connectors verify the authenticity of source data (e.g., checking digital signatures if devices sign their alerts, or at least validating message formats). The normalization stage might do data quality checks (like ensuring patient IDs are valid, timestamps make sense). The ledger inherently ensures any corruption in data would be detectable (if someone tried to meddle, the hashes wouldn't match). Backups of the ledger and event database are maintained (possibly distributed across nodes, especially if blockchain tech is used, enhancing fault tolerance). Thus, the chance of losing event data or having undetected errors is minimized.

- **Access Controls and Privacy Auditing:** Access to view or edit events in the system is restricted by role. The system can log every read access as well (for instance, which user viewed a particular event record), creating an audit trail of access to sensitive information. This is akin to EHR access auditing for HIPAA, now applied to the incident system – supporting the idea that every access is accountable. If someone without authorization attempts access or if any anomalous usage is detected, the system can flag it. This protects patient privacy even within the safety system.

- **Failover and Robustness:** From a safety-of-operation perspective, Sentinel Connect is designed with high availability. In the event of a system outage or network failure, it has local buffering – e.g., device connectors might temporarily store events until connectivity is restored. The ledger can be kept on redundant nodes. This ensures the **safety system itself does not become a single point of failure.** Hospitals can trust it to be reliably capturing events 24/7.

In summary, the safety and audit modules together create a **secure, trustworthy framework**. They ensure that while the system automates detection and reporting of errors (improving speed and compliance), it does so in a controlled manner where humans can intervene and everything is documented. This greatly mitigates the risk of technology-induced errors or liability, making Sentinel Connect a reliable "sentinel" for the hospital's patient safety program.

Use-Case Examples

To illustrate the operation of SmartStop Sentinel Connect™, consider the following real-world sentinel event scenarios and how the system handles them:

Example 1: Opioid Overdose Averted by Smart Pump

Scenario: An IV smart pump is infusing a high-potency opioid (e.g., morphine) via patient-controlled analgesia (PCA). Due to a programming mistake, the concentration was entered incorrectly and the pump is about to deliver a potentially lethal dose. The pump's built-in

safety software (DERS) catches that the dose is outside of safe limits and triggers an "Infusion Error – Dose Limit Exceeded" alarm. The patient also shows signs of respiratory depression.

System Detection: The smart pump's Sentinel Connect **connector** immediately captures the overdose alarm event. Simultaneously, if integrated, the patient's vital sign monitor might send an alert of low respiratory rate – the system can correlate these events (pump alarm + patient vitals) as part of the same incident.

Normalization & Triage: The event is normalized as a **Medication Safety** event with subtype "Overdose – High Risk". The triage engine assigns it the highest severity (potentially fatal opioid overdose) and urgency (occurring now, patient in danger). The composite risk score is **Critical**, triggering instant escalation.

Real-time Response: Even before formal reporting, Sentinel Connect's alert module notifies the ward's nurses and the rapid response team. A "Code Blue" or similar emergency call could be auto-activated via integration with hospital alerting systems, ensuring clinicians administer naloxone antidote and ventilatory support immediately. This shows how the system helps **prevent harm in real time**, not just report it.

Rules Engine Outcome: The rules engine analyzes the event details. The patient required intervention (naloxone) to prevent death, which means this incident qualifies as a serious reportable adverse drug event. Many state regulations would classify this as a **sentinel event (unanticipated serious harm)** that must be reported. For instance, if the state rule says any medication error requiring an intervention to sustain life is reportable, this case meets that threshold. The engine sets "Reportable to State = YES". It also tags it for PSO: since the patient was harmed (or at serious risk), the hospital certainly wants to analyze it under PSO confidentiality as well. So "Report to PSO = YES (as PSWP)" is marked. Additionally, an internal RCA is mandated (the engine knows an RCA must be done for any sentinel event like this within 45 days per Joint Commission guidelines, for example).

Export Actions: The export engine automatically compiles a **State Incident Report**. Let's say the state health department uses an online form; the system fills in patient anonymized ID, date/time, event description ("Opioid overdose, naloxone administered, patient revived"), and other required fields (probably severity, location, etc.). It also references that a full RCA will follow. The risk manager gets a notification to review this report. With one click, they approve and the system submits it electronically (or they can add more info before submission). At the same time, the system packages the de-identified event data for the **PSO**. Using AHRQ Common Formats, it categorizes it as an "Incident – reached patient, harm occurred"[13] and sends it to the affiliated PSO's secure server. Internally, an event entry is created in RLDatix (or the hospital's incident management tool) so all stakeholders can follow up, document the RCA findings, and track actions.

Ledger and Audit: Every step – pump alarm at specific time, alert sent to staff at time, report generated, user X approved report at time, report sent – is logged immutably. Weeks later, during the RCA meeting, the team pulls up the timeline to see exactly how fast the response was and how the process flowed. The ledger shows, for instance, "detected to treatment = 2 minutes, detected to external notification = 2 hours" which might be an improvement over a manual process that might have taken days to identify and report the event.

Outcome: The patient recovered (thanks to prompt detection and response). The hospital satisfied its reporting duty promptly. The event data is now part of a PSO dataset for learning (contributing to perhaps identifying trends in PCA errors). And internally, they'll do an RCA (the system has already populated an RCA template with basic info, saving time). This closed-loop example demonstrates how Sentinel Connect not only catches a deadly error in time but also automates the after-action documentation, letting clinicians focus on patient care in the moment.

Example 2: Wrong-Site Surgery Prevention and Reporting

Scenario: A patient is scheduled for surgery on the left knee. However, a miscommunication in the OR scheduling system results in the surgical team preparing to operate on the right knee. The hospital employs a **surgical site tracking system** with RFID tags – the patient is wearing an RFID tag associated with the surgery plan (which includes the correct site), and the operating room has a scanner.

System Detection: Just before incision, the surgical site tracking system scans the patient's tag and the planned procedure. It detects a mismatch (wrong site) and triggers an alert: "**Wrong Site Surgery Prevented** – Left vs Right discrepancy." The Sentinel Connect connector for the surgical tracker picks up this alert. Additionally, suppose the circulating nurse had also entered a manual incident report into the EHR or pressed a button on a safety checklist app when the near-miss was caught; that too can be ingested.

Normalization & Triage: The event is normalized under **Procedure Safety** with type "Wrong-Site Surgery – Near Miss". The triage engine assesses severity: if it had proceeded it would be devastating (permanent harm), but it was caught in time so the patient wasn't harmed. The severity is still rated extremely high (because the potential was severe harm) but the urgency is lower now since it was prevented just in time. Overall, it's a **High priority** event for investigation (even though no immediate patient rescue is needed, it's a **"hair's breadth from catastrophe"** kind of event that merits serious attention).

Rules Engine Outcome: Wrong-site surgery is universally considered a **never event**. Even though in this case it was *prevented*, it reached the point of near-incision which many regulators would still consider a serious reportable event (some states require reporting of "any wrong-site surgery that was started, whether or not completed" and even near misses might be reportable to patient safety authorities for learning). The rules engine likely marks "Report to State = YES" because it fits the category, albeit perhaps as a near-miss. It also sets "Report to The Joint Commission = advisable" (since The Joint Commission highly prioritizes such events). And definitely "Report to PSO = YES" to analyze how the safety net caught it just in time. If the hospital's policy is to treat near-miss wrong-site surgeries with equal gravity, the engine might also spawn tasks like scheduling a formal *case conference*.

Intervention: In this scenario, the system might present an **override opportunity**: because technically no harm occurred, some jurisdictions might not legally mandate reporting of near misses (they encourage it but don't require). The risk manager receives the event details marked as a "close call". They might use the Sentinel Connect interface to add a comment: "Surgery aborted before incision – no patient harm. Will report as near miss per our policy." They choose to still notify authorities voluntarily to contribute to safety culture. Or if a particular state did not

want near-misses, the manager could override the state report while still sending it to the PSO. Whatever the decision, it's captured and logged.

Export Actions: The system prepares an incident report. Perhaps the state has a category for "serious preventable event – wrong site". The report is filled with what happened, and notably that the patient was unharmed. This is submitted to the state's sentinel event registry within the required timeframe (e.g., some states might still want notification within 24 hours of even the near miss). A copy is generated for The Joint Commission's voluntary reporting system (if the hospital opts to send it, they can transmit it or upload it). For PSO, the common format entry would be "Near miss – unsafe condition that *could* have caused harm"[13]. That data goes into the confidential database for aggregate learning (and indeed, PSOs love to analyze near misses because they signal where systems are catching errors or almost failing).

Follow-up: Internally, an RCA is definitely warranted: how did the scheduling error happen, how did the checklist/RFID catch it, how to prevent even getting that far. Sentinel Connect's integration with the RCA tool sets up a case for investigation. The timeline of detection is recorded (e.g., "patient in OR at 10:00, RFID alert at 10:05, surgery stood down at 10:06"). This precise timeline helps the RCA team analyze how close it was and how effective the stopgap was.

Outcome: The wrong-site surgery was prevented – a major save. The system ensured that even though it was a near miss, it was treated with the seriousness of an actual event by automatically initiating reports and analysis. This demonstrates how Sentinel Connect handles **not just events that cause harm, but also near misses**, which are equally important for improving safety. It captured the event that, historically, might have gone unreported (since no one was hurt, staff might be relieved and move on). Instead, the system's presence creates a culture where even near-misses are systematically recorded and learned from, without significantly adding to staff workload.

Example 3: Inpatient Suicide Attempt

(Another brief example to show behavioral health integration:)

A patient in a behavioral health unit attempts suicide by hanging but is rescued by staff, resulting in serious injury. A **behavioral risk assessment tool** had flagged the patient as high risk earlier, and Sentinel Connect had logged that risk score. When the **bed sensor** detects abnormal pressure (patient attempting to stand on bed rail, etc.) and an alarm triggers, staff respond and find the patient. Sentinel Connect correlates the incident with the high-risk flag. The triage engine marks it as critical (severe harm occurred). The rules engine knows inpatient suicide attempts are sentinel events that **must be reported to The Joint Commission and state** in many cases. It flags those reports. The export engine sends immediate notification to hospital leadership and prepares the regulatory report. Meanwhile, the ledger logs the timeline – notably it captures that the risk was known and an observation rounding was perhaps missed by 5 minutes. This information becomes part of the review. The system's holistic view (from risk prediction to incident) provides a fuller picture to analyze what went wrong in supervision. The safe-harbor tagging applies here too, so the hospital can share this with a PSO for broader learning without fear of legal exposure.

Each of these use cases highlights different aspects: medication devices, surgical systems, psych unit safety – showing the **modular and extensible nature** of Sentinel Connect in handling a variety of sentinel events across hospital domains.

Advantage Over Prior Art

Prior to Sentinel Connect, hospitals typically relied on a patchwork of manual processes and siloed software to manage sentinel events. **Traditional incident reporting systems**, such as RLDatix or home-grown databases, generally depend on staff to manually enter event details into electronic forms[8]. These systems can capture data and produce basic reports, but they **lack automated detection**, integration with real-time devices, and advanced decision support. Key advantages of Sentinel Connect over such prior art include:

- **Automated Event Capture vs. Manual Reporting:** In legacy setups, if an infusion pump alarmed or a patient fell, a nurse would have to notice the incident and then fill out a report (often after attending to the patient). This delays the reporting and might even miss events (near misses especially go unreported due to time or fear factors). Sentinel Connect eliminates this gap by *automatically capturing events from devices and systems as they happen*. It doesn't rely solely on human memory or willingness to report. This leads to more comprehensive incident data (including near misses and minor events that are crucial for prevention analysis) and faster initiation of response protocols.

- **Unified, Federated Registry:** Traditionally, each department might have its own log (the pharmacy tracks medication errors, the surgery department tracks OR mishaps, etc.), or disparate tools don't talk to each other. Sentinel Connect provides a **unified registry** where all sentinel events across the enterprise are stored in one place with a common taxonomy. This federation of data enables organization-wide analytics – patterns that span departments (e.g., similar errors occurring in different units) can be spotted more easily. Prior art lacked such an integrated view; they were often **fragmented systems** that hindered big-picture safety improvements.

- **Real-time Risk Stratification:** Conventional systems log events but generally do not prioritize them in real time. At best, someone might scan incoming reports and decide what's urgent. Sentinel Connect's **priority triage engine** systematically scores risk and urgency, ensuring that truly critical events get immediate attention. This is akin to having an automated triage nurse for patient safety events, something prior incident management tools did not have. It helps hospitals focus resources where it matters most, potentially saving lives by accelerating response to time-critical events.

- **Embedded Regulatory Knowledge:** One of the significant innovations is the rules engine with encoded statutes and PSQIA guidelines. Historically, compliance was a manual effort – risk managers had to recall or look up what needed reporting, fill out forms by interpreting event data, and remember deadlines. Mistakes in compliance (failing to report on time or at all) could result in penalties or accreditation issues. Sentinel Connect **builds this knowledge in**; it acts as a virtual compliance officer that *automatically knows* the reporting rules[4]. This reduces reliance on individual expertise and mitigates the risk of non-compliance due to oversight. No existing incident reporting

product encapsulated regulatory rules to the depth of mapping each event to multi-jurisdictional requirements.

- **Comprehensive Safe Harbor Integration:** Prior art systems rarely integrated with PSOs automatically. Hospitals would have separate processes for feeding data to a PSO (often manual exports or periodic uploads). Sentinel Connect seamlessly distinguishes PSWP and non-PSWP and funnels data appropriately, thus **maximizing the protections of the Patient Safety Act** without extra effort. This encourages more candid internal reporting and analysis, improving safety culture. Competing systems either ignored PSO reporting or left it entirely to users, meaning many learning opportunities were lost. Our system's ability to tag and route events to PSOs by default (when allowable) is a major advantage in building a robust learning health system.

- **Immutable Audit Trail and Traceability:** Traditional databases can be edited – unfortunately, this can lead to intentional or unintentional alteration of records (for example, someone might "fix" an incident entry later or a technical glitch might lose data). By using an immutable ledger, Sentinel Connect provides unprecedented **data integrity**[9]. This not only helps with internal trust ("the data hasn't been tampered with by anyone") but is also extremely valuable in legal/external contexts. If a malpractice claim or investigation arises from an event, the hospital can produce a cryptographically assured log of how the event was handled. Prior art lacks this level of security; most systems have basic audit logs at best, but not a chain-of-custody grade ledger.

- **Interoperability and Cross-Platform Support:** Sentinel Connect was built with interoperability as a core principle. It **interfaces with EHRs (e.g., Epic)** to both gather information (like patient data, orders, etc. that contextualize an event) and potentially feed back information (like noting in the EHR that an incident report was filed for a case). It ties into **medication dispensing systems (Pyxis)** to catch errors at the point of dispensing, and into external risk management platforms (like RLDatix) to ensure continuity for users who rely on those platforms. Competing solutions often lock you into their ecosystem or require heavy custom integration work. In contrast, our system provides out-of-the-box connectors and export templates that cover major platforms, using standards like HL7, FHIR, or CSV/XML data exchange. This means Sentinel Connect can act as a **central hub, augmenting existing systems** rather than forcing a rip-and-replace. For example, if a hospital loves their RLDatix for managing follow-up tasks, they can keep using it; Sentinel Connect will just feed it better data automatically. This flexibility is a strong differentiator.

- **Adaptability and Extensibility:** The invention's modular design allows it to quickly adapt to new types of sentinel events or integrate new innovations (like AI predictive analytics for detecting events before they happen). Prior art systems are often static, requiring vendor changes for any new functionality. With Sentinel Connect, adding a new module (say a connector for a new wearable that detects patient distress) or updating a rule is straightforward. This future-proofs the hospital's investment in safety infrastructure – as healthcare technology evolves (e.g., new implants that send alerts), the system can incorporate them, staying ahead of emerging patient safety issues.

- **Improved Workflow and Efficiency:** By automating data entry and report generation, the system significantly **reduces the workload** on clinical and quality staff. Nurses and doctors spend less time filling out forms and more time on patient care. Risk managers spend less time gathering info and more on analysis and prevention strategies. This efficiency leads to cost savings (less administrative overhead) and faster throughput of safety investigations. Moreover, automated notifications and follow-ups mean fewer things fall through the cracks (no waiting for someone to "remember to report that incident"). The net effect is a more proactive and reliable safety program compared to the largely **reactive, manual prior art**.

- **Data for Analytics and Machine Learning:** With all sentinel events in one structured database, the hospital (or vendor) can apply analytics or machine learning to predict and prevent future events. While not the core of the question, it's worth noting as an advantage: once data is aggregated, patterns like "X type of pump errors increasing on certain ward" can be spotted. Over time, Sentinel Connect could incorporate predictive alerts (e.g., flagging if multiple near misses indicate a pattern). Legacy systems, with fragmented data, made such analysis difficult. Our unified approach sets the stage for **continuous improvement and learning**, which is ultimately the goal of patient safety efforts[14][15].

In summary, SmartStop Sentinel Connect™ stands out from prior art by providing an **integrated, intelligent, and secure sentinel event management framework**. It brings together detection, analysis, and reporting in one loop, whereas older solutions addressed these pieces in isolation. By doing so, it not only ensures better compliance and faster response but fundamentally shifts the paradigm from passive reporting to active safety surveillance.

Modular Expansion for Non-Infusion Sentinels

While the initial impetus for Sentinel Connect includes infusion pump errors (a frequent source of serious medication events), the system is deliberately architected to be **modular and extensible** so it can handle **any type of sentinel event** – including those outside the medication domain. The modular expansion is achieved through both hardware-agnostic input adapters and flexible data schema/rule sets. Here's how the system expands to new sentinel domains:

- **Plug-in Connectors for New Event Sources:** The platform supports adding new **device connectors or integration plugins** without core changes. For example, if the hospital wants to incorporate radiology incidents (like radiation overdose from CT scanners), a connector can be developed to monitor the CT machine's dose logs or the PACS system alerts. Similarly, for **laboratory errors** (like a blood transfusion mismatch), connectors could tie into the lab information system or blood bank system for critical alerts. These connectors feed events into the same normalization pipeline. The system's connector SDK (Software Development Kit) provides standardized interfaces, meaning as soon as the new connector outputs an event in the expected format, all downstream components (triage, rules) can handle it. Thus, expansion is a matter of adding a module and configuring it, rather than retooling the whole system.

- **Extensible Unified Schema:** The unified event schema is designed to be **extensible** with new event types and fields. It is not limited to infusion or falls; it has a taxonomy that can

encompass surgical events, device malfunctions, security incidents (like infant abduction alarms), etc. For any new category, one can define new codes or values in the schema. For instance, adding **"Fire in OR"** as a sentinel event type (which indeed some consider a reportable event) would involve adding that code to the schema and teaching the rules engine what to do with it. The use of industry standards where possible (like leveraging Common Formats which have multiple event types built-in[16]) makes it easier to extend with confidence that definitions align with accepted norms.

- **Configurable Rules for New Statutes/Guidelines:** When expanding to a new domain, new rules can be written and loaded into the rules engine knowledge base. For example, if we add **maternal obstetric events** (like a maternal death or severe morbidity – also sentinel events), we would encode the rule "IF maternal mortality event THEN report to state within X timeframe (because many states mandate maternal death reporting)". The rules engine is essentially a repository of "IF conditions THEN actions" that can be updated as needed. The system's admin interface or configuration files allow adding these rules or modifying existing ones when laws change or when the hospital decides on new internal reporting criteria.

- **Scalable Triage and Scoring Mechanism:** The triage engine's algorithms can be tuned or expanded per event type. Each event type can have a defined severity mapping. For example, when incorporating **falls**, the hospital might decide: a fall with no injury is low severity, with injury is moderate, with fracture is high, etc. These can be configured so that the risk scoring remains accurate across domains. The triage engine can even incorporate different scales for different event classes (it could use the HIT hazard score for technology malfunctions vs. use a different index for clinical errors, yet all normalized to a priority ranking). The expansion modules simply provide the parameters or plugin functions to evaluate severity for those new event types.

- **Additional Export Channels:** As new sentinel event types are handled, there may be corresponding new **reporting destinations**. For instance, expanding to **device-related sentinel events** might involve reporting certain issues to the FDA (e.g., an implant that nearly caused death – FDA's MedWatch could be relevant). The system can integrate new export modules for these channels as well. A "FDA MedWatch exporter" could be added so that whenever the rules engine flags an event as requiring FDA notification (say an adverse device event), the data is formatted for that (perhaps even auto-filling a MedWatch form). Thus, modular expansion covers not just inputs but outputs too.

- **Cross-Industry Adaptability:** Although designed for hospitals, the framework could be extended to other care settings – e.g., nursing homes or ambulatory surgery centers – by developing connectors and rules relevant to those environments (since Common Formats exist for those as well[17]). This could allow a health system to use a unified sentinel event system across different facility types, customizing the modules as needed.

- **Non-Infusion Examples:** Consider a **hemodialysis machine** that has a blood leak alarm (a potentially life-threatening situation). A connector for dialysis machines could feed that alarm in; triage marks it critical; rules say it's reportable to state if it led to patient injury; exports handle notifying ESRD networks or whomever is required. Or consider an **IT system outage** that compromises patient safety (like downtime of clinical systems –

some regulators consider prolonged EHR outage causing delay in care a reportable event). The system could take feeds from IT monitoring tools and incorporate those events for administrative follow-up. These examples show that *any event that could be defined and detected* can be folded into Sentinel Connect.

- **User Interface and Workflow Scaling:** The system's dashboards and workflows are built to handle multiple event types without confusion. They use filters and categorization so users can focus (e.g., a pharmacy safety officer might filter to see only med events, whereas a patient safety officer sees all). As new types are added, they appear as new filters or sections, maintaining usability. The modular design ensures that adding a new event type doesn't overwhelm users with noise – because the triage and filtering will still highlight what's important to whom.

- **Continuous Learning Loop:** Each expansion also benefits from the system's core learning approach. As new modules are added, the data collected can be studied to refine the system further. For instance, after adding a **new fall detection AI**, the system might pick up many alerts; analysis of those alerts might lead to fine-tuning the triage or ignoring certain false alarms. The architecture facilitates this because all data is there to analyze and the components are tunable in isolation.

In essence, SmartStop Sentinel Connect is not a static product limited to one domain of errors. It's a **framework that grows with the hospital's patient safety program**. Infusion errors, medication errors, surgical events, device failures, falls, diagnostic errors, cybersecurity incidents affecting patient care – all can be brought under the umbrella of Sentinel Connect by leveraging its modular expansion capability. This extensibility ensures longevity and relevance: as new patient safety challenges emerge (for example, imagine in the future, AI decision support failures might be a category of error), the system can be updated to catch and manage those as well. The invention's value thus compounds over time, continuously increasing the scope of patient harm it can help prevent or manage.

Footnotes [1] [10] [14] [15] Sentinel Event Policy and Procedures | Joint Commission

https://www.jointcommission.org/en-us/knowledge-library/support-center/standards-interpretation/sentinel-event-policy-and-procedures

[2] [13] [16] [17] PSOPPC: Common Formats - FAQ

https://www.psoppc.org/psoppc_web/publicpages/faq

[3] [4] [11] [12] Sentinel Events | Department of Health and Human Services

https://www.maine.gov/dhhs/dlc/safety-reporting/sentinel-events

[5] [6] [7] Patient Safety and Quality Improvement Act - Wikipedia
https://en.wikipedia.org/wiki/Patient_Safety_and_Quality_Improvement_Act
[8] Event Reporting | Incident Reporting Software | RLDatix
https://rld.rldatix.com/en-apac/solutions/how-we-help/risk/event-reporting/
[9] The role of blockchain in healthcare audits
https://www.paubox.com/blog/the-role-of-blockchain-in-healthcare-audits