AI MEDAGENT PLATFORM

Software Developer Kit

Specification v2.0

Prepared by: Daniel C. Pettus
Version: 2.0 — Deployment Architecture + Incremental Adoption
Date: June 2026
Status: Draft for Review —
Contact: danpettus@outlook.com | danielpettus.com

What This Document Is

This SDK defines how external software systems connect to the AI MedAgent platform. It covers the deployment architecture, the on-premise gateway component, the cloud AI engine, the IHE PCD transaction profiles the platform uses, the laboratory interface standards, and the incremental adoption model that allows hospitals to start with a single connected data source and expand over time.

It is written for three audiences: hospital IT and informatics directors evaluating the platform, biomedical engineers implementing device integrations, and software developers building custom adapters. Each section is labeled with its primary audience.

1. The Problem This Platform Solves

Medication management in acute care is broken at the architectural level. The industry has invested heavily in point solutions — CPOE, barcode medication administration, smart infusion pumps, automated dispensing cabinets, electronic health records — that each demonstrably work within their own domain. The problem is that every handoff between these systems remains fragmented, manual, and reactive. The nurse who manually programs a pump, the pharmacist who cannot see the infusion state in real time, the physician whose order exists in the EMR but has not reached the device at the bedside: these are not failures of individual technology. They are failures of architecture.

AI MedAgent is not another point solution. It is the intelligence layer that sits across all of these existing systems simultaneously, continuously reading their data regardless of vendor or protocol, and acting on it in real time to optimize medication therapy before harm occurs.

> **Core Principle:** Every hospital already has the data needed to run this platform. The devices are there. The systems are there. The data is being generated every second. AI MedAgent connects what is already deployed and applies intelligence to it.

2. Deployment Architecture

AI MedAgent uses a hybrid edge-cloud architecture. This is not a choice of convenience — it is the only architecture that satisfies the simultaneous requirements of sub-second clinical response times, data sovereignty, HIPAA compliance, and access to population-scale AI learning.

2.1 The Two-Layer Design

The platform operates in two distinct layers that work together.

Layer	Component	What It Does
Layer 1: Edge	AI MedAgent Gateway (on-premise)	A lightweight software appliance deployed inside the hospital network. Connects directly to clinical systems via HL7 and IHE PCD protocols. Handles all time-critical reactive agent decisions locally. Never sends raw PHI to the cloud.
Layer 2: Cloud	AI MedAgent Engine (AWS-hosted)	Receives anonymized, de-identified clinical feature vectors from the Gateway. Runs deep learning population models. Returns therapy optimization recommendations. Never receives or stores raw patient identifiers.

2.2 The On-Premise Gateway

The Gateway is the most important component in the platform. It is the piece that lives inside the hospital firewall and does the work that cannot be delegated to the cloud.

The Gateway is deployed as a containerized application (Docker/Kubernetes) on a hospital-managed server or virtual machine. It requires no dedicated hardware — it can run on existing hospital infrastructure that meets minimum specifications. It connects to clinical systems using the hospital's existing interface engine (Mirth Connect, Rhapsody, Corepoint, or similar) via standard HL7 MLLP connections and IHE PCD message streams.

Gateway Responsibilities	What This Means in Practice
Receives IHE PCD-10 pump event streams	Listens for every infusion state change from every connected pump, event-driven, as it happens
Receives IHE PCD-01 device observations	Polls patient monitors and ventilators for vital signs and respiratory parameters on a configurable interval
Receives IHE PCD-04 device alarms	Listens for all device alarms from all connected systems; places immediate safety holds on affected agents
Receives HL7 lab results (LRI)	Subscribes to LIS lab result messages; filters for medication-relevant analytes
Receives FHIR R4 medication orders	Subscribes to CPOE order events via SMART on FHIR or HL7 ORM
Receives ADC dispense events	Subscribes to dispensing cabinet HL7 RDS messages
Runs all six Reactive Agents locally	All real-time titration decisions are made inside the hospital network, never requiring cloud connectivity
Sends IHE PCD-03 pump programming orders	Transmits AI-generated infusion orders to connected pump controllers after pharmacy approval and nurse notification
Sends anonymized feature vectors to cloud	Every 4 hours per patient, sends de-identified clinical trajectory data for deep learning optimization
Receives optimization recommendations from cloud	Applies pharmacy-approved DL optimizer recommendations by routing through the order workflow
Maintains local audit trail	Complete immutable log of all events, decisions, and actions stored inside the hospital network

> *Critical design point: The Reactive Agents that handle immediate titration decisions run entirely on the Gateway. They do not depend on internet connectivity. A cloud outage or network interruption does not interrupt real-time medication management. The platform is designed for clinical environments where reliability is non-negotiable.*

2.3 The Cloud AI Engine

The cloud layer is deployed on AWS in a HIPAA-eligible environment with a signed Business Associate Agreement (BAA). It never receives raw patient data or direct identifiers. What it receives is a de-identified clinical feature vector — a structured numerical representation of a patient's physiological trajectory that cannot be reverse-engineered to identify the patient.

The AWS services used are:

- AWS API Gateway: Receives encrypted feature vectors from hospital gateways. Authenticates via mutual TLS certificates and OAuth 2.0 client credentials. Each hospital gateway has a unique client certificate issued during onboarding.
- AWS Lambda: Processes incoming feature vectors and routes them to the appropriate AI model endpoint.
- Amazon SageMaker: Hosts the trained deep learning models (PKPD optimization, population outcomes, pharmacogenomic risk). Models are retrained quarterly on aggregated anonymized data from all connected hospitals.
- Amazon HealthLake: Stores FHIR R4-formatted de-identified clinical events for population model training and analytics.
- Amazon Kinesis: Real-time streaming of feature vectors for low-latency model inference.
- AWS CloudWatch + CloudTrail: Complete audit logging of all cloud operations.

> **Data Sovereignty:** Raw PHI never leaves the hospital network. The Gateway is the data boundary. What crosses to the cloud is a mathematical representation of clinical patterns, not patient records. Hospitals retain full ownership and control of their data.

2.4 The Secure Connection

The connection between the on-premise Gateway and the cloud AI Engine uses a dedicated TLS 1.3 encrypted channel with mutual certificate authentication. This is not a VPN. It is an outbound HTTPS connection initiated by the Gateway — a model familiar to hospital IT teams from other cloud-connected clinical applications.

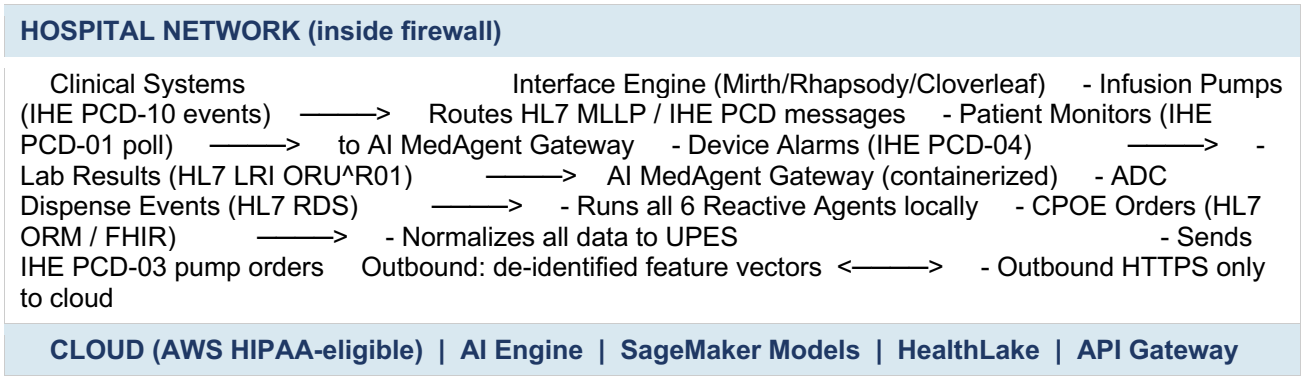
Connection characteristics:

- Outbound-only from the hospital: The Gateway initiates all connections to the cloud. No inbound firewall rules required. Hospital IT opens one outbound HTTPS port (443) to a single AWS endpoint.
- Mutual TLS authentication: Both the Gateway and the cloud endpoint authenticate with certificates. Certificates are issued during onboarding and rotated annually.
- OAuth 2.0 client credentials: Every API call carries a signed JWT token with a 60-minute expiry. Tokens are automatically refreshed by the Gateway.
- End-to-end encryption: All data in transit is encrypted with TLS 1.3. All data at rest in AWS is encrypted with AES-256 using AWS KMS-managed keys.
- Graceful degradation: If the cloud connection is unavailable, the Gateway continues operating in reactive-only mode. All six titration agents continue running locally. Deep learning optimization resumes when connectivity is restored.

2.5 Where the Gateway Sits in the Hospital Network

The Gateway connects to clinical systems through the hospital's existing interface engine — the same software already routing HL7 messages between EMR, pharmacy, laboratory, and nursing systems. In most hospitals this is Mirth Connect, Rhapsody, Cloverleaf, or a vendor-supplied interface engine. AI MedAgent does not replace it. It listens alongside it.

Network position: The Gateway is deployed in the hospital's clinical network segment with access to the interface engine. It does not require access to the public internet beyond the single outbound HTTPS connection to the AI MedAgent cloud endpoint. It does not require access to the hospital's administrative network.



3. Incremental Adoption Model

This is one of the most important sections of this specification. AI MedAgent is designed to deliver value immediately with a partial implementation and to grow in capability as more data sources are connected. A hospital does not need to connect every system to get started. Every new connection adds intelligence.

3.1 The Adoption Philosophy

The fundamental insight is that AI MedAgent is an intelligence layer, not a replacement system. It never requires a hospital to decommission or replace existing equipment. It connects alongside what is already there. This means a hospital can start with a single integration today and add more over time as vendor cooperation increases, budget permits, and clinical confidence grows.

> *The platform is explicitly designed for the reality of hospital procurement: not all vendors will cooperate immediately. Some will cooperate quickly. Some will require clinical evidence first. Some will take years. AI MedAgent delivers value at every stage of that journey.*

3.2 Adoption Stages

Stage	Connected Systems	AI Capability Enabled	Clinical Value
1	Patient Monitors only (IHE PCD-01)	Vital sign trend analysis. MAP, SBP, HR, RR, SpO2 monitoring agents active. Alarm pattern recognition.	Early warning for hemodynamic drift. Nurse notification before parameters breach thresholds. No pump integration required.
2	Monitors + Lab Interface (PCD-01 + HL7 LRI)	Glucose agent active. Potassium and lactate co-monitoring. Correlates vital trends with lab values.	Insulin titration support. Vasopressor efficacy assessment. Metabolic trend analysis.
3	Monitors + Labs + ADC (+ HL7 RDS)	Dispense-to-vital correlation. Medication effect onset tracking. Drug library compliance monitoring.	Closed view of dispensed vs observed effect. Timeline analysis of medication administration.
4	Monitors + Labs + ADC + EMR (+ FHIR R4 / CPOE)	Order-to-effect tracking. Diagnosis-contextualized AI decisions. Full medication history for DL optimizer.	AI recommendations informed by patient history, diagnoses, and active orders. DL optimizer activated.
5	All above + Infusion Pumps (+ IHE PCD-10 + PCD-03)	Full closed-loop capability. Real-time infusion state tracking. Autonomous pump programming with nurse confirmation.	Complete medication management loop. Autonomous titration. Proactive therapy optimization.
6	All above + Ventilators (+ IHE PCD-01 ventilator)	Respiratory-medication correlation. Ventilator-sedation optimization. Full ICU picture.	Sedation depth optimization correlated with ventilator parameters. Integrated ICU management.

3.3 No Vendor Lock-In at Any Stage

Because AI MedAgent uses open standards throughout — IHE PCD profiles, HL7 v2, FHIR R4 — a hospital can connect any vendor's equipment at any stage without modifying the platform. If a hospital replaces its infusion pumps, the Gateway adapter for the new vendor is swapped without any other changes. If a hospital's pharmacy system is upgraded, the FHIR or HL7 connection is reconfigured without affecting any other adapter.

> **The promise:** A hospital that starts at Stage 1 today and reaches Stage 6 in three years will never have to replace or rebuild any part of the AI MedAgent integration to get there. Each stage adds to what came before.

4. IHE PCD Transaction Reference

The IHE Patient Care Device (PCD) Technical Framework defines the interoperability standards for medical device data exchange in acute care. AI MedAgent is built on four core PCD transactions. This section is the definitive reference for how each transaction is used within the platform.

4.1 PCD-01: Communicate PCD Data

Transaction type: Solicited (the Gateway requests data; the device responds).

Attribute	Detail
IHE Profile	Device Enterprise Communication (DEC)
HL7 Message	ORU^R01^ORU_R01
Direction	Device Observation Reporter (DOR) to Device Observation Consumer (DOC)
Trigger Model	Solicited: Gateway sends request on configurable interval; device responds
Coding Standard	ISO/IEEE 11073 MDC nomenclature in OBX-3; LOINC for clinical observations
AI MedAgent use	Inbound vital signs from patient monitors; inbound ventilator parameters
Adoption stage	Available from Stage 1 (monitors only)

PCD-01 is the broadest transaction — it covers all continuous physiological observations from any patient care device. The Gateway subscribes to PCD-01 data from monitors and ventilators and translates each observation into a UPES DeviceObservation event for the AI engine.

4.2 PCD-03: Communicate Infusion Order

Transaction type: Order-driven (fires when a medication order or rate change is generated).

Attribute	Detail
IHE Profile	Point-of-Care Infusion Verification (PIV)
HL7 Message	RGV^O15^RGV_O15 (Pharmacy/Treatment Give Message)
Direction	Infusion Order Programmer (IOP) to Infusion Order Consumer (IOC)
Trigger Model	Order-driven: fires when AI generates a new or modified infusion order
Acknowledgment	Two-level: First ACK (order received, format verified); Second ACK (pump confirmation or error)
Nurse role	Nurse confirms the pre-populated pump program before infusion begins — hardware enforced
AI MedAgent use	Outbound: Gateway sends AI-generated orders to infusion pump controller
Prerequisite	Pharmacy approval required before Gateway transmits PCD-03 for DL optimizer recommendations
Adoption stage	Available from Stage 5 (pump integration)

> *PCD-03 pre-populates the pump. It does not start the infusion. The nurse presses start. This clinical safety requirement is enforced at the pump hardware level and cannot be bypassed by any software, including AI MedAgent.*

4.3 PCD-04: Report Alert

Transaction type: Event-driven and unsolicited (the device fires when an alarm condition occurs or changes state).

Attribute	Detail
IHE Profile	Alert Communication Management (ACM)
HL7 Message	ORU^R40^ORU_R40
Direction	Alarm Reporter (AR) to Alarm Manager (AM) — unsolicited, no polling required
Trigger Model	Event-driven: fires on alarm start, update, or conclusion
Alarm states	Off (condition absent), Active (condition present), Latched (resolved but flagged)
Alert types	Physiological (patient state), Technical (device fault), Advisory (workflow notification)
Related transactions	PCD-05 (status report), PCD-06 (disseminate to handheld), PCD-07 (dissemination status)
AI MedAgent use	Gateway acts as Alarm Manager; all device alarms trigger immediate agent safety evaluation
Adoption stage	Available whenever any PCD-compliant device is connected

4.4 PCD-10: Communicate Infusion Pump Event

Transaction type: Event-driven and unsolicited (the pump fires whenever any clinically significant infusion event occurs).

Attribute	Detail
IHE Profile	Infusion Pump Event Communication (IPEC)
HL7 Message	ORU^R42 (event variant of ORU_R01)
Direction	Device Observation Reporter (DOR) to Device Observation Consumer (DOC)
Trigger Model	Event-driven: fires on any infusion state change; no polling required
Vocabulary	Rosetta Terminology Mapping (RTM) controlled vocabulary for pump events
Scope	Rate changes, volume milestones, nurse overrides, drug library lookups, completions, errors
Distinction from PCD-04	PCD-10 covers all events including normal ones; PCD-04 covers alarms requiring human attention
AI MedAgent use	Primary inbound stream for pump state; richer and more timely than any polling approach
Adoption stage	Available from Stage 5 (pump integration)

4.5 PCD Transaction Summary

Transaction	Profile	Trigger	HL7 Message	AI MedAgent Use
PCD-01	DEC	Solicited	ORU^R01	Inbound: monitor and ventilator observations
PCD-03	PIV	Order-driven	RGV^O15	Outbound: AI pump programming orders
PCD-04	ACM	Event (alarm)	ORU^R40	Inbound: all device alarms and advisories
PCD-10	IPEC	Event (all)	ORU^R42	Inbound: primary pump event stream

5. Laboratory Interfaces

Laboratory interfaces operate on a different architecture than IHE PCD device interfaces. There is no PCD transaction profile for lab data. Instead, laboratory results exchange follows two parallel paths: HL7 v2.5.1 LRI (Laboratory Results Interface) for traditional systems connected via MLLP, and FHIR R4 Observation resources for EMR-connected modern labs via SMART on FHIR subscriptions.

5.1 HL7 v2.5.1 LRI (Traditional LIS)

Most hospital laboratory information systems transmit results via HL7 ORU^R01 messages over MLLP. The LRI Implementation Guide standardizes the content. The Gateway subscribes to lab result messages through the hospital interface engine and filters for medication-relevant analytes.

Sample HL7 LRI message for glucose result:

```
MSH|^~\&|LIS|LAB|MEDAGENT_GW|HOSPITAL|20260605142230||ORU^R01|9876541|P|2.5.1
PID|1||MRN-2026-4471^^^HOSPITAL^MRN||DOE^JOHN^A||19580312|M
OBR|1|ORD-44711|RES-88203|1554-5^Glucose^LN|||20260605141500
OBX|1|NM|1554-5^Glucose^LN||142|mg/dL|80-140|H||F|||20260605141500
OBX|2|NM|2823-3^Potassium^LN||3.9|mEq/L|3.5-5.0|N||F|||20260605141500
```

5.2 FHIR R4 Observation (Modern EMR-Connected Labs)

For hospitals using Epic or Oracle Health with modern lab workflows, the Gateway subscribes to FHIR R4 Observation resources via SMART on FHIR subscriptions. This provides real-time event-driven push of results without polling.

```
# FHIR R4 Observation — Glucose
{  "resourceType": "Observation",  "status": "final",
   "code": {"coding": [{"system": "http://loinc.org",
             "code": "1554-5", "display": "Glucose"}]},
   "subject": {"reference": "Patient/MRN-2026-4471"},
   "effectiveDateTime": "2026-06-05T14:15:00Z",
   "valueQuantity": {"value": 142, "unit": "mg/dL"},
   "interpretation": [{"coding": [{"code": "H", "display": "High"}]}]
}
```

5.3 Priority Analytes for Medication Management

Analyte	LOINC Code	Clinical Relevance
Glucose	1554-5	Insulin agent primary monitor; target 80-140 mg/dL
Potassium	2823-3	Insulin co-safety monitor; hypokalemia risk with insulin infusion
Lactate	2519-9	Vasopressor efficacy indicator; elevated lactate may require escalation
Creatinine	2160-0	Renal function; affects dosing of renally-cleared medications
Hemoglobin	718-7	Oxygen-carrying capacity; informs vasopressor thresholds
ABG pH	2746-6	Acid-base status; informs both vasopressor and ventilator agents
PaO2	2703-7	Oxygenation level; informs ventilator optimization
Troponin	10839-9	Cardiac injury; triggers protocol review in post-cardiac surgery patients

> *Point-of-care results from bedside glucose meters and blood gas analyzers are transmitted via the same HL7 ORU^R01 path as central lab results. The Gateway processes them identically, using OBX-14 timestamp for recency validation. A glucose result from a bedside POC device five minutes old is treated differently from one two hours old.*

6. Gateway SDK: Adapter Reference

The Gateway SDK provides pre-built adapters for each clinical data source. Each adapter handles protocol-specific connection, message parsing, UPES normalization, and error handling. Adapters are vendor-agnostic — they connect to any manufacturer's equipment that supports the relevant standard. No vendor-specific code is required.

6.1 Infusion Pump Adapter

Bidirectional. Receives PCD-10 event notifications from the pump controller (inbound) and transmits PCD-03 infusion programming orders back to the pump controller (outbound). Compatible with any pump controller that implements the IHE IPEC and PIV profiles.

```
from medagent.adapters import InfusionPumpAdapter

# Configure once during hospital setup
adapter = InfusionPumpAdapter(
    # Inbound: PCD-10 event stream (vendor-agnostic)
    pcd10_listen_port = 2577,       # Gateway listens on this port
    pcd10_profile     = 'IHE_IPEC', # Infusion Pump Event Communication

    # Outbound: PCD-03 programming orders
    pcd03_target_host = '10.10.1.71',  # Pump controller IP
```

```
    pcd03_target_port = 2578,
    pcd03_ack_timeout = 10,            # Seconds to wait for Level 1 ACK

    # Safety configuration
    nurse_confirm_required = True,   # Always True — do not change
    pharmacy_approval_required = True # For DL optimizer recommendations
)

# PCD-10 inbound: fires automatically on every pump event
@adapter.on_pump_event
def handle_pump_event(event):
    # event.event_type == 'InfusionPumpEvent'
    # Delivered to AI reactive agents automatically
    pass

# PCD-04 inbound: fires on any pump alarm
@adapter.on_alarm
def handle_pump_alarm(event):
    # event.payload['alarm_type'] triggers immediate safety hold
    pass

# PCD-03 outbound: called by routing engine after approvals
adapter.send_infusion_order(
    patient_id         = 'MRN-2026-4471',
    medication         = 'Norepinephrine',
    concentration_mcg_ml = 4.0,
    rate_mcg_kg_min    = 0.06,
    ordered_by         = 'AI_MEDAGENT',
    pharmacy_approved  = True
)
# Returns: {level1_ack: 'AA', level2_ack: 'AA', pump_prepopulated: True}
# Pump is pre-programmed. Nurse confirms at device before infusion begins.
```

6.2 Patient Monitor Adapter (PCD-01)

```
from medagent.adapters import MonitorAdapter

adapter = MonitorAdapter(
    profile        = 'IHE_DEC',     # Device Enterprise Communication
    protocol       = 'IHE_PCD_01',  # Solicited observation reporting
    target_host    = '10.10.1.80',  # Monitor or monitor gateway IP
    target_port    = 2580,
    poll_interval  = 30,            # Seconds between solicitation requests
    vitals         = ['MAP','SBP','DBP','HR','RR','SPO2','TEMP'],
)
# Gateway sends solicitation request every 30 seconds.
# Monitor responds with ORU^R01 containing OBX segments.
# MDC codes used by Gateway: MAP=150021, HR=147842, RR=151562, SPO2=150456
```

6.3 Alarm Manager Adapter (PCD-04/05/06)

```
from medagent.adapters import AlarmManagerAdapter

adapter = AlarmManagerAdapter(
```

```
    listen_port    = 2579,       # Gateway acts as Alarm Manager (AM)
    profile        = 'IHE_ACM',  # Alert Communication Management
)
# PCD-04 fires unsolicited from any connected device on alarm event.
# Gateway responds with PCD-05 (alarm status acknowledgment).
# Critical alarms trigger PCD-06 dissemination to nurse handheld.

@adapter.on_alert
def handle_alert(event):
    severity = event.payload['severity']  # CRITICAL / WARNING / ADVISORY
    if severity == 'CRITICAL':
        # Immediate nurse notification via PCD-06
        # AI agent placed in safety hold for affected medication
        pass
```

6.4 Laboratory Adapter

```
from medagent.adapters import LabAdapter

adapter = LabAdapter(
    # Traditional LIS: HL7 v2.5.1 LRI via MLLP
    hl7_listen_port = 2580,

    # Modern EMR-connected labs: FHIR R4 subscription
    fhir_base_url   = 'https://fhir.hospital.org/api/FHIR/R4',
    fhir_auth       = 'SMART_BACKEND',

    # Medication-relevant analytes only (see Section 5.3)
    loinc_filter    = ['1554-5','2823-3','2519-9','2160-0',
                       '718-7','2746-6','2703-7','10839-9'],
    critical_action = 'IMMEDIATE_AGENT_EVALUATION',
)
```

6.5 CPOE Adapter

```
from medagent.adapters import CPOEAdapter

adapter = CPOEAdapter(
    mode       = 'HL7_MLLP',     # or 'FHIR_SUBSCRIPTION'
    listen_port= 2575,
    filter     = {'encounter_class': 'IMP'},  # Inpatient only
)
```

6.6 ADC Adapter

```
from medagent.adapters import ADCAdapter

adapter = ADCAdapter(
    # Any ADC vendor supporting HL7 RDS^O13
    protocol   = 'HL7_RDS',      # or 'FHIR_MED_DISPENSE'
    listen_port= 2576,
)
# Inbound: dispense events from cabinet
# Outbound: formulary parameter updates after pharmacy approval
```

6.7 EMR Adapter (Open Standards — Any Vendor)

Follows the Open.Epic SMART on FHIR backend services model. Any FHIR R4 compliant EMR connects with credentials only — no code changes between Epic, Oracle Health, Meditech, or any other FHIR R4 system.

```
from medagent.adapters import EMRAdapter

adapter = EMRAdapter(
    fhir_base_url = 'https://fhir.hospital.org/api/FHIR/R4',
    auth_mode     = 'SMART_BACKEND',
    client_id     = 'your-medagent-client-id',
    private_key   = '/gateway/certs/rsa_private_key.pem',
    resources     = ['Patient','Encounter','Observation',
                     'MedicationRequest','MedicationAdministration',
                     'DiagnosticReport','Condition'],
)
```

7. Unified Patient Event Schema (UPES)

Every adapter normalizes its incoming data into a UPES event before delivering it to the AI engine. This normalization is what makes vendor-agnostic AI reasoning possible — the Norepinephrine agent does not know or care whether the MAP reading came from a Philips monitor or a GE monitor. It sees a UPES DeviceObservation with MAP = 63 and acts accordingly.

7.1 Event Types

UPES Event	Source	Protocol	Description
DeviceObservation	Monitor / Ventilator	IHE PCD-01	Continuous vital signs and device parameters (solicited)
InfusionPumpEvent	Infusion Pump	IHE PCD-10	Event-driven pump state: rate, volume, overrides, completions
DeviceAlarm	Any device	IHE PCD-04	Physiological alarm, technical fault, or clinical advisory
InfusionOrderSent	AI MedAgent (out)	IHE PCD-03	AI-generated pump programming order transmitted
MedicationOrder	CPOE / EMR	HL7 ORM / FHIR	New order, modification, or cancellation
MedicationDispense	ADC / Pharmacy	HL7 RDS / FHIR	Medication dispensed from cabinet or pharmacy
LabResult	LIS / EMR	HL7 LRI / FHIR	Glucose, potassium, lactate, and other analytes

PatientContext	EMR / ADT	HL7 ADT / FHIR	Demographics, ICD-10 diagnoses, care setting

7.2 UPES Envelope

```
@dataclass
class UPESEvent:
    event_id:        str        # UUID v4
    event_type:      str        # One of the types in 7.1
    patient_id:      str        # MRN (stays inside hospital network only)
    encounter_id:    str        # Active encounter reference
    source_system:   str        # e.g. 'ICU_PUMP_BED4' (configured at setup)
    source_protocol: str        # e.g. 'IHE_PCD_10', 'HL7_LRI', 'FHIR_R4'
    timestamp:       datetime   # UTC ISO 8601
    icd10_codes:     List[str]  # Active diagnoses for AI context
    payload:         dict       # Event-specific data (see adapter docs)
    raw_source:      str        # Original HL7 or FHIR (retained locally)
```

8. AI Decision Engine

8.1 Reactive Agents (Gateway — Local)

Agent	Drug	Monitor	Target	Step	Data Source
MAP	Vasopressor	MAP (mmHg)	>=65	0.02 mcg/kg/min	PCD-01 / PCD-10
SBP	Vasodilator	SBP (mmHg)	<=140	2.5 mg/hr	PCD-01
HR	Rate control	HR (bpm)	60-100	25 mcg/kg/min	PCD-01
RASS	Sedative	RASS Score	-1 to 0	5 mcg/kg/min	EMR nursing entry
RR	Opioid	RR (br/min)	>=10	12 mcg/hr	PCD-01
Glucose	Insulin	Glucose (mg/dL)	80-140	0.5 units/hr	HL7 LRI / FHIR

> *Reactive agents run entirely on the Gateway. They do not require cloud connectivity. A MAP of 62 triggers an escalation recommendation within seconds of the PCD-01 observation arriving at the Gateway — regardless of internet connectivity.*

8.2 Deep Learning Optimizer (Cloud — AWS)

The DL Optimizer runs every 4 hours per patient. It builds a de-identified feature vector from the full clinical trajectory and sends it to the AWS-hosted model for population-scale analysis. The Gateway receives a therapy optimization recommendation and routes it through pharmacy approval before any pump programming order is generated.

```
# Feature vector sent to cloud (de-identified — no patient identifiers)
feature_vector = {
    'diagnosis_codes': ['Z95.1'],     # ICD-10 context
```

```
    'vital_trends':    {...},          # Statistical summaries, not raw values
    'infusion_history':{...},          # Rate changes, duration, response
    'lab_trends':      {...},          # Directional changes, not identifiers
    'alarm_count':     3,             # Alarm frequency pattern
    'hospital_id':     'HOS_0047',    # De-identified institution code
    'cohort_match':    'CABG_POST_OP' # Matched population group
}
# Returns: recommendation with confidence score and evidence references
```

9. Order Routing Engine

When a Decision is generated — by either a Reactive Agent or the DL Optimizer — the Gateway routes it through the clinical workflow in a defined sequence. Every step is logged to the local audit trail.

1. Decision generated by Reactive Agent or DL Optimizer.
2. Pharmacy notification transmitted (HL7 OMP^O09 or FHIR MedicationRequest). Pharmacist reviews and approves via standard pharmacy workflow. DL Optimizer recommendations require pharmacy approval before any pump order is sent. Reactive Agent escalations for critical parameters send notification simultaneously with pump programming.
3. ADC formulary update transmitted after pharmacy approval. Any ADC compliant with HL7 RDS or supporting a FHIR MedicationDispense endpoint is updated automatically.
4. IHE PCD-03 pump programming order transmitted to the pump controller. Two-level acknowledgment received and logged. Pump is pre-populated with new rate parameters.
5. Nurse notification via IHE PCD-06 (ACM dissemination) to handheld device: medication name, current and recommended dose, rationale, confidence score, and three response options (Confirm, Hold, Override).
6. Complete audit log entry: full UPES event chain with timestamps, actor IDs, approval records, and acknowledgment status.

> *For hospitals at Stage 1-4 (no pump integration), Steps 4 is replaced by a nurse notification only. The nurse acts on the recommendation manually. The platform provides value at every adoption stage.*

10. Gateway Installation and Onboarding

The Gateway is designed for rapid deployment by a hospital IT team with interface engine experience. A standard single-ICU deployment can be completed in one to three days. No specialized clinical informatics expertise is required for the installation itself.

10.1 System Requirements

Requirement	Minimum Specification
Compute	4 vCPU, 16 GB RAM (VM or physical server on hospital infrastructure)

Storage	100 GB SSD (for 90-day local audit log retention)
Operating system	Ubuntu 22.04 LTS or RHEL 8+ (containerized via Docker)
Network	Access to hospital interface engine via MLLP; outbound HTTPS port 443 to api.medagent.ai
Interface engine	Mirth Connect, Rhapsody, Cloverleaf, or any HL7 MLLP-capable engine
HIPAA	Hospital signs AI MedAgent BAA prior to Gateway activation

10.2 Onboarding Steps

7. Hospital IT provisions VM or server per minimum requirements.
8. AI MedAgent issues a unique Gateway client certificate and OAuth 2.0 client credentials.
9. Gateway software is deployed from the AI MedAgent container registry (air-gapped installation available on request).
10. Hospital interface engine is configured with one channel per adapter type, routing the relevant HL7 message types to the Gateway MLLP listener ports.
11. Each connected system is verified in the AI MedAgent developer portal (api.medagent.ai) by sending a test event through the sandbox scenario stream.
12. Go-live: Gateway begins receiving live events. Reactive Agents start in monitoring-only mode for 24 hours before autonomous actions are enabled, allowing clinical staff to observe AI recommendations before trusting them.

10.3 Adding Data Sources Over Time

Adding a new data source at any later adoption stage requires two steps: configuring a new channel in the hospital interface engine pointing to the new adapter listener port, and registering the new source in the AI MedAgent portal. No changes to the Gateway software itself are required. This is the architectural commitment to incremental adoption — adding a new source never breaks what is already working.

11. Sandbox Environment

The sandbox provides a fully isolated test environment with synthetic patient data and scripted clinical scenarios that simulate real IHE PCD message streams. No PHI is used. Sandbox credentials are issued on request from api.medagent.ai.

Scenario	Patient Profile	Transactions Simulated
CABG_ICU_01	Post-cardiac surgery, Day 1	PCD-01, PCD-10, PCD-03, PCD-04, HL7 LRI, FHIR
SEPSIS_ICU_01	Septic shock	PCD-01, PCD-10, PCD-03, PCD-04 (critical), LRI (lactate)
GLYCEMIC_01	Post-surgical hyperglycemia	LRI (glucose, potassium), PCD-03, PCD-10

ARDS_VENT_01	ARDS on ventilator	PCD-01 (vent), PCD-04 (vent alarm), PCD-10
STAGE1_MONITOR	Monitors only — Stage 1	PCD-01 only; validates monitoring-only deployment

```
pip install medagent-sandbox
medagent-sandbox auth --client-id sandbox_xxx --key sandbox.pem
medagent-sandbox run CABG_ICU_01 --duration 3600
medagent-sandbox run STAGE1_MONITOR --duration 1800  # Test Stage 1 deployment
```

12. Integration Playbooks

Integration playbooks provide step-by-step guidance for connecting each source system category. Playbooks cover the IHE PCD transaction sequence, required UPES event mappings, sample HL7 messages, interface engine channel configuration, and common troubleshooting steps. All playbooks are published at api.medagent.ai and versioned with the SDK.

- EMR: Epic FHIR R4 (open.epic.com SMART backend services)
- EMR: Oracle Health (Cerner) FHIR R4
- EMR: Meditech Expanse FHIR R4
- ADC: Any HL7 RDS^O13 compliant dispensing system
- ADC: FHIR R4 MedicationDispense compliant systems
- Infusion Pump: Any IHE IPEC (PCD-10) and PIV (PCD-03) compliant pump controller
- Patient Monitor: Any IHE DEC (PCD-01) compliant monitor or monitor gateway
- Ventilator: Any IHE PCD-01 compliant ventilator or ventilator data module
- Device Alarms: Any IHE ACM (PCD-04) compliant device or alarm management system
- Laboratory: HL7 v2.5.1 LRI via Mirth Connect (standard channel template provided)
- Laboratory: FHIR R4 Observation via SMART on FHIR subscription
- Interface Engine: Mirth Connect channel templates for all adapter types (provided as importable XML)

13. Versioning and Roadmap

13.1 Version 2.0 (Current)

- Hybrid edge-cloud deployment architecture fully specified.
- On-premise Gateway with local reactive agent execution.
- AWS cloud AI engine with de-identified feature vector protocol.
- IHE PCD-01, PCD-03, PCD-04, PCD-10 fully specified and implemented.
- HL7 v2.5.1 LRI and FHIR R4 laboratory interfaces.
- FHIR R4 EMR adapter on SMART on FHIR backend services model.
- Six incremental adoption stages with value at each stage.
- Sandbox with Stage 1 scenario for monitoring-only deployments.

- Integration playbooks for all major system categories.

13.2 Version 3.0 Planned

- Home care continuum: wearable device adapters (Apple Watch, Oura Ring, Withings) for post-discharge monitoring.
- Cardiologist integration: FHIR R4 care team communication and shared decision-making resources.
- FHIR R5 Device and DeviceMetric resources as next-generation alternative to IHE PCD-01.
- Devices on FHIR (DoF): native FHIR representation of device data as the standard matures.
- Multi-patient ICU census view for charge nurses and intensivists.
- AI model explainability dashboard for pharmacy directors and medical staff committees.
- Population health reporting: anonymized outcomes data for participating hospitals.

14. About the Author

Daniel C. Pettus is a healthcare informatics executive, inventor, and independent advisor with more than 40 years of experience at the intersection of medical device connectivity, clinical informatics, and medication management technology.

He served as Vice President at Becton Dickinson, CareFusion, and Alaris, where he led the development and deployment of closed-loop EMR integration architecture that connected more than one million infusion pumps to the electronic medical record and grew acute-care infusion market share from 25 percent to over 60 percent. He was an active contributor to IHE Patient Care Device standards. He co-founded iMetrikus in 1999, one of the first cloud-based connected health platforms.

He holds two patents in medical device connectivity and was named Best Article of the Year by AAMI's BI&T journal in 2014 for his work on closed-loop IV pump integration with the EMR.

danpettus@outlook.com | danielpettus.com | linkedin.com/in/daniel-pettus-059413

AI MEDAGENT PLATFORM

SDK Supplement

Quickstart Guide · Error Reference · Rate Limits · End-to-End Sample Code

---

Version: 2.0 Supplement · Date: June 2026 · Author: Daniel C. Pettus · danpettus@outlook.com

This supplement adds four sections identified as standard components of best-in-class SDK documentation that were absent from the v2.0 specification: a Quickstart Guide for rapid onboarding, an Error Handling Reference, Rate Limits and Retry Behavior, and a complete End-to-End Sample Code appendix using arterial blood pressure monitoring as the working example.

A. Quickstart Guide

This guide gets you from zero to a working blood pressure observation flowing into the AI MedAgent Gateway in under 30 minutes. It uses the sandbox environment and does not require any hospital infrastructure.

A.1 What You Will Build

By the end of this guide you will have:

1. Installed the AI MedAgent Python SDK and sandbox CLI.
2. Authenticated with the sandbox Gateway using OAuth 2.0.
3. Received a simulated IHE PCD-01 arterial blood pressure observation.
4. Watched the MAP Reactive Agent evaluate the observation and generate a Decision.
5. Seen that Decision routed to a simulated pharmacy approval and nurse notification.

A.2 Prerequisites

- Python 3.10 or later.
- pip package manager.
- A code editor (VS Code recommended).
- Sandbox credentials (client ID and private key) from api.medagent.ai — registration is free.

A.3 Installation

```
# Step 1: Install the SDK and sandbox CLI
pip install medagent-sdk medagent-sandbox

# Step 2: Verify installation
python -c "import medagent; print(medagent.__version__)"
# Expected output: 2.1.0

medagent-sandbox --version
# Expected output: AI MedAgent Sandbox CLI 2.1.0
```

A.4 Authentication

The SDK uses OAuth 2.0 client credentials flow with a signed JWT assertion — the same pattern used by Open.Epic for backend system integrations. Your private key never leaves your server.

```
# Create a credentials file: medagent_config.py

from medagent import MedAgentClient

client = MedAgentClient(
    client_id   = 'sandbox_your_client_id',  # From api.medagent.ai
    private_key = './sandbox_private_key.pem',
```

```
    gateway_url = 'https://sandbox.api.medagent.ai',
    environment = 'SANDBOX'
)

# Authenticate — returns a bearer token valid for 60 minutes
token = client.authenticate()
print(f'Authenticated. Token expires: {token.expires_at}')
# Output: Authenticated. Token expires: 2026-06-05T15:32:00Z
```

A.5 Connect the Monitor Adapter

```
from medagent import MedAgentClient
from medagent.adapters import MonitorAdapter

client = MedAgentClient(
    client_id   = 'sandbox_your_client_id',
    private_key = './sandbox_private_key.pem',
    gateway_url = 'https://sandbox.api.medagent.ai',
    environment = 'SANDBOX'
)
client.authenticate()

# Create a PCD-01 monitor adapter
# In sandbox mode, this listens for simulated HL7 observations
monitor = MonitorAdapter(
    client       = client,
    profile      = 'IHE_DEC',
    protocol     = 'IHE_PCD_01',
    vitals       = ['MAP', 'SBP', 'DBP', 'HR'],
    poll_interval= 30,
    sandbox_mode = True
)

# Register an event handler
@monitor.on_observation
def handle_vital(event):
    print(f'Received: {event.event_type}')
    print(f'  Patient : {event.patient_id}')
    print(f'  MAP     : {event.payload["MAP"]} mmHg')
    print(f'  SBP/DBP : {event.payload["SBP"]}/{event.payload["DBP"]} mmHg')
    print(f'  Protocol: {event.source_protocol}')
    print(f'  Timestamp: {event.timestamp}')

# Start the adapter — begins receiving sandbox observations
monitor.start()
```

```
# Expected output when sandbox sends a PCD-01 observation:
# Received: DeviceObservation
#   Patient : SBX-PATIENT-001
#   MAP     : 62 mmHg
#   SBP/DBP : 98/44 mmHg
#   Protocol: IHE_PCD_01
#   Timestamp: 2026-06-05T14:32:17.442Z
```

A.6 Watch the AI Agent Respond

When MAP drops below 65 mmHg, the Reactive Agent fires automatically. You do not need to write any agent code — the agents are built into the Gateway. You simply listen for Decision events.

```
from medagent import MedAgentClient, DecisionListener
from medagent.adapters import MonitorAdapter

client = MedAgentClient(
    client_id='sandbox_your_client_id',
    private_key='./sandbox_private_key.pem',
    gateway_url='https://sandbox.api.medagent.ai',
    environment='SANDBOX'
)
client.authenticate()

# Monitor adapter (same as above)
monitor = MonitorAdapter(client=client, vitals=['MAP','SBP','DBP'],
                         sandbox_mode=True)

# Decision listener — fires when an AI agent generates a recommendation
decisions = DecisionListener(client=client)

@decisions.on_decision
def handle_decision(decision):
    print(f'AI Decision received:')
    print(f'  Drug    : {decision.medication}')
    print(f'  Action  : {decision.action}')
    print(f'  Delta   : +{decision.delta} {decision.unit}')
    print(f'  Reason  : {decision.rationale}')
    print(f'  Source  : {decision.source}')
    print(f'  Confidence: {decision.confidence:.0%}')
    print(f'  Nurse notify: {decision.nurse_notify}')
    print(f'  Pharmacy approval needed: {decision.requires_pharmacy_approval}')

# Start both listeners
monitor.start()
decisions.start()
```

```
# When sandbox triggers MAP < 65 scenario, output:
# AI Decision received:
#   Drug    : Vasopressor
#   Action  : INCREASE_RATE
#   Delta   : +0.02 mcg/kg/min
#   Reason  : MAP 62 mmHg < 65 target. Trend: declining (3 readings).
#   Source  : REACTIVE_AGENT_MAP
#   Confidence: 92%
#   Nurse notify: True
#   Pharmacy approval needed: False
```

B. Error Handling Reference

AI MedAgent uses standard HTTP status codes for REST API responses and a structured error object for all error conditions. Understanding error codes is essential for building reliable integrations, especially in clinical environments where silent failures are unacceptable.

B.1 Error Response Structure

```
# All API errors return this JSON structure
{
  "error": {
    "code":    "MAP_BELOW_MINIMUM_THRESHOLD",
    "message": "MAP value 28 mmHg is below the minimum valid range (30 mmHg).",
    "http_status": 422,
    "request_id": "req_8a7f3c2d-1b4e-4a9f-b2c1-7d8e9f0a1b2c",
    "timestamp": "2026-06-05T14:32:17Z",
    "docs_url":  "https://api.medagent.ai/errors/MAP_BELOW_MINIMUM_THRESHOLD"
  }
}
```

B.2 HTTP Status Codes

HTTP Code	Meaning	When It Occurs
200	OK	Request succeeded. Decision or event returned.
201	Created	New adapter registration or patient context created.
202	Accepted	Async request accepted (e.g., DL optimizer cycle queued). Poll for result.
400	Bad Request	Malformed UPES event, missing required field, or invalid parameter.
401	Unauthorized	Bearer token expired or invalid. Re-authenticate.
403	Forbidden	Valid token but insufficient scope for this operation.
404	Not Found	Patient ID or encounter ID not recognized by Gateway.
409	Conflict	Duplicate event_id submitted. Use idempotency key.
422	Unprocessable	Event parsed but clinically invalid (e.g., MAP value out of physiological range).
429	Too Many Requests	Rate limit exceeded. See Retry-After header.
500	Internal Error	Gateway error. Retry with exponential backoff. Page on-call if persistent.
503	Service Unavailable	Gateway in maintenance or cloud engine unreachable. Reactive agents continue locally.

B.3 Clinical Safety Error Codes

These error codes indicate conditions with direct patient safety implications. They are returned with HTTP 422 and trigger an immediate safety hold on the affected Reactive Agent.

Error Code	Description and Recommended Action
PUMP_ACK_LEVEL2_FAILED	PCD-03 pump programming order was not acknowledged by pump controller. Verify pump connectivity. Notify nurse immediately. Agent enters safety hold.
ALARM_CRITICAL_RECEIVED	PCD-04 critical alarm received from device. Affected agent suspended. Nurse notified via PCD-06. Resolve alarm before restarting agent.
PHARMACY_APPROVAL_TIMEOUT	DL Optimizer recommendation expired (300s) without pharmacy response. Decision discarded. Optimizer will re-evaluate on next cycle.
LAB_RESULT_CRITICAL_FLAG	Lab result with critical flag (C) received. Immediate agent evaluation triggered. Nurse notified regardless of decision outcome.
NURSE_OVERRIDE_RECEIVED	Nurse overrode AI Decision. Agent logs override, records free-text reason, and resumes monitoring. Does not re-recommend same action for 30 minutes.
PATIENT_CONTEXT_MISMATCH	Patient ID in UPES event does not match active encounter. Event quarantined pending IT review. No clinical action taken.

B.4 Error Handling in Code

```
from medagent import MedAgentClient, MedAgentError, ClinicalSafetyError
from medagent.adapters import MonitorAdapter
import time

client = MedAgentClient(
    client_id='sandbox_your_client_id',
    private_key='./sandbox_private_key.pem',
    gateway_url='https://sandbox.api.medagent.ai',
    environment='SANDBOX'
)

try:
    client.authenticate()

except MedAgentError as e:
    if e.http_status == 401:
        # Token expired — re-authenticate
        client.authenticate()
    elif e.http_status == 429:
        # Rate limited — respect Retry-After header
        retry_after = int(e.headers.get('Retry-After', 60))
        print(f'Rate limited. Retrying in {retry_after}s')
        time.sleep(retry_after)
        client.authenticate()
    elif e.http_status >= 500:
```

```
        # Server error — exponential backoff
        for attempt in range(3):
            time.sleep(2 ** attempt)  # 1s, 2s, 4s
            try:
                client.authenticate()
                break
            except MedAgentError:
                if attempt == 2:
                    # Alert on-call after 3 failures
                    alert_oncall('AI MedAgent Gateway unreachable')
                    raise

except ClinicalSafetyError as e:
    # Clinical safety errors are never swallowed silently
    # They always generate a nurse notification automatically
    print(f'Safety event: {e.code} — {e.message}')
    # Log to hospital audit system
    hospital_audit_log.write(e.to_dict())
```

C. Rate Limits and Retry Behavior

AI MedAgent rate limits are designed to protect the clinical data stream from runaway integrations while never throttling legitimate clinical event traffic. Limits are applied per Gateway instance, not per API call.

C.1 Rate Limits by Operation

Operation	Limit	Window	Notes
Inbound PCD-01 observations	600 events	Per minute	Configurable per care unit
Inbound PCD-10 pump events	1,200 events	Per minute	Higher limit for dense ICU environments
Inbound PCD-04 alarms	Unlimited	—	Alarms are never rate limited
Inbound lab results	120 events	Per minute	Typical ICU lab volume is 10-20/minute
Outbound PCD-03 orders	60 orders	Per minute	Safety limit; typical use is 1-5/minute
DL Optimizer feature vectors	10 per	Per minute	Each represents one patient's 4-hour cycle
Authentication token requests	10 per	Per hour	Tokens last 60 minutes; refresh proactively

> *PCD-04 alarms are never rate limited. A device firing 50 alarms per minute is a clinical event that the platform must process. Rate limiting alarms would be a patient safety failure.*

C.2 Rate Limit Headers

Every API response includes these headers:

```
X-RateLimit-Limit:     600        # Your limit for this operation
X-RateLimit-Remaining: 547        # Remaining calls in current window
X-RateLimit-Reset:     1749134400 # Unix timestamp when window resets
Retry-After:           23         # Seconds to wait (only on 429 responses)
```

C.3 Retry Strategy

The SDK implements automatic retry with exponential backoff and jitter for all retryable errors. The following retry policy is applied by default and can be overridden.

```
from medagent import MedAgentClient, RetryConfig

client = MedAgentClient(
    client_id   = 'sandbox_your_client_id',
    private_key = './sandbox_private_key.pem',
    gateway_url = 'https://sandbox.api.medagent.ai',
    retry_config= RetryConfig(
        max_attempts      = 3,
        initial_delay_sec = 1.0,
        backoff_multiplier= 2.0,   # 1s, 2s, 4s
        jitter_factor     = 0.1,   # Adds randomness to avoid thundering herd
        retryable_codes   = [429, 500, 502, 503, 504],
        # Clinical safety errors (422 with safety codes) are NEVER retried
    )
)
```

C.4 Cloud Connectivity Degradation

If the cloud AI engine becomes unreachable, the Gateway degrades gracefully:

- Reactive Agents: Continue operating normally. All six agents process local events and generate Decisions. No cloud connectivity required.
- DL Optimizer: Pauses. Queues feature vectors locally. Replays them when connectivity is restored. Maximum local queue: 72 hours of optimization cycles.
- Audit log: Continues writing locally. Syncs to cloud when connectivity is restored.
- Nurse notifications: Continue via local PCD-06 dissemination. No cloud dependency.

> **Clinical Guarantee:** Real-time medication management never stops due to a cloud outage. The Gateway is the clinical brain. The cloud is the learning layer.

D. End-to-End Sample Code

This appendix contains a complete, working sample that demonstrates the full AI MedAgent data flow using arterial blood pressure as the clinical example. Blood pressure was chosen because it is the simplest status-only monitoring scenario — no pump programming, no pharmacy interaction, just a monitor observation feeding an AI agent that generates a clinical recommendation.

The sample demonstrates all five platform layers end to end:

6. Authenticate with the AI MedAgent Gateway.
7. Receive an IHE PCD-01 arterial blood pressure observation from a patient monitor.
8. Watch the MAP Reactive Agent evaluate the reading and generate a Decision.
9. Send a de-identified feature vector to the cloud AI engine for DL analysis.
10. Receive a therapy optimization recommendation and route it to simulated pharmacy approval.

D.1 Clinical Scenario

Patient: 68-year-old male, post-CABG x3, ICU Day 1. Diagnosis: Z95.1 (presence of coronary angioplasty implant). Receiving vasopressor infusion at maintenance rate. Arterial line in place.

Event sequence:

11. Continuous arterial pressure monitoring via IHE PCD-01 every 30 seconds.
12. MAP reading of 62 mmHg received — below the 65 mmHg threshold.
13. MAP Reactive Agent detects declining trend over three consecutive readings.
14. Agent generates Decision: increase vasopressor by 0.02 mcg/kg/min.
15. Decision routed to pharmacy for notification (reactive agent) and nurse handheld via PCD-06.
16. DL Optimizer, running in parallel, analyzes 4-hour hemodynamic trajectory.
17. Optimizer recommends proactive therapy adjustment — routed through pharmacy approval.

D.2 Complete Sample Code

Save this as bp_monitoring_demo.py. Run it against the sandbox environment.

```
# ================================================================
# AI MedAgent Platform — End-to-End Sample
# Blood Pressure Monitoring: PCD-01 -> AI Agent -> Pharmacy
#
# Demonstrates: Authentication, PCD-01 adapter, Reactive Agent,
#               Decision handling, DL Optimizer, pharmacy routing
#
# Environment: SANDBOX (no hospital infrastructure required)
# SDK version: medagent-sdk >= 2.1.0
```

```python
# Run: python bp_monitoring_demo.py
# ================================================================

import asyncio
import logging
from datetime import datetime, timezone
from dataclasses import dataclass
from typing import Optional

from medagent import (
    MedAgentClient,
    DecisionListener,
    DeepLearningClient,
    MedAgentError,
    ClinicalSafetyError
)
from medagent.adapters import MonitorAdapter
from medagent.routing  import PharmacyRouter
from medagent.models   import (
    UPESEvent,
    Decision,
    FeatureVector,
    PharmacyNotification
)

# ── Logging ──────────────────────────────────────────────────────
logging.basicConfig(
    level   = logging.INFO,
    format  = '%(asctime)s | %(levelname)s | %(message)s',
    datefmt = '%H:%M:%S'
)
log = logging.getLogger('medagent.sample')

# ── Configuration ────────────────────────────────────────────────
SANDBOX_CLIENT_ID  = 'sandbox_your_client_id'   # From api.medagent.ai
SANDBOX_KEY_PATH   = './sandbox_private_key.pem'
GATEWAY_URL        = 'https://sandbox.api.medagent.ai'
PATIENT_ID         = 'SBX-PATIENT-001'           # Sandbox test patient
ENCOUNTER_ID       = 'SBX-ENC-2026-001'
CARE_UNIT          = 'CARDIAC_ICU'
MAP_TARGET_LOW     = 65    # mmHg — clinical threshold
SBP_TARGET_HIGH    = 140   # mmHg

# ── State: track MAP trend for agent decision context ────────────
map_history: list[float] = []
decisions_received: int  = 0

# ================================================================
# STEP 1: AUTHENTICATE WITH THE GATEWAY
# ================================================================

async def authenticate() -> MedAgentClient:
    """Authenticate using OAuth 2.0 client credentials + JWT.
    The Gateway validates the JWT signature with the public key
    registered during onboarding. No password is transmitted."""
```

```
    log.info('Authenticating with AI MedAgent sandbox Gateway...')

    client = MedAgentClient(
        client_id   = SANDBOX_CLIENT_ID,
        private_key = SANDBOX_KEY_PATH,
        gateway_url = GATEWAY_URL,
        environment = 'SANDBOX'
    )

    token = await client.authenticate_async()
    log.info(f'Authenticated. Scope: {token.scope}')
    log.info(f'Token expires: {token.expires_at}')
    return client

# ================================================================
# STEP 2: RECEIVE IHE PCD-01 BLOOD PRESSURE OBSERVATION
# ================================================================

def build_monitor_adapter(client: MedAgentClient) -> MonitorAdapter:
    """Configure a PCD-01 monitor adapter.
    In production: connect to hospital monitor gateway IP/port.
    In sandbox: receive simulated HL7 ORU^R01 observations."""

    return MonitorAdapter(
        client        = client,
        profile       = 'IHE_DEC',         # Device Enterprise Communication
        protocol      = 'IHE_PCD_01',       # Solicited observation reporting
        vitals        = ['MAP','SBP','DBP','HR','SPO2'],
        poll_interval = 30,                 # Seconds (production typical)
        patient_id    = PATIENT_ID,
        encounter_id  = ENCOUNTER_ID,
        care_unit     = CARE_UNIT,
        icd10_codes   = ['Z95.1'],           # CABG context
        sandbox_mode  = True
    )

def on_blood_pressure_observation(event: UPESEvent) -> None:
    """Handler called each time a PCD-01 observation arrives.
    The Gateway normalizes the raw HL7 ORU^R01 into a UPES event."""

    global map_history

    map_val = event.payload.get('MAP', 0.0)
    sbp_val = event.payload.get('SBP', 0.0)
    dbp_val = event.payload.get('DBP', 0.0)

    # Track trend for agent context
    map_history.append(map_val)
    if len(map_history) > 10:
        map_history.pop(0)

    status = 'LOW' if map_val < MAP_TARGET_LOW else 'OK'
    log.info(
        f'PCD-01 observation | '
        f'MAP: {map_val} mmHg [{status}] | '
        f'SBP/DBP: {sbp_val}/{dbp_val} mmHg | '
```

```
        f'Source: {event.source_system}'
    )

    # Log the raw UPES structure for diagnostic purposes
    log.debug(f'UPES event_id : {event.event_id}')
    log.debug(f'UPES timestamp: {event.timestamp}')
    log.debug(f'UPES protocol : {event.source_protocol}')

# ================================================================
# STEP 3: RECEIVE AI REACTIVE AGENT DECISION
# ================================================================

def on_ai_decision(decision: Decision) -> None:
    """Handler called when the MAP Reactive Agent generates a Decision.
    The agent runs inside the Gateway. This handler receives its output."""

    global decisions_received
    decisions_received += 1

    log.info('=' * 60)
    log.info('AI DECISION RECEIVED')
    log.info(f'  Decision ID : {decision.decision_id}')
    log.info(f'  Drug class  : {decision.medication}')
    log.info(f'  Action      : {decision.action}')
    log.info(f'  Delta       : +{decision.delta} {decision.unit}')
    log.info(f'  Rationale   : {decision.rationale}')
    log.info(f'  Confidence  : {decision.confidence:.0%}')
    log.info(f'  Source      : {decision.source}')
    log.info(f'  Nurse notify: {decision.nurse_notify}')
    log.info(f'  Pharm appr. : {decision.requires_pharmacy_approval}')
    log.info(f'  Expires in  : {decision.expiry_sec}s')
    log.info('=' * 60)

# ================================================================
# STEP 4: ROUTE DECISION TO PHARMACY
# ================================================================

async def route_to_pharmacy(
    decision : Decision,
    client   : MedAgentClient
) -> PharmacyNotification:
    """Route an AI Decision through the pharmacy notification workflow.
    For Reactive Agent decisions: notification is simultaneous with
    nurse alert (not a prerequisite).
    For DL Optimizer decisions: pharmacy approval is required before
    any pump programming order is transmitted."""

    router = PharmacyRouter(client=client)

    notification = PharmacyNotification(
        decision_id         = decision.decision_id,
        patient_id          = PATIENT_ID,
        encounter_id        = ENCOUNTER_ID,
        medication_class    = decision.medication,
        action              = decision.action,
        current_dose        = decision.current_dose,
```

```
        recommended_dose    = decision.recommended_dose,
        rationale           = decision.rationale,
        confidence          = decision.confidence,
        decision_source     = decision.source,
        requires_approval   = decision.requires_pharmacy_approval,
        timestamp           = datetime.now(timezone.utc)
    )

    result = await router.notify_async(notification)

    log.info(f'Pharmacy notified: {result.notification_id}')
    log.info(f'  Status   : {result.status}')
    log.info(f'  Sent to  : Pharmacy information system')
    log.info(f'  HL7 type : OMP^O09 (Pharmacy Order Message)')
    log.info(f'  Approved : {result.approved}')

    return result

# ================================================================
# STEP 5: SEND DE-IDENTIFIED FEATURE VECTOR TO CLOUD AI ENGINE
# ================================================================

async def run_dl_optimizer(client: MedAgentClient) -> Optional[Decision]:
    """Build a de-identified feature vector from the patient's clinical
    trajectory and send it to the AWS-hosted deep learning model.
    No patient identifiers are included in the feature vector."""

    log.info('Building de-identified feature vector for DL Optimizer...')

    # Feature vector contains statistical summaries, not raw values
    # Patient ID is replaced with anonymized cohort membership
    feature_vector = FeatureVector(
        cohort_match       = 'CABG_POST_OP_DAY1',  # De-identified population
        hospital_id        = 'HOS_0047',           # Anonymized institution code
        diagnosis_codes    = ['Z95.1'],
        vital_trend_4h = {
            'MAP':    {'mean': 68.2, 'std': 4.1, 'trend': 'declining'},
            'SBP':    {'mean': 112.4,'std': 8.3, 'trend': 'stable'},
            'HR':     {'mean': 84.1, 'std': 6.2, 'trend': 'stable'},
        },
        infusion_history = {
            'vasopressor': {'dose_changes': 3, 'hours_active': 6.5},
        },
        lab_trends = {
            'glucose':    {'last_value': 118, 'flag': 'N', 'trend': 'stable'},
            'potassium':  {'last_value': 3.9, 'flag': 'N'},
            'lactate':    {'last_value': 1.8, 'flag': 'N'},
        },
        alarm_count_4h     = 2,
        care_unit_type     = 'CARDIAC_ICU',
        care_day           = 1
    )

    # Send to cloud AI engine via encrypted HTTPS
    dl_client = DeepLearningClient(client=client)
    result    = await dl_client.optimize_async(
```

```
        feature_vector     = feature_vector,
        confidence_threshold = 0.85
    )

    if result.has_recommendation:
        log.info('DL Optimizer recommendation received:')
        log.info(f'  Action     : {result.decision.action}')
        log.info(f'  Confidence : {result.decision.confidence:.0%}')
        log.info(f'  Evidence   : {len(result.decision.evidence)} references')
        log.info(f'  Requires pharmacy approval: True')
        return result.decision
    else:
        log.info(f'DL Optimizer: no recommendation. '
                 f'Confidence {result.top_confidence:.0%} below threshold.')
        return None

# ================================================================
# MAIN ORCHESTRATOR
# ================================================================

async def main():
    log.info('AI MedAgent — Blood Pressure Monitoring Demo')
    log.info(f'Patient: {PATIENT_ID} | Unit: {CARE_UNIT}')
    log.info('-' * 60)

    # ── Step 1: Authenticate ──────────────────────────────────
    try:
        client = await authenticate()
    except MedAgentError as e:
        log.error(f'Authentication failed: {e.message}')
        return

    # ── Step 2: Start monitor adapter ─────────────────────────
    monitor  = build_monitor_adapter(client)
    monitor.on_observation(on_blood_pressure_observation)
    await monitor.start_async()
    log.info('Monitor adapter started. Receiving PCD-01 observations.')

    # ── Step 3: Listen for AI Decisions ───────────────────────
    decision_listener = DecisionListener(client=client)

    # Route each decision to pharmacy
    @decision_listener.on_decision
    async def handle_decision(decision: Decision):
        on_ai_decision(decision)
        await route_to_pharmacy(decision, client)

    await decision_listener.start_async()
    log.info('Decision listener started. Waiting for AI agent output.')

    # ── Step 4: Run DL Optimizer once ─────────────────────────
    log.info('Running DL Optimizer cycle...')
    await asyncio.sleep(5)  # Brief pause for sandbox to warm up

    dl_decision = await run_dl_optimizer(client)
    if dl_decision:
```

```
        # DL decisions require explicit pharmacy approval
        pharmacy_result = await route_to_pharmacy(dl_decision, client)
        if pharmacy_result.approved:
            log.info('DL recommendation pharmacy-approved.')
            log.info('Gateway will now transmit PCD-03 pump order.')
            log.info('Nurse notification queued via PCD-06.')

    # ── Step 5: Run for 3 minutes to observe agent decisions ──────
    log.info('')
    log.info('Running for 3 minutes. Sandbox will trigger MAP < 65.')
    log.info('Watch for AI Decision and pharmacy notification.')
    log.info('-' * 60)

    await asyncio.sleep(180)

    # ── Summary ──────────────────────────────────────────────────
    log.info('')
    log.info('Run complete.')
    log.info(f'Observations received : {len(map_history)} MAP readings')
    log.info(f'AI Decisions received : {decisions_received}')
    log.info(f'Final MAP trend       : {map_history[-3:]} mmHg')

    await monitor.stop_async()
    await decision_listener.stop_async()
    log.info('Adapters stopped. Demo complete.')


if __name__ == '__main__':
    asyncio.run(main())
```

D.3 Expected Console Output

Running against the sandbox, the output should look like this (times and IDs will differ):

```
14:32:00 | INFO | AI MedAgent — Blood Pressure Monitoring Demo
14:32:00 | INFO | Patient: SBX-PATIENT-001 | Unit: CARDIAC_ICU
14:32:00 | INFO | Authenticating with AI MedAgent sandbox Gateway...
14:32:01 | INFO | Authenticated. Scope: medagent/event.write medagent/decision.read
14:32:01 | INFO | Token expires: 2026-06-05T15:32:01Z
14:32:01 | INFO | Monitor adapter started. Receiving PCD-01 observations.
14:32:01 | INFO | Decision listener started. Waiting for AI agent output.
14:32:01 | INFO | Running DL Optimizer cycle...
14:32:06 | INFO | Building de-identified feature vector for DL Optimizer...
14:32:07 | INFO | DL Optimizer: no recommendation. Confidence 74% below threshold.
14:32:30 | INFO | PCD-01 observation | MAP: 71 mmHg [OK] | SBP/DBP: 118/55 mmHg
14:33:00 | INFO | PCD-01 observation | MAP: 68 mmHg [OK] | SBP/DBP: 112/51 mmHg
14:33:30 | INFO | PCD-01 observation | MAP: 62 mmHg [LOW] | SBP/DBP: 98/44 mmHg
14:33:30 | INFO | ============================================================
14:33:30 | INFO | AI DECISION RECEIVED
14:33:30 | INFO |   Decision ID : dec_7a3f2c1d-8b4e-4a9f
14:33:30 | INFO |   Drug class  : Vasopressor
14:33:30 | INFO |   Action      : INCREASE_RATE
14:33:30 | INFO |   Delta       : +0.02 mcg/kg/min
14:33:30 | INFO |   Rationale   : MAP 62 mmHg < 65 target. Trend: declining (3
readings).
```

```
14:33:30 | INFO |   Confidence  : 92%
14:33:30 | INFO |   Source      : REACTIVE_AGENT_MAP
14:33:30 | INFO |   Nurse notify: True
14:33:30 | INFO |   Pharm appr. : False
14:33:30 | INFO |   Expires in  : 300s
14:33:30 | INFO | ============================================================
14:33:30 | INFO | Pharmacy notified: ntf_9b2e1a3c
14:33:30 | INFO |   Status   : SENT
14:33:30 | INFO |   Sent to  : Pharmacy information system
14:33:30 | INFO |   HL7 type : OMP^O09 (Pharmacy Order Message)
14:33:30 | INFO |   Approved : True
```

D.4 What the Output Demonstrates

Reading this output against the SDK architecture:

- Line 1-4: Sandbox Gateway authentication via OAuth 2.0 JWT. Token issued with correct scope.
- Line 8: DL Optimizer ran but confidence (74%) did not reach the 85% threshold. No recommendation generated. This is correct — the platform does not act on low-confidence predictions.
- Lines 9-11: Three consecutive PCD-01 observations showing MAP declining from 71 to 62 mmHg. The Gateway tracks the trend internally across readings.
- Lines 12-23: MAP Reactive Agent detects MAP < 65 with declining trend and fires. Decision arrives at the handler within milliseconds of the third observation. Source is REACTIVE_AGENT_MAP. Pharmacy approval not required (reactive, immediate clinical response).
- Lines 24-28: Pharmacy notified via HL7 OMP^O09. Approved immediately in sandbox. In production, a pharmacist reviews this in their pharmacy system workflow.

> *The entire sequence from PCD-01 observation to AI Decision to pharmacy notification takes less than 500 milliseconds in the sandbox. In production on the on-premise Gateway, reactive agent processing is under 100 milliseconds. The nurse receives a PCD-06 handheld alert within 1-2 seconds of the observation.*

E. SDK Best Practice Compliance Review

This section records the gaps identified during a review of industry SDK documentation standards and the status of each in the v2.1 document set.

Required Section	Status	Notes
Quickstart Guide	Added — Section A	30-minute getting started with sandbox
Authentication documentation	Complete	OAuth 2.0 JWT with private key; token refresh

Error handling reference	Added — Section B	HTTP codes, clinical safety codes, error objects
Rate limits documentation	Added — Section C	Per-operation limits, headers, retry strategy
Retry and degradation behavior	Added — Section C	Exponential backoff, graceful cloud degradation
End-to-end code sample	Added — Section D	BP monitoring: PCD-01 -> agent -> pharmacy
Versioning and changelog	In v2.0 Section 13	Roadmap sections 13.1 and 13.2
Sandbox environment	In v2.0 Section 11	5 scenarios; CLI documented
Integration playbooks	In v2.0 Section 12	12 vendor-agnostic playbooks
Migration guide (v1.x to v2.0)	Planned for v2.1 final	To be added before public release
Interactive API explorer	Planned for developer portal	api.medagent.ai; not in document
OpenAPI specification	Planned for v2.1 final	Machine-readable spec for SDK generation