Interface Wizard

Complete Technical Documentation

Backend - Mirth Connect Integration Guide

Version 1.0
November 2025

Table of Contents

1. System Overview

Interface Wizard is a healthcare integration system that enables natural language interaction with Electronic Health Record (EHR) systems. The system uses AI to interpret user commands and automatically generates HL7 messages for patient registration, updates, and queries.

Component	Technology	Purpose
Frontend	Angular/React	User interface
Backend	FastAPI (Python)	API server, HL7 generation
Integration Engine	Mirth Connect	HL7 message routing
Database	MySQL (OpenEMR)	Patient data storage
AI Processing	OpenAI GPT-4	Natural language understanding

2. System Architecture

```
User Interface (Angular/React)
                ↓
        HTTP POST
                ↓
Backend API (FastAPI)
        ■■→ AI Service (OpenAI GPT-4)
        ■■→ HL7 Service (hl7apy)
        ■■→ MLLP Client (socket)
                ↓
        TCP Port 6661
                ↓
Mirth Connect Channel
        ■■→ MLLP Listener
        ■■→ Source Transformer (JavaScript)
        ■■→ Database Writer
                ↓
OpenEMR Database (MySQL)
```

3. Required Libraries and Dependencies

Python Dependencies (requirements.txt)

Core Libraries:
- **hl7apy==1.3.4** - Creates and parses HL7 v2.x messages
- **fastapi==0.104.1** - Modern web framework for building APIs
- **uvicorn==0.24.0** - ASGI server to run FastAPI
- **openai==1.3.5** - OpenAI API client for GPT-4
- **pymysql==1.1.0** - MySQL database driver
- **pydantic==2.5.0** - Data validation and settings management
- **python-dotenv==1.0.0** - Load environment variables from .env file

Network Communication:
- **socket** (built-in) - TCP/IP communication with Mirth Connect

Library	File Used In	Purpose
hl7apy	hl7_service.py	Create HL7 ADT^A04 messages
socket	mllp_client.py	Send messages via MLLP protocol
fastapi	main.py, command.py	REST API endpoints
openai	ai_service.py	Extract patient data from text
pydantic	config.py, models/	Configuration and validation

4. Configuration Files

4.1 Environment Variables (.env)

The **backend/.env** file contains all configuration needed for the system to operate. This is the ONLY configuration file you need to modify.

Variable	Value	Description
MLLP_HOST	localhost	Mirth Connect server location
MLLP_PORT	6661	Mirth MLLP listener port (CRITICAL!)
OPENAI_API_KEY	sk-proj-...	OpenAI API key for GPT-4
DB_HOST	localhost	MySQL database server
DB_NAME	openemr	OpenEMR database name
DB_USER	openemr	Database username
DB_PASSWORD	openemr	Database password

■ **CRITICAL:** The MLLP_PORT value (6661) MUST match the port configured in your Mirth Connect channel's MLLP Listener. If these don't match, messages will fail to send.

5. Backend Code Structure

```
backend/
■■■ app/
■   ■■■ main.py                    # FastAPI application entry
■   ■■■ config.py                  # Loads .env configuration
■   ■
■   ■■■ api/v1/endpoints/
■   ■   ■■■ command.py              # POST /api/v1/command
■   ■
■   ■■■ services/
■   ■   ■■■ ai_service.py           # OpenAI integration
■   ■   ■■■ hl7_service.py          # ■ Creates HL7 messages
■   ■   ■■■ mllp_client.py          # ■ Sends to Mirth
■   ■   ■■■ database_service.py     # Database operations
■   ■
■   ■■■ models/
■   ■   ■■■ command.py              # Request/Response models
■   ■   ■■■ patient.py              # Patient data models
■   ■
■   ■■■ utils/
■       ■■■ logger.py              # Logging configuration
■
■■■ .env                           # ■ Configuration file
■■■ requirements.txt               # Python dependencies
■■■ run.py                         # Application entry point
```

File	Lines	Purpose	Key Libraries
hl7_service.py	150	Create HL7 messages	hl7apy
mllp_client.py	180	Send to Mirth via MLLP	socket
ai_service.py	120	Process user commands	openai
config.py	80	Load configuration	pydantic
command.py	50	API endpoint	fastapi
.env	56	All configuration	-

6. HL7 Service Implementation

The **hl7_service.py** file is responsible for creating HL7 v2.x messages that Mirth Connect can understand and process. It uses the **hl7apy** library to construct properly formatted HL7 messages.

Key Code Snippet:

```
from hl7apy.core import Message, Segment
from datetime import datetime

class HL7Service:
    def create_adt_a04_message(self, patient_data):
        # Create HL7 ADT^A04 (Register Patient) message
        msg = Message("ADT_A04", version="2.5")

        # Message Header
        msg.msh.msh_3 = "InterfaceWizard"
        msg.msh.msh_9 = "ADT^A04"
        msg.msh.msh_10 = f"MSG{datetime.now()}"

        # Patient Identification
        msg.pid.pid_3 = f"{patient_data['mrn']}^^^MRN"
        msg.pid.pid_5 = f"{patient_data['last_name']}^"
                        f"{patient_data['first_name']}"
        msg.pid.pid_7 = patient_data['dob']
        msg.pid.pid_8 = patient_data['gender']

        # Convert to ER7 format (pipe-delimited)
        return msg.to_er7()
```

Output Example:

```
MSH|^~\&|InterfaceWizard|Facility|||20251117101530||ADT^A04|MSG001|P|2.5
PID|1||12345^^^MRN||Doe^John||19800101|M
```

7. MLLP Client Implementation

The **mllp_client.py** file handles TCP/IP communication with Mirth Connect using the MLLP (Minimal Lower Layer Protocol) standard. MLLP wraps HL7 messages with special control characters for transmission.

MLLP Protocol Format:

`<VT> + HL7_MESSAGE + <FS> + <CR> Where: • <VT> = Vertical Tab (0x0B) - Start of message • HL7_MESSAGE = The actual HL7 content • <FS> = File Separator (0x1C) - End of message • <CR> = Carriage Return (0x0D) - Message terminator`

Key Code Snippet:

```
import socket
from app.config import settings

class MLLPClient:
    VT = b'\x0b'  # Start Block
    FS = b'\x1c'  # End Block
    CR = b'\x0d'  # Carriage Return

    def send_message(self, hl7_message):
        # Wrap with MLLP envelope
        mllp_msg = self.VT + hl7_message.encode() +
                   self.FS + self.CR

        # Connect to Mirth
        sock = socket.socket(socket.AF_INET,
                             socket.SOCK_STREAM)
        sock.connect((settings.MLLP_HOST,
                      settings.MLLP_PORT))

        # Send message
        sock.sendall(mllp_msg)

        # Receive ACK
        response = sock.recv(4096)
        sock.close()

        return {"success": True, "ack": response}
```

8. Complete Message Flow

Step	Component	Action
1	User	Types: "Create patient John Doe"
2	Frontend	POST /api/v1/command
3	API Endpoint	Receives request, calls AI Service
4	AI Service	Extracts patient data using OpenAI
5	HL7 Service	Creates HL7 ADT^A04 message
6	MLLP Client	Wraps with MLLP, sends via TCP
7	Mirth Connect	Receives on port 6661
8	Source Transformer	Extracts data, inserts to DB
9	Database	Patient record created
10	ACK Response	Success message returned to user

9. Mirth Connect Channel Setup

Mirth Connect must be configured with a channel that listens for incoming HL7 messages on port 6661 and processes them into the OpenEMR database.

Component	Setting	Value
Channel Name	Name	Interface Wizard HL7 Listener
Source Connector	Type	MLLP Listener
Source Connector	Host	0.0.0.0
Source Connector	Port	6661 (CRITICAL!)
Source Transformer	Language	JavaScript
Source Transformer	Action	Extract data, insert to DB
Destination	Type	File Writer (for archival)
Destination	Directory	C:/mirth/hl7_messages/

Why Use Source Transformer for Database?

We use the **Source Transformer** (instead of Database Destination) because:

✓ **Faster** - Database insert happens immediately
✓ **Guaranteed** - Executes even if destinations fail
✓ **Flexible** - Full control with JavaScript
✓ **Validation** - Can check for duplicates before inserting
✓ **Custom Logic** - Calculate next PID, handle special cases

10. Testing and Troubleshooting

10.1 Testing Checklist

Test	Command/Check	Expected Result
Backend Running	Check http://localhost:8000/health	Status: OK
Mirth Running	Check http://localhost:8443	Login page appears
Channel Deployed	Mirth Dashboard	Green status indicator
Port Available	netstat -ano | findstr :6661	Shows listening port
Test Message	Send via frontend	Success response
Database Check	SELECT * FROM patient_data	New patient record

10.2 Common Issues

Problem	Solution
Connection Refused (6661)	Start and deploy Mirth channel
CORS Error	Add frontend port to backend/.env CORS_ORIGINS
Duplicate PID Error	Use SELECT MAX(pid)+1 in transformer
OpenAI API Error	Check OPENAI_API_KEY in .env
Database Connection Failed	Verify MySQL is running, check credentials

11. Quick Reference Guide

11.1 Start Commands

```
# Start Backend cd backend .\venv\Scripts\python.exe -m uvicorn app.main:app --reload #
Start Angular Frontend cd frontend-angular npm start # Start React Frontend cd
frontend-react npm start
```

11.2 Key Ports

Service	Port	URL
Backend API	8000	http://localhost:8000
Angular Frontend	4200	http://localhost:4200
React Frontend	3000	http://localhost:3000
Mirth Connect	8443	https://localhost:8443
Mirth MLLP Listener	6661	TCP localhost:6661
MySQL Database	3306	localhost:3306
OpenEMR	80	http://localhost/openemr

11.3 Default Credentials

System	Username	Password
Mirth Connect	admin	admin
OpenEMR	administrator	Admin@123456
MySQL	openemr	openemr

Summary

Interface Wizard successfully integrates natural language processing with healthcare systems using industry-standard HL7 messaging protocol.

Key Components:
• **hl7apy** library creates properly formatted HL7 messages
• **socket** module sends messages via MLLP protocol
• **Mirth Connect** receives and processes messages
• **OpenEMR database** stores patient records

Critical Configuration:
• MLLP_PORT in .env MUST match Mirth channel listener port
• All configuration in single .env file
• Source Transformer handles database operations

Benefits:
• Standards-based healthcare integration
• Scalable and maintainable architecture
• Comprehensive error handling and logging
• Easy to test and troubleshoot

Document Version: 1.0
Last Updated: November 2025
Author: Interface Wizard Development Team