# AI Communication Optimizer – Technical Integration Guide

## 1. Introduction

This document describes how to integrate the AI Communication Optimizer prototype into internal Quadient applications. It covers architecture, API usage, deployment procedures, security considerations, rate limits, and UI embedding options. The goal is to provide a clear reference that allows engineering teams to adopt the service with minimal friction.

## 2. System Overview

The AI Communication Optimizer consists of two main components:

• Backend API (FastAPI) responsible for message analysis and rewriting.

• Streamlit UI for demonstration and testing.

The API exposes two endpoints: /analyze and /rewrite. The backend uses lightweight NLP tools for analysis and OpenAI API for rewriting, allowing high-quality text generation with minimal latency.

## 3. Project Structure

The project is organized as follows:

ai-comm-optimizer/

backend/ – FastAPI application, NLP logic, rewrite service

ui/ – Streamlit demo interface

venv/ – isolated Python environment

requirements.txt – dependencies

README.md – project description

This separation allows the backend to be deployed as an independent microservice.

## 4. Backend API

Base URL (local demo):

http://127.0.0.1:8000

Endpoints:

POST /analyze

Analyzes sentiment, readability, entity extraction, and overall communication clarity.

Payload:

```
{

"text": "Your message here"

}
```

Response:

```
{

"analysis": {

"sentiment": 0.2,

"entities": [["Team", "ORG"]],

"word_count": 15,

"readability": 15.0

}

}
```

POST /rewrite

Rewrites a customer message into a more professional, friendly, or concise version.

Payload:

```
{

"text": "Your message here",
"tone": "professional"
}
```

Response:

```
{
"rewritten_text": "Improved version of the message..."
}
```

5. Security

Authentication

For production use within Quadient, recommend:

• API Key or OAuth2 (client credentials)

• Restrict access by IP (internal VPN only)

• Use HTTPS only

Secret Management

• Store API keys (OpenAI, internal secrets) in environment variables

• For Azure / cloud deployment: use KeyVault, AWS Secrets Manager, or Vault

Data Protection

• No logs should contain customer text bodies

• Sensitive data must be anonymized before processing when possible

• Enable request size limits to prevent accidental data uploads

6. Rate Limits

Recommended safeguards:

• 10 requests per second per user (local)

• 100 requests per minute global for rewrite

• Maximum text size: 4000 characters

• Apply circuit breakers if external LLM provider responds slowly

7. Deployment

Local Development

```
cd "D:\AI Models\ai-comm-optimizer"

.\venv\Scripts\activate

cd backend

uvicorn app:app --reload
```

Production Deployment (recommended options):

Option A: Docker container

• Build a container from the backend

• Deploy on Kubernetes or Quadient internal container environment

Option B: Azure App Service / AWS Lambda

• Lightweight and scalable

• Use environment variables for OpenAI credentials

Option C: Quadient On-Prem Node

• Deploy directly to an internal server

• Use systemd service for uptime

8. UI Embedding

The UI is a Streamlit app used only for demonstration purposes.

Launching the demo:

```
cd "D:\AI Models\ai-comm-optimizer"

.\venv\Scripts\activate

cd ui

streamlit run demo.py
```

Integration Options:

• Embed the service inside Inspire Designer for communication testing

• Integrate with Parcel Locker management tools (internal UI widget)

• Build a plugin for Quadient’s custom management dashboards

• Call the backend API directly from web or desktop applications

## 9. Example Integration Flow

1. A user writes a customer email inside any Quadient product.
2. The product sends the draft message to /analyze.
3. The UI highlights sentiment issues or unclear phrasing.
4. The user selects a tone ("professional", "friendly", "concise").
5. The product sends the message to /rewrite.
6. The improved version is returned and placed into the editor.

## 10. Error Handling

Typical errors:

• 400 – Invalid text payload

• 401 – Unauthorized (missing API key)

• 429 – Rate limit exceeded

• 500 – Upstream LLM provider error

Client integration recommendations:

• Retry with exponential backoff

• Cache frequent responses

• Validate text before submission

## 11. Logging and Monitoring

Monitor:

• Total requests

• Average processing time

• Rewrite failures

• External API latency (OpenAI)

Use:

• Prometheus/Grafana dashboards or Azure Insights

## 12. Future Extensions

• Tone detection instead of manual selection

• Multi-language rewriting

• Integration with Quadient Inspire workflows

• Automated compliance checks for regulated industries

This document serves as a deployment-ready guide for internal Quadient engineering teams to integrate the AI Communication Optimizer prototype into their systems.