MCP Email Workflow: Executive Summary

1. Model Context Protocol (MCP) Overview
MCP is an open protocol to connect LLM applications to external tools, services, and workflows. An MCP server exposes tool endpoints (e.g., send email, list messages) that agents can call. This standardises integrations, enabling modular, reusable, and auditable workflows. MCP is relevant for email automation where LLMs need to send, read, and summarise emails without managing low-level SMTP/IMAP code.

2. Email MCP Server: Tool Set

Tool	Purpose	Input/Output
send_email	Send messages	to, subject, body, attachments → message_id, status
draft_email	Create drafts	to, subject, body → draft_id
list_messages	Retrieve emails	folder, filter, limit → List[MessageSummary]
get_message	Fetch full email	message_id → MessageDetail
reply_to_message	Send reply	message_id, body → status

3. Integration Patterns
Even if not using LangChain, your framework can adopt the MCP pattern: - Wrap MCP server endpoints as client functions (e.g., Django module). - Agents or Celery tasks call the wrapper for sending, reading, or summarising emails. - Standardises email workflows, separates agent logic from provider specifics, supports auditing and scaling.

4. Benefits & Risks
Benefits: standardisation, modularity, separation of concerns, auditing, scalability, agentic workflows.
Risks: MCP server security, access control gaps, agent hallucination, deliverability limits, attachment/data leakage. Mitigation: audit servers, least privilege, logging, monitoring, secure credentials, guardrails on agent logic.

5. Implementation Steps (High-Level)
1. Decide: Build vs adopt MCP server for email. 2. Define tool schema (send_email, list_messages, get_message, reply_to_message). 3. Deploy MCP server (secure endpoints, logging, limits). 4. Client/wrapper in framework (call MCP tools via HTTP/API). 5. Integrate into agent logic or Celery tasks. 6. Apply security, governance, compliance (access control, attachment limits, monitoring). 7. Test sandbox before production rollout.

6. Recommendations
- Start with a minimal set of email tools. - Use wrapper modules in your Django + Celery stack. - Log all tool invocations for auditing. - Start with internal test mailboxes, then scale to stakeholders. - Monitor deliverability and replies, automate summarisation if needed. - Apply strict access control and security safeguards. - Consider future extension to agent-monitored inbox + automated follow-ups.